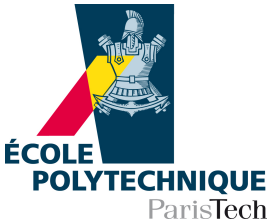
Scaling Algorithms and Tropical Methods in Numerical Matrix Analysis: Application to the Optimal Assignment Problem and to the Accurate Computation of Eigenvalues

Meisam Sharify



Thèse présentée pour obtenir le titre de

DOCTEUR DE L'ÉCOLE POLYTECHNIQUE

Spécialité: **Mathématiques Appliquées**

par

Meisam Sharify

Scaling Algorithms and Tropical Methods in Numerical Matrix Analysis:

Application to the Optimal Assignment Problem and to the Accurate Computation of Eigenvalues

Jury

Marianne Akian	Président du jury
Stéphane Gaubert	Directeur
Laurence Grammont	Examinateur
Laura Grigori	Examinateur
Andrei Sobolevski	Rapporteur
Françoise Tisseur	Rapporteur
Paul Van Dooren	Examinateur

September 2011

Abstract

Tropical algebra, which can be considered as a relatively new field in Mathematics, emerged in several branches of science such as optimization, synchronization of production and transportation, discrete event systems, optimal control, operations research, etc. The first part of this manuscript is devoted to the study of the numerical applications of tropical algebra.

We start by considering the classical problem of estimating the roots of a univariate complex polynomial. We prove several new bounds for the modulus of the roots of a polynomial exploiting tropical methods. These results are specially useful when considering polynomials whose coefficients have different orders of magnitude.

We next consider the problem of computing the eigenvalues of a matrix polynomial. Here, we introduce a general scaling technique, based on tropical algebra, which applies in particular to the companion form. This scaling is based on the construction of an auxiliary tropical polynomial function, depending only on the norms of the matrices. The roots (non-differentiability points) of this tropical polynomial provide a priori estimates of the modulus of the eigenvalues. This is justified in particular by a new location result, showing that under assumption involving condition numbers, there is one group of "large" eigenvalues, which have a maximal order of magnitude, given by the largest root of the auxiliary tropical polynomial. A similar result holds for a group of small eigenvalues. We show experimentally that this scaling improves the backward stability of the computations, particularly in situations when the data have various orders of magnitude.

We also study the problem of computing the tropical eigenvalues (non-differentiability points of the characteristic polynomial) of a tropical matrix polynomial. From the combinatorial perspective, this problem can be interpreted as finding the maximum weighted matching function in a bipartite graph whose arcs are valued by convex piecewise linear functions of a variable, λ. We developed an algorithm which computes the tropical eigenvalues in polynomial time.

In the second part of this thesis, we consider the problem of solving very

large instances of the optimal assignment problems (so that standard sequential algorithms cannot be used). We propose a new approach exploiting the connection between the optimal assignment problem and the entropy maximization problem. This approach leads to a preprocessing algorithm for the optimal assignment problem which is based on an iterative method that eliminates the entries not belonging to an optimal assignment. We consider two variants of the preprocessing algorithm, one by using the Sinkhorn iteration and the other by using Newton iteration. This preprocessing algorithm can reduce the initial problem to a much smaller problem in terms of memory requirements.

We also introduce a new iterative method based on a modification of the Sinkhorn scaling algorithm, in which a deformation parameter is slowly increased We prove that this iterative method, referred to as the *deformed-Sinkhorn iteration*, converges to a matrix whose nonzero entries are exactly those belonging to the optimal permutations. An estimation of the rate of convergence is also presented.

Abstract(French)

L'Algèbre tropicale peut être considérée comme un domaine relativement nouveau en mathématiques. Elle apparait dans plusieurs domaines telles que l'optimisation, la synchronisation de la production et du transport, les systèmes à événements discrets, le contrôle optimal, la recherche opérationnelle, etc. La première partie de ce manuscrit est consacrée a l'étude des applications de l'algèbre tropicale à l'analyse numérique matricielle.

Nous considérons tout d'abord le problème classique de l'estimation des racines d'un polynôme univarié. Nous prouvons plusieurs nouvelles bornes pour la valeur absolue des racines d'un polynôme en exploitant les méthodes tropicales. Ces résultats sont particulièrement utiles lorsque l'on considère des polynômes dont les coefficients ont des ordres de grandeur différents.

Nous examinons ensuite le problème du calcul des valeurs propres d'une matrice polynomiale. Ici, nous introduisons une technique de mise à l'échelle générale, basée sur l'algèbre tropicale, qui s'applique en particulier à la forme compagnon. Cette mise à l'échelle est basée sur la construction d'une fonction polynomiale tropicale auxiliaire, ne dépendant que de la norme des matrices. Les racines (les points de non-différentiabilité) de ce polynôme tropical fournissent une pré-estimation de la valeur absolue des valeurs propres. Ceci se justifie en particulier par un nouveau résultat montrant que sous certaines hypothèses faites sur le conditionnement, il existe un groupe de valeurs propres bornées en norme. L'ordre de grandeur de ces bornes est fourni par la plus grande racine du polynôme tropical auxiliaire. Un résultat similaire est valable pour un groupe de petites valeurs propres. Nous montrons expérimentalement que cette mise à l'échelle améliore la stabilité numérique, en particulier dans des situations où les données ont des ordres de grandeur différents.

Nous étudions également le problème du calcul des valeurs propres tropicales (les points de non-différentiabilité du polynôme caractéristique) d'une matrice polynômiale tropicale. Du point de vue combinatoire, ce problème est équivalent à trouver une fonction de couplage: la valeur d'un couplage de poids maximum

dans un graphe biparti dont les arcs sont valués par des fonctions convexes et linéaires par morceaux. Nous avons développé un algorithme qui calcule ces valeurs propres tropicales en temps polynomial.

Dans la deuxième partie de cette thèse, nous nous intéressons à la résolution de problèmes d'affectation optimale de très grande taille, pour lesquels les algorithmes séquentiels classiques ne sont pas efficaces. Nous proposons une nouvelle approche qui exploite le lien entre le problème d'affectation optimale et le problème de maximisation d'entropie. Cette approche conduit à un algorithme de prétraitement pour le problème d'affectation optimale qui est basé sur une méthode itérative qui élimine les entrées n'appartenant pas à une affectation optimale. Nous considérons deux variantes itératives de l'algorithme de prétraitement, l'une utilise la méthode Sinkhorn et l'autre utilise la méthode de Newton. Cet algorithme de prétraitement ramène le problème initial à un problème beaucoup plus petit en termes de besoins en mémoire.

Nous introduisons également une nouvelle méthode itérative basée sur une modification de l'algorithme Sinkhorn, dans lequel un paramètre de déformation est lentement augmenté. Nous prouvons que cette méthode itérative(*itération de Sinkhorn déformée*) converge vers une matrice dont les entrées non nulles sont exactement celles qui appartiennent aux permutations optimales. Une estimation du taux de convergence est également présentée.

Acknowledgments

I would like to express my deep gratitude to my thesis advisor Stéphane Gaubert for his constant guidance, support and encouragement in each and every step of my PhD studies such as mathematical analysis, design of algorithms, numerical analysis, writing articles and even preparing the presentations. Stéphane's profound knowledge and vision in mathematics helped me improve my understanding of the subject at hand. I have always appreciated his kindness and modesty which made it enjoyable to work with him.

I am also indebted to Marianne Akian who also supported me throughout the entire PhD period. It was a good opportunity for me to collaborate with her during my PhD studies.

My gratefulness goes to Laura Grigori for offering me the opportunity to work on the parallel optimal assignment problem and for her co-supervision on the second part of this thesis.

I am grateful to the thesis reviewers, Françoise Tisseur and Andrei Sobolevski for their helpful comments and for the time they devoted to carefully read the dissertation. I would like to thank the other members of the thesis committee, Laurence Grammont and Paul Van Dooren.

I would like to thank Wallis Filippi and Sandra Schnakenbourg, the secretaries in CMAP for their constant help during my PhD studies.

I would also like to extend my thanks to my officemates, Jean-Baptiste Bellet who helped me to integrate into the French system, and who made the work easier by his sense of humor; Denis Villemonais, for his mathematical remarks and Abdul Wahab for his suggestions on my thesis. I would like to acknowledge the generous help of my friend, Majid for reviewing the English version of the manuscript and his encouraging support.

And finally, my deepest gratitude goes to my family for all their love and support over the years. I want to thank them for just being there when I needed them, supporting and encouraging me during tough times.

Contents

CHAPTER 1

Introduction

1.1 Numerical applications of tropical algebra

Tropical algebra can be considered as a relatively new field in Mathematics. The adjective tropical is given in the honor of the Brazilian mathematician, Imre Simon, who was one of the pioneers of the field [Pin98]. Imre Simon introduced the semiring $(\mathbb{N} \cup \{+\infty\}, \min, +)$ in the context of automata theory in theoretical computer science [Sim78]. In the late 80's in France, the term *algèbres exotiques* was used (for example a seminar in 1987, which took place in Issy-les-Moulineaux, France under the title: "'Algèbres Exotiques et Systèmes à Evénements Discrets"'). Cuninghame-Green [CGM80] introduced the name "max-algebra". The name "max-plus" has been more recently used in particular in the control and discrete event systems communities [BCOQ92, CpQ99, McE06, JvdWJ06]. Maslov and its school [MS92, KM97, LMS01] introduced the name "idempotent analysis". It is reported in [Gau09] that at the BRIMS HP-Labs workshop on Idempotency in Bristol(1994), which was organized by J. Gunawardena, a discussion took place on how the field should be named. The names "'max-plus"', "'exotic"', "'tropical"', "'idempotent"' were considered. At that time, strictly speaking, tropical referred to the semiring $(\mathbb{N} \cup \{+\infty\}; \min; +)$, whereas the semirings $(\mathbb{N} \cup \{-\infty\}; \max; +)$ and $(\mathbb{Z} \cup \{-\infty\}; \max; +)$ were sometimes called boreal and equatorial. Also the terms, "'max-plus"' and "'min-plus"' semiring refers to $(\mathbb{R} \cup \{-\infty\}; \max; +)$ and $(\mathbb{R} \cup \{+\infty\}; \min; +)$. Nowadays, tropical is used as a

general term, whereas max-plus, min-plus and max-times semirings are models of tropical structures.

This field emerged in several branches of science such as optimization [Vor67, Zim77, But10], synchronization of production and transportation [CG60], discrete event systems [CMQV84, BCOQ92], optimal control [KM97, CGQ01, AGL08], operations research [GM84], tropical geometry [Vir01, Mik05, FPT00, RGST05] etc.

The scope of this work is Numerical analysis and Combinatorics. The first part of this thesis is devoted to the study of the numerical applications of tropical algebra. We start in Chapter 3 by considering the classical problem of estimating the roots of a polynomial. Some of the known bounds for the modulus of the roots of a polynomial, in particular, the generalizations and refinements of the classical bound of Cauchy, Hadamard, Specht and Ostrowski [Mar66, Had93, Spe38, Ost40a, Ost40b] turn out to be of tropical nature.

In problems of numerical analysis, it is of primary importance to have a priori estimates of the order of magnitude of the quantities to be computed, like the roots of a polynomial, or the eigenvalues of a matrix, in order to perform appropriate scalings.

The roots of tropical polynomial can be defined as the set of nondifferentiability points of a convex piecewise linear function, or equivalently, as the slopes of a certain Newton polygon in which the log of the modulus of the coefficients of the polynomial are thought of as a valuation. To any polynomial with complex coefficients, one can associate a tropical polynomial, depending only on the modulus of the coefficients of the original polynomials. A theorem of Hadamard and Ostrowski shows that the modulus of the complex roots can be bounded in terms of the tropical roots. One interest of this result is that the tropical roots can be computed in linear time, since it turns out to be a special instance of convex hull computation in dimension 2 in which the points are already sorted. We describe such an implementation. Then, we provide some new bounds for the modulus of the roots exploiting tropical methods.

These results are specially useful when considering polynomials whose coefficients have a large difference in order of magnitude. These polynomials maybe considered as difficult examples for the numerical algorithms; however, the modulus of the roots of this family of polynomials can be well estimated by the tropical method.

Another problem that we considered here, in Chapter 4, is the problem of computing the eigenvalues of a matrix polynomial. A common way to solve this problem is to convert P into a "linearized" matrix pencil with the same spectrum as P and solve the eigenproblem of the later problem [MS73]. The problem of finding the good linearizations and the good scalings, in the sense that the relative error of an eigenvalue or the backward error of an eigenpair should be

small and that certain properties of symmetry when they are present should be preserved, has received a considerable attention, see [FLVD04, Tis00, HLT07, HMT06, AA09, TDM10]. Here, we introduce a general scaling technique, based on tropical algebra, which applies in particular to the companion form. This scaling relies only on the norms of the matrices. We show experimentally that this scaling improves the backward stability of the computations, particularly in situations when the data have various orders of magnitude. We also proved that in non-degenerate cases, when the maximal tropical root is well separated from the other ones, then, there is a group of "large" eigenvalues, which have a maximal order of magnitude, and we bound explicitly the modulus of these eigenvalues in terms of the maximal tropical root. A similar result holds for a group of small eigenvalues by taking the smallest tropical root.

We also study, in Chapter 5 the problem of computing the tropical eigenvalues of a tropical matrix polynomial. From the combinatorial perspective, this problem can be interpreted as finding the maximum weighted matching function in a bipartite graph whose arcs are convex Piecewise linear functions of a variable, λ. Our motivation for this problem is to use these information in the computation of the classical eigenvalues of a matrix polynomial. Indeed, in degenerate cases (when certain matrices are ill conditioned), the scaling of Chapter 4 based only on the norms of the matrices behaves poorly. However, the tropical eigenvalues (which depend on the modulus of all the entries of the matrices, and not only on their norms), provide more accurate a priori estimates of the classical eigenvalues. This is inspired by a work of Akian, Bapat and Gaubert [ABG05, ABG04] where the tropical eigenvalues were shown to determine the order of magnitude (valuation) of the eigenvalues of a perturbed matrix pencil. We developed an algorithm, which computes the tropical eigenvalues in $O(n^4 d)$ time where d is the degree of the input matrix polynomial and n is the dimension of the matrices. This algorithm is a generalization of the idea of the algorithm proposed by Burkard and Butkovic [BB03].

1.2 Optimal Assignment Problem

In the second part of this thesis, we consider the optimal assignment problem, which is among of the most classical ones in combinatorial optimization. Several applications of this problem arise in different fields of applied science such as bioinformatics for the protein structure alignment problem [Hol93, LCL04], VLSI design [HCLH90], image processing and computer vision [CWC+96] and the pivoting problem in the solution of large linear systems of equations[ON96, DK00, LD03].

We propose a new approach exploiting the connection between the optimal assignment problem and the entropy maximization problem. We consider a one-

parameter family of relative entropy maximization problem, in which the exponential of the weights of the optimal assignment problem play the role of a reference measure. reward to be maximize is nothing but the reward of the assignment problem, augmented by an entropy term, the importance of which depends on the deformation parameter. We show that the solution of the relative entropy maximization problem converges to an optimal solution of the optimal assignment problem with an error term, which is exponentially small in the deformation parameter.

This approach leads to a preprocessing algorithm for the optimal assignment problem, which is developed in Chapter 7. The latter algorithm is based on an iterative method that eliminates the entries not belonging to an optimal assignment. We consider two variants of the preprocessing algorithm, one by using the Sinkhorn iteration [SK67] and the other by using Newton iteration [KR07]. The advantage of Sinkhorn iteration is that it can be efficiently implemented in parallel [ADRU08]. On the other hand, the advantage of Newton method is the speed of it's convergence. The implemented code and several experimental results for both variants are presented.

An interesting application of this new approach is the solution of large scale dense optimal assignment problems. Several efforts have been made to solve this problem [BT09, LO94]. A well-known application arises from the approximation algorithms and heuristics for solving the Asymmetric Traveling Salesman Problem or the Vehicle Routing Problem. There are also some applications in object recognition and computer vision. An application to cosmology (reconstruction of the early universe) can be found in the work of Brenier et al. [BFH$^+$03]. Models of large dense random assignment problems are also considered in [MPV87, Ch. VII] from the point of view of statistical physics.

By using the preprocessing method one can reduce the initial problem to a much smaller problem in terms of memory requirements. Specially, for very large dense optimal assignment problems, which can not be stored in one machine, the parallel Sinkhorn iteration can be used to reduce the size of the problem so that the reduced problem becomes executable on a sequential machine

We also introduce a new iterative method based on a modification of the Sinkhorn scaling algorithm, in which the deformation parameter is slowly increased (this procedure is reminiscent from simulated annealing, the parameter p playing the role of the inverse of the temperature). We prove that, the iterative method, referred to as *deformed-Sinkhorn iteration*, converges to a matrix whose nonzero entries are exactly those belonging to the optimal permutations. An estimation of the rate of convergence is also presented.

1.3 Thesis Outline

This thesis is divided in two parts. Part I is devoted to the numerical applications of tropical geometry. Also, a combinatorial problem in this field has been investigated. The sketch of this part is as follows:

- In Chapter 2, we provide some background on tropical linear algebra. We show that the tropical roots can be computed in linear time, $O(n)$, where n is the degree of $p(x)$. Also, we discuss the problem of perturbation of eigenvalues of matrix polynomials.
- In Chapter 3, we study the connection between the roots of a polynomial and the tropical roots of an associated max-times polynomial.
- In Chapter 4, We introduce a general scaling technique, based on tropical algebra for the problem of computing the eigenvalues of a matrix polynomial in order to increase the stability of the computation.
- In Chapter 5, we study the problem of computing the tropical eigenvalues of a tropical matrix polynomial.

Part II is devoted to the optimal assignment problem:

- In Chapter 6 We provide a short background on the optimal assignment problem and entropy maximization problem. The main theoretical results showing that the solution of a deformed entropy maximization problem asymptotically leads to the solution of optimal assignment problem, when the deformed parameter tends to infinity, is presented in this chapter. The theoretical results about the exponential convergence speed are also presented here.
- In Chapter 7 We present a preprocessing algorithm for the optimal assignment problem. Two variants of the algorithm, one based on Sinkhorn iteration [SK67] and the other based on Newton method [KR07] have been studied here. Also *deformed-Sinkhorn iteration* method is introduced and studied.

In Appendix A, we present the Scilab implementation of an algorithm, which computes the tropical roots in linear time. Appendix B includes Matlab and Scilab implementations of tropical scaling for a matrix polynomial eigenvalue problem. In Appendix C, we present the Scilab implementation of an algorithm, which computes the tropical eigenvalues. In Appendix D, we provide a Matlab implementation of the Newton method, which is appeared in the work of Knight and Ruiz [KR07].

Part I

Tropical Algebra and Numerical Methods

CHAPTER 2

Tropical mathematics and linear algebra

In this chapter we will provide some preliminary definitions and terminologies which will be used in the next chapters. Most of these definitions can be found in [BCOQ92].

2.1 Max-plus, Min-plus and Max-times semifields

Definition 2.1.1. A *semiring* is a set $\mathcal{S}$ with two binary operations, addition, denoted by $+$, and multiplication, denoted by $\cdot$ or by concatenation, such that:

- $\mathcal{S}$ is an abelian monoid under addition (with neutral element denoted by 0 and called zero);
- $\mathcal{S}$ is a semigroup under multiplication (with neutral element denoted by 1 and called unit);
- multiplication is distributive over addition on both sides;
- $s0 = 0s = 0$ for all $s \in \mathcal{S}$.

Example 2.1.1. Some basic examples of semirings consisting of the set $\mathbb{N}$, or the set $\mathbb{Q}^+$ of non-negative rational numbers, or of the set $\mathbb{R}^+$ of non-negative real numbers occupied with the usual addition and multiplication.

Definition 2.1.2. A semifield $\mathcal{K}$ is a semiring in which all the nonzero elements have a multiplicative inverse.

Definition 2.1.3. A semiring or an abelian monoid $\mathcal{S}$ is called *idempotent* if $a + a = a$ for all $a \in \mathcal{S}$.

Definition 2.1.4. A semiring $\mathcal{S}$ is called *zero-sum free* or *antinegative* if $a+b = 0$ implies $a = b = 0$ for all $a, b \in \mathcal{S}$.

Remark 1. An idempotent semiring is zero-sum free.

Definition 2.1.5. A semiring $\mathcal{S}$ is called *commutative* if the multiplication is commutative, i.e. $a \cdot b = b \cdot a$ for all $a, b \in \mathcal{S}$.

The max-plus semiring $\mathbb{R}_{\max}$, is the set $\mathbb{R} \cup \{-\infty\}$, equipped with maximum as addition, and the usual addition as multiplication. It is traditional to use the notation $\oplus$ for max (so $2 \oplus 3 = 3$), and $\otimes$ for $+$ (so $1 \otimes 1 = 2$). We denote by $\mathbb{0}$ the zero element of the semiring, which is such that $\mathbb{0} \oplus a = a$, here $\mathbb{0} = -\infty$, and by $\mathbb{1}$ the unit element of the semiring, which is such that $\mathbb{1} \otimes a = a \otimes \mathbb{1} = a$, here $\mathbb{1} = 0$. We refer the reader to [BCOQ92, KM97, ABG06] for more background.

The min-plus semiring, $\mathbb{R}_{\min}$, is defined as the set $\mathbb{R} \cup \{+\infty\}$, equipped with minimum as addition, and the usual addition as multiplication. This semiring is isomorphic to $\mathbb{R}_{\max}$ by the map $x \mapsto -x$. Thus, the zero element of this semiring is $+\infty$ and the unit element is 0.

A variant of the $\mathbb{R}_{\max}$ semiring is the max-times semiring $\mathbb{R}_{\max,\times}$, which is the set of nonnegative real numbers $\mathbb{R}^+$, equipped with max as addition, and $\times$ as multiplication. This semiring is isomorphic to $\mathbb{R}_{\max}$ by the map $x \mapsto \log x$. So, every notion defined over $\mathbb{R}_{\max}$ has an $\mathbb{R}_{\max,\times}$ analogue that we shall not redefine explicitly. In the sequel, the word "tropical" will refer indifferently to any of these algebraic structures.

Proposition 2.1.1. *The algebraic structures* $\mathbb{R}_{\max}$, $\mathbb{R}_{\min}$ *and* $\mathbb{R}_{\max,\times}$ *are idempotent commutative semifields.*

Proof. The proof is straightforward. □

2.2 Tropical polynomials

Consider a max-plus (formal) polynomial of degree n in one variable, i.e. a formal expression $P = \bigoplus_{0 \le k \le n} P_k X^k$ in which the coefficients P_k belong to $\mathbb{R}_{\max}$, and the associated numerical polynomial, which, with the notation of the

classical algebra, can be written as $p(x) = \max_{0\leq k\leq n} P_k + kx$. Cuninghame-Green and Meijer showed [CGM80] that the analogue of the fundamental theorem of algebra holds in the max-plus setting, i.e., that $p(x)$ can be written uniquely as $p(x) = P_n + \sum_{1\leq k\leq n} \max(x, c_k)$, where $c_1, \ldots, c_n \in \mathbb{R}_{\max}$ are the *roots*, i.e., the points at which the maximum attained at least twice. This is a special case of more general notions which have arisen recently in tropical geometry [IMS07]. The *multiplicity* of the root c is the cardinality of the set $\{k \in \{1, \ldots, n\} \mid c_k = c\}$. Define the *Newton polygon* $\Delta(P)$ of P to be the upper boundary of the convex hull of the set of points (k, P_k), $k = 0, \ldots, n$. This boundary consists of a number of linear segments. An application of Legendre-Fenchel duality (see [ABG05, Proposition 2.10]) shows that the opposite of the slopes of these segments are precisely the tropical roots, and that the multiplicity of a root coincides with the horizontal width of the corresponding segment. (actually, min-plus polynomials are considered in [ABG05], but the max-plus case reduces to the min-plus case by an obvious change of variable). Since the Graham scan algorithm [Gra72] allows us to compute the convex hull of a finite set of points by making $O(n)$ arithmetical operations and comparisons, provided that the given set of points is already sorted by abscissa, we get the following result.

Proposition 2.2.1. *The roots of a max-plus polynomial in one variable can be computed in linear time.* □

The case of a max-times polynomial will be reduced to the max-plus case by replacing every coefficient by its logarithm. The exponentials of the roots of the transformed polynomial are the roots of the original polynomial.

Example 2.2.1. Consider the max-times polynomial $\mathsf{t}p(x) = 1 \oplus 15x^2 \oplus 8x^3 \oplus 70x^4 \oplus 10^{-1}x^7$. The Newton polygon corresponding to this polynomial is shown in Figure 2.1. The vertices of the Newton polygon are $[0, 0]$, $[2, \log(15)]$, $[4, \log(70)]$, $[7, \log(10^{-1})]$ and the tropical roots are $\alpha_1 = \exp(-\frac{log(15)}{2}) = \frac{1}{\sqrt{15}} \approx 0.258$ with multiplicity 2, $\alpha_2 = \exp(-\frac{\log(70)-\log(15)}{2}) = \sqrt{\frac{15}{70}} \approx 0.463$ with multiplicity 2 and $\alpha_3 = \exp(-\frac{\log(10^{-1})-\log(70)}{3}) = \sqrt[3]{700} \approx 8.879$ with multiplicity 3.

In the sequel, we will refer to the roots of a max-plus or max-times polynomial by tropical roots. The Scilab code to compute the tropical roots in linear time, is presented in Appendix A. In chapter 3 we will consider the tropical roots of a max-times polynomial corresponding to a formal polynomial. We will show that the tropical roots can provide a priori estimation of the modulus of the formal roots when the tropical roots are well separated.

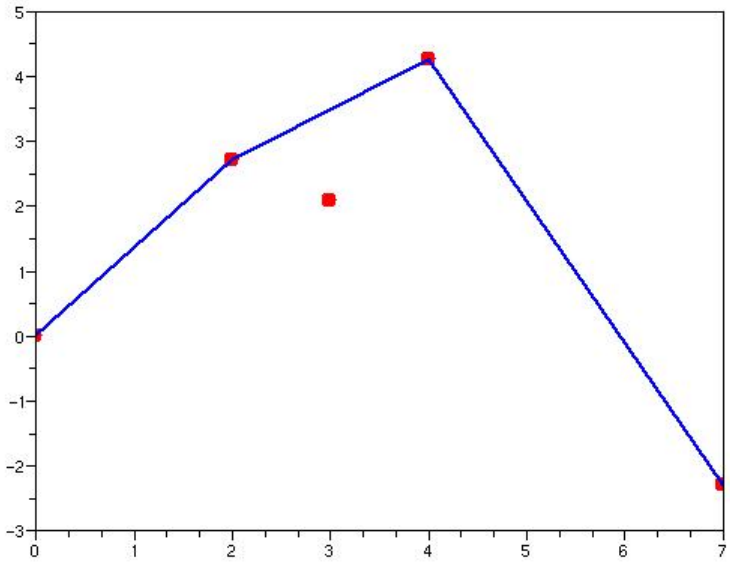


Figure 2.1: Newton polygon corresponding to the max-times polynomial $\mathtt{t}p(x) = 1 \oplus 15x^2 \oplus 8x^3 \oplus 70x^4 \oplus 10^{-1}x^7$.

2.3 Matrices and tropical algebra

Definition 2.3.1. A semimodule $\mathcal{M}$ over an idempotent semifield $\mathcal{K}$ with operations $\oplus$ and $\otimes$, zero element, $\mathbb{0}$, and identity element, $\mathbb{1}$, is a set endowed with

- an internal operation also denoted by $\oplus$ with a zero element also denoted by $\mathbb{0}$;
- an external operation defined on $\mathcal{K} \times \mathcal{M}$ with values in $\mathcal{M}$ indicated by the simple juxtaposition of the scalar and vector symbols;

which satisfy the following properties:

- $\oplus$ is associative, commutative;
- $\alpha(x \oplus y) = \alpha x \oplus \alpha y$;
- $(\alpha \oplus \beta)x = \alpha x \oplus \beta x$;
- $\alpha(\beta x) = (\alpha\beta)x$;
- $\mathbb{1}x = x$;

- $\mathbb{0}x = \mathbb{0}$;

for all $\alpha, \beta \in \mathcal{S}$ and all $x, y \in \mathcal{M}$.

Remark 2. A semimodule can be viewed as a vector space without additive inverse.

Example 2.3.1. The set of vectors, $(\mathbb{R}_{\max})^n$, is a semimodule over $\mathbb{R}_{\max}$ for which the zero element is $(\mathbb{0}, \mathbb{0}, \ldots, \mathbb{0})$.

Definition 2.3.2. A semimodule with an additional internal operation also denoted by $\otimes$ is called an idempotent algebra if $\otimes$ is associative, if it has an identity element also denoted by $\mathbb{1}$, and if it is distributive with respect to $\oplus$.

The set $\mathbb{R}_{\max}^{n\times n}$ denotes the set of $n \times n$ matrices with coefficients in $\mathbb{R}_{\max}$ endowed with the following two internal operations:

- the componentwise addition denoted $\oplus$;
- the matrix multiplication, $\otimes$, defined as $(A \otimes B)_{ij} = \oplus_{k=1}^{n} (A)_{ik} \otimes (B)_{kj}$

and the external operation:

- $\forall \alpha \in \mathbb{R}_{\max}, \forall A \in \mathbb{R}_{\max}^{n\times n}, \alpha A = (\alpha(A)_{ij})$.

It is an idempotent algebra with

- the zero matrix, again denoted $\mathbb{0}$, which has all its entries equal to $\mathbb{0}$;
- the identity matrix, denoted by $\mathbb{I}$, which has the diagonal entries equal to $\mathbb{1}$ and the other entries equal to $\mathbb{0}$.

2.4 Eigenvalues and Eigenvectors in $\mathbb{R}_{\max}^{n\times n}$

For a given matrix $A \in \mathbb{R}_{\max}^{n\times n}$, let $\mathcal{G}(A)$ denotes the graph corresponding to the matrix A with set of nodes 1, ..., n in which $(A)_{ij} \neq \mathbb{0}$ is the weight of arc from node i to node j. The matrix A^* is defined as

$$A^* = \mathbb{I} \oplus A \oplus A^2 \oplus \cdots \oplus A^n \oplus \ldots \ .$$

The meaning of $(A^*)_{ij}$ is the maximum weight of all paths of any weight from i to j. A necessary and sufficient condition for the existence of the matrix A^* as an element of $\mathbb{R}_{\max}^{n\times n}$ is that the digraph, $\mathcal{G}(A)$ does not contain any circuit with positive weight.

A path, p, of length k from i to j is a sequence of vertices $i_0, i_1, \ldots, i_k$ where $i = i_0, j = i_k$ such that the arcs $(i_0, i_1), \ldots, (i_{k-1}, i_k)$ belonging to the graph. The weight of the path is defined to be $|p|_w = (A)_{i_0 i_1} + \cdots + (A)_{i_{k-1} i_k}$ We denote by $|p|$ the length of the path, p. A circuit is a path, $(i_0, \ldots, i_k)$, such that $i_0 = i_k$. It is elementary if the vertices $i_0, \ldots, i_k$ are distinct.

Theorem 2.4.1 (Theorem 3.20 [BCOQ92]). *For a given $A \in \mathbb{R}_{\max}^{n\times n}$, if $\mathcal{G}(A)$ has no circuit with positive weight, then*

$$A^* = \mathbb{I} \oplus A \oplus \ldots \oplus A^{n-1} \ ,$$

We say that a nonzero $\lambda \in \mathbb{R}_{\max}$ is a (geometric) tropical eigenvalue of the matrix A if there exists a vector $x \in \mathbb{R}_{\max}^n \setminus \{\mathbb{0}\}$ (the associated eigenvector) such that $A \otimes x = \lambda \otimes x$.

Theorem 2.4.2 (Theorem 3.23 [BCOQ92]). *If A is irreducible, meaning that if $\mathcal{G}(A)$ is strongly connected, there exists one and only one eigenvalue (but possibly several eigenvectors). This eigenvalue is equal to the maximum circuit mean of the graph, i.e.*

$$\lambda = \max_{\zeta} \frac{|\zeta|_w}{|\zeta|}, \tag{2.1}$$

where $|\zeta|_w$ and $|\zeta|$ denote the weight and the length of a circuit ζ of $\mathcal{G}(A)$ respectively and the maximum is taken over the set of elementary circuits of $\mathcal{G}(A)$.

When the matrix is reducible (not irreducible), there are at most n (geometric) eigenvalues. Indeed, any of these eigenvalues is necessarily the maximal circuit mean of a diagonal block of A corresponding to a strongly connected component of $\mathcal{G}(A)$, but not every strongly connected component determine an eigenvalue. The maximal circuit mean is always an eigenvalue (even if A is reducible), and it is the maximal one. The eigenvalues and eigenvectors were characterized in [Gau92], see also [BSvdD95, BCGG09]. An account in English of [Gau92] can be found in [ABG06].

Several algorithms have been used to compute the eigenvalues such as Karp's algorithm [Kar78], which works in $O(nm)$ time where n is the dimension and m is number of non$-\mathbb{0}$ entries of an input matrix. A good survey on the maximal cycle mean problem can be found in [DG97]. An algorithm based on Howard's policy iteration have been developed by Cochet-terrasson et al. [CtCG+98]. This algorithm, appears to be experimentally more efficient.

To define the eigenspace we need to use the notion of critical graph. An arc (i, j) is critical if it belongs to a circuit $(i_1, \ldots, i_k)$ whose mean weight attains the max in 2.1. Then, the nodes i, j are critical. Critical nodes and arcs form the critical graph. A critical class is a strongly connected component of the critical graph. Let $C_1^c, \ldots, C_r^c$ denote the critical classes. Let $(B)_{ij} = (A)_{ij} - \lambda_{\max}(A)$. Using Theorem 2.4.1, $B^* = \mathbb{I} \oplus B \oplus \ldots \oplus B^{n-1}$. If j is in a critical class, we call the column $B^*_{\cdot j}$ of B^* critical. The following result can be found e.g. in [BCOQ92, CG94].

Theorem 2.4.3 (Eigenspace). *Let $A \in \mathbb{R}_{\max}^{n\times n}$ denote an irreducible matrix. The critical columns of B^* span the eigenspace of A. If we select only one column, arbitrarily, per critical class, we obtain a weak basis of the eigenspace.*

For a more complete survey see [Gau98].

An analogue of the notion of determinant, involving a formal "minus" sign, has been studied by several authors [GM84, BCOQ92, AGG09] However, the simplest approach is to consider the permanent instead of the determinant. The permanent of a matrix A with entries in an arbitrary semiring can be defined as

$$\operatorname{per} A := \sum_{\sigma \in S_n} \prod_{i=1}^{n} (A)_{i\sigma(i)} \ ,$$

where S_n denotes the set of all permutations. When the semiring is $\mathbb{R}_{\max}$, so that $(A)_{ij} \in \mathbb{R} \cup \{-\infty\}$. In this case, $\operatorname{per} A := \max_{\sigma \in S_n} \sum_{i=1}^{n} (A)_{i\sigma(i)}$, which is nothing but the value of an optimal assignment problem with weights $(A)_{ij}$.

The formal tropical characteristic polynomial is defined to be the

$$p_A(\lambda) = \operatorname{per}(A \oplus \lambda \otimes \mathbb{I}) \ ,$$

where the entries of the matrix $A \oplus \lambda \otimes \mathbb{I}$ are interpreted as formal polynomials with coefficients in $\mathbb{R}_{\max}$. The associated numerical tropical characteristic polynomial is the function

$$\lambda \mapsto p_A(\lambda), \qquad \mathbb{R}_{\max} \to \mathbb{R}_{\max} \ ,$$

which associates to a parameter $\lambda \in \mathbb{R} \cup \{-\infty\}$, the value of the optimal assignment problem in which the weights are given by the matrix $B := A \oplus \lambda \otimes \mathbb{I}$, i.e., $(B)_{ij} = (A)_{ij}$ for $i \neq j$ and $(B)_{ii} = \max((A)_{ii}, \lambda)$.

Following [ABG05, ABG04] the (algebraic) tropical eigenvalues are defined as the tropical roots of the tropical characteristic polynomial p_A, i.e., as the nondifferentiability points of the function $\lambda \mapsto p_A(\lambda)$. Every geometrical eigenvalue is an algebraic eigenvalue, but the converse does not hold in general. The maximal circuit mean (the maximal geometrical eigenvalue) is also the maximal algebraic eigenvalue.

Example 2.4.1. Consider the following matrix

$$A = \begin{pmatrix} 2 & 7 & 9 \\ \mathbb{0} & 4 & 1 \\ \mathbb{0} & 0 & 3 \end{pmatrix} \ .$$

This graph is reducible with two geometric tropical eigenvalues, $2, 4$. The characteristic polynomial of A is

$$p_A(\lambda) = (2 \oplus \lambda) \otimes \big((4 \oplus \lambda) \otimes (3 \oplus \lambda) \oplus 1\big) = (2 \oplus \lambda) \otimes (4 \oplus \lambda) \otimes (3 \oplus \lambda) \ .$$

Thus, the algebraic tropical eigenvalues of $p_A(\lambda)$ are $2, 3, 4$.

In the sequel we refer to algebraic tropical eigenvalues by tropical eigenvalue.

2.5 Perturbation of eigenvalues of matrix pencils

Let $A(t,\lambda) = (A(t,\lambda))_{ij}$ be a square matrix defined as follows,

$$(A(t,\lambda))_{ij} = \sum_{s\in\mathbb{Z}} \sum_{r\in\mathbb{Q}} A_{ijrs} t^r \lambda^s \ ,$$

where $(A)_{ij}$ is a polynomial in t and λ and A_{ijrs} are elements of a certain field and the summations are assumed to involve a finite number of terms. K. Morota [Mur90] studied the computation of Puiseux series solutions $\lambda = \lambda(t)$ to the equation $\det(A(t,\lambda)) = 0$. This problem arises in sensitivity analysis of eigenvalues of a matrix, A, when it is subject to a perturbation t [KK90]. Recall that a Puiseux series is a formal series of the form, $\sum_{n=m}^{\infty} a_n t^{n/k}$ where $k \geq 1$ is an integer and m is also an integer.

Another study of a similar problem with a focus on the valuation (leading exponents) of Puiseux series has been done by M. Akian et al. [ABG04]. They have considered the problem of perturbation of eigenvalues for a perturbed polynomial matrix defined as follows:

$$\mathcal{A}_\epsilon = \mathcal{A}_{\epsilon,0} + \lambda\mathcal{A}_{\epsilon,1} + \ldots + \lambda^d \mathcal{A}_{\epsilon,d} \ ,$$

where for each $0 \leq k \leq d$, $\mathcal{A}_{\epsilon,k}$ is an $n \times n$ matrix whose coefficients, $(\mathcal{A}_{\epsilon,k})_{ij}$ are complex valued continuous functions of a nonnegative parameter ϵ, and λ is indeterminate. Thus, the (finite) eigenvalues $\mathcal{L}_\epsilon$ of $\mathcal{A}_\epsilon$ are by definition the roots of the polynomial $\det(\mathcal{A}_\epsilon)$. They assumed that for every $0 \leq k \leq d$, matrices $a_k = ((a_k)_{ij}) \in C^{n\times n}$ and $A_k = ((A_k)_{ij}) \in (\mathbb{R} \cup \{+\infty\})^{n\times n}$ are given, so that

$$(\mathcal{A}_{\epsilon,k})_{ij} = (a_k)_{ij}\epsilon^{(A_k)_{ij}} + o(\epsilon^{(A_k)_{ij}}), \quad \textit{for all } 1 \leq i,j \leq n \ ,$$

when ϵ tends to 0. When $(A_k)_{ij} = +\infty$, this means by convention that $(A_{\epsilon,k})_{ij}$ is zero in a neighborhood of zero. They look for an asymptotic equivalent of the form $\mathcal{L}_\epsilon \sim \eta\epsilon^\Gamma$, with $\eta \in \mathbb{C} \setminus \{0\}$ and $\Gamma \in \mathbb{R}$, for every eigenvalue $\mathcal{L}_\epsilon$ of $\mathcal{A}_\epsilon$. They have shown that the first order asymptotics of the eigenvalues, γ of A_ϵ can be computed generically by methods of min-plus algebra and optimal assignment algorithms. The scaling, which we present in Chapter 4, is inspired from this analysis of asymptotic behavior of eigenvalues of a matrix polynomial.

CHAPTER 3

Locations of the roots of a polynomial by means of tropical algebra

Let $\zeta_1 \dots \zeta_n$ denote the roots of a polynomial, $p(x) = \sum_{i=1}^{n} a_i x^i, \quad a_i \in \mathbb{C}$, ranked by increasing modulus. We associate to $p(x)$ the max-times polynomial

$$\mathsf{t}p(x) = \bigoplus_{0 \le k \le n} |a_k| x^k = \max_{0 \le k \le n} |a_k| x^k \ .$$

Let $\alpha_1 < \ldots < \alpha_p$ denote the tropical roots of $\mathsf{t}p(x)$ with multiplicities $m_1 \dots m_p$, respectively, where $\sum_{i=1}^{p} m_i = n$. (See Chapter 2, §2.2 for the definition, recall in particular that the tropical roots are the non-differentiability points of the function $\mathsf{t}p(x)$). Also, for $i = 2, \dots, p$, let $\delta_{i-1} = \frac{\alpha_{i-1}}{\alpha_i}$ be the parameters measuring the separation between the tropical roots. We prove that if $\delta_{i-1}, \delta_i < \frac{1}{9}$, then $p(x)$ has exactly m_i roots for which

$$\frac{1}{3}\alpha_i < |\zeta_j| < 3\alpha_i, \qquad k < j \le k + m_i \ ,$$

where $k = 0$ for $i = 1$ and $k = m_1 + \cdots + m_{i-1}$ for $i > 2$. We also show that under a more restrictive condition, i.e. $\delta_{i-1}, \delta_i < \frac{1}{2^{m_i+2}+2}$, the previous bound can be

improved as follows

$$\frac{1}{2}\alpha_i < |\zeta_j| < 2\alpha_i \qquad k < j \leq k + m_i \ .$$

(the constant 2 cannot be improved in general due to a result of Cauchy). For the smallest and largest tropical roots the conditions over δ_i can be improved to $\delta_1 < \frac{1}{2^{m_1+1}+2}$ and $\delta_{p-1} < \frac{1}{2^{m_p+1}+2}$ for $i = 1$ and $i = p$ respectively.

When the tropical roots corresponding to a formal polynomial have different orders of magnitudes, or more precisely, when the values of δ_i are small enough, the usual numerical methods such as the ones implemented in the *roots function* in Matlab or Scilab, may fail to compute the roots correctly; however, the tropical roots can provide an a priori estimation of the modulus of the roots in linear time. This leads to an immediate application of these theoretical results to root finding methods.

3.1 Introduction

Solving a polynomial equation, $p(x) = a_0 + a_1 x + \cdots + a_n x^n = 0$, perhaps is the most classical problem in Mathematics. The study of this problem, focused on small degrees and for specific coefficients, comes back to the Sumerian (third millennium B.C.) and Babylonian (about 2000 B.C.) [Pan97]. This problem has been studied during the centuries by Egyptians, in ancient Greece by Pythagoreans and later by Persian mathematicians such as Omar Khayyam who presented a geometrical solution for the cubic polynomials [Pan97, AM62]. Later on, in the 16th century, the arithmetic solution to the degree three and four polynomials have been achieved. However, all the attempts to find an arithmetic solution for any polynomial with degree greater than 4 were failed till the time when Ruffini in 1813 and Abel in 1827 proved the nonexistence of such a formula for the class of polynomials of degree greater than 4 and later by Galois in 1832 [Pan97]. The fundamental theorem of algebra, which was stated by several mathematicians and finally proved in 19th century, states that $p(x) = a_0 + a_1 x + \cdots + a_n x^n = 0$ always has a complex solution for any positive degree n. This result yields the existence of a factorization $p(x) = a_n \prod_{i=1}^{n}(x - \zeta_i), a_n \neq 0$, where $\zeta_1, \ldots, \zeta_n$ are the roots of $p(x)$.

Due to nonexistence of any exact method to find the roots, the main effort was to design the numerical algorithms, which approximate the roots. These efforts yields the development of several methods such as Weyl's algorithm, Graeffe's iteration, Schönhage's algorithm [Sch82], etc. For a survey we refer to [Pan97].

3.2 Classical bounds on the modulus of the roots of a polynomial by using tropical roots

Let $p(x) = \sum_{k=0}^{n} a_k x^k, \quad a_i \in \mathbb{C}$ be a polynomial of degree n in the complex plane. Also let $\zeta_1 \dots \zeta_n$, denote the roots of $p(x)$ ordered by increasing modulus. We define the max-times polynomial

$$\mathsf{t}p(x) = \bigoplus_{0 \le k \le n} |a_k| x^k = \max_{0 \le k \le n} |a_k| x^k \ ,$$

corresponding to $p(x)$. Due to Proposition 2.2.1, the tropical roots of $\mathsf{t}p(x)$, $\alpha_1 \le \alpha_2 \le \ldots \le \alpha_n$, repeated with multiplicities, can be computed in linear time, $O(n)$. In the sequel, we refer to the complex roots of $p(x)$ as the *roots* and to $\alpha_1, \dots, \alpha_n$ by the *tropical roots of* $p(x)$.

J. S. Hadamard gave a bound on the modulus of the roots of a polynomial using a systematic method in which the modulus of the coefficients of the polynomial are bounded by a variant of the classical Newton polygon construction. The Newton polygon used by Hadamard can be seen to be the Legendre-Fenchel transform of the tropical polynomial used here. Hence, the bound given by Hadamard in [Had93] (page 201, third inequality) can be restated as follows in tropical terms:

$$|\zeta_1 \zeta_2 \dots \zeta_k| \ge \frac{\alpha_1 \cdots \alpha_k}{k+1} \ . \tag{3.1}$$

In particular, $|\zeta_1| \ge \frac{\alpha_1}{2}$ is an equivalent form of the classical bound of Cauchy.

The result of Hadamard (proved in passing in a memoir devoted to the Riemann zeta function) remained apparently not so well known. In particular, in 1938, W. Specht [Spe38] proved that

$$|\zeta_1 \zeta_2 \cdots \zeta_k| \ge \frac{{\alpha_1}^k}{k+1} \ , \tag{3.2}$$

which is weaker since $\alpha_1 \le \alpha_2, \dots, \alpha_k$.

Alexander Ostrowski, in 1940, proved several bounds on the roots of a polynomial in his comprehensive report entitled "'Recherches sur la méthode de Graeffe" [Ost40a, Ost40b], in which he used again the Newton polygon introduced by Hadamard. He called the slopes of this polygon (corresponding to the tropical roots) the *inclinaisons numériques* and he proved the following theorem to estimate the modulus of the roots.

Theorem 3.2.1 (Ostrowski [Ost40a])**.** *Let $p(x) = \sum_{i=0}^{n} a_i x^i$ be a polynomial of degree n where $\zeta_1, \dots, \zeta_n$ denote the roots of $p(x)$ ordered by increasing modulus. Also, let $\alpha_1 \le \alpha_2 \le \ldots \le \alpha_n$ (counted with multiplicities) denote the tropical roots of the associated max-times polynomial $\mathsf{t}p(x)$. Then,*

- *for $k = 1$* $\quad \frac{1}{2}\alpha_1 < |\zeta_1| \le n\alpha_1$,

- *for* $k = n \quad \frac{1}{n}\alpha_n \leq |\zeta_n| < 2\alpha_n$,
- *for* $k = 2, \ldots, n-1$,

$$(1 - (\frac{1}{2})^{\frac{1}{k}})\alpha_k \leq |\zeta_k| \leq \frac{\alpha_k}{1 - (\frac{1}{2})^{\frac{1}{n-k+1}}} \tag{3.3}$$

In this way, he recovered the inequality (3.1) of Hadamard. He also included a private conversation with M. G. Pólya who showed the following inequality, in which the coefficient is improved:

$$|\zeta_1\zeta_2\cdots\zeta_k| \geq \sqrt{\frac{k^k}{(k+1)^{k+1}}}\alpha_1\cdots\alpha_k \ .$$

According to his report, if $\frac{\alpha_k}{\alpha_{k+1}}$ is less than $\frac{1}{9}$, then $|\zeta_k| < |\zeta_{k+1}|$ which is a sufficient condition for ζ_k to be separated from ζ_{k+1}.

According to Ostrowski, another result about this bound has been proved by G. Valiron [VAL14] that is, if $\frac{\alpha_k}{\alpha_{k+1}} < \frac{1}{9}$ then $p(x)$ has exactly k roots in the circle $|z| < \sqrt{\alpha_k\alpha_{k+1}}$.

In a recent work, B. Kirshtein shows that the classical algorithm of Graffe-Lobachevski, which is used to find the roots of a univariate polynomial, calculates a tropical polynomial whose tropical roots coincides with the logarithms of the moduli of the roots of the input complex polynomial [Kir09].

Assume that $\alpha_{k-1} < \alpha_k = \cdots = \alpha_{k+m-1} < \alpha_{k+m}$. Due to Theorem 3.2.1, the modulus of the m corresponding roots of $p(x)$ bounded from lower and upper by α_k where the left and right constants in Inequality 3.3 will be different for $\alpha_k, \ldots, \alpha_{k+m-1}$. Also it can be seen that Inequality 3.3 is not tight from left(right) when k is increasing(decreasing). However, we will prove, in the next section, that when the values $\frac{\alpha_{k-1}}{\alpha_k}$ and $\frac{\alpha_{k+m-1}}{\alpha_{k+m}}$ are small enough i.e when α_k is well separated from the other tropical roots, a tight inequality with the same constant will hold for all m tropical roots.

3.3 Location of the roots of a polynomial in terms of the tropical roots

In this section, we provide some new bounds on the modulus of the roots of a polynomial by considering not only the tropical roots but also their multiplicities. We will prove that when a tropical root, α, with multiplicity m, of a polynomial, $p(x)$, is well separated from the other tropical roots, then, $p(x)$ has m roots with the same order of magnitude as α.

In the sequel we assume that $p(x)$ has no zero root. We shall use the following classical theorem of Rouché,

Theorem 3.3.1 (Rouché's theorem)**.** *Let the functions f and g be analytic in the simply connected region R, let Γ be a Jordan curve in R, and let $|f(z)| > |g(z)|$ for all $z \in \Gamma$. Then the functions $f + g$ and f have the same number of zeros in the interior of Γ.*

Let $p(x) = \sum_{k=0}^{n} a_i x^i, a_i \in \mathbb{C}$ be a polynomial with the corresponding tropical roots $\alpha_1 < \alpha_2 < \ldots < \alpha_p$. Let $m_1, \ldots m_p$ denote the multiplicity of these tropical roots, respectively, where $\sum_{i=1}^{p} m_i = n$. Recall from Section 2.2 that the *Newton polygon*, $\Delta(P)$, of P is defined to be the upper boundary of the convex hull of the set of points $(k, \log|a_k|)$, $k = 0, \ldots, n$. Figure 3.1 shows the Newton polygon of $p(x)$. Here, $k_0 = 0, k_1, \ldots, k_p$ denote the X-coordinates (abscissa) of the *vertices* (extreme points of edges) of the Newton polygon. The opposite of the slopes of the linear segments in the diagram are precisely the logarithms of the tropical roots. The multiplicity of a root coincides with the width of the corresponding segment measured on the horizontal (X) axis. So, $m_1 = k_1, m_2 = k_2 - k_1, \ldots, m_p = k_p - k_{p-1}$. The next lemma provides some

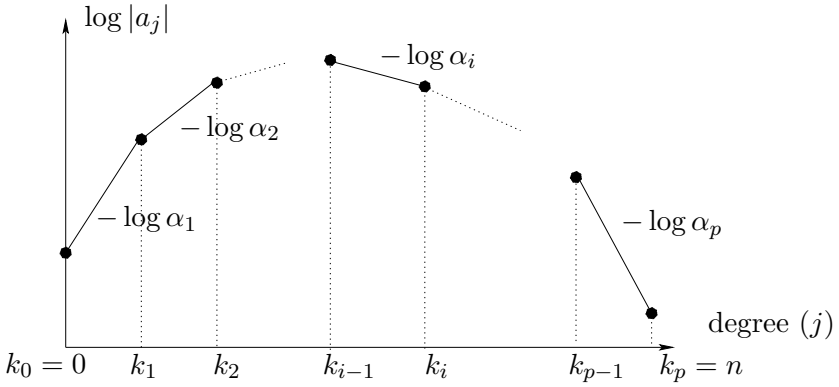


Figure 3.1: Newton polygon corresponding to $p(x)$.

bounds on the coefficients of $p(x)$ based on tropical roots.

Lemma 3.3.2. *Let $\alpha_1 < \ldots < \alpha_p$ denote the corresponding tropical roots of a polynomial $p(x) = \sum_{k=0}^{n} a_i x^i$, Also let $k_0 = 0, k_1, \ldots k_p$ be the X-coordinates of the vertices of the Newton polygon of $p(x)$ shown in Figure 3.1. The following statements hold.*

(i) $\alpha_i = \left(\frac{|a_{k_{i-1}}|}{|a_{k_i}|}\right)^{\frac{1}{k_i - k_{i-1}}}$

(ii) $|a_{k_{i-1}}|{\alpha_i}^{k_{i-1}} = |a_{k_i}|{\alpha_i}^{k_i}$ *for all* $i = 1 \ldots p$

(iii) $|a_j| \leq |a_{k_i}|{\alpha_i}^{k_i - j}$ *for all* $1 \leq i \leq p; \quad k_{i-1} \leq j \leq k_i;$

(iv) $|a_j| \leq |a_{k_i}|{\alpha_{i+1}}^{-(j-k_i)}$ *for all* $1 \leq i \leq p; \quad k_i \leq j \leq k_{i+1};$

(v) $|a_j| \leq |a_{k_i}|{\alpha_i}^{k_i-j}$ *for all* $1 \leq i \leq p; \quad 0 \leq j \leq k_i;$

(vi) $|a_j| \leq |a_{k_i}|{\alpha_{i+1}}^{-(j-k_i)}$ *for all* $1 \leq i \leq p; \quad k_i \leq j \leq n;$

Proof. The proof of the statements *i*, *ii*, *iii* and *iv* are straightforward. For inequality *v*

$$\begin{aligned} |a_j| &\leq |a_{k_{u+1}}|\alpha_{u+1}^{k_{u+1}-j} \quad \text{for } k_u \leq j \leq k_{u+1} \leq k_i \quad \text{due to } iii \\ &\leq |a_{k_{u+2}}|\alpha_{u+2}^{k_{u+2}-k_u}\alpha_{u+1}^{k_{u+1}-j} \quad \text{due to } ii \\ &\leq |a_{k_i}|\alpha_i^{k_i-k_{i-1}} \dots \alpha_{u+2}^{k_{u+2}-k_u}\alpha_{u+1}^{k_{u+1}-j} \\ &\leq |a_{k_i}|\alpha_i^{k_i-j} \end{aligned}$$

and for the last inequality

$$\begin{aligned} |a_j| &\leq |a_{k_u}|\alpha_{u+1}^{k_u-j} \quad \text{for } k_i \leq k_u \leq j \leq k_{u+1} \quad \text{due to } iv \\ &\leq |a_{k_{u-1}}|\alpha_u^{k_{u-1}-k_u}\alpha_{u+1}^{k_u-j} \quad \text{due to } ii \\ &\leq |a_{k_i}|\alpha_{i+1}^{k_i-k_{i+1}} \dots \alpha_{u+1}^{k_u-j} \\ &\leq |a_{k_i}|\alpha_{i+1}^{k_i-j} \end{aligned}$$

□

Definition 3.3.1 (α_i-normalized polynomial)**.** Let α_i be the ith tropical root of a polynomial $p(x)$ and let $x = \alpha_i y$ be an scaling on the variable x. We call $q(y)$, an α_i-*normalized* polynomial corresponding to $p(x)$ which is defined as follows

$$q(y) = (|a_{k_{i-1}}|\alpha_i^{k_{i-1}})^{-1}(\sum_{j=0}^{n} a_j(\alpha_i y)^j) \ .$$

Remark 3. It follows from the definition that for the α_i-normalized polynomial $q(y) = \sum_{i=1}^{n} b_i$ due to Lemma 3.3.2 we have,

$$|b_{k_{i-1}}| = |b_{k_i}| = |(|a_{k_{i-1}}|\alpha_i^{k_{i-1}})^{-1}a_{k_{i-1}}\alpha_i^{k_{i-1}}| = 1 \ ,$$

and

$$|b_j| \leq 1 \quad \text{for all } k_{i-1} \leq j \leq k_i \ .$$

The following theorem provides the main result of this chapter.

Theorem 3.3.3. *Let $\zeta_1, \dots, \zeta_n$ be the roots of a polynomial, $p(x) = \sum_{k=0}^{n} a_i x^i$ ordered by increasing modulus. Also let $\alpha_1 < \alpha_2 < \dots < \alpha_p$ denote the tropical roots of $p(x)$ with multiplicity $m_1, m_2, \dots, m_p$ respectively where $\sum_{i=1}^{p} m_i = n$. Let $\delta_1 = \frac{\alpha_1}{\alpha_2}, \dots, \delta_{p-1} = \frac{\alpha_{p-1}}{\alpha_p}$ be the parameters, which measure the separation between the tropical roots. Then,*

(i) $p(x)$ has exactly m_i roots in the annulus $\frac{1}{2}\alpha_i \leq |\zeta| < 2\alpha_i$ if,

- $\delta_i, \delta_{i-1} < \frac{1}{2^{m_i+2}+2}$ *for* $1 < i < p$

- $\delta_1 < \frac{1}{2^{m_1+1}+2}$ *for* $i = 1$
- $\delta_{p-1} < \frac{1}{2^{m_p+1}+2}$ *for* $i = p$

(ii) $p(x)$ *has exactly* m_i *roots in the annulus* $\frac{1}{3}\alpha_i \leq |\zeta| < 3\alpha_i$ *if,*

- $\delta_i, \delta_{i-1} < \frac{1}{9}$ *for* $1 < i < p$
- $\delta_1 < \frac{1}{9}$ *for* $i = 1$
- $\delta_{p-1} < \frac{1}{9}$ *for* $i = p$

Consider the ith tropical root of $p(x)$ and let $q(y)$ be the α_i-normalized polynomial corresponding to $p(x)$. The idea of the proof is to decompose $q(y)$ to three polynomials as follows,

$$q_i^-(y) = (|a_{k_{i-1}}|\alpha_i^{k_{i-1}})^{-1}(\sum_{j=0}^{k_{i-1}-1} a_j {\alpha_i}^j y^j) \tag{3.4}$$

$$q_i(y) = (|a_{k_{i-1}}|\alpha_i^{k_{i-1}})^{-1}(\sum_{j=k_{i-1}}^{k_i} a_j {\alpha_i}^j y^j) \tag{3.5}$$

$$q_i^+(y) = (|a_{k_{i-1}}|\alpha_i^{k_{i-1}})^{-1}(\sum_{j=k_i+1}^{n} a_j {\alpha_i}^j y^j) \tag{3.6}$$

so that $q_i(y)$ is the normalized polynomial corresponding to the ith edge of the Newton polygon. Then we find the appropriate disks, such that $|q_i(y)| > |q_i^+(y) + q_i^-(y)|$ holds on their boundary under the conditions for δ_i, which are mentioned in the theorem. In this way, by Rouché's theorem, $q_i(y)$ and $q(y)$ will have the same number of roots inside the disk. The proof of the theorem relies on the following lemmas.

Lemma 3.3.4. *Let* α_i *be the* i*th tropical root of a polynomial,* $p(x)$ *with multiplicity* m_i*. Also, let* $q(y)$ *be the* α_i-*normalized polynomial and* $q_i(y)$ *be the polynomial defined in Equation 3.5 corresponding to the* i*th edge of the Newton polygon of* $p(x)$*. Then,* $q_i(y)$ *has* m_i *nonzero roots, which lies in the annulus* $1/2 < |z| < 2$*.*

Proof. Define

$$s(y) = y^{-k_{i-1}} q_i(y) = (|a_{k_{i-1}}|\alpha_i^{k_{i-1}})^{-1}(\sum_{j=k_{i-1}}^{k_i} a_j {\alpha_i}^j y^{j-k_{i-1}}) \ ,$$

to be a polynomial with $m_i = k_i - k_{i-1}$ nonzero roots. As it is mentioned in Remark 3, the modulus of the coefficients of $s(y)$ will not be greater than 1. Also, due to Cauchy's bound all the roots of $s(y)$ lies in the disk of

$$|z| < 1 + \max_{k_{i-1} \leq j \leq k_i} |s_j/s_{k_i}| = 2 \ ,$$

where s_j presents the jth coefficient of $s(y)$. The lower bound can be achieved by applying the Cauchy bound on the reciprocal polynomial of $s(y)$, i.e. $s^*(y) = y^{m_i}\overline{s(\overline{y}^{-1})}$. □

The next lemma provides some bounds on the absolute value of $q_i^-(y)$, $q_i(y)$, $q_i^+(y)$.

Lemma 3.3.5. *Let α_i be the ith tropical root of a polynomial, $p(x)$ with multiplicity m_i. Assume that $q(y)$ is the α_i-normalized polynomial and $q_i^-(y)$, $q_i(y)$ and $q_i^+(y)$ be the polynomials defined in 3.4, 3.5 and 3.6. Also let $0, k_1, \dots k_p$ be the X-coordinates of the vertices of the Newton polygon of $p(x)$ shown in Figure 3.1. The following inequalities hold.*

(i) $|q_i^-(y)| \leq \frac{|y|^{k_{i-1}}}{\delta_{i-1}^{-1}|y|-1}$

(ii) $|q_i^+(y)| \leq \frac{\delta_i|y|^{k_i+1}}{1-\delta_i|y|}$

(iii) $|q_i(y)| \geq |y|^{k_{i-1}}(\frac{1-2|y|+|y|^{k_i-k_{i-1}+1}}{1-|y|})$ *for* $|y| < 1$

(iv) $|q_i(y)| \geq |y|^{k_i}(\frac{|y|-2+|y|^{k_{i-1}-k_i}}{|y|-1})$ *for* $|y| > 1$

Before proving this lemma, we give its geometrical interpretation, in Figure 3.2. The α_i-normalized polynomial $q(y)$ is such that the edge of the Newton polygon corresponding to α_i lies on the horizontal axis. The polynomials $q_i^{\pm}$ are bounded by geometric series, corresponding to the half-lines with slopes $-\log\delta_{i-1}$ and $\log\delta_i$, as shown by Inequalities (i) and (ii). For small (resp. large) values of $|y|$, the leading monomial of q_i is the one with the smallest (resp. highest) degree, corresponding to the left (resp. right) extreme point of the horizontal segment, as shown by Inequalities (iii) and (iv).

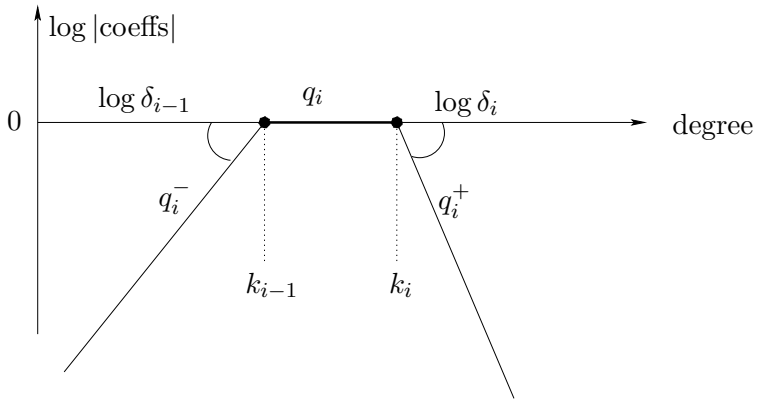


Figure 3.2: Illustration of Lemma 3.3.5.

Proof. We have

$$\begin{aligned}
|q_i^-(y)| &= (|a_{k_{i-1}}|\alpha_i^{k_{i-1}})^{-1}(|\sum_{j=0}^{k_{i-1}-1} a_j {\alpha_i}^j y^j|) \\
&\leq (|a_{k_{i-1}}|\alpha_i^{k_{i-1}})^{-1}(\sum_{j=0}^{k_{i-1}-1} |a_{k_{i-1}}|\alpha_{i-1}^{k_{i-1}-j}{\alpha_i}^j|y|^j) \quad \textit{due to 3.3.2} \\
&\leq \delta_{i-1}^{k_{i-1}}(\sum_{j=0}^{k_{i-1}-1} \delta_{i-1}^{-j}|y|^j) \\
&= \delta_{i-1}^{k_{i-1}}\frac{(\delta_{i-1}^{-1}|y|)^{k_{i-1}}-1}{\delta_{i-1}^{-1}|y|-1} = \frac{|y|^{k_{i-1}}-{\delta_{i-1}}^{k_{i-1}}}{\delta_{i-1}^{-1}|y|-1} \\
&\leq \frac{|y|^{k_{i-1}}}{\delta_{i-1}^{-1}|y|-1} \ .
\end{aligned}$$

Similarly,

$$\begin{aligned}
|q_i^+(y)| &= (|a_{k_{i-1}}|\alpha_i^{k_{i-1}})^{-1}(|\sum_{j=k_i+1}^{n} a_j {\alpha_i}^j y^j|) \\
&\leq (|a_{k_i}|\alpha_i^{k_i})^{-1}(\sum_{j=k_i+1}^{n} |a_{k_i}|\alpha_{i+1}^{k_i-j}{\alpha_i}^j|y|^j) \quad \textit{due to 3.3.2} \\
&\leq \delta_i^{-k_i}(\sum_{j=k_i+1}^{n} (\delta_i|y|)^j) \\
&\leq \delta_i^{-k_i}(\delta_i|y|)^{k_i+1}\frac{1-(\delta_i|y|)^{n-k_i}}{1-\delta_i|y|} = \delta_i|y|^{k_i+1}\frac{1-(\delta_i|y|)^{n-k_i}}{1-\delta_i|y|} \\
&\leq \frac{\delta_i|y|^{k_i+1}}{1-\delta_i|y|}
\end{aligned}$$

Finally,

$$\begin{aligned}
|q_i(y)| &= (|a_{k_{i-1}}|\alpha_i^{k_{i-1}})^{-1}(|\sum_{j=k_{i-1}}^{k_i} a_j {\alpha_i}^j y^j|) \quad \textit{for } |y|<1 \\
&\geq (|a_{k_{i-1}}|\alpha_i^{k_{i-1}})^{-1}(|a_{k_{i-1}}|{\alpha_i}^{k_{i-1}}|y|^{k_{i-1}} - |\sum_{j=k_{i-1}+1}^{k_i} a_j {\alpha_i}^j y^j|) \\
&\geq |y|^{k_{i-1}} - (|a_{k_{i-1}}|\alpha_i^{k_{i-1}})^{-1}\sum_{j=k_{i-1}+1}^{k_i} |a_{k_{i-1}}|\alpha_i^{-(j-k_{i-1})}{\alpha_i}^j|y|^j \\
&\geq |y|^{k_{i-1}} - \sum_{j=k_{i-1}+1}^{k_i} |y|^j \geq |y|^{k_{i-1}} - |y|^{k_{i-1}+1}(\frac{1-|y|^{k_i-k_{i-1}}}{1-|y|}) \\
&= (|y|)^{k_{i-1}}(\frac{1-2|y|+|y|^{k_i-k_{i-1}+1}}{1-|y|}) \ ;
\end{aligned}$$

$$
\begin{aligned}
|q_i(y)| &\geq |y|^{k_{i-1}}(|y|^{k_i-k_{i-1}} - \sum_{j=1}^{k_i-k_{i-1}-1} |y|^j) \quad \textit{for } |y|>1 \\
&\geq |y|^{k_{i-1}}(|y|^{k_i-k_{i-1}} - \frac{|y|^{k_i-k_{i-1}}-1}{|y|-1}) \\
&= |y|^{k_i} - \frac{|y|^{k_i}-|y|^{k_{i-1}}}{|y|-1} \\
&= |y|^{k_i}(\frac{|y|-2+|y|^{k_{i-1}-k_i}}{|y|-1}) \ .
\end{aligned}
$$

□

In the next lemma we will consider the conditions on δ_i and δ_{i-1} under which $|q_i(y)| > |q_i^+(y) + q_i^-(y)|$ holds.

Lemma 3.3.6. *Let α_i be the ith tropical root of a polynomial, $p(x)$ with multiplicity m_i. Assume that $q(y)$ is the α_i-normalized polynomial and $q_i^-(y)$, $q_i(y)$ and $q_i^+(y)$ be the polynomials defined in 3.4, 3.5 and 3.6. Also let $0, k_1, \dots k_p$ be the X-coordinates of the vertices of the Newton polygon of $p(x)$ shown in Figure 3.1. Then the inequality*

$$|q_i(y)| > |q_i^+(y) + q_i^-(y)| \ , \tag{3.7}$$

holds

(i) on the circle $|y| = \frac{1}{2}$, when $\delta_{i-1} < \frac{1}{2^{m_i+2}+2}$ and $\delta_i < \frac{2}{3}$

(ii) on the circle $|y| = \frac{1}{3}$, for any $\delta_i < 1$, whenever $\delta_{i-1} < \frac{1}{9}$. Moreover, among all the radii $r < \frac{1}{2}$, the choice of the radius $r = \frac{1}{3}$ has the property of maximizing the value of δ_{i-1} such that the strict inequality 3.7 holds on a circle of radius r.

(iii) on the circle $|y| = 2$, for $\delta_{i-1} < \frac{2}{3}$, whenever $\delta_i < \frac{1}{2^{m_i+2}+2}$

(iv) on the circle $|y| = 3$, for any $\delta_{i-1} < 1$, whenever $\delta_i < \frac{1}{9}$. Moreover, among all the radii $r > 2$, the choice of the radius $r = 3$ has the property of maximizing the value of δ_i such that the strict inequality 3.7 holds on a circle of radius r.

Proof. Due to the inequalities i, ii and iii presented in Lemma 3.3.5, to satisfy the inequality 3.7 for a disk $r = |y| \leq \frac{1}{2}$, it is sufficient that

$$
\begin{aligned}
r^{k_{i-1}}(\frac{1-2r+r^{k_i-k_{i-1}+1}}{1-r}) &> \tfrac{\delta_i r^{k_i+1}}{1-\delta_i r} + \tfrac{r^{k_{i-1}}}{\delta_{i-1}^{-1} r - 1} \Leftrightarrow \\
\frac{1-2r+r^{m_i+1}}{1-r} &> \tfrac{\delta_i r^{m_i+1}}{1-\delta_i r} + \tfrac{\delta_{i-1}}{r-\delta_{i-1}}
\end{aligned} \tag{3.8}
$$

Setting $r = \frac{1}{2}$ in last inequality we have

$$(\frac{1}{2})^{m_i} > \frac{(\frac{1}{2})^{m_i}\delta_i}{2-\delta_i} + \frac{2\delta_{i-1}}{1-2\delta_{i-1}} \ ,$$

which is valid when $\delta_i < \frac{2}{3}$ and $\delta_{i-1} < \frac{1}{2^{m_i+2}+2}$.

Consider again the inequality 3.8, which can be rewritten as follows:

$$\frac{1-2r}{1-r} + r^{m_i+1}(\frac{1}{1-r} - \frac{\delta_i}{1-\delta_i r}) > \frac{\delta_{i-1}}{r-\delta_{i-1}} \ .$$

Since $\delta_i < 1$, $(\frac{1}{1-r} - \frac{\delta_i}{1-\delta_i r}) > 0$ holds. Also for $r < 1$, $r^{m_i+1} \to 0$ when $m_i \to \infty$. Indeed, the latter inequality is verified for all m_i iff

$$\frac{1-2r}{1-r} > \frac{\delta_{i-1}}{r-\delta_{i-1}} \ ,$$

which yields $\delta_{i-1} < \frac{r-2r^2}{2-3r}$. The maximum value of $\frac{r-2r^2}{2-3r}$ is $\frac{1}{9}$, which will be achieved when $r = \frac{1}{3}$.

The same argument can be made when $|y| \geq 2$. For the disk $r = |y| \geq 2$. Due to the inequalities i, ii and iv presented in Lemma 3.3.5, the sufficient condition to satisfy the inequality 3.7 is that

$$\begin{aligned} r^{k_i}(\frac{r-2+r^{k_{i-1}-k_i}}{r-1}) &> \frac{\delta_i r^{k_i+1}}{1-\delta_i r} + \frac{r^{k_{i-1}}}{\delta_{i-1}^{-1}r-1} \Leftrightarrow \\ \frac{r-2+r^{-m_i}}{r-1} &> \frac{\delta_i r}{1-\delta_i r} + \frac{r^{-m_i}\delta_{i-1}}{r-\delta_{i-1}} \end{aligned} \tag{3.9}$$

So, for $r = |y| = 2$,

$$2^{-m_i} > \frac{2\delta_i}{1-2\delta_i} + \frac{2^{-m_i}\delta_{i-1}}{2-\delta_{i-1}} \ ,$$

which is satisfied when $\delta_{i-1} < \frac{2}{3}$ and $\delta_i < \frac{1}{2^{m_i+2}+2}$.

When $|y| > 2$, the inequality 3.9 can be rewritten as

$$\frac{r-2}{r-1} + r^{-m_i}(\frac{1}{r-1} - \frac{\delta_{i-1}}{r-\delta_{i-1}}) > \frac{\delta_i r}{1-\delta_i r} \ .$$

Since $\delta_{i-1} < 1$, $(\frac{1}{r-1} - \frac{\delta_{i-1}}{r-\delta_{i-1}}) > 0$ holds. Also for $r > 1$, $r^{-m_i} \to 0$ when $m_i \to \infty$. Thus, the latter inequality is verified for all m_i iff

$$\frac{r-2}{r-1} > \frac{\delta_i r}{1-\delta_i r} \ ,$$

which yields $\delta_i < \frac{r-2}{2r^2-3r}$. The maximum value of $\frac{r-2}{2r^2-3r}$ is $\frac{1}{9}$, which will be achieved when $r = 3$. □

For the smallest tropical root, α_1, $q_i^-(y) = 0$, so, when $r = |y| = \frac{1}{2}$ the inequality 3.8 becomes $(\frac{1}{2})^{m_1} > \frac{(\frac{1}{2})^{m_1}\delta_1}{2-\delta_1}$ which is valid for all $\delta_1 < 1$. When $r = |y| \geq 2$, the inequality 3.9 becomes $\frac{r-2+r^{-m_1}}{r-1} > \frac{\delta_1 r}{1-\delta_1 r}$ which yields

$$\delta_1 < \frac{r-2+r^{-m_1}}{2r^2-3r+r^{-m_1+1}} \quad . \tag{3.10}$$

Thus, for $r = 2$, the latter inequality holds for all m_1 iff δ_1 is less than $\frac{1}{2^{m_1+1}+2}$.

For $r > 2$, since $\frac{r-2+r^{-m_1}}{2r^2-3r+r^{-m_1+1}} \to \frac{r-2}{2r^2-3r}$ when $m_1 \to \infty$, the inequality 3.10 holds for all m_1 iff $\delta_1 < \frac{r-2}{2r^2-3r}$. The maximum value of $\frac{r-2}{2r^2-3r}$, which is $\frac{1}{9}$, is achieved when $r = 3$. This is also illustrated in Figure 3.3 for several values of m_1 when r varies in the interval $(2, 4)$.

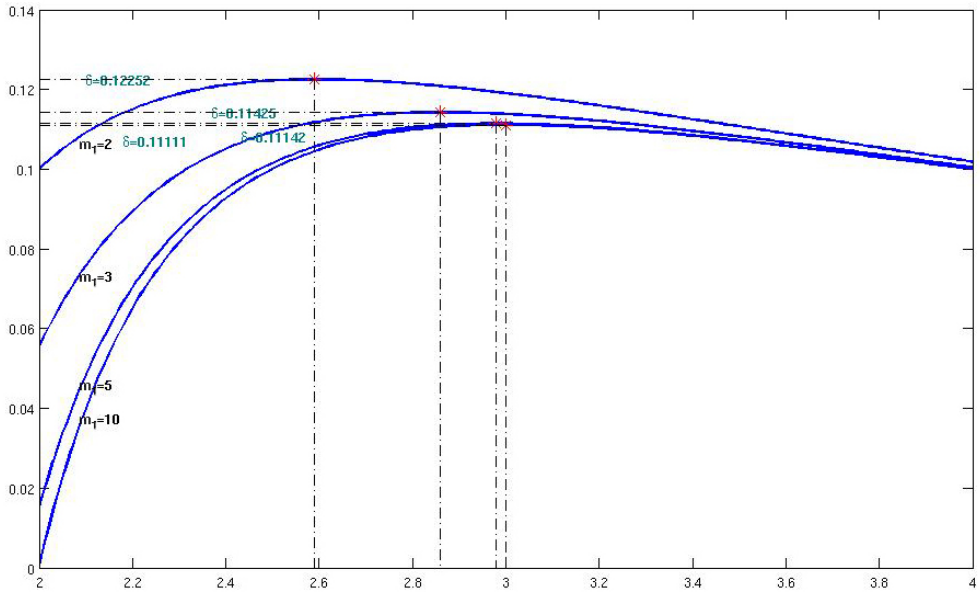


Figure 3.3: The illustration of the upper bound for δ_1 in inequality 3.10 for several values of m_1 when r varies in $(2, 4)$.

These results yield the following lemma:

Lemma 3.3.7. *Let α_1 be the smallest tropical root of a polynomial, $p(x)$ with multiplicity m_1. Assume that $q(y)$ is the α_1-normalized polynomial. Then the inequality 3.7 holds,*

(i) on the circle $|y| = \frac{1}{2}$, for any $\delta_1 < 1$.

(ii) on the circle $|y| = 2$, iff $\delta_1 < \frac{1}{2^{m_1+1}+2}$.

(iii) on the circle $|y| = 3$, whenever $\delta_1 < \frac{1}{9}$. Moreover, among all the radii $r > 2$, the choice of the radius 3 has the property of maximizing the value of δ_1 such that the strict inequality 3.7 holds on a circle of radius r.

For the largest tropical root, α_p, $q_i^+(y) = 0$, so, when $r = |y| = 2$ the inequality 3.9 becomes $2^{-m_p} > \frac{2^{-m_p}\delta_{p-1}}{2-\delta_{p-1}}$ which is valid for all $\delta_{p-1} < 1$.

When $r = |y| \leq \frac{1}{2}$, the inequality 3.8 becomes $\frac{1-2r+r^{m_p+1}}{1-r} > \frac{\delta_{p-1}}{r-\delta_{p-1}}$ which implies

$$\delta_{p-1} < \frac{r - 2r^2 + r^{m_p+2}}{2 - 3r + r^{m_p+1}} \ . \tag{3.11}$$

Thus, for $r = \frac{1}{2}$, the latter inequality holds for all m_p iff $\delta_{p-1} < \frac{1}{2^{m_p+1}+2}$.

For $r < \frac{1}{2}$, since $\frac{r-2r^2+r^{m_p+2}}{2-3r+r^{m_p+1}} \to \frac{r-2r^2}{2-3r}$ when $m_p \to \infty$, the inequality 3.11 holds for all m_p iff $\delta_{p-1} < \frac{r-2r^2}{2-3r}$. The maximum value of $\frac{r-2r^2}{2-3r}$, which is $\frac{1}{9}$, is achieved when $r = \frac{1}{3}$. These results implies the following lemma:

Lemma 3.3.8. *Let α_p be the largest tropical root of a polynomial, $p(x)$ with multiplicity m_p. Assume that $q(y)$ is the α_p-normalized polynomial. Then the inequality 3.7 holds,*

(i) on the circle $|y| = \frac{1}{2}$, iff $\delta_{p-1} < \frac{1}{2^{m_p+1}+2}$.

(ii) on the circle $|y| = \frac{1}{3}$, for any $\delta_{p-1} < \frac{1}{9}$. Moreover, among all the radii $r < \frac{1}{2}$, the choice of the radius $r = \frac{1}{3}$ has the property of maximizing the value of δ_{p-1} such that the strict inequality 3.7 holds on a circle of radius r.

(iii) on the circle $|y| = 2$, for $\delta_{p-1} < 1$.

To conclude the proof of Theorem 3.3.3, let $q(y) = (|a_0|^{-1})(\sum_{i=0}^n a_i {\alpha_1}^i y^i)$ be the α_i-normalized polynomial decomposed into three polynomial $q_i^-(y), q_i(y)$ and $q_i^+(y)$. Due to Lemma 3.3.6, for any $1 < i < p$, the condition $|q_i(y)| > |q_i^-(y) + q_i^+(y)|$ holds over the disks $|y| = 1$ and $|y| = \frac{1}{2}$, when $\delta_i, \delta_{i-1} < \frac{1}{2^{m_i+2}+2}$. According to the Rouché theorem, the latter implies that $q_i(y)$ and $q(y)$ have the same number of roots inside the disks $|y| = \frac{1}{2}$ and $|y| = 2$. Due to Lemma 3.3.4, $q_i(y)$ has m_i roots in the annulus $\frac{1}{2} \leq |y| \leq 2$ which implies that $q(y)$ also has the same number of roots in this annulus. The proof is achieved, since $y = \alpha_i x$, $p(x)$ has m_i roots in the annulus $\frac{1}{2}\alpha_i \leq |x| \leq 2\alpha_i$. The same argument can be made for the other cases.

3.4 Application

Consider the following polynomial,

$$p(x) = 0.1 + 0.1x + (1.000D + 40)x^7 + (1.000D - 10)x^{11} \ .$$

The associated Newton polygon is shown on Figure 3.4. There are two tropical roots, $\alpha^- := 10^{-41/7} \simeq 1.39D - 6$ with multiplicity 7 and $\alpha^+ := 10^{50/4} \simeq$

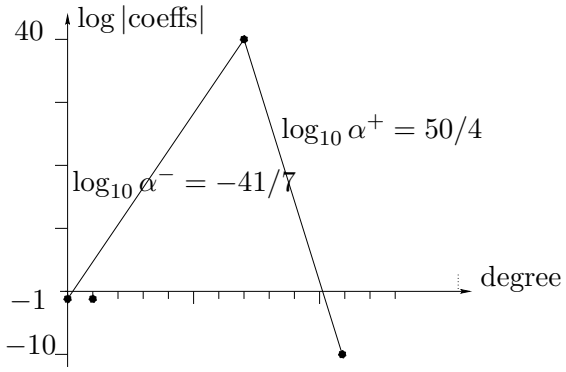


Figure 3.4: Newton polygon of $p(x) = 0.1 + 0.1x + (1.000D + 40)x^7 + (1.000D - 10)x^{11}$.

$3.16D + 12$, with multiplicity 4. Then, $\delta_1 < 10^{-18}$ and due to Theorem 3.3.3, $p(x)$ has 7 roots with

$$\frac{1}{2} \times (1.39D - 6) < |z| < 2 \times (1.39D - 6) \ ,$$

and 4 roots with the order of magnitude

$$\frac{1}{2} \times 3.16D + 12 < |z| < 2 \times 3.16D + 12 \ .$$

However, the results of calling the root function in Matlab version 7.11.0 which is shown in Figure 3.4 is different. In other words, Matlab fails to compute correctly the group of small roots. These kind of examples can be easily made by setting

```
>> abs(roots(p))

ans =

   3.1623e+12
   3.1623e+12
   3.1623e+12
   3.1623e+12
            0
            0
            0
            0
            0
            0
            0

fx >>
```

Figure 3.5: The result of calling root function on $p = 0.1+0.1x+1.000D+40x^7+1.000D-10x^11$ in Matlab.

small enough values for δ_is. This observation shows that the theoretical results of this chapter can be used in numerical methods, at least for the verification of the results. Since the computation of tropical roots can be done in linear time, the execution time of the verification test is negligible.

3.5 Conclusion

In this chapter we considered the relation between the tropical roots and the classical (complex) roots of a given polynomial. We showed that the tropical roots can provide a priori estimation of the modulus of the roots. This principle is at the origin of the scaling of matrix polynomials which will be introduced in the next chapter.

CHAPTER 4

Tropical scaling of polynomial eigenvalue problem*

The eigenvalues of a matrix polynomial can be determined classically by solving a generalized eigenproblem for a linearized matrix pencil, for instance by writing the matrix polynomial in companion form. We introduce a general scaling technique, based on tropical algebra, which applies in particular to this companion form. This scaling relies on the computation of "tropical roots". We give explicit bounds, in a typical case, indicating that these roots provide estimates of the order of magnitude of the different eigenvalues, and we show by experiments that this scaling improves the backward stability of the computations, particularly in situations in which the data have various orders of magnitude. In the case of quadratic polynomial matrices, we recover in this way a scaling due to Fan, Lin, and Van Dooren, which coincides with the tropical scaling when the two tropical roots are equal. If not, the eigenvalues generally split in two groups, and the tropical method leads to make one specific scaling for each of the groups.

*The results of this chapter have been partly reported in [1, 7, 5, 6, 4].

4.1 Introduction

Consider the classical problem of computing the eigenvalues of a matrix polynomial

$$P(\lambda) = A_0 + A_1\lambda + \cdots + A_d\lambda^d \ ,$$

where $A_l \in \mathbb{C}^{n\times n}, l = 0 \ldots d$ are given. Recall that the eigenvalues are defined as the solutions of $\det(P(\lambda)) = 0$. If λ is an eigenvalue, the associated right and left eigenvectors x and $y \in \mathbb{C}^n$ are the non-zero solutions of the systems $P(\lambda)x = 0$ and $y^*P(\lambda) = 0$, respectively. A common way to solve this problem, is to convert P into a "linearized" matrix pencil

$$L(\lambda) = \lambda X + Y, \quad X, Y \in \mathbb{C}^{nd\times nd} \ ,$$

with the same spectrum as P and solve the eigenproblem for L, by standard numerical algorithms like the QZ method [MS73]. If D and D' are invertible diagonal matrices, and if α is a non-zero scalar, we may consider equivalently the scaled pencil $DL(\alpha\lambda)D'$.

The problem of finding the good linearizations and the good scalings has received a considerable attention. The backward error and conditioning of the matrix pencil problem and of its linearizations have been investigated in particular in works of Tisseur, Li, Higham, and Mackey, see [Tis00, HLT07, HMT06, AA09].

A scaling on the eigenvalue parameter to improve the normwise backward error of a quadratic matrix polynomial was proposed by Fan, Lin, and Van Dooren [FLVD04]. This scaling only relies on the norms $\gamma_l := \|A_l\|$, $l = 0, 1, 2$. In this chapter, we introduce a new family of scalings, which also rely on these norms. The degree d is now arbitrary.

As it is mentioned in chapter 2, these scalings originate from the work of Akian, Bapat, and Gaubert [ABG05, ABG04], in which the entries of the matrices A_l are functions, for instance Puiseux series, of a (perturbation) parameter t. The valuations (leading exponents) of the Puiseux series representing the different eigenvalues were shown to coincide, under some genericity conditions, with the points of non-differentiability of the value function of a parametric optimal assignment problem , a result, which can be interpreted in terms of amoebas [PT05, IMS07].

The scaling that we propose in this chapter is based on the tropical roots which relies only on the norms $\gamma_l = \|A_l\|$. A better scaling may be achieved by considering the tropical eigenvalues, which will be introduced in the next chapter. But computing these eigenvalues requires $O(nd)$ calls to an optimal assignment algorithm, whereas the tropical roots considered here can be computed in $O(d)$ time.

As an illustration, consider the following quadratic matrix polynomial

$$P(\lambda) = \lambda^2 10^{-18} \begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix} + \lambda \begin{pmatrix} -3 & 10 \\ 16 & 45 \end{pmatrix} + 10^{-18} \begin{pmatrix} 12 & 15 \\ 34 & 28 \end{pmatrix} \quad .$$

By applying the QZ algorithm on the first companion form of $P(\lambda)$ we get the eigenvalues -Inf,- 7.731e-19 , Inf, 3.588e-19, by using the scaling proposed in [FLVD04] we get -Inf, -3.250e-19, Inf, 3.588e-19. However by using the tropical scaling we can find the four eigenvalues properly: - 7.250e-18 ± 9.744e-18i, -2.102e+17 ± 7.387e+17i. The result was shown to be correct (actually, up to a 14 digits precision) with PARI, in which an arbitrarily large precision can be set. The above computations were performed in Matlab (version 7.3.0).

This chapter is organized as follows. Section 4.2 states preliminary results concerning matrix pencils, linearization and normwise backward error. In Section 4.3, we describe our scaling method. In Section 4.4, we give a theorem locating the eigenvalues of a matrix polynomial, which provides some theoretical justification of the method. Finally in Section 4.5, we present the experimental results showing that the tropical scaling can highly reduce the normwise backward error of an eigenpair. We consider the quadratic case in Section 4.5.1 and the general case in Section 4.5.2. For the quadratic case, we compare our results with the scaling proposed in [FLVD04].

4.2 Matrix pencil and normwise backward error

Let us come back to the eigenvalue problem for the matrix pencil $P(\lambda) = A_0 + A_1\lambda + \cdots + A_d\lambda^d$. There are many ways to construct a "linearized" matrix pencil $L(\lambda) = \lambda X + Y, \quad X, Y \in \mathbb{C}^{nd\times nd}$ with the same spectrum as $P(\lambda)$, see [MMMM06] for a general discussion. In particular, the first companion form $\lambda X_1 + Y_1$ is defined by

$$X_1 = \mathrm{diag}(A_k, I_{(k-1)n}), \qquad Y_1 = \begin{pmatrix} A_{k-1} & A_{k-2} & \dots & A_0 \\ -I_n & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots \\ 0 & \dots & -I_n & 0 \end{pmatrix} \quad .$$

In the experimental part of this work, we are using this linearization.

we shall consider, as in [Tis00], normwise backward error To measure the stability of a numerical algorithm computing an eigenpair. The latter arises when considering a perturbation

$$\Delta P = \Delta A_0 + \Delta A_1\lambda + \cdots + \Delta A_d\lambda^d \quad .$$

The backward error of an approximate eigenpair $(\tilde{x}, \tilde{\lambda})$ of P is defined by

$$\eta(\tilde{x}, \tilde{\lambda}) = \min\{\epsilon : (P(\tilde{\lambda}) + \Delta P(\tilde{\lambda}))\tilde{x} = 0, \|\Delta A_l\|_2 \le \epsilon\|E_l\|_2, l = 0, \dots m\} \quad .$$

The matrices E_l representing tolerances. The following computable expression for $\eta(\tilde{x}, \tilde{\lambda})$ is given in the same reference,

$$\eta(\tilde{x}, \tilde{\lambda}) = \frac{\|r\|_2}{\tilde{\alpha}\|\tilde{x}\|_2} \ ,$$

where $r = P(\tilde{\lambda})\tilde{x}$ and $\tilde{\alpha} = \sum |\tilde{\lambda}|^l \|E_l\|_2$. In the sequel, we shall take $E_l = A_l$.

Our aim is to reduce the normwise backward error, by a scaling of the eigenvalue $\lambda = \alpha\mu$, where α is the scaling parameter. This kind of scaling for quadratic matrix polynomial was proposed by Fan, Lin and Van Dooren [FLVD04]. We next introduce a new scaling, based on the tropical roots.

4.3 Construction of the tropical scaling

Consider the matrix pencil modified by the substitution $\lambda = \alpha\mu$

$$\tilde{P}(\mu) = \tilde{A}_0 + \tilde{A}_1\mu + \cdots + \tilde{A}_d\mu^d \ ,$$

where $\tilde{A}_i = \beta\alpha^i A_i$. The tropical scaling, which we next introduce is characterized by the property that α and β are such that $\tilde{P}(\mu)$ has at least two matrices $\tilde{A}_i$ with an (induced) Euclidean norm equal to one, whereas the Euclidean norm of the other matrices are all bounded by one. The theorem on the location of the eigenvalues, which is stated in the next section provides some justification for the present scaling.

We associate to the original pencil the max-times polynomial

$$\mathtt{t}p(x) = \max(\gamma_0, \gamma_1\lambda, \cdots, \gamma_d\lambda^d) \ ,$$

where

$$\gamma_i := \|A_i\| \ ,$$

(the symbol $\mathtt{t}$ stands for "tropical"). Let $\alpha_1 \leq \alpha_2 \leq \ldots \leq \alpha_d$ be the tropical roots of $\mathtt{t}p(x)$ counted with multiplicities. For each α_i, the maximum is attained by at least two mononomials. Subsequently, the transformed polynomial $q(x) := \beta_i \mathtt{t}p(\alpha_i x)$, with $\beta_i := (\mathtt{t}p(\alpha_i))^{-1}$ has two coefficients of modulus one, and all the other coefficients have modulus less than or equal to one. Thus $\alpha = \alpha_i$ and $\beta = \beta_i$ will satisfy the goal.

The idea is to apply this scaling for all the tropical roots of $\mathtt{t}p(x)$ and each time, to compute n out of nd eigenvalues of the corresponding scaled matrix pencil, because replacing $P(\lambda)$ by $P(\alpha_i\mu)$ is expected to decrease the backward error for the eigenvalues of order α_i, while possibly increasing the backward error for the other ones.

More precisely, let $\alpha_1 \leq \alpha_1 \leq \ldots \leq \alpha_d$ denote the tropical roots of $\mathtt{t}p(x)$. Also let

$$\underbrace{\mu_1, \ldots, \mu_n}, \underbrace{\mu_{n+1}, \ldots, \mu_{2n}}, \ldots, \underbrace{\mu_{(d-1)n+1}, \ldots, \mu_{nd}} \ ,$$

be the eigenvalues of $\tilde{P}(\mu)$ sorted by increasing modulus, computed by setting $\alpha = \alpha_i$ and $\beta = \mathtt{t}p(\alpha_i)^{-1}$ and partitioned in d different groups. Now, we choose the ith group of n eigenvalues, multiply by α_i and put in the list of computed eigenvalues. By applying this iteration for all $i = 1 \dots d$, we will get the list of the eigenvalues of $P(\lambda)$. Taking into account this description, we arrive at Algorithm 1. It should be understood here that in the sequence $\mu_1, \dots, \mu_{nd}$ of eigenvalues above, only the eigenvalues of order α_i are hoped to be computed properly. Indeed, in some extreme cases in which the tropical roots have very different orders of magnitude (as in the example shown in the introduction), the eigenvalues of order α_i turn out to be accurate whereas the groups of higher orders have some eigenvalues Inf or Nan.

Algorithm 4.1 Computing the eigenvalues using the tropical scaling

INPUT: Matrix pencil $P(\lambda)$
OUTPUT: List of eigenvalues of $P(\lambda)$
Compute the corresponding tropical polynomial $\mathtt{t}p(x)$
Find the tropical roots of $\mathtt{t}p(x)$
for all tropical root such as α_i **do**
  Compute the tropical scaling based on α_i
  Compute the eigenvalues using the QZ algorithm and sort them by increasing modulus
  Choose the ith group of the eigenvalues
end for

To illustrate the algorithm, let $P(\lambda) = A_0 + A_1\lambda + A_2\lambda^2$ be a quadratic matrix polynomial and let $\mathtt{t}p(\lambda) = \max(\gamma_0, \gamma_1\lambda, \gamma_2\lambda^2)$ be the tropical polynomial corresponding to this quadratic matrix polynomial.

We refer to the tropical roots of $\mathtt{t}p(x)$ by $\alpha^+ \geq \alpha^-$. If $\alpha^+ = \alpha^-$, which happens when $\gamma_1^2 \leq \gamma_0\gamma_2$ then, $\alpha = \sqrt{\frac{\gamma_0}{\gamma_2}}$ and $\beta = \mathtt{t}p(\alpha)^{-1} = \gamma_0^{-1}$. This case coincides with the scaling of [FLVD04] in which $\alpha^* = \sqrt{\frac{\gamma_0}{\gamma_2}}$.

When $\alpha^+ \neq \alpha^-$, we will have two different scalings based on $\alpha^+ = \frac{\gamma_1}{\gamma_2}$, $\alpha^- = \frac{\gamma_0}{\gamma_1}$ and two different β corresponding to the two tropical roots:

$$\beta^+ = \mathtt{t}p(\alpha^+)^{-1} = \frac{\gamma_2}{\gamma_1^2}, \qquad \beta^- = \mathtt{t}p(\alpha^-)^{-1} = \frac{1}{\gamma_0} \ .$$

To compute the eigenvalues of $P(\lambda)$ by using the first companion form linearization, we apply the scaling based on α^+, which yields

$$\lambda \begin{pmatrix} \frac{1}{\gamma_2}A_2 & \\ & I \end{pmatrix} + \begin{pmatrix} \frac{1}{\gamma_1}A_1 & \frac{\gamma_2}{\gamma_1^2}A_0 \\ -I & 0 \end{pmatrix} \ ,$$

to compute the n largest eigenvalues. We apply the scaling based on α^-, which

yields

$$\lambda \begin{pmatrix} \frac{\gamma_0}{\gamma_1^2} A_2 & \\ & I \end{pmatrix} + \begin{pmatrix} \frac{1}{\gamma_1} A_1 & \frac{1}{\gamma_2} A_0 \\ -I & 0 \end{pmatrix} ,$$

to compute the n smallest eigenvalues.

In general, let $\alpha_1 \leq \alpha_1 \leq \ldots \leq \alpha_d$ be the tropical roots of $\mathtt{t}p(x)$ counted with multiplicities. To compute the ith largest group of eigenvalues, we perform the scaling for α_i, which yields the following linearization:

$$\lambda \begin{pmatrix} \beta\alpha_i^d A_d & & & \\ & I & & \\ & & \ddots & \\ & & & I \\ & & & & I \end{pmatrix} + \begin{pmatrix} \beta\alpha_i^{d-1} A_{d-1} & \ldots & \beta\alpha_i A_1 & \beta A_0 \\ -I & 0 & \ldots & 0 \\ 0 & -I & \ddots & \vdots \\ \vdots & & \ddots & 0 \\ 0 & \ldots & -I & 0 \end{pmatrix} ,$$

where $\beta = \mathtt{t}p(\alpha_i)^{-1}$. Doing the same for all the distinct tropical roots, we can compute all the eigenvalues.

Remark 4. The interest of Algorithm 4.1 lies in the backward stability (since it allows us to solve instances in which the data have various order of magnitudes). However, its inconvenient is to call several times (once for each distinct tropical eigenvalue, and so, at most d times) the QZ algorithm. To increase the speed, we may partition the different tropical eigenvalues in groups consisting each of eigenvalues of the same order of magnitude, and then, the speed factor we would loose would be reduced to the number of different groups.

4.4 Splitting of the eigenvalues in tropical groups

In this section we provide theoretical results showing that the tropical roots can provide an a priori estimation of the modulus of the eigenvalues of a matrix polynomial problem.

We shall need to compare spectra, which may be thought of as unordered sets, therefore, we define the following metric (eigenvalue variation), which appeared in [GH08]. We shall use the notation spec for the spectrum of a matrix or a pencil.

Definition 4.4.1. Let $\lambda_1, \ldots \lambda_n$ and $\mu_1 \ldots \mu_n$ denote two sequences of complex numbers. The variation between λ and μ is defined by

$$v(\lambda, \mu) := \min_{\pi \in S_n} \{ \max_i |\mu_{\pi(i)} - \lambda_i| \} ,$$

where S_n is the set of permutations of $\{1, 2, \ldots, n\}$. If $A, B \in \mathbb{C}^{n \times n}$, the eigenvalue variation of A and B is defined by $v(A, B) := v(\operatorname{spec} A, \operatorname{spec} B)$.

Recall that the quantity $v(\lambda, \mu)$ can be computed in polynomial time as soon as λ and μ are known, by solving a bottleneck assignment problem.

We shall need the following theorem of Bathia, Elsner, and Krause [BEK90].

Theorem 4.4.1 ([BEK90]). *Let $A, B \in \mathbb{C}^{n\times n}$. Then $v(A,B) \leq 4\times 2^{-1/n}(\|A\|_2 + \|B\|_2)^{1-1/n}\|A-B\|_2^{1/n}$.*

A similar inequality holds, more generally, with a different constant for any operator norm [BEK90]; However, in this section, we consider only the spectral norm. We shall use the notation "cond" to refer to the condition number of a given matrix with respect to spectral norm.

We associate to a matrix polynomial, $P(\lambda) = A_0 + \lambda A_1 + \ldots + \lambda^d A_d$, a max-times polynomial, $\mathtt{t}p(x) = \gamma_0 \oplus \gamma_1 x \oplus \cdots \oplus \gamma_d x^d$, where $\gamma_i := \|A_i\|_2$. The Newton polygon of $\mathtt{t}p(x)$ is shown in Figure 4.1. Here, $\alpha_{\min} = \alpha_1 < \alpha_2 < \ldots < \alpha_p = \alpha_{\max}$ denote the tropical roots of $\mathtt{t}p(x)$. Also, $k_0 = 0, k_1, \ldots, k_p$ denote the horizontal coordinates of the vertices belonging to the Newton polygon. We define $P_{\alpha_{\max}}(\lambda) = A_{k_{p-1}}\lambda^{k_{p-1}} + \cdots + A_{k_p}\lambda^{k_p}$ to be the matrix polynomial corresponding to the last edge of the Newton polygon. In the following theorem we compare the spectrum of $P(\lambda)$ with the spectrum of $P_{\alpha_{\max}}(\lambda)$. We also show that every nonzero eigenvalue of $P_{\alpha_{\max}}(\lambda)$ can be bounded from upper and below by the largest tropical root, $\alpha_{\max}$.

Theorem 4.4.2 (Tropical splitting of eigenvalues). *Let $P(\lambda) = A_0 + \lambda A_1 + \ldots + \lambda^d A_d$ be a matrix polynomial of degree d and $\mathtt{t}p(x) = \gamma_0 \oplus \gamma_1 x \oplus \cdots \oplus \gamma_d x^d$ be the corresponding max-times polynomial where $\gamma_i := \|A_i\|_2$. Also, let $\alpha_{\min} = \alpha_1 < \alpha_2 < \ldots < \alpha_p = \alpha_{\max}$ denote the tropical roots of $\mathtt{t}p(x)$ with multiplicities $m_1, \ldots m_p$, respectively, where $\sum_{i=1}^p m_i = d$. We assume that $\mathtt{t}p(x)$ has more than one tropical root. Let $k_0 = 0, k_1, \ldots, k_p$ denote the horizontal coordinates of the vertices belonging to the Newton polygon of $\mathtt{t}p(x)$ which is shown in Figure 4.1. so that, $m_1 = k_1, m_2 = k_2 - k_1, \ldots, m_p = k_p - k_{p-1}$. Also, let $\delta_1 = \frac{\alpha_1}{\alpha_2}, \ldots, \delta_{p-1} = \frac{\alpha_{p-1}}{\alpha_p}$ be the parameters, which measure the separation between the tropical roots. Assume that A_d and $A_{k_{p-1}}$ are nonsingular. Then,*

$$v(\operatorname{spec} P(\lambda), \operatorname{spec} P_{\alpha_{\max}}(\lambda)) \leq C\alpha_{\max}\left(\frac{\delta_{p-1}}{1-\delta_{p-1}}\right)^{\frac{1}{nd}} (\operatorname{cond} A_d)^{\frac{1}{nd}} \quad , \tag{4.1}$$

where $C = 4\times 2^{-\frac{1}{nd}}(2 + \operatorname{cond} A_d(2m_p + \frac{\delta_{p-1}}{1-\delta_{p-1}}))^{1-\frac{1}{nd}}$. Also, every nonzero eigenvalue, λ, of $P_{\alpha_{\max}}(\lambda)$ satisfies

$$\alpha_{\max}(1 + d\operatorname{cond} A_{k_{p-1}})^{-1} \leq |\lambda| \leq \alpha_{\max}(1 + d\operatorname{cond} A_d) \ . \tag{4.2}$$

Remark 5. When A_0 and A_{k_1} are nonsingular, a similar argument can be made for the matrix polynomial corresponding to the first edge of the Newton polygon, $P_{\alpha_{\min}}(\lambda) = A_0 + \cdots + A_{k_1}\lambda^{k_1}$, by considering $\alpha_{\min}$ and $\lambda^d P(\lambda^{-1})$.

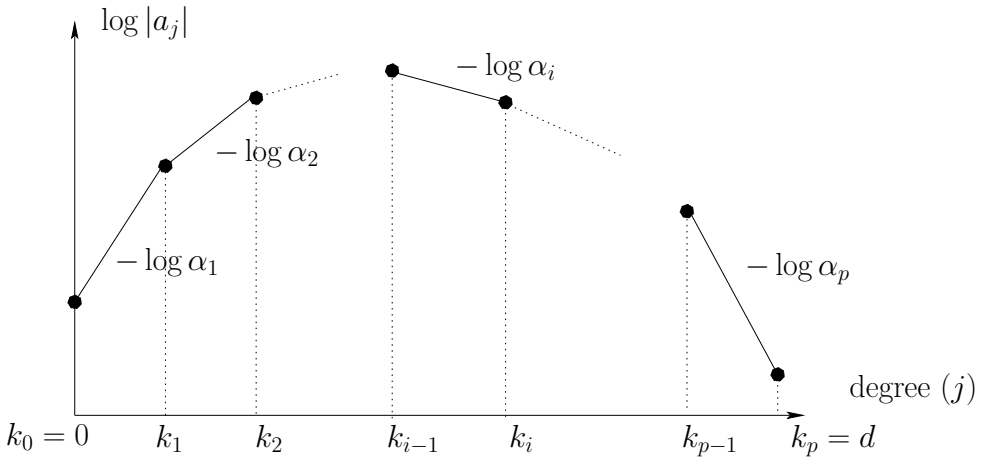


Figure 4.1: Newton polygon corresponding to $\mathtt{t}p(x)$.

The proof relies on the next lemmas.

Lemma 4.4.3. *Let* $\tilde{P}(\mu) = (\mathtt{t}p(\alpha_{\max}))^{-1}P(\alpha_{\max}\lambda) = \tilde{A}_0 + \tilde{A}_1\mu + \cdots + \tilde{A}_d\mu^d$ *be the scaled matrix polynomial by using the largest tropical root,* $\alpha_{\max}$*. Then, the following inequalities hold*

$$\|\tilde{A}_i\|_2 \leq \delta_{p-1}^{k_{p-1}-i} \quad \textit{for } i = 1 \dots k_{p-1} - 1 \ , \tag{4.3}$$

and

$$\|\tilde{A}_i\|_2 \leq 1 \quad \textit{for } i = k_{p-1} \dots d \ . \tag{4.4}$$

Proof. Due to Proposition 3.3.2, $\gamma_i \leq \gamma_{k_{p-1}}\alpha_{p-1}^{k_{p-1}-i}$ for all $0 \leq i < k_{p-1}$. Thus,

$$\begin{aligned}(\mathtt{t}p(\alpha_p))^{-1}\gamma_i\alpha_p^i &\leq (\gamma_{k_{p-1}}\alpha_p^{k_{p-1}})^{-1}\gamma_{k_{p-1}}\alpha_{p-1}^{k_{p-1}-i}\alpha_p^i \\ &= (\frac{\alpha_{p-1}}{\alpha_p})^{k_{p-1}-i} = \delta_{p-1}^{k_{p-1}-i}\end{aligned}$$

which proves the first statement. Since $(\mathtt{t}p(\alpha_p))^{-1}\gamma_i\alpha_p^i \leq 1$ for all $k_{p-1} \leq i \leq d$ the second statement is also established. □

Lemma 4.4.4. *Let* $\tilde{P}(\mu)$ *be the scaled matrix polynomial defined in Lemma 4.4.3. Also let* $\tilde{P}_{\alpha_{\max}}(\mu) = (\mathtt{t}p(\alpha_{\max}))^{-1}P_{\alpha_{\max}}(\alpha_{\max}\lambda)$ *be the scaled matrix polynomial corresponding to the last edge of the Newton polygon shown in Figure 4.1. Assume that* A_d *is nonsingular and let*

$$L_{\tilde{P}(\mu)} = \begin{pmatrix} \tilde{A}_d^{-1}\tilde{A}_{d-1} & \tilde{A}_d^{-1}\tilde{A}_{d-2} & \dots & \tilde{A}_d^{-1}\tilde{A}_0 \\ -I_n & 0 & \dots & 0 \\ \vdots & \ddots & \vdots & \vdots \\ 0 & \dots & -I_n & 0 \end{pmatrix} \ ,$$

$$L_{\tilde{P}_{\alpha_{\max}}(\mu)} = \begin{pmatrix} \tilde{A}_d^{-1}\tilde{A}_{d-1} & \dots & \tilde{A}_d^{-1}\tilde{A}_{k_{p-1}} & \dots & 0 & 0 \\ -I_n & 0 & \dots & \dots & 0 & 0 \\ \vdots & \ddots & \vdots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & \dots & \dots & \dots & -I_n & 0 \end{pmatrix},$$

denote the first companion forms of $\tilde{P}(\mu)$ *and* $\tilde{P}_{\alpha_{\max}}(\mu)$ *respectively. Define* $\Delta L = L_{\tilde{P}(\mu)} - L_{\tilde{P}_{\alpha_{\max}}(\mu)}$. *Then, the following statements hold:*

- $\|\Delta L\|_2 = \|L_{\tilde{P}(\mu)} - L_{\tilde{P}_{\alpha_{\max}}(\mu)}\|_2 \leq \operatorname{cond} \tilde{A}_d \frac{\delta_{p-1}}{1-\delta_{p-1}}$
- $\|L_{\tilde{P}(\mu)}\|_2 \leq 1 + \operatorname{cond} A_d(m_p + \frac{\delta_{p-1}}{1-\delta_{p-1}})$
- $\|L_{\tilde{P}_{\alpha_{\max}}(\mu)}\|_2 \leq 1 + m_p \operatorname{cond} A_d$

Proof. It follows from the inequality 4.3 and trivial norm inequalities that:

$$\begin{aligned} \|\Delta L\|_2 = \|L_{\tilde{P}(\mu)} - L_{\tilde{P}_{\alpha_{\max}}(\mu)}\|_2 &\leq \operatorname{cond} \tilde{A}_d \sum_{i=0}^{k_{p-1}-1} \|\tilde{A}_i\|_2 \\ &\leq \operatorname{cond} \tilde{A}_d \sum_{i=0}^{k_{p-1}-1} \delta_{p-1}^{k_{p-1}-i} \leq \operatorname{cond} \tilde{A}_d \frac{\delta_{p-1}}{1-\delta_{p-1}} \end{aligned}$$

To prove the second statement we have,

$$\begin{aligned} \|L_{\tilde{P}(\mu)}\|_2 &\leq 1 + \|(\tilde{A}_d)^{-1}\|_2 \sum_{i=0}^{d-1} \|\tilde{A}_i\|_2 \\ &\leq 1 + (\operatorname{cond} \tilde{A}_d)(d - k_{p-1} + \sum_{i=0}^{k_{p-1}} \delta_{p-1}^{k_{p-1}-i}) \quad \textit{since} \quad \operatorname{cond} \tilde{A}_d = \operatorname{cond} A_d \\ &\leq 1 + \operatorname{cond} A_d(m_p + \frac{\delta_{p-1}}{1-\delta_{p-1}}) \quad \textit{due to Eqs. 4.3 and 4.4} \end{aligned}$$

A similar argument can be made to prove the last statement. □

Remark 6. When $\mathtt{t}p(x)$ has only two tropical roots and the multiplicity of the smallest tropical root is one, $k_{p-1} - 1 = 0$. In this case, $\|\Delta L\|_2 \leq \delta \operatorname{cond} A_d$, where $\delta = \frac{\alpha_{\min}}{\alpha_{\max}}$ And $\|L_{\tilde{P}(\mu)}\|_2 \leq 1 + \operatorname{cond} A_d(m_p + \delta)$.

The statement 4.1 of Theorem 4.4.2 can be achieved by applying theorem 4.4.1 and lemmas 4.4.3 and 4.4.4. Next lemma, will prove Equation 4.2.

Lemma 4.4.5 (Corollary of [HT03, Lemma 2.2])**.** *Let* $P(\lambda) = A_0 + A_1\lambda + \ldots + A_d\lambda$ *be a matrix polynomial of degree* d *and assume that* $\|A_0\|_2, \|A_d\|_2 = 1$ *and* $\|A_1\|_2 \ldots \|A_{d-1}\|_2 \leq 1$. *Also, assume that* A_1 *and* A_d *are nonsingular. Then, the*

modulus of the eigenvalues of $P(\lambda)$ can be bounded by the condition numbers of A_0 and A_d as follows:

$$(1 + d \operatorname{cond} A_0)^{-1} \leq |\lambda| \leq 1 + d \operatorname{cond} A_d \ . \tag{4.5}$$

Proof. The eigenvalues of $P(\lambda)$ coincide with the eigenvalues of the first companion form of $A_d^{-1}P(\lambda)$, that is,

$$L = \begin{pmatrix} A_d^{-1}A_{d-1} & A_d^{-1}A_{d-2} & \dots & A_d^{-1}A_0 \\ -I_n & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots \\ 0 & \dots & -I_n & 0 \end{pmatrix} \ .$$

The following inequality can be easily verified by the properties of the norms

$$\|L\|_2 \leq 1 + \sum_{i=0}^{d-1} \|A_i\|_2 \|A_d^{-1}\|_2 \leq 1 + d \operatorname{cond} A_d \ .$$

Since $\rho(A) = \max_{\lambda \in \sigma(A)} |\lambda| \leq \|A\|_2$ where $\sigma(A)$ denotes the spectrum of A, we have

$$|\lambda| \leq 1 + d \operatorname{cond} A_d \ .$$

The left inequality can be achieved by considering $A_0^{-1}P(\frac{1}{\lambda})$ and making the same argument. □

Corollary 4.4.6. *When $\mathtt{t}p(x)$ has only one tropical root, α, then, due to the scaling equation, $\mu = \alpha\lambda$ the modulus of the eigenvalues of $P(\lambda)$ are bounded by the tropical root as the following*

$$(1 + d \operatorname{cond} A_0)^{-1}\alpha \leq |\lambda| \leq \alpha(1 + d \operatorname{cond} A_d) \ . \tag{4.6}$$

Remark 7. Let $P(\lambda) = A_0 + A_1\lambda$ be a matrix polynomial of degree one. Then, the inequality 4.5 can be improved to

$$(\operatorname{cond} A_0)^{-1} \leq |\lambda| \leq \operatorname{cond} A_1 \ .$$

Remark 8. If A_0 and A_d are well conditioned then the order of magnitude of the eigenvalues of $P(\lambda)$ are expected to be of order one.

Corollary 4.4.7 (Quadratic matrix polynomial)**.** *for a quadratic matrix polynomial, $P(\lambda) = \lambda^2 A_2 + \lambda A_1 + A_0$, let $\alpha_{\min}, \alpha_{\max}$ be two tropical roots of $\mathtt{t}p(x)$ and $\delta = \frac{\alpha_{\min}}{\alpha_{\max}}$. So, $P_{\alpha_{\max}} = A_1\lambda + A_2\lambda^2$ and due to Remark 6,*

$$v(\operatorname{spec} P, P_{\alpha_{\max}}) \leq C\alpha_{\max}\delta^{1/2n} \ ,$$

where

$$C := 4 \times 2^{-1/2n}\big(2 + 2 \operatorname{cond} A_2 + \delta \operatorname{cond} A_2\big)^{1-1/2n}\big(\operatorname{cond} A_2\big)^{1/2n} \ .$$

Also, due to Remark 7, the nonzero eigenvalues of $P_{\alpha_{\max}}$ are bounded by

$$\alpha_{\max}(\operatorname{cond} A_1)^{-1} \leq |\lambda| \leq \alpha_{\max} \operatorname{cond} A_2 \ .$$

Thus, when the parameter δ measuring the separation between the two tropical roots is sufficiently small, and when the matrices A_2, A_1 are well conditioned, then, there are precisely n eigenvalues of the order of the maximal tropical. By applying the same result to the reciprocal pencil, we deduce, under the same separation condition, that when A_1, A_0 are well conditioned, there are precisely n eigenvalues of the order of the minimal tropical root.

Remark 9. In view of the asymptotic results of [ABG04], the exponentials of the tropical eigenvalues , which will be introduced in the next chapter, are expected to provide a better estimation of the moduli of the complex roots. This alternative approach is the object of a further work, however, the comparative interest of the tropical roots considered here lies in their simplicity: they only depend on the norms of $A_0, \ldots, A_d$, and can be computed in linear time from these norms. They can also be used as a measure of ill-posedness of the problem (when the tropical roots have different orders of magnitude, the standard methods in general fail).

4.5 Experimental Results

4.5.1 Quadratic polynomial matrices

Consider first $P(\lambda) = A_0 + A_1\lambda + A_2\lambda^2$ and its linearization $L = \lambda X + Y$. Let z be the eigenvector computed by applying the QZ algorithm to this linearization. Both $\zeta_1 = z(1:n)$ and $\zeta_2 = z(n+1:2n)$ are eigenvectors of $P(\lambda)$. We present our results for both of these eigenvectors; η_s denotes the normwise backward error for the scaling of [FLVD04], and η_t denotes the same quantity for the tropical scaling.

Our first example coincides with Example 3 of [FLVD04] where $\|A_2\|_2 \approx 5.54 \times 10^{-5}, \|A_1\|_2 \approx 4.73 \times 10^{3}, \|A_0\|_2 \approx 6.01 \times 10^{-3}$ and $A_i \in \mathbb{C}^{10\times 10}$. We used 100 randomly generated pencils normalized to get the mentioned norms and we computed the average of the quantities mentioned in the following table for these pencils. Here we present the results for the 5 smallest eigenvalues, however for all the eigenvalues, the backward error computed by using the tropical scaling is of order 10^{-16}, which is the precision of the computation. The computations were carried out in SCILAB 4.1.2. The code can be found in Appendix B.

$\lvert\lambda\rvert$	$\eta(\zeta_1,\lambda)$	$\eta(\zeta_2,\lambda)$	$\eta_s(\zeta_1,\lambda)$	$\eta_s(\zeta_2,\lambda)$	$\eta_t(\zeta_1,\lambda)$	$\eta_t(\zeta_2,\lambda)$
2.98E-07	1.01E-06	4.13E-08	5.66E-09	5.27E-10	6.99E-16	1.90E-16
5.18E-07	1.37E-07	3.84E-08	8.48E-10	4.59E-10	2.72E-16	1.83E-16
7.38E-07	5.81E-08	2.92E-08	4.59E-10	3.91E-10	2.31E-16	1.71E-16
9.53E-07	3.79E-08	2.31E-08	3.47E-10	3.36E-10	2.08E-16	1.63E-16
1.24E-06	3.26E-08	2.64E-08	3.00E-10	3.23E-10	1.98E-16	1.74E-16

In the second example, we consider a matrix pencil with $\|A_2\|_2 \approx 10^{-6}$, $\|A_1\|_2 \approx 10^3$, $\|A_0\|_2 \approx 10^5$ and $A_i \in \mathbb{C}^{40\times 40}$. Again, we use 100 randomly generated pencils with the mentioned norms and we compute the average of all the quantities presented in the next table. We present the results for the 5 smallest eigenvalues. This time, the computations shown are from MATLAB 7.3.0, actually, the results are insensitive to this choice, since the versions of MATLAB and SCILAB we used both rely on the QZ algorithm of Lapack library (version 3.0). More details about the code can be found in Appendix B.

$\lvert\lambda\rvert$	$\eta(\zeta_1,\lambda)$	$\eta(\zeta_2,\lambda)$	$\eta_s(\zeta_1,\lambda)$	$\eta_s(\zeta_2,\lambda)$	$\eta_T(\zeta_1,\lambda)$	$\eta_T(\zeta_2,\lambda)$
1.08E+01	2.13E-13	4.97E-15	8.98E-12	4.19E-13	5.37E-15	3.99E-16
1.75E+01	5.20E-14	4.85E-15	7.71E-13	4.09E-13	6.76E-16	3.95E-16
2.35E+01	4.56E-14	5.25E-15	6.02E-13	4.01E-13	5.54E-16	3.66E-16
2.93E+01	4.18E-14	5.99E-15	5.03E-13	3.97E-13	4.80E-16	3.47E-16
3.33E+01	3.77E-14	5.28E-15	4.52E-13	3.84E-13	4.67E-16	3.53E-16

4.5.2 Polynomial matrices of degree d

Consider now the matrix polynomial $P(\lambda) = A_0 + A_1\lambda + \cdots + A_d\lambda^d$, and let $L = \lambda X + Y$ be the first companion form linearization of this pencil. If z is an eigenvector for L then $\zeta_1 = z(1:n)$ is an eigenvector for $P(\lambda)$. In the following computations, we use ζ_1 to compute the normwise backward error of Matrix pencil, however this is possible to use any $z(kn+1:n(k+1))$ for $k = 0\ldots d-1$.

To illustrate our results, we apply the algorithm for 20 different randomly generated matrix pencils and then compute the backward error for a specific eigenvalue of these matrix pencils. The 20 values x-axis, in Fig. 4.2 and 4.3, identify the random instance while the y-axis shows the $\log_{10}$ of backward error for a specific eigenvalue. Also we sort the eigenvalues in a decreasing order of their absolute value, so λ_1 is the maximum eigenvalue.

We firstly consider the randomly generated matrix pencils of degree 5 where the order of magnitude of the Euclidean norm of A_i is as follows:

$\|A_0\|$	$\|A_1\|$	$\|A_2\|$	$\|A_3\|$	$\|A_4\|$	$\|A_5\|$
$O(10^{-3})$	$O(10^{2})$	$O(10^{2})$	$O(10^{-1})$	$O(10^{-4})$	$O(10^{5})$

Fig. 4.2 shows the results for this case where the dotted line shows the backward error without scaling and the solid line shows the backward error using the tropical scaling. We show the results for the minimum eigenvalue, the "central" 50^{th} eigenvalue and the maximum one from top to down. In particular, the picture at the top shows a dramatic improvement in the stability of the computation of the smallest eigenvalue, whereas for the largest eigenvalues, the scaling typically improves the backward error by a factor 10. For the central eigenvalue, the improvement we get is intermediate. The second example concerns the randomly generated matrix pencil with degree 10 while the order of the norm of the coefficient matrices are as follows:

$\|A_0\|$	$\|A_1\|$	$\|A_2\|$	$\|A_3\|$	$\|A_4\|$	$\|A_5\|$
$O(10^{-5})$	$O(10^{-2})$	$O(10^{-3})$	$O(10^{-4})$	$O(10^{2})$	$O(1)$

$\|A_6\|$	$\|A_7\|$	$\|A_8\|$	$\|A_9\|$	$\|A_{10}\|$
$O(10^{3})$	$O(10^{-3})$	$O(10^{4})$	$O(10^{2})$	$O(10^{5})$

In this example, the order of the norms differ from 10^{-5} to 10^{5} and the space dimension of A_i is 8. Figure 4.3 shows the results for this case where the dotted line shows the backward error without scaling and the solid line shows the backward error using tropical scaling. Again we show the results for the minimum eigenvalue, the 40th eigenvalue and the maximum one from top to down.

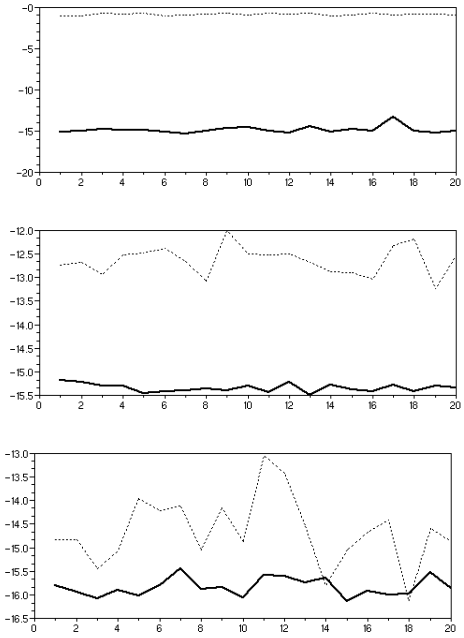


Figure 4.2: Backward error for smallest, medium and largest eigenvalues from top to bottom. The vertical axis shows the $\log_{10}$ of backward error and the horizontal axis shows 20 different randomly generated matrices.

4.6 Conclusion

In this chapter we proposed a new family of scaling based on tropical methods to increase the precision of the computation of the eigenvalues of matrix polynomials. We show that, the presented scaling can be easily applied in numerical solutions. We also provide theoretical justification for the quadratic case. In the next chapter we will introduce the tropical eigenvalues for matrix polynomials. These tropical eigenvalues can provide a better scaling specially when the matrices, A_is, are not well conditioned. The time complexity of computing these

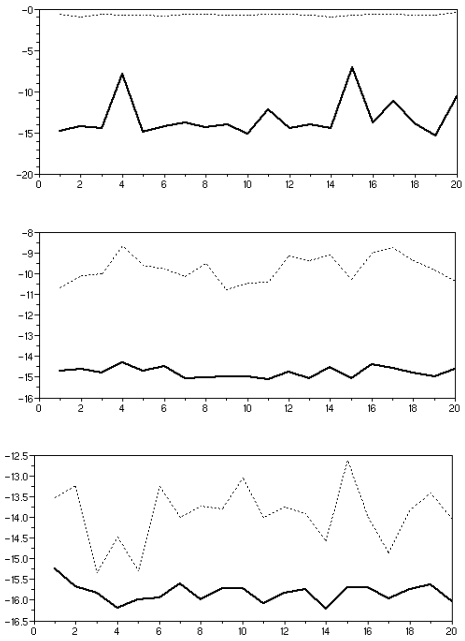


Figure 4.3: Backward error for smallest, medium and largest eigenvalues from top to bottom. The vertical axis shows the $\log_{10}$ of backward error and the horizontal axis shows 20 different randomly generated matrices.

tropical eigenvalues is $O(n^4 d)$. Thus, studying the efficiency of the scaling based on tropical eigenvalue can be the matter of future works from the numerical point of view.

CHAPTER 5

Finding the tropical eigenvalues of a max-plus matrix polynomial

We study the problem of computing the tropical eigenvalues of a tropical matrix polynomial. From the combinatorial perspective, this problem can be interpreted as finding the maximum weighted matching function in a bipartite graph whose weights are convex piecewise linear functions of a variable, λ. Several algorithms to compute the tropical eigenvalues of a matrix have been proposed in [BM00, BB03, GK10]. The algorithm that we develop in this chapter computes the tropical eigenvalues of a generalized problem, i.e. a matrix polynomial, $\mathtt{t}P(\lambda) = A_0 \oplus \lambda \otimes A_1 \oplus \ldots \lambda^d \otimes A_d$ where $A_i \in \mathbb{R}_{\max}^{n \times n}$. This is analogous to the generalization of the eigenvalue problem in the classical linear algebra. This algorithm extends an idea of Burkard and Butkovic [BB03] who considered the special case $\mathtt{t}P(\lambda) = A \oplus \lambda \mathbb{I}$ where $\mathbb{I}$ is the tropical identity matrix. The present algorithm computes all the tropical eigenvalues in $O(n^4 d)$ time where d is the degree of the input matrix polynomial and n is the dimension of the matrices. The Scilab implementation of this algorithm can be found in Appendix C.

5.1 Introduction

The eigenvalues of a matrix A can be computed by finding the roots of its characteristic polynomial, $\det(A-\lambda I)$ where det denotes the determinant and I presents the identity matrix. As it is mentioned in Chapter 2, in the max-plus algebra, an analogue of the notion of determinant, involving a formal "minus" sign, has been studied by several authors [GM84, BCOQ92, AGG09]. However, the simplest approach is to consider the permanent instead of the determinant. The permanent of a matrix A is classically defined as

$$\operatorname{per} A := \sum_{\sigma \in S_n} \prod_{i=1}^{n} (A)_{i\sigma(i)} \ ,$$

where S_n denotes the set of all permutations. When the semiring is $\mathbb{R}_{\max}$, so that $(A)_{ij} \in \mathbb{R} \cup \{-\infty\}$, the permanent, $\operatorname{per} A := \max_{\sigma \in S_n} \sum_{i=1}^{n} (A)_{i\sigma(i)}$, is the value of an optimal assignment problem with weights $(A)_{ij}$. So, in the max-plus algebra, the *formal* tropical characteristic polynomial is defined to be,

$$p_A(\lambda) = \operatorname{per}(A \oplus \lambda \otimes \mathbb{I}) \ , \tag{5.1}$$

where the entries of the matrix $A \oplus \lambda \otimes \mathbb{I}$ are interpreted as formal polynomials with coefficients in $\mathbb{R}_{\max}$ [CG83]. Recall that $\mathbb{I}$ is a matrix of dimension $n \times n$ where all the diagonal entries are 0 and all off-diagonal entries are $-\infty$.

The *numerical* tropical characteristic polynomial is the function

$$p_A : \mathbb{R}_{\max} \to \mathbb{R}_{\max} \qquad \lambda \mapsto p_A(\lambda) \ ,$$

which associates to a parameter $\lambda \in \mathbb{R} \cup \{-\infty\}$, the value of the optimal assignment problem in which the weights are given by the matrix $B = A \oplus \lambda \otimes \mathbb{I}$, i.e.

$$B = \begin{pmatrix} (A)_{11} \oplus \lambda & (A)_{12} & \dots & (A)_{1n} \\ (A)_{21} & (A)_{22} \oplus \lambda & \dots & (A)_{2n} \\ \dots & \dots & & \dots \\ (A)_{n1} & (A)_{n2} & \dots & (A)_{nn} \oplus \lambda \end{pmatrix} \ ,$$

so that $(B)_{ij} = (A)_{ij}$ for $i \neq j$ and $(B)_{ii} = \max((A)_{ii}, \lambda)$.

Following [ABG05, ABG04] the algebraic tropical eigenvalues(refer to Section 2.4), which we refer to, in the sequel, as the tropical eigenvalues, are defined as the tropical roots of the tropical characteristic polynomial $p_A(\lambda)$, i.e., as the nondifferentiability points of the function $\lambda \mapsto p_A(\lambda)$. To the author's knowledge, there is yet no polynomial method for finding all coefficients of the formal tropical characteristic polynomial; however, the numerical tropical characteristic polynomial and the tropical eigenvalues can be computed in polynomial time.

Butkovič and Murfitt [BM00] developed an $O(n^5)$ method to compute all the tropical eigenvalues of an $n \times n$ matrix with entries from $\mathbb{Q} \cup \{-\infty\}$. Later on,

Burkard and Butkovic [BB03] proposed an algorithm, which computes all the tropical eigenvalues of a max-plus matrix in $O(n^2(m + n \log n))$ time where m is the number of finite entries of $A \in \mathbb{R}_{\max}^{n\times n}$. Recently, Gassner and Klinz [GK10], studied the problem of solving the parametric minimum assignment for a matrix B where $(B)_{ij} = (A_0)_{ij} - \lambda(A_1)_{ij}$, $(A_0)_{ij} \in \mathbb{R}$ and $(A_1)_{ij} \in \{0, 1\}$. They developed an algorithm, which works in $O(n(m + n \log n))$ time. They also adopted their algorithm, to run in the same time complexity, to compute the tropical eigenvalues of a min-plus matrix. This new algorithm is n times faster than the one proposed in [BB03].

An obvious generalization of the mentioned problem, is the problem of computing the tropical eigenvalues of a max-plus matrix polynomial. This is analogous to the classical generalization of the eigenvalue problem in classical linear algebra. A max-plus matrix polynomial can be defined as,

$$\mathtt{t}P(\lambda) = A_0 \oplus \lambda \otimes A_1 \oplus \cdots \oplus \lambda^d \otimes A_d \quad A_i \in \mathbb{R}_{\max}^{n\times n} \quad \textit{for } i = 1 \ldots d \ .$$

The formal tropical characteristic polynomial of $\mathtt{t}P(\lambda)$ is defined as

$$f(\lambda) = \operatorname{per}(A_0 \oplus \lambda \otimes A_1 \oplus \cdots \oplus \lambda^d \otimes A_d) \ ,$$

which is a generalization of the one in Equation 5.1. The associated numerical tropical characteristic polynomial, $f(\lambda)$, is defined as

$$f : \mathbb{R}_{\max} \to \mathbb{R}_{\max} \quad f(\lambda) = \operatorname{per}(\mathtt{t}P(\lambda)) \ , \tag{5.2}$$

which associates to a parameter $\lambda \in \mathbb{R} \cup \{-\infty\}$, the value of the optimal assignment problem in which the weights are given by the matrix $\mathtt{t}P(\lambda)$ where $(\mathtt{t}P(\lambda))_{ij} = (A_0)_{ij} \oplus \lambda(A_1)_{ij} \oplus \cdots \oplus \lambda^d(A_d)_{ij}$. The tropical eigenvalues of $\mathtt{t}P(\lambda)$ are defined as the tropical roots of $f(\lambda)$, i.e. the points at which the maximum attained at least twice.

In this chapter we develop an algorithm, which computes the tropical eigenvalues of a max-plus matrix polynomial in $O(n^4 d)$ time where d is the degree of a given matrix polynomial and n is the dimension of A_is. This algorithm is a generalization of the idea of the algorithm, which is proposed by Burkard and Butkovic [BB03]. Comparing with [BB03], a difficulty is that the leading monomial, which is λ^n, when considering $\operatorname{per}(A \oplus \lambda\mathbb{I})$ is not known anymore. However we shall see in Proposition 5.5 that the leading monomial has a unique algebraic characteristic, which will allow us to compute it in polynomial time. We shall also show that the right and left derivatives of the function $f(\lambda)$ at any point can be calculated by solving an auxiliary optimal assignment problem.

Our motivation for this problem is to use the tropical eigenvalues in the computation of the classical eigenvalues of a matrix polynomial. Indeed, in degenerate cases (when certain matrices are ill conditioned), the scaling of Chapter 4 based

only on the norms of the matrices behaves poorly. However, the tropical eigenvalues (which depend on the modulus of all the entries of the matrices, and not only on their norms), provide better a priori estimates of the classical eigenvalues. This is inspired by a work of Akian, Bapat and Gaubert [ABG05, ABG04] where the tropical eigenvalues were shown to determine the order of magnitude (valuation) of the eigenvalues of a perturbed matrix pencil.

In the next section, we provide some preliminaries, Then, in section 5.3, we present the algorithm to compute all the tropical eigenvalues.

5.2 Preliminaries

Let

$$\mathtt{t}P(\lambda) = A_0 \oplus \lambda \otimes A_1 \oplus \ldots \lambda^d \otimes A_d \quad A_i \in \mathbb{R}_{\max}^{n\times n} \quad \textit{for } i = 1 \ldots d \ ,$$

be a max-plus matrix polynomial. Also, let, $f(\lambda)$ denote the characteristic polynomial of $\mathtt{t}P(\lambda)$. Therefore, $f(\lambda)$ is a convex piecewise linear function. If $f(\lambda) \equiv -\infty$, which happens when there is no permutation with finite value for $\mathtt{t}P(\lambda)$, then $\mathtt{t}P(\lambda)$ does not have any tropical eigenvalue, because $\mathrm{per}(\mathtt{t}P(\lambda))$ is the tropically zero polynomial. In the sequel, we shall assume that $\mathtt{t}P(\lambda)$ has at least one permutation with finite value. This restriction is analogous to considering the regular case in the theory of matrix pencils. For a matrix A, we denote by "maxper(A)" the value

$$\mathrm{maxper}(A) = \max_{\sigma\in S_n} \sum_{i=1}^{n} (A)_{i\sigma(i)} \ ,$$

and by "minper(A)",

$$\mathrm{minper}(A) = \min_{\sigma\in S_n} \sum_{i=1}^{n} (A)_{i\sigma(i)} \ .$$

The Computation of "maxper" or "minper" can be done by solving the optimal assignment problem, which is among the most classical problems in combinatorics. Several algorithms such as Hungarian method have been developed to solve it. We will discuss these algorithms later in Chapter 6.

A principal submatrix of a matrix $A \in \mathbb{R}_{\max}^{n\times n}$ is defined as,

$$\begin{pmatrix} (A)_{i_1i_1} & (A)_{i_1i_2} & \ldots & (A)_{i_1i_k} \\ (A)_{i_2i_1} & (A)_{i_2i_2} & \ldots & (A)_{i_2i_k} \\ \vdots & \vdots & & \vdots \\ (A)_{i_ki_1} & (A)_{i_ki_2} & \ldots & (A)_{i_ki_k} \end{pmatrix} \ ,$$

where $1 \le i_1 < \ldots < i_k \le n$. Let

$$p_A(\lambda) = \delta_0 \oplus (\delta_1 \otimes \lambda) \oplus \ldots \oplus (\delta_{n-1} \otimes \lambda^{n-1}) \oplus \lambda^n \ ,$$

be the tropical characteristic polynomial of matrix A. It is proved by Cuninghame-Green [CG83], that for $k = 0, \ldots, n-1$,

$$\delta_k = \bigoplus_{B \in p_k(A)} \text{maxper}(B) \ , \tag{5.3}$$

where $p_k(A)$ is the set of all $(n-k)\times(n-k)$ principal submatrices of A. In this way, $\delta_0 = \text{maxper}(A)$ and $\delta_{n-1} = \max((A)_{11}, \ldots, (A)_{nn})$. However other coefficients cannot be computed efficiently from the Equation 5.3 since the number of matrices in $P_k(A)$ is $\binom{n}{k}$. Burkard and Butkovic [BB03] called a monomial, $\delta_k \otimes \lambda^k$, of $p_A(\lambda)$ *inessential* if

$$\delta_k \otimes \lambda^k \leq \oplus_{i \neq k} \delta_i \otimes \lambda^i \ ,$$

for every real λ, otherwise it is *essential*. Thus, the inessential terms of $p_A(\lambda)$ can be ignored in computing the function $p_A(\lambda)$. They proposed an algorithm, which computes all the essential terms in $O(n^2(m + n\log n))$ where m is the number of finite entries of A. It is evident that the tropical eigenvalues, i.e. the nondifferentiability points of $p_A(\lambda)$, are the intersections of the essential terms. The rest of this chapter, provides a generalization of the algorithm, developed in [BB03], which computes all the essential terms of $f(\lambda)$.and subsequently the tropical eigenvalues of $\mathtt{t}P(\lambda)$.

5.3 Computing all the tropical eigenvalues

Consider the following max-plus matrix polynomial

$$\mathtt{t}P(\lambda) = A_0 \oplus \lambda \otimes A_1 \oplus \ldots \lambda^d \otimes A_d \quad A_i \in \mathbb{R}_{\max}^{n\times n} \quad \textit{for } i = 1 \ldots d \ .$$

Let,

$$f(\lambda) = \delta_0 \otimes \lambda^{v_0} \oplus \ldots \oplus \delta_t \otimes \lambda^{v_t} \ , \tag{5.4}$$

be the tropical characteristic polynomial of $\mathtt{t}P(\lambda)$ by considering only the essential terms where $0 \leq v_0 < \ldots < v_t \leq nd$. We refer to v_0 and v_t as the *valuation*, denoted by "val", and the *degree*, denoted by "deg", of $f(\lambda)$ respectively. Due to the max-plus "fundamental theorem of algebra", $f(\lambda)$ has $v_t - v_0$ nonzero tropical roots and v_0 zero tropical roots.

5.3.1 Computing the first and the last essential terms

We first explain the computation of the first and the last essential terms that is $\delta_0 \otimes \lambda^{v_0}$ and $\delta_t \otimes \lambda^{v_t}$. Let M be the matrix defined as follows,

$$(M)_{ij} = \begin{cases} \max_{(A_k)_{ij} \neq \mathbb{0}} k & \text{if } \exists (A_k)_{ij} \neq \mathbb{0} \\ \mathbb{0} & \text{otherwise} \end{cases} \tag{5.5}$$

Proposition 5.3.1. *We have* $\deg(f(\lambda)) = \text{maxper}(M)$.

Proof. Due to the definition, $(M)_{ij}$ denotes the degree of the tropical polynomial

$$(\mathtt{t}P(\lambda))_{ij} = (A_{l_0})_{ij} \oplus (\lambda \otimes (A_{l_1})_{ij}) \oplus \ldots \oplus (\lambda^{l_k} \otimes (A_{l_k})_{ij}) \ , \tag{5.6}$$

where $0 \leq l_0 < \ldots < l_k \leq d$ and $(A_{l_0})_{ij}, \ldots, (A_{l_k})_{ij} \neq \mathbb{0}$. So, for any permutation, σ, $\sum_{i=1}^{n}(M)_{i\sigma(i)}$ computes the degree of a tropical polynomial, $\bigotimes_{i=1}^{n}(\mathtt{t}P(\lambda))_{i\sigma(i)}$, and subsequently, $\text{maxper}(M)$ computes the degree of $f(\lambda)$. □

The same idea can be used to compute the valuation of $f(\lambda)$. More precisely, let M' be the matrix defined as follows,

$$(M')_{ij} = \begin{cases} \min_{(A_k)_{ij} \neq \mathbb{0}} k & \text{if } \exists (A_k)_{ij} \neq \mathbb{0} \\ +\infty & \text{otherwise} \end{cases} \tag{5.7}$$

so that, $(M')_{ij}$ denotes the valuation of the max-plus polynomial in Equation 5.6.

Proposition 5.3.2. *We have* $\text{val}(f(\lambda)) = \text{minper}(M')$.

Proof. The proof is the same as that of proposition 5.3.1. □

To compute the coefficient of the monomial with maximum degree, i.e. δ_t in Equation 5.4, we define a saturated graph by taking all the arcs belonging to the optimal permutation of M as follows:

$$G_M = \{(i,j) | (i,j) \text{ belongs to a maximum permutation in } M\} \ . \tag{5.8}$$

Let $\hat{A}_M$ be the matrix defined as follows,

$$(\hat{A}_M)_{ij} = \begin{cases} (A_k)_{ij} & \text{if } (i,j) \text{ belongs to a maximum permutation in } M \\ & \qquad\qquad \& \quad (M)_{ij} = k \\ -\infty & \text{otherwise} \end{cases} \tag{5.9}$$

Although there is in general an exponential number of optimal permutations, we shall see that we can compute G_M in polynomial time.

Proposition 5.3.3. *The coefficient of the monomial with maximum degree, i.e.* δ_t *in Equation 5.4, can be computed as follows,*

$$\delta_t = \text{maxper}(\hat{A}_M) \ .$$

Proof. For any permutation σ and any tropical monomial,

$$\bigotimes_{k=1}^{n}((A_{i_k})_{k\sigma(k)} \otimes \lambda^{i_k}) \ , \tag{5.10}$$

with degree $\deg(f(\lambda))$, since all the arcs $(k, \sigma(k)) \in G_M$, we have, $(\hat{A}_M)_{k\sigma(k)} = (A_{i_k})_{k\sigma(k)}$. So, this tropical monomial represents the weight of a non-$\mathbb{0}$ permutation in the matrix $\hat{A}_M$. Conversely, any non-$\mathbb{0}$ permutation, σ, of $\hat{A}_M$, presents a tropical monomial with degree $\deg(f(\lambda))$, such as the one indicated in (5.10) where $(A_{i_k})_{k\sigma(k)} = (\hat{A}_M)_{k\sigma(k)}$ and $(M)_{k\sigma(k)} = i_k$. Since δ_t is the maximal coefficient of all monomials with degree $\deg(f(\lambda))$, the statement is achieved. □

The same method can be used to compute the coefficient of the monomial with the smallest degree by defining

$$G_{M'} = \{(i,j) | (i,j) \text{ belongs to a minimum permutation in } M'\} \ , \tag{5.11}$$

and

$$(\hat{A}_{M'})_{ij} = \begin{cases} (A_k)_{ij} & \text{if } (i,j) \text{ belongs to a minimum permutation in } M' \\ & \qquad\qquad \& \quad (M')_{ij} = k \\ -\infty & \text{otherwise} \end{cases} \tag{5.12}$$

Proposition 5.3.4. *The coefficient of the monomial with minimum degree, i.e. δ_0 in Equation 5.4, can be computed as follows,*

$$\delta_0 = \text{maxper}(\hat{A}_{M'}) \ .$$

Proof. The proof is the same as that of Proposition 5.3.3 □

We now show that we can compute the digraph G_M consisting of all the arcs belonging to the optimal permutations of the matrix M by a linear time post-processing, after calling an optimal assignment solver.

Consider the primal linear programming formulation of the optimal assignment for the matrix M;

$$\begin{array}{lll} \max & \sum_{i=1}^n \sum_{j=1}^n (M)_{ij}(X)_{ij} & \\ s.t. & \sum_{j=1}^n (X)_{ij} = 1 & (i = 1, \ldots, n) \\ & \sum_{i=1}^n (X)_{ij} = 1 & (j = 1, \ldots, n) \\ & (X)_{ij} \geq 0 & (i, j = 1, \ldots, n) \end{array}$$

Then the dual problem can be written as

$$\begin{array}{ll} \min & \sum_{i=1}^n (U)_i + \sum_{j=1}^n (V)_j \\ s.t. & (U)_i + (V)_j \geq (M)_{ij} \end{array}$$

If X is a feasible solution of the primal problem, and if (U, V) is a feasible solution of the dual problem, then, the *complementary slackness* condition shows that X and (U, V) are both optimal if and only if

$$(X)_{ij}((M)_{ij} - (U)_i - (V)_j) = 0 \quad (i, j = 1, 2, \ldots, n) \ .$$

Fix now an arbitrary optimal dual solution (U^*, V^*), and, following [ABG05, ABG04] define the *saturation digraph* $\mathrm{Sat}(M, U^*, V^*)$ to be the graph with set of nodes $1, \ldots, n$ and an arc from $i \to j$ whenever the $(M)_{ij} - (U^*)_i - (V^*)_j = 0$. It follows that the set of optimal solutions of the primal problem is precisely the set of bistochastic matrices X such that the digraph of X is included in the saturation digraph. In particular, a permutation matrix is optimal if and only if its digraph is included in the same saturation digraph.

Assume now, without loss of generality (we may always permute the rows or columns of M), that the identity is an optimal permutation. Then, one readily checks that an arc (i, j) belongs to an optimal permutation if and only if either $i = j$ or (i, j) belongs to a circuit in $\mathrm{Sat}(M, U^*, V^*)$. Indeed, by the complementary slackness condition, any arc of an optimal permutation belongs to a circuit of $\mathrm{Sat}(M, U^*, V^*)$, and conversely, if we find a circuit in the latter digraph, we can always complete it by loops to make a permutation, which is optimal.

The previous discussion can be summarized as follows.

Proposition 5.3.5. *Assume that the optimal assignment for the matrix M is feasible (meaning that there is at least one permutation of finite weight), and let (U^*, V^*) denote any optimal solution of the dual problem. Then, the graph G_M (the arcs of which belong to optimal permutations) is included in* $\mathrm{Sat}(M, U^*, V^*)$. *Moreover, if the identity permutation is optimal, G_M consists of those arcs of the digraph* $\mathrm{Sat}(M, U^*, V^*)$, *which belong to circuits of this digraph.* □

Most optimal assignment algorithms, and in particular the Hungarian algorithm, yield as an output both an optimal permutation and a pair of optimal dual variables (U^*, V^*). The digraph $\mathrm{Sat}(M, U^*, V^*)$ can be computed in linear time from this output. Then, G_M can be computed in an additional linear time by computing the strongly connected components of $\mathrm{Sat}(M, U^*, V^*)$. Therefore, computing G_M only requires a linear time post-processing.

Remark 10. In all the applications of the present chapter, the digraph G_M could be replaced by the digraph $\mathrm{Sat}(M, U^*, V^*)$ without changing the final results. For instance, if we replace the condition that (i, j) belongs to an optimal permutation in the definition of $\hat{A}_M$ above by the condition that $(i, j) \in \mathrm{Sat}(M, U^*, V^*)$, then, we modify only those entries of the matrix $\hat{A}_M$, which do not belong to any permutation in the digraph of this matrix. These entries do not play any role in the computations (which only involve the weights of permutations of the matrix $\hat{A}_M$). Hence, whereas the introduction of the "intrinsic" digraph G_M is useful for theoretical purposes, for algorithmic purposes, one may be content with $\mathrm{Sat}(M, U^*, V^*)$.

Example 5.3.1. As a simple illustration, consider the matrix

$$M = \begin{pmatrix} 0 & -1 \\ -1 & 0 \end{pmatrix} .$$

The identity is the only optimal permutation, so G_M consists of the two loops $1 \to 1$ and $2 \to 2$. Observe that $U^* = (0,0)$ and $V^* = (0,0)$ are optimal dual variables, and that the corresponding saturation digraph, the arcs of which correspond to the zero entries of M, is precisely G_M. However, the dual variables $U^* = (0,1)$ and $V^* = (0,-1)$ yield the following scaled matrix with entries $(M)_{ij} - (U)_i - (V)_j$

$$\begin{pmatrix} 0 & 0 \\ -2 & 0 \end{pmatrix} .$$

Then, the arc $1 \to 2$, corresponding to the top-right zero entry of this matrix, is added to the saturation digraph. Similarly, the choice $U^* = (0,-1)$ and $V^* = (0,1)$ replaces the arc $1 \to 2$ by the arc $2 \to 1$. This illustrates the fact that for any choice of the optimal dual variable (U^*, V^*), the digraph $\mathrm{Sat}(M, U^*, V^*)$ contains the optimal permutations, plus artificial (dummy) arcs which do depend on (U^*, V^*) but do not belong to any optimal permutation.

Example 5.3.2. Consider the following quadratic max-plus matrix polynomial

$$\mathtt{t}P(\lambda) = \begin{pmatrix} \mathbb{0} & \mathbb{0} \\ 8 & 15 \end{pmatrix} \oplus \lambda \otimes \begin{pmatrix} 5 & 10 \\ 3 & \mathbb{0} \end{pmatrix} \oplus \lambda^2 \otimes \begin{pmatrix} 7 & 9 \\ 1 & 4 \end{pmatrix} ,$$

then

$$M = \begin{pmatrix} 2 & 2 \\ 2 & 2 \end{pmatrix}, \quad M' = \begin{pmatrix} 1 & 1 \\ 0 & 0 \end{pmatrix} ,$$

so, $\deg(f(\lambda)) = \mathrm{maxper}(M) = 4$ and $\mathrm{val}(f(\lambda)) = \mathrm{minper}(M') = 1$. Also,

$$G_M = \{(1,1),(1,2),(2,1),(2,2)\} ,$$
$$G_{M'} = \{(1,1),(1,2),(2,1),(2,2)\} ,$$

$$\hat{A}_M = \begin{pmatrix} 7 & 9 \\ 1 & 4 \end{pmatrix}, \quad \hat{A}_{M'} = \begin{pmatrix} 5 & 10 \\ 8 & 15 \end{pmatrix} ,$$

and subsequently $\delta_4 = \mathrm{maxper}(\hat{A}_M) = 11$ and $\delta_0 = \mathrm{maxper}(\hat{A}_{M'}) = 20$.

5.3.2 Computing all the other essential terms

Let $f_{v_r}(\lambda) = \delta_r \otimes \lambda^{v_r}$ and $f_{v_s}(\lambda) = \delta_s \otimes \lambda^{v_s}$ be two terms of the function $f(\lambda)$ where $v_r < v_s$. The next proposition provides a sufficient condition to check whether the intersection of these two linear segments is a tropical root.

Proposition 5.3.6 (Proposition 3.2 in [BB03])**.** *Let $\mathtt{t}P(\lambda)$ be a matrix polynomial and $f_{v_r}(\lambda) = \delta_r \otimes \lambda^{v_r}$ and $f_{v_s}(\lambda) = \delta_s \otimes \lambda^{v_s}$ be two terms of its tropical characteristic polynomial, $f(\lambda)$ where $v_0 \le v_r < v_s \le v_t$. Also, let $\bar{\lambda}$ be the intersection of these two linear segments such that $f_{v_r}(\bar{\lambda}) = f_{v_s}(\bar{\lambda})$. Assume that $f(\bar{\lambda}) = f_{v_r}(\bar{\lambda})$, then, there is not any essential term such as $\delta_k \otimes \lambda^{v_k}$ where $v_r < v_k < v_s$ which follows that $\bar{\lambda}$ is a tropical eigenvalue of $\mathtt{t}P(\lambda)$.*

Proof. Since $f_{v_r}(\bar{\lambda}) = f(\bar{\lambda})$, we have $\delta_r + v_r\bar{\lambda} \geq \delta_k + v_k\bar{\lambda}$ for every $0 < k < t$. It follows that $f(\lambda) \geq f_{v_r}(\lambda) = \delta_r + v_r\lambda \geq \delta_k + v_k\lambda = f_{v_k}(\lambda)$ for every $\lambda < \bar{\lambda}$ and $v_k > v_r$. Similarly, $f(\lambda) \geq f_{v_s}(\lambda) \geq f_{v_k}(\lambda)$ for every $\lambda > \bar{\lambda}$ and $v_k < v_s$; thus $f(\lambda) \geq f_{v_k}(\lambda)$ for all λ and $v_r < v_k < v_s$. □

Remark 11. For any value $\bar{\lambda} \in \mathbb{R}$, the matrix, $\mathtt{t}P(\bar{\lambda})$ can be computed in $O(md)$ time where m is the number of finite entries and d is the degree of $\mathtt{t}P(\lambda)$. Then by applying an optimal assignment algorithm on $\mathtt{t}P(\bar{\lambda})$, the value of $f(\bar{\lambda})$ can be computed.

Let $f'_+(\bar{\lambda})$ and $f'_-(\bar{\lambda})$ denote right and left directional derivatives of the function, f, at point $\bar{\lambda} \in \mathbb{R}$. Thus, if $\bar{\lambda}$ is a tropical eigenvalue, i.e. a non-differentiability point, then $f'_+ \neq f'_-$ otherwise $f'_+ = f'_-$ and $f(\bar{\lambda}) = f_k(\bar{\lambda}) = k\bar{\lambda} + \delta_k$ in a neighborhood of $\bar{\lambda}$ for some k between v_0 and v_t.

Theorem 5.3.7 (Computing $f'_+(\bar{\lambda}), f'_-(\bar{\lambda})$)**.** *Let $\bar{\lambda}$, be a point in $\mathbb{R}$. Also let $G_{\bar{\lambda}}$ be a graph defined as follows,*

$$G_{\bar{\lambda}} = \{(i,j) | (i,j) \text{ belongs to a maximum permutation in } \mathtt{t}P(\bar{\lambda})\} \ , \tag{5.13}$$

and $M^{\bar{\lambda}}_{\max}, M^{\bar{\lambda}}_{\min}$ be the matrices defined as follows,

$$(M^{\bar{\lambda}}_{\max})_{ij} = \begin{cases} \max_{(A_k)_{ij} \neq \mathbb{0}} k & \text{if } (i,j) \in G_{\bar{\lambda}} \quad \& \quad (A_{\bar{\lambda}})_{ij} = (A_k)_{ij} + k\bar{\lambda} \\ \mathbb{0} & \text{otherwise} \end{cases} \tag{5.14}$$

$$(M^{\bar{\lambda}}_{\min})_{ij} = \begin{cases} \min_{(A_k)_{ij} \neq \mathbb{0}} k & \text{if } (i,j) \in G_{\bar{\lambda}} \quad \& \quad (A_{\bar{\lambda}})_{ij} = (A_k)_{ij} + k\bar{\lambda} \\ +\infty & \text{otherwise} \end{cases} \tag{5.15}$$

then,

$$f'_+(\bar{\lambda}) = \mathrm{maxper}(M^{\bar{\lambda}}_{\max}) \ , \tag{5.16}$$

$$f'_-(\bar{\lambda}) = \mathrm{minper}(M^{\bar{\lambda}}_{\min}) \ . \tag{5.17}$$

In particular, if $\bar{\lambda}$ is a tropical eigenvalue then $f'_+(\bar{\lambda})$ and $f'_-(\bar{\lambda})$ represents the slopes of the right and left segments of the graph of f at point $(\bar{\lambda}, f(\bar{\lambda}))$ respectively. If $\bar{\lambda}$ is not a tropical eigenvalue, then, the slope of the segment passing from $\bar{\lambda}$, coincides with $f'_+(\bar{\lambda}) = f'_-(\bar{\lambda})$.

Proof. Due to the definition, $(M^{\bar{\lambda}}_{\max})_{ij}$ coincides with the right-derivative at point $\bar{\lambda}$ of the tropical polynomial,

$$\begin{aligned} (\mathtt{t}P(\lambda))_{ij} &= (\lambda^{l_1} \otimes (A_{l_1})_{ij}) \oplus \ldots \oplus (\lambda^{l_k} \otimes (A_{l_k})_{ij}) \\ &= \max(\lambda l_1 + (A_{l_1})_{ij}, \ldots, \lambda l_k + (A_{l_k})_{ij}) \ . \end{aligned}$$

Indeed, the rule of "differentiation of a supremum" (see Exercise 8.31 of [RW98]) shows that the right-derivative at a given point of a finite supremum of functions

coincides with the supremum of the right-derivatives of those functions, which attain the former supremum at that point. This gives precisely

$$\big((\mathtt{t}P(\bar{\lambda}))_{ij}\big)'_+ = (M^{\bar{\lambda}}_{\max})_{ij} \ .$$

Consider now the weight of the permutation σ,

$$f^{(\sigma)}(\lambda) = \bigotimes_{i=1}^{n} (\mathtt{t}P(\lambda))_{i\sigma(i)} = \sum_{i=1}^{n} (\mathtt{t}P(\lambda))_{i\sigma(i)} \ . \tag{5.18}$$

Since taking the right-derivative commutes with the addition, it follows from the preceding discussion that $(f^{(\sigma)})'_+(\bar{\lambda})$, i.e. the right-derivative of the map $f^{(\sigma)}$ at point $\bar{\lambda}$, satisfies

$$(f^{(\sigma)})'_+(\bar{\lambda}) = \sum_{i=1}^{n} (M^{\bar{\lambda}}_{\max})_{i\sigma(i)} \ .$$

Also,

$$f(\lambda) = \sup_{\sigma} f^{(\sigma)}(\lambda) \ , \tag{5.19}$$

where the sum is taken over all permutations σ having a finite weight in the sense of (5.18). Applying again the rule of "differentiation of a supremum", we get that the right-derivative of f is given by

$$f'_+(\bar{\lambda}) = \sup_{\sigma} (f^{(\sigma)})'_+(\bar{\lambda}) \ ,$$

where the supremum is now restricted to those permutations σ that attain the maximum in (5.19) at point $\bar{\lambda}$. This shows that

$$f'_+(\bar{\lambda}) = \mathrm{maxper}(M^{\bar{\lambda}}_{\max}) \ .$$

The characterization of the left-derivative relies on a dual argument.

The final assertion of the theorem follows readily since f is a piecewise affine map the nondifferentiability points of which are precisely the tropical eigenvalues. □

Thus, $f(\lambda) = \lambda f'_{\pm}(\lambda) + c^{\bar{\lambda}}_{\pm}$, holds for all $\lambda > \bar{\lambda}$ (in the "+" case) and $\lambda < \bar{\lambda}$ (in the "−" case) close enough to $\bar{\lambda}$, where $c^{\bar{\lambda}}_{\pm}$ is a constant, which simply can be computed by

$$c^{\bar{\lambda}}_{\pm} = f(\bar{\lambda}) - f'_{\pm}(\bar{\lambda})\bar{\lambda} \ .$$

Remark 12. If for all $(i,j) \in G_{\bar{\lambda}}$, there is only one slope k such that $(A_{\bar{\lambda}})_{ij} = (A_k)_{ij} + k\bar{\lambda}$, then, f is differentiable at point $\bar{\lambda}$: $f'_-(\bar{\lambda}) = f'_+(\bar{\lambda})$.

A sketch of the algorithm, which computes all the tropical roots is given as follows.

1: **function** trop_eigenvalues(m_l, c_l, m_r, c_r)

2: **assume:** The graph of f contains two segments of the lines $\lambda \mapsto m_l\lambda + c_l$ and $\lambda \mapsto m_r\lambda + c_r$, where $m_l < m_r$.
3: **output:** The tropical eigenvalues λ of f such that $f'_+(\lambda) \leq m_r$ and $f'_-(\lambda) \geq m_l$.
4: Find the point $(\bar{\lambda}, \bar{y})$ of intersection of the lines: $\begin{cases} y = m_l\lambda + c_l \\ y = m_r\lambda + c_r \end{cases}$
5: **if** $f(\bar{\lambda}) = \bar{y}$ **then**
6: **return** ($\bar{\lambda}$ is a tropical eigenvalue with multiplicity $m_r - m_l$)
7: **else**
8: Compute the right and left derivatives $f'_\pm(\bar{\lambda})$, and the constant coefficients $c^{\bar{\lambda}}_\pm$ such that $f'_\pm(\bar{\lambda})\bar{\lambda} + c^{\bar{\lambda}}_\pm = \bar{y}$
9: **if** $f'_-(\bar{\lambda}) < f'_+(\bar{\lambda})$ **then**
10: **output:** ($\bar{\lambda}$ is a tropical eigenvalue with multiplicity $f'_+(\bar{\lambda}) - f'_-(\bar{\lambda})$)
11: **end if**
12: **call** trop_eigenvalues$(m_l, c_l, f'_-(\bar{\lambda}), c^{\bar{\lambda}}_-)$
13: **call** trop_eigenvalues$(f'_+(\bar{\lambda}), c^{\bar{\lambda}}_+, m_r, c_r)$
14: **end if**
15: **end function**

We will present a detailed non recursive instantiation of the Algorithm (Algorithm 5.1). Also a Scilab implementation of the algorithm can be found in Appendix C.

We start the algorithm by calling trop_eigenvalues$(v_0, \delta_0, v_t, \delta_t)$ which computes the intersection of the following two lines:

$$\bar{\lambda} = \begin{cases} \delta_0 + v_0\lambda \\ \delta_t + v_t\lambda \end{cases}$$

If $f(\bar{\lambda}) = f_{v_0}(\bar{\lambda})$, then, due to the Proposition **??**, we are done and $\mathtt{t}P(\lambda)$ has one nonzero tropical eigenvalue, $\bar{\lambda}$, with multiplicity $v_t - v_0$. Otherwise, we compute $f'_\pm(\bar{\lambda})$ to decide whether $\bar{\lambda}$ is a tropical eigenvalue. Then, we compute the left and right tangent lines at point $\bar{\lambda}$, $\lambda \mapsto f'_\pm(\bar{\lambda})\bar{\lambda} + c^{\bar{\lambda}}_\pm$, and call recursively the function to compute the tropical roots λ such that $f'_+(\lambda) \leq f'_-(\bar{\lambda})$ and $f'_-(\lambda) \geq f'_+(\bar{\lambda})$. By repeating this iteration, we compute all the tropical eigenvalues.

To summarize, for every point such as $\bar{\lambda}$, we find a tropical eigenvalue or we compute two new points. Since all the slopes are the integers in the interval $[v_0, v_t]$, where $0 \leq v_0, v_t \leq nd$, the total number of points will be at most $O(nd)$. Also for every point, we make three calls to an optimal assignment algorithm. Since, optimal assignment can be computed in $O(n^3)$ for a dense matrix and in $O(n(m + n\log n)$ for a sparse matrix, we arrive at the following theorem

Theorem 5.3.8. *Let,*

$$\mathtt{t}P(\lambda) = A_0 \oplus \lambda \otimes A_1 \oplus \ldots \oplus \lambda^d \otimes A_d \quad A_i \in \mathbb{R}^{n\times n}_{\max} \quad \textit{for } i = 1 \ldots d \ ,$$

be a max-plus matrix polynomial of degree d. Then, all the tropical eigenvalues of $\mathtt{t}P(\lambda)$ can be computed in $O(n^4 d)$. If the matrix M, which is defined by equation 5.5, is sparse with m nonzero entries then, all the tropical eigenvalues can be computed in $O(n^2 d(m + n\log n))$ time.

Remark 13. Of course, a dual result holds for min-plus matrix polynomial.

Example 5.3.3. Consider the following quadratic max-plus matrix polynomial

$$\mathtt{t}P(\lambda) = \begin{pmatrix} 1 & 6 & \mathbb{0} \\ 4 & 3 & 2 \\ 8 & \mathbb{0} & 5 \end{pmatrix} \oplus \lambda \otimes \begin{pmatrix} 2 & 8 & 10 \\ 5 & 6 & 7 \\ \mathbb{0} & \mathbb{0} & \mathbb{0} \end{pmatrix} \oplus \lambda^2 \otimes \begin{pmatrix} 6 & \mathbb{0} & \mathbb{0} \\ 3 & 4 & 8 \\ 12 & 9 & \mathbb{0} \end{pmatrix} \ .$$

The matrices M and M' and, the graphs $G_M, G_{M'}$ can be computed as follows

$$M = \begin{pmatrix} 2 & 1 & 1 \\ 2 & 2 & 2 \\ 2 & 2 & 0 \end{pmatrix}, \quad M' = \begin{pmatrix} 0 & 0 & +\infty \\ 0 & 0 & 0 \\ 0 & \mathbb{0} & 0 \end{pmatrix} \ ,$$

$$G_M = \{(1,1),(3,2),(2,3)\} \ ,$$
$$G_{M'} = \{(1,1),(2,2),(3,3),(1,2),(2,1),(3,3),(2,3),(3,1)\} \ .$$

Also,

$$\hat{A}_M = \begin{pmatrix} 6 & \mathbb{0} & \mathbb{0} \\ \mathbb{0} & \mathbb{0} & 8 \\ \mathbb{0} & 9 & \mathbb{0} \end{pmatrix}, \quad \hat{A}_{M'} = \begin{pmatrix} 1 & 6 & \mathbb{0} \\ 4 & 3 & 2 \\ 8 & \mathbb{0} & 5 \end{pmatrix} \ ,$$

which results: $v_0 = 0, \delta_0 = 16$ and $v_t = 6, \delta_t = 23$. The intersection of the lines $y = 16$ and $y = 6\lambda + 23$ is $\lambda_1 = \frac{16-23}{6} = -\frac{7}{6}$. So,

$$\mathtt{t}P(-\frac{7}{6}) = \begin{pmatrix} \frac{22}{6} & \frac{41}{6} & \mathbf{\frac{53}{6}} \\ 4 & \mathbf{\frac{29}{6}} & \frac{35}{6} \\ \mathbf{\frac{58}{6}} & \frac{40}{6} & 5 \end{pmatrix} \ ,$$

where the bold entries belong to the optimal permutation. $f(\lambda_1) = \frac{70}{3} \neq 16$, yields that λ_1 is not a tropical eigenvalue. So we continue by computing the graph $G_{-\frac{7}{6}} = \{(1,3),(2,2),(3,1)\}$ and the following matrices:

$$M_{\max}^{-\frac{7}{6}} = \begin{pmatrix} \mathbb{0} & \mathbb{0} & 1 \\ \mathbb{0} & 1 & \mathbb{0} \\ 2 & \mathbb{0} & \mathbb{0} \end{pmatrix}, \quad M_{\min}^{-\frac{7}{6}} = \begin{pmatrix} +\infty & +\infty & 1 \\ +\infty & 1 & +\infty \\ 2 & +\infty & +\infty \end{pmatrix} \ .$$

Therefor, $f'_+(-\frac{7}{6}) = f'_-(-\frac{7}{6}) = 4$, $c_+^{-\frac{7}{6}} = c_-^{-\frac{7}{6}} = 28$. By computing the intersections, two new points will be added to the list which yields

$$L = \{(-3,0,16,4,28),(\frac{5}{2},4,28,6,23)\}$$

Next, we choose $(-3, 0, 16, 4, 28)$ from L. Then,

$$\mathtt{t}P(-3) = \begin{pmatrix} 1 & \mathbf{6} & \mathbf{7} \\ 4 & \mathbf{3} & \mathbf{4} \\ \mathbf{8} & 3 & 5 \end{pmatrix} ,$$

and $f(-3) = 18 \neq 16$. Again -3 is not a tropical eigenvalue. So we continue by computing $G_{-3} = \{(1,2),(1,3),(2,2),(2,3),(3,1)\}$ and

$$M^{-3}_{\max} = \begin{pmatrix} \mathbb{0} & 0 & 1 \\ \mathbb{0} & 0 & 1 \\ 0 & \mathbb{0} & \mathbb{0} \end{pmatrix}, \quad M^{-3}_{\min} = \begin{pmatrix} +\infty & 0 & 1 \\ +\infty & 0 & 1 \\ 0 & +\infty & +\infty \end{pmatrix} .$$

Therefore, $f'_+(-3) = f'_-(-3) = 1$, $c^{-3}_+ = c^{-3}_- = 21$. By computing the intersections, two new points are added to L, which yields,

$$L = \{(\frac{5}{2}, 4, 28, 6, 23), (-5, 0, 16, 1, 21), (-\frac{7}{3}, 1, 21, 4, 28)\} .$$

Next, we choose $(-5, 0, 16, 1, 21)$. Since $v^r_{-5} - v^l_{-5} = 1$ then -5 is a tropical eigenvalue with multiplicity 1.

Since L is not empty we continue by selecting $(\frac{5}{2}, 4, 28, 6, 23)$ from L and computing

$$\mathtt{t}P(\frac{5}{2}) = \begin{pmatrix} 11 & \mathbf{\frac{21}{2}} & \frac{25}{2} \\ 8 & 9 & \mathbf{13} \\ \mathbf{17} & 14 & 5 \end{pmatrix} ,$$

which results $f(\frac{5}{2}) = \frac{81}{2}$. Again $\frac{5}{2}$ is not a tropical eigenvalue and we compute $G_{\frac{5}{2}} = \{(1,2),(2,3),(3,2)\}$ and

$$M^{\frac{5}{2}}_{\max} = \begin{pmatrix} \mathbb{0} & 1 & \mathbb{0} \\ \mathbb{0} & \mathbb{0} & 2 \\ 2 & \mathbb{0} & \mathbb{0} \end{pmatrix}, \quad M^{\frac{5}{2}}_{\min} = \begin{pmatrix} +\infty & 1 & +\infty \\ +\infty & +\infty & 2 \\ 2 & +\infty & +\infty \end{pmatrix} ,$$

which yields $f'_+(\frac{5}{2}) = f'_-(\frac{5}{2}) = 5$, $c^{\frac{5}{2}}_+ = c^{-3}_- = 28$. Thus, two new intersection points will be added to L which results

$$L = \{(-\frac{7}{3}, 1, 21, 4, 28), (5, 4, 28, 5, 23)(0, 5, 23, 6, 23)\} .$$

Next, we choose $(-\frac{7}{3}, 1, 21, 4, 28)$. So,

$$\mathtt{t}P(-\frac{7}{3}) = \begin{pmatrix} \frac{4}{3} & 6 & \mathbf{\frac{23}{3}} \\ 4 & \mathbf{\frac{11}{3}} & \frac{14}{3} \\ \mathbf{8} & \frac{13}{3} & 5 \end{pmatrix} ,$$

and subsequently $f(-\frac{7}{3}) = \frac{58}{3}$. Since $-\frac{7}{3}$ is not a tropical eigenvalue we compute $G_{-\frac{7}{3}} = \{(1,3),(2,2),(3,1)\}$ and

$$M_{\max}^{-\frac{7}{3}} = \begin{pmatrix} \mathbb{0} & \mathbb{0} & 1 \\ \mathbb{0} & 1 & \mathbb{0} \\ 0 & \mathbb{0} & \mathbb{0} \end{pmatrix}, \quad M_{\min}^{-\frac{7}{3}} = \begin{pmatrix} +\infty & +\infty & 1 \\ +\infty & 1 & +\infty \\ 0 & +\infty & +\infty \end{pmatrix} ,$$

which results $f'_+(-\frac{7}{3}) = f'_-(-\frac{7}{3}) = 2$, $c_+^{\frac{5}{2}} = c_-^{-3} = 24$. Therefore, two new points are added to L,

$$L = \{(-2,2,24,4,28),(-3,1,21,2,24),(5,4,28,5,23)(0,5,23,6,23)\} .$$

This time we choose $(-2,2,24,4,28)$, which yields

$$\mathrm{t}P(-2) = \begin{pmatrix} 2 & 6 & \mathbf{8} \\ 4 & \mathbf{4} & 5 \\ \mathbf{8} & 5 & 5 \end{pmatrix} .$$

Since $f(-2) = 20 = 24 + 2 \times (-2)$, -2 is a tropical eigenvalue with multiplicity $4 - 2 = 2$. The algorithm continues by fetching the remaining entries of the list

$$L = \{(-3,1,21,2,24),(5,4,28,5,23)(0,5,23,6,23)\} ,$$

which yields to recognize three tropical eigenvalues, since $v^r_{-3} - v^l_{-3} = 1$, $v^r_5 - v^l_5 = 1$ and $v^r_0 - v^l_0 = 1$. Overall, the tropical eigenvalues are -5 with multiplicity 1, -3 with multiplicity 1, -2 with multiplicity 2, 0 with multiplicity 1 and 5 with multiplicity 1 and $f(\lambda)$ is

$$f(\lambda) = \begin{cases} 16 & \lambda \leq -5 \\ \lambda + 21 & -5 < \lambda \leq -3 \\ 2\lambda + 24 & -3 < \lambda \leq -2 \\ 4\lambda + 28 & -2 < \lambda \leq 0 \\ 5\lambda + 28 & 0 < \lambda \leq 5 \\ 6\lambda + 23 & \lambda > 5 \end{cases}$$

Figure 5.1 demonstrates the diagram of $f(\lambda)$.

Algorithm 5.1 Compute the tropical eigenvalues of $\mathtt{t}P(\lambda)$.

Input: $\mathtt{t}P(\lambda) = A_0 \oplus \lambda \otimes A_1 \oplus \ldots \lambda^d \otimes A_d \quad A_i \in \mathbb{R}_{\max}^{n\times n} \quad$ *for* $i = 1 \ldots d$
Output: tropical eigenvalues of $\mathtt{t}P(\lambda)$

• Compute the matrices M, M' defined in Equations 5.5 and 5.7, the graphs $G_M, G_{M'}$, which are defined in the equations 5.8 and 5.11 and the matrices $\hat{A}_M, \hat{A}_{M'}$ defined in the equations 5.9 and 5.12 and subsequently compute the valuation, $v_0 = \text{minper}(M')$, the degree, $v_t = \text{maxper}(M)$ of $f(\lambda)$ and the constants $\delta_0 = \text{maxper}\,\hat{A}_{M'}, \delta_t = \text{maxper}\,\hat{A}_M$.

• Compute the intersection of the two lines $\delta_0 + v_0\lambda$ and $\delta_t + v_t\lambda$ as $\lambda_1 = \frac{\delta_0-\delta_t}{v_t-v_0}$ and set $L = \{(\lambda_1, v_0, \delta_0, v_t, \delta_t,)\}$
while L is not empty **do**
  • Choose any arbitrary element $(\lambda_i, v_i^l, c_i^l, v_i^r, c_i^r)$ from L and remove it from the list.
  if $v_i^r = v_i^l + 1$ **then**
    • Output: λ_i as a tropical eigenvalues with multiplicity one
  else
    Compute $f(\lambda_i)$
    if $f(\lambda_i) = c_i^l + v_i^l\lambda_i$ **then**
      • Output: λ_i as a tropical eigenvalue with multiplicity $v_i^r - v_i^l$
    else
      • Compute the graph G_{λ_i} defined in Equation 5.13, the matrix $M_{\max}^{\lambda_i}$ defined in Equation 5.14 and $M_{\min}^{\lambda_i}$ defined in Equation 5.15. Then, compute $f'_+(\lambda_i) = \text{maxper}(M_{\max}^{\lambda_i})$, $f'_-(\lambda_i) = \text{minper}(M_{\min}^{\lambda_i})$, $c_\pm^{\lambda_i} = f(\lambda_i) - f'_\pm(\lambda_i) * \lambda_i$
      if $f'_-(\lambda_i) < f'_+(\lambda_i)$ **then**
        • Output: λ_i as a tropical eigenvalue with multiplicity $f'_+(\lambda_i) - f'_-(\lambda_i)$
      end if
      • Add two points to the list, L as follows
      $(\frac{c_i^l - c_-^{\lambda_i}}{f'_-(\lambda_i) - v_i^l}, v_i^l, c_i^l, f'_-(\lambda_i), c_-^{\lambda_i})$,
      $(\frac{c_i^r - c_+^{\lambda_i}}{f'_+(\lambda_i) - v_i^r}, f'_+(\lambda_i), c_+^{\lambda_i}, v_i^r, c_i^r)$
    end if
  end if
end while

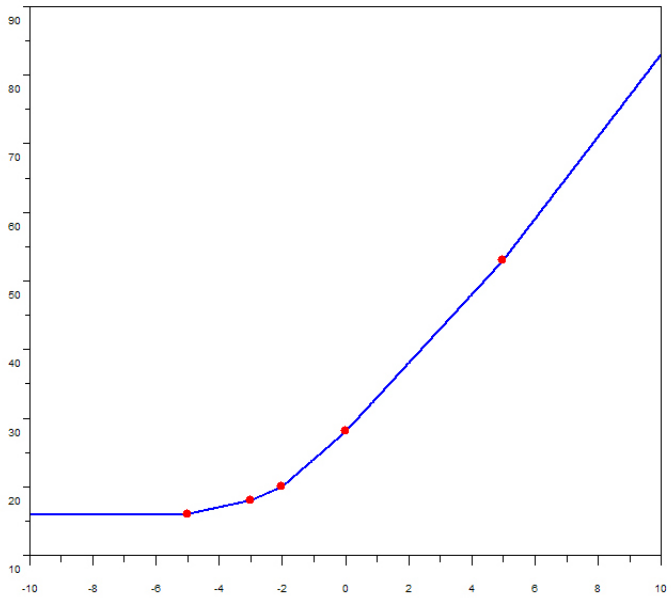


Figure 5.1: The diagram of $f(\lambda)$.

Part II

Optimal Assignment Problem

CHAPTER 6

Entropy maximization problem and max-product assignment problem*

In this chapter we consider the connection between the optimal assignment problem and the Entropy maximization problem. Due to the wide variety of applications of these problems, both ones have received a considerable attention in the fields of computer science and convex optimization and several algorithms have been developed to solve them.

The first part of this chapter is a short survey about the optimal assignment problem which includes the definition, several types of this problem and algorithms which have been developed to solve it. Next, we provide a short survey of entropy maximization problem and the related applications. In Section 6.3, we consider an entropy maximization problem with a deformation parameter. We show that the matrix maximizing the entropy converges, as the deformation parameter goes to infinity, to a matrix whose nonzero entries are precisely the ones belonging to optimal assignments. We also show that the solution of this entropy problem can be found by applying the scaling algorithms to the original matrix. These theorems let us use the scaling algorithms as the solutions of optimal as-

*The results of this chapter have been partly reported in [2, 3, 8].

signment problem. We also show, In Section 6.4, that the speed of convergence to the final solution, when the deformation parameter increases, is exponential. The theoretical results of this chapter lead to the development of new iterative methods to solve optimal assignment problem and related combinatorial optimization problems, which will be presented in the next chapter.

6.1 Optimal assignment problem

6.1.1 Definition

The assignment problem can be classically described as assigning n jobs to n machines without assigning more than one job to a machine and ensuring that all jobs are completed. A feasible solution of this problem is a bijective mapping, σ, between two sets U and V of n elements. In this way, σ is a permutation of set $1, \ldots, n$, which assign entry $i \in U$ to entry $\sigma(i) \in V$. Each permutation can be presented by a permutation matrix $X = (x_{ij})$, which is defined as $x_{ij} = 1$ if $j = \sigma(i)$ and other entries are zero.

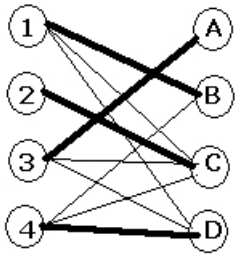


Figure 6.1: An assignment between two sets.

From the graph point of view, this problem can be explained as finding a perfect matching in a bipartite graph $G =< U, V; E >$ when the number of vertices of U and V are equal. A classical description of this problem has been given by Hermann Weyl in 1949. Consider n young ladies as the set U and n young men as the set V. Lady i is a friend of man j if there is an arc $(i, j) \in E$. A perfect matching is a collection of marriages in which all ladies and all men are married, and only friends are married.

Hall's theorem [Hal35], also known as marriage theorem, provides a necessary and sufficient condition for existence of perfect matching in a graph. For a set of vertices, X, let $N(X)$ be the set of all vertices adjacent to some element of X. Hall's theorem indicates that a bipartite graph, $G =< U, V; E >$, has a perfect matching if and only if $|X| \leq N(X)$ for every subset X of U.

This problem can also be modeled as a Network flow problem. For a bipartite

graph, $G =< U, V; E >$, the network $N =< W, A >$ can be defined as $W = \{s, t\} \cup U \cup V$, $A = E \cup \{(s, i)|i \in U\} \cup \{(i, t)|t \in V\}$ with the capacity function C defined as $c_{si} = 1$ for all $i \in U$, $c_{it} = 1$ for all $i \in U$ and for $(u_i, v_j) \in E$, $c_{u_i v_j} = 1$. In this way, s is the source node and t can be considered as a sink. Any maximum flow in this network corresponds to a matching in the graph G and if the value of maximum flow is n then, it corresponds to a perfect matching in G.

6.1.2 Linear optimal assignment problem

In general, the optimal assignment problem can be defined as finding the best assignment over all the possible ones due to an objective function. Let C be an $n \times n$ cost matrix, which determines the cost of assigning i to j. The objective function can be defined as

$$\min_{\sigma \in S_n} \sum_{i=1}^{n} c_{i\sigma(i)} \ ,$$

where S_n denotes the symmetric group of n elements. An assignment problem due to this objective function called linear assignment problem.

The formulation of this problem by using a permutation matrix, X, can be given as the following:

$$\begin{array}{lll} \min & \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} x_{ij} & \\ s.t. & \sum_{j=1}^{n} x_{ij} = 1 & (i = 1, \ldots, n) \\ & \sum_{i=1}^{n} x_{ij} = 1 & (j = 1, \ldots, n) \\ & x_{ij} \in \{0, 1\} & (i, j = 1, \ldots, n) \end{array} \ .$$

This problem is also known as *Minimum Weighted Bipartite Matching* or shortly *Minimum Weighted Matching*. Without loss of generality, the weight matrix is assumed to be nonnegative. Since, the cases with negative weight, can be reduced to the nonnegative cases by adding $-\min_{ij} c_{ij}$ to all the elements of C.

For a nonnegative weight matrix, any solution to the previous problem can be used to find the optimal solution of the following objective function,

$$\max_{\sigma \in S_n} \prod_{i=1}^{n} c_{i\sigma(i)} \ .$$

Since finding the minimum weighted matching for a cost matrix, B, which is defined as $b_{ij} = -\log c_{ij}$, yields to finding a solution for the later problem, which is also known as Max-Product Maximum Weight Matching.

Remark 14. Besides linear optimal assignment problem, the quadratic optimal assignment problem, QAP, and the multi-index assignment problem have also been well studied in the literature. QAP is a generalization of the linear one for which, in addition to the weight matrix, the distance matrix is also involved.

This problem firstly introduced by Koopmans and Beckmann [KB57] to model a facility location problem. A short description of this problem can be given as follows: for a set of n facilities, let f_{ij} denotes the flow between facility i and facility j. Also let d_{ij} denotes the distance between location i and j. Assume that the total cost depends on the flow between the facilities multiplied by their distance. Then the goal is to solve:

$$\min_{\sigma \in S_n} \sum_{i=1}^{n} \sum_{j=1}^{n} f_{ij} d_{\sigma(i)\sigma(j)} \ .$$

The QAP is proved to be *NP-complete* [SG76], thus, several heuristic algorithms have been proposed to find a sub-optimal solution [BDM09, § 7].

The multi-index assignment problem is also another generalization of the linear optimal assignment problem. In the three dimension case, it can be discussed as a timetabling problem in which the assignment of n courses to n rooms and to n time slots is required. More precisely, for a three dimensional cost matrix c_{ijk}, which denotes the cost of assigning course i to room j at time slot k, we are interested in finding the assignment σ of the courses to the rooms and an assignment ϕ of the courses to the time slots. In this way the objective function can be described as the following:

$$\min_{\sigma, \phi \in s_n} \sum_{i=1}^{n} c_{i\sigma(i)\phi(i)} \ .$$

It is proved, by Karp in 1972, that, this problem, which is known as *axial 3-index assignment problem* is NP-complete [Kar72]. For a better survey, we refer an interested reader to chapter 7 and chapter 10 of the recent book of Burkard et. al. [BDM09].

6.1.3 Applications and Solutions for the linear assignment problem

The optimal assignment problem has been applied to a number of concrete problems. A well known application has emerged in bioinformatic for protein structure alignment problem. The latter consists in aligning two proteins based on their 3-D structures [Hol93, LCL04]. Other important applications can be found in shape matching and object recognition [BMP02], image processing and computer vision [CWC+96] and VLSI design [HCLH90].

We shall be specifically interested in large scale dense optimal assignment problems. The latter arise in several applications. A well-known application arises from the approximation algorithms and heuristics for solving the Asymmetric Traveling Salesman Problem [CJMZ01]. An application to cosmology (reconstruction of the early universe) can be found in the work of Brenier et al. [BFH+03]. Models of large dense random assignment problems are also considered in [MPV87, Ch. VII] from the point of view of statistical physics.

Another important application of this problem appears in the solution of very large sparse linear systems of equations. In the context of sparse LU factorization, the purpose of the large-diagonal permutation is to decrease the probability of encountering small pivots during factorization, and hence avoid pivoting during the factorization [ON96, DK00, LD03].

Due to the variety of its applications, the optimal assignment problem has received a considerable attention and several algorithms have been proposed to solve it. The first polynomial time algorithm was proposed by H. W. Kuhn in 1955 [Kuh55]. For a linear optimization problem stated in Section 6.1.2 the corresponding dual problem can be defined as finding the vectors U and V(also called Hungarian pairs) as the following

$$\begin{array}{ll} \max & \sum_{i=1}^{n} u_i + \sum_{j=1}^{n} v_j \\ s.t. & u_i + v_j \leq c_{ij} \end{array} .$$

According to the *complementary slackness* condition, a pair of feasible solution x of the primal problem and of feasible solutions (U, V) of the dual problem are both optimal if and only if

$$x_{ij}(c_{ij} - u_i - v_j) = 0 \quad (i, j = 1, 2, \ldots, n) \ .$$

(We already encountered this condition in Chapter 5, §5.3). This algorithm computes Hungarian pairs while at the same time, finds the optimal permutation. The original algorithm works in $O(n^4)$; However it was improved later on to run in $O(n^3)$ time [DK69, EK70], which is still optimal for the dense matrices. For the sparse case, the algorithm of Edmonds and Karp [EK70] which runs in $O(n^3)$ later improved by Fredman and Tarjan [FT87]. This new algorithm, which is developed based on Fibonacci Heaps for the shortest paths computations, runs in $O(n(m + nlogn))$. A good survey on the algorithms, which have been developed for this problem can be found in [BDM09].

6.2 Entropy maximization problem

The term Entropy firstly appeared in thermodynamics to measure the amount of energy in a thermodynamic system as a function of the temperature of the system and the heat that enters the system. However, later the same term used by Shannon to measure the uncertainty associated with a random variable. This problem can be better described by how to find a probability distribution of a random variable when only the knowledge of certain moments such as expected values of the distribution is known. According to Laplace's principle of insufficient reasoning [KK92a] when there is no knowledge about the certain moments, then the uniform distribution should be chosen since this distribution maximize

the uncertainty. but when there are some information, then the definition of uncertainty becomes important. Shannon, provides some axioms, which he believes that should be satisfied by any function, which measures the uncertainty.

For a random variable with n possible outcomes, $X \equiv (x_1, \ldots, x_n)$, and $p \equiv (p_1, \ldots, p_n)^T$ as the probability distribution associated to theses outcomes, the function $S_n(p) = -k \sum_{j=1}^n p_j \ln p_j, \quad k > 0$ will satisfy all the axioms; However, Shannon used $S_n(p) = -\sum_{j=1}^n p_j \ln p_j$ as the entropy function [SW49]. In this way, for $g_1(X), \ldots, g_m(X)$ as m functions of random variable X with known expected values $E[g_1(X)] = a_1, \ldots E[g_m(X)] = a_m$, the problem of finding a distribution due to the knowledge of expected values can be mathematically formulated as follows:

$$\begin{array}{ll} \max & -\sum_{j=1}^n p_j \ln p_j \\ s.t. & \sum_{j=1}^n p_j g_i(x_j) = a_i, \quad i = 1, \ldots, m, \\ & \sum_{j=1}^n p_j = 1, \\ & p_j \geq 0, \quad j = 1, \ldots, n \end{array} .$$

Several versions of this problem have been appeared in a wide variety of applications in different fields such as thermodynamics and statistical mechanics [Jay57], Finance [BK96] and linear programming [Erl81]. For a good survey on this problem we refer an interested reader to the book of Fang [FRT97]

6.3 Deformed Entropy maximization problem and matrix scaling

The diagonal scaling problem can be generally defined as finding diagonal matrices D_r and D_c with positive diagonal entries such that the scaled matrix $D_r A D_c$ has prescribed row and column sums. Due to the variety of its applications, this problem has been well studied [MS69, Bru74, SK67]. A comparison of the proposed algorithms to solve this problem, can be found in [SZ90]. A remarkable special case arises when the row and column sums of the matrix $X = D_r A D_c$ are required to be identically one, so that X is bistochastic. Then, the following theorem provides a sufficient condition for the existence of a diagonal scaling.

Theorem 6.3.1 (Sinkhorn [SK67]). *Let A be an $n \times n$ nonnegative matrix with total support (every positive entry belongs to a diagonal). Then there exist diagonal matrices D_r and D_c such that $D_r A D_c$ is bistochastic. Moreover, if A is fully indecomposable, then D_r and D_c are unique up to a constant factor.*

Now, consider the following optimization problem, which consists in finding an $n \times n$ bistochastic matrix $X = (x_{ij})$ maximizing the following relative entropy

$$\max_{X \in B_n} J_p(X), \qquad J_p(X) := \sum_{ij} x_{ij} b_{ij} + p^{-1} S(X); \quad b_{ij} = \log a_{ij} \ , \tag{6.1}$$

where

$$S(X) := -\sum_{ij} x_{ij} \log x_{ij} \ ,$$

is the entropy function, $p > 0$ is a parameter and B_n denotes the set of $n \times n$ bistochastic matrices. The convention $0 \times (-\infty) = 0$ is understood when interpreting the product $x_{ij} b_{ij}$.

We shall assume that the matrix $A := (a_{ij})$ has total support, so that the diagonal matrices D_r and D_c are known to exist. We denote by $G(A) := \{(i,j) \mid a_{ij} > 0\}$ the *pattern* (set of non-zero entries) of the matrix A.

The general relation between the entropy maximization and scaling problems is well known, see e.g. [Sch89] for an overview. We shall need in particular the following result.

Proposition 6.3.2 (Corollary of [BLN94, Th. 3.1])**.** *Let A be a matrix with total support. Then, the solution $X(p)$ of the entropy maximization problem indicated in Equation 6.1 is unique and it is characterized by the existence of two positive vectors, U and V, such that $x_{ij} = a_{ij}^p u_i v_j$ for all i, j.*

Thus, the characterization of the proposition shows that X is obtained from the pth Hadamard power $A^{(p)} := (a_{ij}^p)$ by a diagonal scaling.

The previous proposition is a special case of Theorem 3.1 of [BLN94], which is established in a more general infinite dimensional setting (for $p = 1$; but the result for an arbitrary p follows trivially from it). We shall need in the sequel a few elements of the proof, which we next include.

First, the function J_p is upper semi-continuous, and B_n is compact, hence, the maximum of J_p over B_n is attained. If there is at least one permutation σ such that $\sum_i b_{i\sigma(i)} > -\infty$, the associated permutation matrix $X = (x_{ij})$, with $x_{ij} = 1$ if $j = \sigma(i)$, and $x_{ij} = 0$ otherwise, is such that $J_p(X) > -\infty$. Then since the maximum of J_p is attained, its value must be finite. Moreover, since the objective function is strictly concave and the feasible set is convex, the point of maximum $X(p)$ is unique.

We claim that $X(p)$ has the same pattern (set of positions of non-zeros entries) as the matrix A.

To see this, let Y be a bistochastic matrix with the same pattern as A, i.e. $y_{ij} > 0$ iff $a_{ij} > 0$. Assume by contradiction that $X(p)$ does not have the same pattern as A, so that $x_{ij}(p) = 0$ and $y_{ij}(p) > 0$ for some (i,j). Then because the right derivative of the function $t \mapsto -t \log t$ at $t = 0^+$ is infinite, the right derivative of $t \mapsto J_p(X(p) + t(Y - X(p)))$ at $t = 0^+$ is easily seen to be infinite, and so, $J_p(X(p) + t(Y - X(p))) > 0$ and $X(p) + t(Y - X(p)) \in B_n$ hold for t small enough, contradicting the optimality of $X(p)$. Hence, the claim is established.

Consider now the Lagrange function

$$L(X, U, V) = J_p(X) + \sum_i u_i (\sum_j x_{ij} - 1) + \sum_j v_j (\sum_i x_{ij} - 1) \ ,$$

where $U = (u_i)$ and $V = (v_j)$ are vectors of Lagrange multipliers. The stationarity condition implies that if X is an optimal solution of the entropy maximization problem indicated in Equation 6.1, then there must exist two vectors of multipliers U and V such that, for all $(i, j) \in G(A)$,

$$\frac{\partial L}{\partial x_{ij}} = b_{ij} - p^{-1}(1 + \log x_{ij}) + u_i + v_j = 0 \ .$$

It follows that

$$x_{ij}(p) = \exp(p(b_{ij} + u_i + v_j) - 1) \ , \qquad \forall (i, j) \in G(A) \ ,$$

showing that X is obtained from the pth Hadamard power $A^{(p)} := (a_{ij}^p)$ by a diagonal scaling.

Using the latter characterization of $X(p)$, we observe that:

$$J_p(X(p)) = -\sum_i \log u_i - \sum_j \log v_j \ .$$

We now study the convergence of $X(p)$ as p tends to infinity. We shall consider the face F of the polytope of bistochastic matrices consisting of the optimal solutions of the linear programming formulation of the optimal assignment problem

$$\max_{x \in B_n} \sum_{ij} x_{ij} b_{ij} = \max_{\sigma \in \mathfrak{S}_n} \sum_i b_{i\sigma(i)} \ .$$

Theorem 6.3.3. *As p tends to infinity, the matrix $X(p)$ converges to the unique matrix X^* maximizing the entropy among the ones that belong to the face F consisting of the convex hull of optimal permutation matrices. In particular, if the solution of the optimal assignment problem is unique, then $X(p)$ converges to the associated bistochastic matrix.*

Proof. Since $X(p)$ is the point of maximum of J_p,

$$\begin{aligned} J_p(X(p)) \quad &= \sum_{ij} x_{ij}(p) b_{ij} + p^{-1} S(X(p)) \\ &\geq J_p(X^*) = \sum_{ij} x_{ij}^* b_{ij} + p^{-1} S(X^*) \\ &= \max_{\sigma \in \mathfrak{S}_n} \sum_i b_{i\sigma(i)} + p^{-1} S(X^*) \ . \end{aligned}$$

Consider a sequence $(p_k)_{k \geq 1}$ converging to infinity, and assume that $X(p_k)$ converges to some matrix Z, which must belong to B_n. Setting $p = p_k$ in the previous inequality and taking the limit as k tends to infinity, we get $\sum_{ij} z_{ij} b_{ij} \geq \max_{\sigma \in \mathfrak{S}_n} \sum_i b_{i\sigma(i)}$, which shows that Z belongs to the face F. Observe that

$$p_k^{-1}(S(X(p_k)) - S(X^*)) = (J_{p_k}(X(p_k)) - J_{p_k}(X^*)) + \left(\sum_{ij} x_{ij}^* b_{ij} - \sum_{ij} x_{ij}(p_k) b_{ij} \right) \ ,$$

is the sum of two nonnegative terms, because $X(p_k)$ is a point of maximum of J_{p_k}, and $X^* \in F$ is a convex hull of matrices representing optimal permutations. It follows that $S(X(p_k)) - S(X^*) \geq 0$, and so, if Z is any accumulation point of $X(p_k)$ as k tends to infinity, $S(Z) - S(X^*) \geq 0$, showing that Z is of maximal entropy among the matrices in F. Since the entropy function is strictly convex, X^* is is the only point with the latter property, and so every accumulation point of $X(p_k)$ is equal to X^*, showing that $X(p)$ converges to X^* as $p \to \infty$. □

Corollary 6.3.4. *If there is only one optimal permutation, then $X(p)$ converges to the corresponding permutation matrix.*

6.4 The speed of convergence

We have already shown in Theorem 6.3.3 that the maximal entropy solution $X(p)$ converges as p tends to infinity, to a matrix $X(\infty)$, which is a convex hull of optimal permutation matrices. In particular, $X(p)$ converges to an optimal permutation matrix if the optimal permutation is unique. The following theorem shows the exponential speed of convergence when p tends to infinity.

Theorem 6.4.1. *Assume that the matrix A has total support and that $\log a_{ij} \in \mathbb{Q}$, for all (i,j) such that $a_{ij} > 0$. Then, there exists a positive constant c such that, for all $i, j \in [n]$,*

$$|x_{ij}(p) - x_{ij}(\infty)| = O(\exp(-cp)) \ .$$

To establish Theorem 6.4.1, recall that a real Puiseux series in the variable t is an expression of the form

$$f = \sum_{k \geq \bar{k}} c_k t^{k/r} \ , \tag{6.2}$$

where $r \in \mathbb{N}$ is positive, $\bar{k} \in \mathbb{Z}$, $c_k \in \mathbb{R}$ for all k, and the sum is taken over all $k \in \mathbb{Z}$ such that $k \geq \bar{k}$. We denote by $\mathbb{R}_{\mathrm{cvg}}\{\{t\}\}$ the set of real Puiseux series that are absolutely convergent for all t of small enough positive modulus.

Lemma 6.4.2. *For all $i, j \in [n]$, there exists a Puiseux series of the form (6.2), such that*

$$x_{ij}(p) = f(\exp(-p)) = \sum_{k \geq \bar{k}} c_k \exp(-pk/r) \ ,$$

the latter series being absolutely convergent for all large enough p.

In order to establish this result, we shall use some tools from the theory of real ordered fields, for which we refer the reader to [BPR06, chapter 2].

Let us consider the following statement: If a nonnegative matrix A has total support, then there exists a nonnegative matrix X with row and column sums 1,

and there exist diagonal matrices D and D' with positive diagonal entries such that

$$A = DXD' \ .$$

According to Sinkhorn's theorem [SK67] and to Proposition 6.3.2, this statement is true when the entries of A, X, D, D' belong to the field of real numbers. Moreover, this statement belongs to the first order theory of the real closed field $(\mathbb{R}, +, \times, 0, 1, >)$. By Tarski's theorem [Tar51], any first order statement that is valid in a special real closed field must also be valid in any real closed field. In particular, the above statement holds over the field of convergent real Puiseux series, which is known to be a real closed field. Indeed, the fact that formal Puiseux series constitute a real closed field is standard, the proof that the same is true in the case of convergent Puiseux series can be found in [BK76, § 10].

Thus for a matrix $A(t) \in \mathbb{R}_{\mathrm{cvg}}\{\{t\}\}^{n\times n}$ with total support, there exists diagonal matrices $D(t), D'(t), \in \mathbb{R}_{\mathrm{cvg}}\{\{t\}\}^{n\times n}$ together with a unique bistochastic matrix $X(t) \in \mathbb{R}_{\mathrm{cvg}}\{\{t\}\}^{n\times n}$ such that $A(t) = D(t)X(t)D'(t)$.

We choose now the matrix $A(t) = (a_{ij}(t))$ such that $a_{ij}(t) = t^{\log a_{ij}}$ where $\log a_{ij} \in \mathbb{Q}$. Then, the entries of the corresponding matrix $X(t)$ have the form

$$\hat{x}_{ij}(p) \qquad = \sum_{k=\bar{k}_{ij}}^{+\infty} c_{ijk} t^{k/r_{ij}} \ ,$$

and this series is convergent for a suitably small positive t. Make now the substitution $t = \exp(-p)$. We deduce that for all suitably large p,

$$x_{ij}(p) = \sum_{k=\bar{k}_{ij}}^{+\infty} c_{ijk} \exp(-pk/r_{ij}) \ . \tag{6.3}$$

Since $x(p)_{ij}$ has a finite limit as $p \to \infty$, understanding that $\bar{k}_{ij}$ is the first index k for which the coefficient c_{ijk} is non-zero, we necessarily have $\bar{k}_{ij} \geq 0$, so that $x_{ij}(\infty)$ can be identified to the constant term in the latter series. Setting $c = \min_{i,j} (\bar{k}_{ij} + 1)/r_{ij}$ we get

$$|x_{ij}(p) - x_{ij}(\infty)| = O(\exp(-cp)) \ ,$$

which proves Theorem 6.4.1.

Remark 15. The assumption that $\log a_{ij} \in \mathbb{Q}$ in Theorem 6.4.1 is inconvenient. It could be avoided by replacing the field of converging Puiseux series by a field of converging generalized Dirichlet series, along the lines of [Mar]. However, this would require working out the convergence issues, which are not treated in [Mar].

Remark 16. The formulation (6.1) is somehow reminiscent of interior point methods, in which the entropy $S(X) = -\sum_{ij} x_{ij} \log x_{ij}$ is replaced by a log-barrier

function (the latter would be $\sum_{ij} \log x_{ij}$ in the present setting). The present $X(p)$ thought of as a function of $p \to \infty$ is analogous to the *central path*, and as does the central path, $X(p)$ converges to a face containing optimal solutions. However, the entropy $S(X)$ does not satisfies the axioms of the theory of self-concordant barriers on which the analysis of interior point methods is based. Indeed, the speed of convergence in $O(\exp(-cp))$ appears to be of a totally different nature by comparison with the speed of $O(1/p)$ observed in interior point methods [NN94].

Example 6.4.1. The constant c appearing in Theorem 6.4.1 can be small if there are several nearly optimal permutations, and then a large value of p may be needed to approximate $X(\infty)$. However, in such cases, a much smaller value of p turns out to be enough for the method described in the next sections, the aim of which is to eliminate a priori entries not belonging to (nearly) optimal permutations. This is illustrated by the following matrix, in which the identity permutation is optimal, and the transposition $(1, 2)$ is nearly optimal:

$$A = \begin{pmatrix} 1 & 0.99 & 0.99 \\ 0.99 & 1 & 1/3 \\ 0.25 & 0.5 & 1 \end{pmatrix} .$$

For $p = 10$, we have the following matrix, the significant entries of which indicate precisely the optimal and nearly optimal permutations:

$$\begin{pmatrix} 0.5195148 & 0.4595136 & 0.0210196 \\ 0.4804643 & 0.5195864 & 0.0000004 \\ 0.0000209 & 0.0209000 & 0.9789800 \end{pmatrix} .$$

The convergence of $X(p)$ to $X(\infty)$ is illustrated in Figures 6.2. Observe that the graph of $\log x_{ij}(p)$ as a function of p is approximately piecewise affine. In fact, each piece corresponds to a monomial in the Puiseux series expansion (6.3) (see [Vir01] for an explanation of this fact). The path $p \mapsto X(p)$ converges quickly to the face containing the two nearly optimal permutations and slowly to the unique optimal permutation.

Remark 17. Finding the speed of convergence c is an open question. We conjecture that when the optimal permutation is unique, c is given as follows:

$A' := DPA$, where D is a diagonal matrix and P is a permutation, such that the identity permutation is optimal and has unit weights in A'.Then

$$\exp(-c) = \gamma(A) := \max_{i_1 \dots i_k} (A'_{i_1 i_2} \cdots A'_{i_k i_1})^{1/k} ,$$

where the max is taken over all elementary circuits, which are not loops. We remark that $\gamma(A) < 1$ iff the optimal permutation is unique.

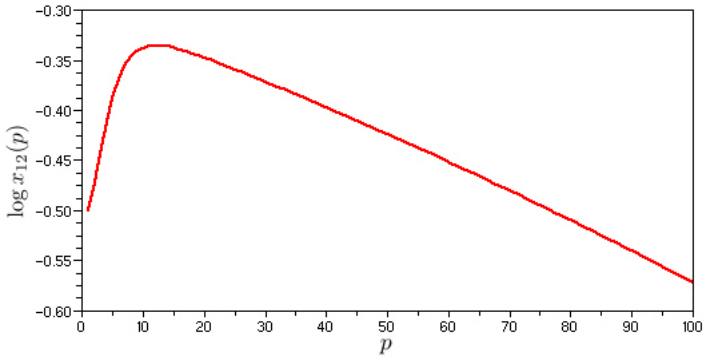


Figure 6.2: The variation of $\log_{10} x_{12}(p)$ as a function of p.

6.5 Conclusion

The main idea, which has been developed in this chapter was to think of the optimal assignment problem as the limit of a deformation of an entropy maximization problem. For an $n \times n$ bistochastic matrix $X = (x_{ij})$ and a deformation parameter p, in the following relative entropy problem

$$J_p(X) := -\sum_{1 \le i,j \le n} x_{ij}(\log(x_{ij}/a_{ij}^p) - 1) \ ,$$

we proved that when p goes to infinity, the solution of the entropy maximization problem converges to a point in the convex hull of the matrices representing optimal permutations. We also proved that for $X(p)$ as the solution of the latter problem for some value of p and for $X(\infty)$ as the solution when $p \to \infty$

$$|x_{ij}(p) - x_{ij}(\infty)| = O(\exp(-cp)) \ ,$$

for $c > 0$. This shows an exponential convergence to the optimal solution when p increases. Finding the exact speed of convergence, c, is still an open problem; However, we formulate a conjecture about it.

CHAPTER 7

Scaling algorithms for optimal assignment problem*

In this chapter, we present a preprocessing method, which is suitable for parallel computation, to solve large optimal assignment problems. We think of the optimal assignment problem as a limit of a deformation of an entropy maximization problem. For every value of the deformation parameter, the matrix of maximal entropy can be computed by Sinkhorn iteration. This leads to a parallel preprocessing for the optimal assignment problem, which allows to delete entries that do not belong to optimal assignments, so that the reduced problem becomes executable on a sequential machine.

7.1 Introduction

We showed in the previous chapter that the solution of optimal assignment problem can be computed as a solution of the following deformed entropy maximization problem when p goes to infinity.

$$\max_{X \in B_n} J_p(X), \qquad J_p(X) := \sum_{ij} x_{ij} b_{ij} + p^{-1} S(X); \quad b_{ij} = \log a_{ij} \tag{7.1}$$

*The results of this chapter have been partly reported in [2, 3, 8].

Here

$$S(X) := -\sum_{ij} x_{ij} \log x_{ij} \ ,$$

is the entropy function and B_n denotes the set of $n \times n$ bistochastic matrices. We also showed that the speed of convergence is exponential as p increases.

In this chapter, we investigate a preprocessing algorithm, which can be used to solve large scale optimal assignment problems. The preprocessing is based on an iterative method that eliminates the entries not belonging to an optimal assignment. This reduces the initial problem to a much smaller problem in terms of memory requirements. This is illustrated in Figures 7.1 and 7.2. Here, the original matrix is an Euclidean random matrix, that is, a matrix whose entries are the Euclidean distance between two sets of n random points in the Euclidean space. Figures 7.1 illustrates the graph corresponding to a 50×50 Euclidean random matrix and Figure 7.2 illustrates the graph corresponding to a matrix after applying the preprocessing algorithm. It can be seen that most of the unnecessary arcs have been removed and subsequently the size of problem is reduced.

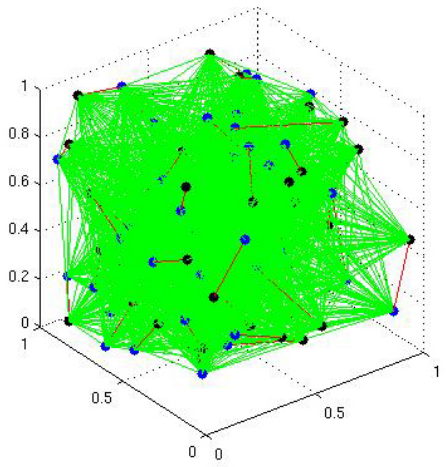


Figure 7.1: The graph corresponding to an Euclidean random matrix where the dimension is 50.

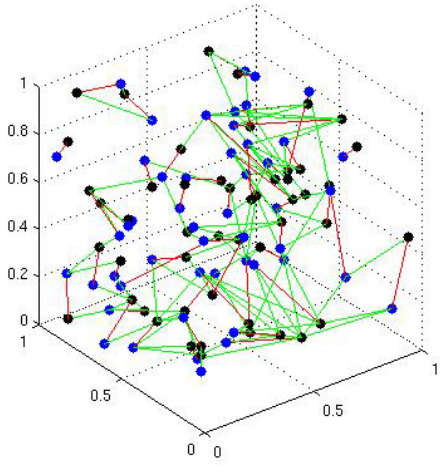


Figure 7.2: The graph corresponding to the reduced matrix by applying the preprocessing algorithm.

The idea of this algorithm is to take p large enough, then apply a diagonal scaling algorithm to $A^{(p)}$ until convergence to a bistochastic matrix X, and finally delete the small entries of X. Here, the exponential of $A^{(p)}$ leads to numerical overflow for large values of p. However, we shall show that it is possible to implement this iteration in a numerically stable way. The present algorithm assumes the existence of at least one matching, since otherwise, the Sinkhorn iteration [SK67] may not converge. However, we note that matrix balancing

(Sinkhorn iteration) can also be used to detect the existence of a perfect matching, as shown by Linial, Samorodnitsky and Wigderson [LSW00].

We consider two variants of the algorithm, one by using the Sinkhorn iteration as the diagonal scaling algorithm and the other one by using Newton iteration. The advantage of Sinkhorn iteration is that, it can be efficiently implemented in parallel [ADRU08]. Thus we show that for very large dense optimal assignment problems the data of which can not be stored in one machine, the parallel Sinkhorn iteration can be used to reduce the size of the problem and then, the reduced problem can be solved by any classical method. On the other hand, the advantage of Newton method is the speed of the convergence to bistochastic matrix.

For both variants, we present several numerical results of various full and sparse matrices from gallery of Matlab and *The University of Florida Sparse Matrix Collection*. We show that the Sinkhorn iteration can be efficiently used to decrease the size of the dense matrices, up to 99% in a small number of iterations. For Newton iteration, we show that it is not only efficient for dense matrices but also efficient for sparse symmetric matrices.

Note also that the present approach yields approximate dual variables, which provide an approximate optimality certificate for the assignment, which is found (Section 7.3.1).

An interesting application of this new approach, is the solution of large scale dense optimal assignment problems. Several efforts have been made to solve this problem [BT09, LO94]. A well-known application arises from the approximation algorithms and heuristics for solving the Asymmetric Traveling Salesman Problem or the Vehicle Routing Problem. There are also some applications in object recognition and computer vision. An application to cosmology (reconstruction of the early universe) can be found in the work of Brenier et al. [BFH+03]. For a survey on the applications of large dense linear assignment problems, we refer the reader to [BT09]. Models of large dense random assignment problems are also considered in [MPV87, Ch. VII] from the point of view of statistical physics.

In the last section, we introduce a new iterative method, which is based on a modification of the Sinkhorn scaling algorithm, in which the deformation parameter is slowly increased (this procedure is reminiscent from simulated annealing, the parameter p playing the role of the inverse of the temperature). We prove that this iteration, which we refer to as *deformed-Sinkhorn iteration*, converges to a matrix whose entries that belong to the optimal permutations are nonzero, while all the other entries are zero. An estimation of the rate of convergence is also presented, but this appears to be mostly of theoretical interest since in practice, the convergence of this variant appears to be slow.

7.2 Preprocessing algorithm

7.2.1 Main idea

For a fixed $p > 0$, the solution for the entropy maximization problem displayed in equation (7.1) can be computed by any scaling algorithm such as Sinkhorn iteration [SK67] or Newton method [KR07]. Using Theorem 6.4.1, it can be seen that if the original matrix has only one optimal permutation, the order of magnitude of all the entries, which belong to the optimal permutation will be $1 \pm O(\exp(-cp))$ while the order of magnitude of all other entries are $O(\exp(-cp))$. As an example, consider the following 5 by 5 random matrix with the bold entries belonging to optimal permutation.

$$A = \begin{pmatrix} 0.292 & 0.502 & \mathbf{0.918} & 0.281 & 0.686 \\ 0.566 & \mathbf{0.437} & 0.044 & 0.128 & 0.153 \\ 0.483 & 0.269 & 0.482 & \mathbf{0.778} & 0.697 \\ 0.332 & 0.633 & 0.264 & 0.212 & \mathbf{0.842} \\ \mathbf{0.594} & 0.405 & 0.415 & 0.112 & 0.406 \end{pmatrix}$$

By applying the Sinkhorn iteration on $A^{(50)}$ The following matrix can be computed.

$$X(50) = \begin{pmatrix} 3.4E-27 & 1.5E-08 & \mathbf{1.0E{+}00} & 7.4E-26 & 4.7E-06 \\ 4.8E-02 & \mathbf{9.4E{-}01} & 4.6E-56 & 4.0E-32 & 7.9E-28 \\ 2.5E-13 & 4.6E-19 & 9.3E-12 & \mathbf{1.0E{+}00} & 1.0E-02 \\ 1.5E-23 & 1.2E-02 & 6.2E-27 & 4.3E-31 & \mathbf{9.8E{-}01} \\ \mathbf{9.5E{-}01} & 4.1E-02 & 6.2E-07 & 1.0E-34 & 2.3E-06 \end{pmatrix}$$

Thus, for sufficiently large values of p, when $X(p)$ is an $\epsilon-$bistochastic matrix, meaning that, some distance between $X(p)$ and a bistochastic matrix is less than ϵ, one may delete all the small entries, which are less than a threshold t, chosen consistent with ϵ, while keeping all others. In this way the size of the original problem in terms of memory requirements will be reduced to a much smaller one.

For a column(row) stochastic matrix, that is a matrix for which the sum of all columns(rows) are one, the distance to the set of bistochastic matrices will be measured by $\max_i |r_i - 1|$ where r_i indicates the ith row(column) sum.

Determining the coarsest accuracy ϵ and the maximal threshold t which are needed to find an optimal permutation would require to know the maximal entropy solution $X(\infty)$ characterized in Theorem 6.3.3. This information is in general not available. However, the worst case can be considered to be the one where $X(\infty)$ is uniform, with all entries equal $1/n$ (and $n!$ optimal permutations). Since we need to preserve the optimal permutations, this leads to a conservative choice $\epsilon = t = 1/n$, which we adopted in the present experimental results. The choice of the value of p will be discussed in Section 7.3.2. This leads to Algorithm 7.1.

Algorithm 7.1 An optimal assignment preprocessing for fixed p

input: A, p
$n \leftarrow size(A, 1)$
$\epsilon, t \leftarrow 1/n$
comment: Prescaling
if $\frac{\max(A)}{\min(A)} > e$ **then**
  $m \leftarrow \frac{1}{\log(\max(A)/\min(A))}$, $c \leftarrow e^{\frac{\log(\min(A))}{\log(\max(A)/\min(A))}}$
  $A \leftarrow \frac{1}{c} A^{(m)}$
else
  $A \leftarrow \frac{1}{\min(A)} A$
end if
$B \leftarrow A^{(p)}$
comment: Main loop
repeat
  Apply one iteration of any diagonal scaling algorithm to B so $B \leftarrow DB'D$, where D, D' are diagonal matrices
until B is $\epsilon-$bistochastic
Delete all the entries of B, which are less than a threshold t

7.2.2 Prescaling

The naive computation of $A^{(p)}$ is numerically unstable for large values of p. This can be avoided by the prescaling step in the Algorithm 7.1. We set $\max(A) = \max_{ij} a_{ij}, \min(A) = \min_{a_{ij}>0} a_{ij}$. By applying this prescaling, all the nonzero scaled entries will be placed in the $[1, e]$ interval. In the case when $\frac{\max(A)}{\min(A)} > e$, the prescaling has another interesting property, that is, the scaled matrix is invariant of entrywise powers of input matrix. In other words, if we apply the prescaling to the matrix $A^{(q)}$, for all $q \geq 1$, the matrix obtained after the prescaling step turns out to be independent of the choice of q. When $\frac{\max(A)}{\min(A)} < e$ the entries of A have already been located in the interval $\min(A)[1, e]$, then we do not need to perform the previous prescaling since the denominator in the formula defining m will be small if $\max(A)$ is close to $\min(A)$. We shall also see in Section 7.3.1 that the iterations can be implemented robustly for large values of p by working with log-coordinates. Next, we provide more details on the proposed algorithm.

7.3 Sinkhorn iteration

A simple way to compute the diagonal matrices D, D' is Sinkhorn iteration [SK67]. This algorithm starts from a given matrix A, divides every row by its sum, then every column of the new matrix by its sum, and so on, until the matrix obtained in this way converges to a bistochastic matrix. The advantage of this algorithm is that, it can be efficiently implemented in parallel [ADRU08] and it can be applied

to any non-negative matrix, which has at least one nonzero permutation. The disadvantage is that, it is generally slower than other methods.

Recall first that the open cone $C = \{x \in \mathbb{R}^n : x_i > 0, \forall i\}$ consisting of positive vectors of $\mathbb{R}^n$ is equipped with Hilbert's projective metric, defined by

$$d(x, x') = \log \max_{i,j} \frac{x_i x'_j}{x'_i x_j} \ .$$

Note that $d(x, x')$ is zero if and only if the vectors x and x' are proportional. We refer to [BR97, § 6] for more background. In particular, if A is a positive matrix, a theorem of Birkhoff shows that the map $x \mapsto Ax$ is a contraction in Hilbert's projective metric, with a contraction rate

$$\kappa(A) := \sup\{\frac{d(Ay, Ay')}{d(y, y')} : y, y' \in C, y, y' \text{ non proportional}\} = \frac{\theta(A)^{1/2} - 1}{\theta(A)^{1/2} + 1} \ ,$$

where

$$\theta(A) = \exp \sup\{d(Ay, Ay') : y, y' \in C\} = \max_{i,j,p,l} \frac{a_{ir} a_{jl}}{a_{jr} a_{il}} \ .$$

The following result is a consequence of this theorem.

Proposition 7.3.1 (Franklin and Lorenz [FL89])**.** *For a positive matrix A, the global rate of convergence of Sinkhorn iteration is bounded above by $\kappa(A)^2$.*

This general bound is applicable only for positive matrices and it can be coarse in practice. Recently, Knight [Kni08] provided a local rate of convergence. Due to his work, for classical Sinkhorn iteration the local rate of convergence of a fully indecomposable matrix, is bounded by σ_2^2 where σ_2 is the second singular value of the bistochastic matrix to which the iteration converges. Hence, the following result allows us to estimate the local convergence rate of Sinkhorn iteration, as $p \to \infty$.

Proposition 7.3.2. *Assume that there is only one optimal permutation. Then, there is a constant $c > 0$ such that*

$$1 - O(\exp(-cp)) \leq \sigma_2(X(p)) \leq 1 \qquad \text{as } p \to \infty \ .$$

Assume now that the matrix $X(\infty)$ is fully indecomposable (which implies that there are several optimal permutations). Then,

$$\sigma_2(X(p)) \to \sigma_2(X(\infty)) < 1 \qquad \text{as } p \to \infty \ .$$

Proof. Due to the perturbation theorem of Mirsky [Mir60], for any unitarily invariant norm $\|.\|$ and $n \times n$ matrices, X and $\tilde{X}$ with singular values $\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_p$ and $\tilde{\sigma_1} \geq \tilde{\sigma_2} \geq \ldots \geq \tilde{\sigma_p}$ respectively, we have,

$$\| \operatorname{diag}(\tilde{\sigma_i} - \sigma_i) \| \leq \|\tilde{X} - X\| \ .$$

So, for $X(p)$ and $X(\infty)$,

$$|\sigma_2(X(p)) - \sigma_2(X(\infty))| \leq \|X(p) - X(\infty)\|_2 \leq O(\exp(-cp)) \ ,$$

for which the constant c depends on the coefficients of the Puiseux series and possibly on the dimension of $X(p)$. Thus, if the original matrix has only one optimal permutation, $\sigma_2(X(\infty)) = 1$, which implies that

$$1 - O(\exp(-cp)) \leq \sigma_2(X(p)) \ .$$

Moreover according to the Birkhoff-von Neumann theorem [Bir46], for any norm $\|.\|$ on $\mathbb{R}^n$, which is invariant under permutation of the coordinates and for any bistochastic matrix X, $\|X\| = 1$ and subsequently

$$1 - O(\exp(-cp)) \leq \sigma_2(X(p)) \leq 1 \ .$$

When $X(\infty)$ is fully indecomposable, since the multiplication of two fully indecomposable matrices is also fully indecomposable, $M = X(\infty)X^T(\infty)$ is fully indecomposable. Note also that for all $1 \leq i \leq n$, $m_{ii} = \sum_{j=1}^{n} x_{ij}^2 > 0$, which implies that M is primitive. Then, according to the Perron-Frobenius theorem, all the eigenvalues of M distinct from $\rho(M)$ have a modulus strictly smaller than $\rho(M) = 1$, which yields $\sigma_2(X(\infty)) < 1$. □

7.3.1 Logarithmic p-Sinkhorn iteration

As it was discussed before, computing the pth Hadamard power of A may cause some numerical difficulties. To avoid this problem a prescaling has been proposed, after which all the matrix entries are in $[1, e]$ interval. A theoretical disadvantage of this prescaling is that the increase of p is limited since $e^p < l$, where l is the largest number, in the numerical range. However, we next give a log-coordinate implementation of Sinkhorn iteration which avoids this limitation. This will provide as a by product a certificate allowing one to check the approximate optimality of a permutation.

Let $A \in \mathbb{R}^{n \times n}$ be a real non-negative matrix, which has total support. For a given p, consider the following iteration for a sequence of vectors $U_k, V_k \in \mathbb{R}^n$

$$V_0 = \mathbb{1} \tag{7.2}$$
$$U_{k+1} = \mathcal{I}(A^{(p)} V_k) \tag{7.3}$$
$$V_{k+1} = \mathcal{I}(A^{(p)T} U_{k+1}) \tag{7.4}$$

where $\mathbb{1}$ is a vector $[1, 1, \ldots, 1]^T$ of dimension n and $\mathcal{I}$ is an operator, which inverses the entries of a vector.

Proposition 7.3.3. *For a nonnegative matrix, A, which has total support, the iteration defined by equations 7.2, 7.3 and 7.4 coincides with Sinkhorn iteration.*

Proof. Let W_k and Z_k respectively, be column scaled and row scaled matrices defined as the following:

$$W_k = \operatorname{diag}(U_k) A^{(p)} \operatorname{diag}(V_k) \ ,$$
$$Z_k = \operatorname{diag}(U_{k+1}) A^{(p)} \operatorname{diag}(V_k) \ .$$

Also, let $\mathcal{C}$ denote the column scaling operator in which all the columns of a matrix are divided by it's sums and $\mathcal{R}$ be the similar operator for rows. It is easy to verify that, $\mathcal{R}(DB) = \mathcal{R}(B)$ and $\mathcal{C}(BD) = \mathcal{C}(B)$ for any diagonal matrix D. According to the definition

$$Z_k = \mathcal{R}(A^{(p)} \operatorname{diag}(V_k)) = \mathcal{R}(\operatorname{diag}(U_k) A^{(p)} \operatorname{diag}(V_k)) = \mathcal{R}(W_k) \ .$$

A similar statement can be proved for W_k, that is, $W_K = \mathcal{C}(Z_K)$, which completes the proof. □

Assume that $\bar{U_k} = (u_i^k) = p^{-1} \log U_k$ and $\bar{V_k} = (v_i^k) = p^{-1} \log V_k$, then, the logarithmic form of this iteration can be written as:

$$\bar{u}_i^{k+1} = -\frac{1}{p} \log \sum_j \exp p(\log a_{ij} + \bar{v}_j^k) \ ,$$
$$\bar{v}_i^{k+1} = -\frac{1}{p} \log \sum_j \exp p(\log a_{ji} + \bar{u}_j^{k+1}) \ .$$

Let

$$\hat{x}_{ij} = \log a_{ij} + \bar{v}_j^k - \max_j (\log a_{ij} + \bar{v}_j^k) \ ,$$
$$\hat{y}_{ji} = \log a_{ji} + \bar{u}_j^{k+1} - \max_j (\log a_{ji} + \bar{u}_j^{k+1}) \ ,$$

for which $\hat{x}_{ij}, \hat{y}_{ji} \leq 0$. The logarithmic iteration can be reformulated by using $\hat{x}_{ij}$ and $\hat{y}_{ji}$ as the following:

$$\bar{u}_i^{k+1} = -\max_j (\log a_{ij} + \bar{v}_j^k) - \frac{1}{p} \log \sum_j \exp p\hat{x}_{ij} \tag{7.5}$$

$$\bar{v}_i^{k+1} = -\max_j (\log a_{ji} + \bar{u}_j^{k+1}) - \frac{1}{p} \log \sum_j \exp p\hat{y}_{ji} \tag{7.6}$$

The last iteration can be computed for a sufficiently large p, without having numerical difficulties. We note that a related trick was used by Malajovich and Zubelli [MZ01] in a different context.

Proposition 7.3.4 (Approximate optimality certificate)**.** *Let $\bar{U}, \bar{V}$ and $\hat{X}$ be produced by the p-Sinkhorn iteration. Also, let $\zeta_i := \frac{1}{p} \log \sum_j \exp p\hat{x}_{ij}$ and let* Val(OAP) *denote the logarithmic of the value of an optimal permutation. Then,*

$$\text{Val(OAP)} \leq -\sum_{i=1}^n \bar{u}_i - \sum_{j=1}^n \bar{v}_j - \sum_{i=1}^n \zeta_i \ . \tag{7.7}$$

Proof. Observe that at each step of the Sinkhorn iteration:

$$\log a_{ij} + \bar{v}_j^k \leq -\bar{u}_i^{k+1} - \zeta_i \ , \qquad 1 \leq i \leq n \ .$$

Let σ denote an optimal permutation. Choosing $j = \sigma(i)$ in the previous inequality, and summing over $1 \leq i \leq n$, we get (7.7). □

In practice, this proposition will be used to check the validity of the preprocessing, by comparing the logarithm of the value of the permutation which is eventually found with the upper bound (7.7).

7.3.2 Experimental results

The experiments, which are presented here have been obtained by using Sinkhorn iteration in Algorithm 7.1 as a diagonal scaling method. We used Matlab version 7.10.0. The detailed Matlab implementation of the algorithm is presented below.

Finding the best value for p seems to be tricky since increasing p yields a slow convergence and at the same time, it yields the lower percentage of remaining entries. This fact also can be seen in Figures 7.3, 7.4 which illustrate the percentage of the remaining entries and the required number of Sinkhorn iterations, for several values of p for the "lotkin" 1000 by 1000 matrix from the gallery of Matlab. In the following experiments, we set the parameter p to 100 which leads

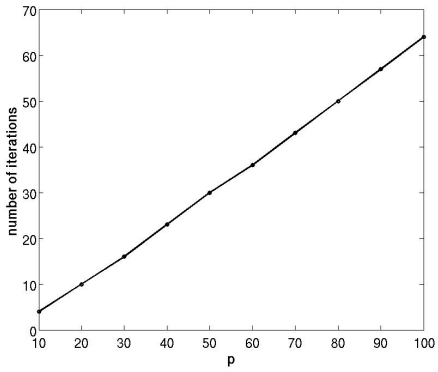


Figure 7.3: The number of iterations as a function of p.

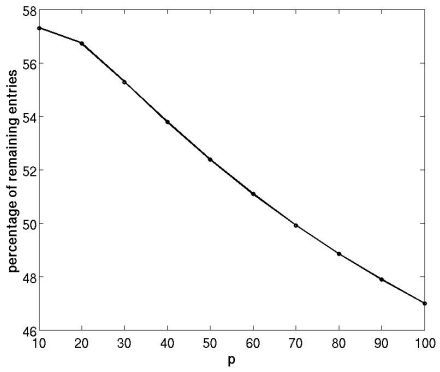


Figure 7.4: The percentage of remaining entries as a function of p.

to a reasonable decrease in the size of the problem and generally does not yield to a slow convergence, however it could be any reasonably large value. Recall that the convergence is measured by $\max_i |r_i - 1|$, where r_i denotes the ith row (column) sum for a column (row) stochastic matrix.

Table 7.1 displays the results for dense matrices from the gallery of test matrices of Matlab. For these experiments the dimension is 5000. The columns

Matlab code for p-Sinkhorn iteration

```
function [it,A]=psinkhorn(A)
    n=size(A,1);
    t=1/n;
    p=100;
    Min=min(A(A>0));
    Max=max(A(A>0));
    if (Max/Min)>exp(1)      %prescaling
        m=1/(log(Max)-log(Min));
        c=exp(log(Min)/(log(Max)-log(Min)));
        A=(1/c)*(A.^m);
    else
        m=1/log(Max);
        A=A.^m;
    end
    A=A.^(p);
    d=(1/n)+1;
    it=0;
    while (d> 1/n)           %main loop
        A=diag(sparse((A*ones(n,1)).^(-1)))*A;
        A=A*diag(sparse((A'*ones(n,1)).^(-1)));
        d=max(abs(sum(A')-1));
        it=it+1;
    end;
    [indx,indy]=find(A>t);
    A=sparse(indx,indy,1,n,n).*A;
end
```

from left to right are: gallery name, number of nonzeros, number of iterations, the logarithmic value of optimal assignment and the percentage of remaining entries after deleting small entries. The same results are also presented for a random matrix, referred to as "'rand"'(the random function of Matlab) and an Euclidean random matrix referred to as "'Euclidean"'. The latter, which is of interest in statistical physics, is a matrix whose entries are functions of random points in an Euclidean space [Par02]. More precisely, we draw at random $2n$ points $x_1, \ldots, x_n; y_1, \ldots, y_n$ uniformly in the unit cube of $\mathbb{R}^3$. Then, we consider the matrix $A = (a_{ij})$ where $a_{ij} = \exp(-d(x_i, y_j))$ and d is the Euclidean distance. In this way, a permutation σ, which maximizes $\prod_{i=1}^{n} a_{ij}$ is the same permutation which minimizes the distance between these two sets of points.

Table 7.1: Sinkhorn iteration for dense matrices from the gallery of test matrices of Matlab and for random and random Euclidean distance matrices.

Gallery	nnz	No. it.	Val(OAP)	Rem. En.(%)
cauchy	25000000	79	$4.54725E+00$	47.95
minij	25000000	473	$1.25025E+07$	26.57
moler	25000000	304	$4.99950E+07$	28.43
orthog	25000000	304	$4.99950E+07$	28.43
pei	25000000	1	$5.50000E+04$	00.02
prolate	25000000	42	$2.00000E+03$	00.66
randcorr	25000000	1	$5.00000E+03$	00.02
toeppd	25000000	1	$1.24767E+07$	00.02
chebvand	24997500	2	$5.00000E+03$	38.67
circul	25000000	1	$2.50000E+07$	19.48
cycol	25000000	3	$1.73422E+04$	13.23
lotkin	25000000	73	$5.54715E+00$	48.59
rand	25000000	2	$4.99837E+03$	28.38
Euclidean	25000000	417	$4.77693E+03$	01.49
chebspec	25000000	1084	$5.33411E+07$	01.98
lehmer	25000000	3537	$5.00000E+03$	18.58
gcdmat	25000000	11174	$1.25025E+07$	00.06

As Table 7.1 shows, For more than 58% of the cases, the algorithm converges very fast (in less than 80 iterations) and for 82% of the cases the algorithm converges in less than 500 iterations(which is less than 0.1 of the dimension of the input matrix). Also for more than 41% of the cases the original problem reduced to a new problem, which has less than 2% of the original entries and in 82% it reduces to a new problem with less than 30% of the input entries. Since, the Sinkhorn iteration can be implemented in parallel, this method can be efficiently applied to large dense optimal assignment problems as a parallel preprocessing to reduce the size of the original problem.

We also tested several sparse matrices from *The University of Florida Sparse Matrix Collection*. The results, which are presented in Table 7.2, show that using Sinkhorn iteration as a diagonal scaling method in Algorithm 7.1 generally makes a slow convergence for sparse matrices.

7.4 Newton Iteration

Solving the diagonal matrix scaling problem by using Newton iteration has been considered first in the work of Khachian and Kahalantari [KK92b] for positive semidefinite symmetric matrices. They have considered the more general problem of finding a positive zero of the mapping

$$f(x) = b + Ax - x^{-1} \ ,$$

where A is a given matrix of dimension n and b is a fixed $n-$dimensional vector. They proposed a path-following Newton algorithm of complexity $O(\sqrt{n}L)$ where L is the binary length of the input.

Recently, Knight and Ruiz have considered a Newton algorithm for the non-negative matrices [KR07]. For a symmetric matrix A, they considered the diag-

Table 7.2: Sinkhorn iteration for sparse matrices from *The University of Florida Sparse Matrix Collection.*

Gallery	n	nnz	No. it.	Val(OAP)	Rem. En.(%)
af23560	23560	460598	22195	$9.74612E+05$	70.32
bayer04	20545	85537	655255	$6.42574E+10$	80.57
bbmat	38744	1771722	15421	$1.35879E+06$	32.83
ecl32	51993	380415	120688	$1.63238E+06$	81.95
g7jac200sc	59310	717620	164538	$8.11767E+05$	86.40
gemat11	4929	33108	6373	$1.54838E+04$	84.70
graham1	9035	335472	17795	$2.64107E+06$	51.59
hcircuit	105676	513072	444744	$4.89275E+04$	88.59
hydr1	5308	22680	92812	$9.23658E+06$	79.38
jpwh_991	991	6027	1	$5.18100E+03$	16.44
mahindas	1258	7682	8390	$2.64476E+03$	32.27
onetone1	36057	335552	72335	$1.50614E+07$	88.03
onetone2	36057	222596	73250	$1.50614E+07$	85.70
orani678	2529	90158	8482	$2.43343E+03$	05.22
sherman3	5005	20033	6007	$7.31238E+07$	85.66
sherman5	3312	20793	3313	$2.00876E+05$	29.57
2cubes_sphere	101492	1647264	57403	$5.10796E+13$	95.91
Andrews	60000	760154	1	$7.00154E+05$	07.89
apache2	715176	4817870	248887	$1.36293E+10$	26.18
boneS01	127224	5516602	1	$1.13217E+09$	02.31
cfd1	70656	1825580	836	$7.06560E+04$	26.15
denormal	89400	1156224	59627	$3.55688E+03$	07.73
Dubcova3	146689	3636643	1405	$1.78266E+05$	46.57
ecology1	1000000	4996000	1	$3.86818E+07$	20.02
filter3D	106437	2707179	26814	$2.17778E+02$	79.95
finan512	74752	596992	73701	$3.27922E+05$	19.73
G2_circuit	150102	726674	245564	$1.00644E+08$	42.42
GaAsH6	61349	3381809	1096	$2.70486E+06$	28.82
gas_sensor	66917	1703365	18454	$4.38605E+03$	90.36
H2O	67024	2216736	10570	$6.66094E+06$	03.03
helm2d03	392257	2741935	1	$1.40807E+06$	14.31
Lin	256000	1766400	1	$1.35416E+08$	14.49
nasasrb	54870	2677324	8863	$1.17425E+12$	62.37
offshore	259789	4242673	37024	$2.74591E+18$	99.87
parabolic_fem	525825	3674625	155111	$2.09716E+05$	71.46
qa8fm	66127	1660579	1	$1.66971E+01$	03.98
rail_79841	79841	553921	57795	$1.79469E+00$	15.14
s3dkq4m2	90449	4427725	5793	$6.34128E+07$	73.77
shallow_water2	81920	327680	1	$2.07196E+15$	25.00

onal matrix scaling problem as finding vector x such that

$$f(x) = D(x)Ax - \mathbb{1} = 0 \ ,$$

where $D(x) = diag(x)$. If A is nonsymmetric, then the following matrix will be considered as the input of the algorithm.

$$S = \begin{pmatrix} 0 & A \\ A^T & 0 \end{pmatrix}$$

They showed that the Newton iteration can be written as

$$A_k x_{k+1} = Ax_k + D(x_k)^{-1}\mathbb{1} \ ,$$

where $A_k = A + D(x_k)^{-1}D(Ax_k)$. Thus in each iteration a linear system of equations should be solved for which they used the Conjugate Gradient method.

In the nonsymmetric case, the latter linear system is singular, however it is proved that the system is consistent whenever A has support ($A \geq 0$ has support if it has a positive diagonal). Our experiments, which will be presented later shows that, the method works fast for dense nonsymmetric matrices. However with the default tuning parameters, it does not work fast in sparse nonsymmetric cases. More details about this method can be found in [KR07]; However, We present the exact Matlab implementation of the algorithm in Appendix D Here, we used the later method in Algorithm 7.1 to find the scaling matrices. We also set the parameter p to 100, which is the same as Sinkhorn iteration.

In the following tables, No. it. denotes the total number of operations, each of them takes $O(n^2)$ time to be done. This includes all the iterations of Conjugate Gradient method for each Newton step. Tables 7.3 and 7.4 show the results for dense symmetric and nonsymmetric matrices with dimension 5000. For both cases the algorithm converges rapidly in a small number of iterations. The percentage of the remaining entries is reasonably less than the original problem. In fact, in more than 38% of the cases, the original problem reduced to a much smaller problem, which has less than 2% of the original entries and in 72% of the cases the problem reduces to a problem with less than 30% of the original entries.

Table 7.3: Newton iteration for dense symmetric matrices.

Gallery	nnz	No. it.	Val(OAP)	Rem. En.(%)
cauchy	25000000	156	$-4.10569E+04$	47.95
fiedler	24995000	175	$3.91202E+04$	35.73
gcdmat	25000000	152	$3.75911E+04$	00.06
lehmer	25000000	166	$0.00000E+00$	18.58
minij	25000000	167	$3.75911E+04$	26.57
moler	25000000	167	$4.45149E+04$	28.43
orthog	25000000	164	$-1.9561E+04$	48.10
pei	25000000	151	$1.19895E+04$	00.02
prolate	25000000	155	$-4.58145E+03$	00.66
randcorr	25000000	151	$0.00000E+00$	00.02
toeppd	25000000	151	$3.91132E+04$	00.02

Table 7.4: Newton iteration for dense nonsymmetric matrices.

Gallery	nnz	No. it.	Val(OAP)	Rem. En.(%)
chebspec	25000000	251	$4.03274E+04$	01.98
chebvand	24997500	166	$-1.19254E-03$	38.67
circul	25000000	161	$4.25860E+04$	19.48
cycol	25000000	162	$6.19386E+03$	11.81
lotkin	25000000	257	$-4.10477E+04$	48.59
rand	25000000	164	$-1.63137E+00$	28.39
Euclidean	25000000	314	$-2.30779E+02$	01.49

Tables 7.5 and 7.6 show the result of this algorithm on several sparse symmetric and nonsymmetric matrices from *The University of Florida Sparse Matrix Collection*. These results show that the algorithm generally works very well for sparse symmetric matrices while the convergence for sparse nonsymmetric matrices is not fast.

Table 7.5: Newton iteration for sparse symmetric matrices.

Gallery	n	nnz	No. it.	Val(OAP)	Rem. En.(%)
2cubes_sphere	101492	1647264	155	$1.29645E+06$	95.91
Andrews	60000	760154	151	$1.45202E+05$	07.89
apache2	715176	4817870	155	$6.65166E+06$	26.18
boneS01	127224	5516602	153	$1.13622E+06$	02.31
denormal	89400	1156224	153	$-2.88379E+05$	07.73
Dubcova3	146689	3636643	159	$-8.55189E+03$	46.57
ecology1	1000000	4996000	153	$3.61494E+06$	20.02
filter3D	106437	2707179	161	$-7.01011E+05$	79.95
finan512	74752	596992	151	$1.03471E+05$	19.67
G2_circuit	150102	726674	153	$6.58486E+05$	41.77
GaAsH6	61349	3381809	162	$2.32268E+05$	28.82
gas_sensor	66917	1703365	160	$-4.89303E+05$	90.37
H2O	67024	2216736	153	$3.08149E+05$	03.02
helm2d03	392257	2741935	153	$5.01026E+05$	14.31
Lin	256000	1766400	153	$1.60526E+06$	14.49
nasasrb	54870	2677324	161	$8.56473E+05$	62.37
offshore	259789	4242673	161	$4.84144E+06$	99.87
parabolic_fem	525825	3674625	153	$-4.83938E+05$	71.46
qa8fm	66127	1660579	153	$-5.51168E+05$	03.98
rail_79841	79841	553921	151	$-8.54968E+05$	15.09
s3dkq4m2	90449	4427725	161	$5.21115E+04$	73.77
shallow_water2	81920	327680	151	$1.95771E+06$	25.00
ship_003	121728	3777036	161	$3.05969E+06$	85.85
shipsec8	114919	3303553	164	$1.94819E+06$	82.96
t3dh_e	79171	4352105	156	$-1.28870E+06$	27.32
thermomech_TK	102158	711558	151	$4.85968E+05$	15.49
tmt_sym	726713	5080961	158	$1.00529E+06$	71.46
filter3D	106437	2707179	161	$-7.01011E+05$	79.95
G3_circuit	1585478	7660826	153	$6.72048E+06$	72.19
H2O	67024	2216736	153	$3.08149E+05$	03.02
SiO2	155331	11283503	153	$7.14208E+05$	17.34
thermal2	1228045	8580313	154	$1.63908E+06$	80.32

Table 7.6: Newton iteration for sparse nonsymmetric matrices.

Gallery	n	nnz	No. it.	Val(OAP)	Rem. En.(%)
af23560	23560	460598	2248	$8.74776E+04$	70.32
bayer04	20545	85537	183275	$-5.45190E+04$	80.21
bbmat	38744	1771722	2234	$4.73786E+04$	32.83
ecl32	51993	380415	23389	$-2.73185E+05$	81.66
g7jac200sc	59310	717620	47245	$3.93891E+04$	86.40
gemat11	4929	33108	2780	$4.07095E+03$	84.70
graham1	9035	335472	4014	$-1.84675E+04$	51.59
hcircuit	105676	513072	34980	$-3.83585E+05$	88.59
hydr1	5308	22680	73772	$5.25311E+03$	78.65
jpwh_991	991	6027	151	$1.47688E+03$	16.44
lhr71c	70304	1528092	2227871	$-7.63013E+04$	83.56
mahindas	1258	7682	3485	$-6.49190E+01$	31.71
onetone1	36057	335552	23601	$1.13220E+05$	87.97
onetone2	36057	222596	24122	$1.13220E+05$	85.64
orani678	2529	90158	5073	$-1.57076E+02$	05.20
sherman3	5005	20033	168	$-2.62102E+04$	85.63
sherman5	3312	20793	1696	$6.67064E+03$	29.55

7.5 Deformed Sinkhorn iteration

In the previous section, we computed $X(p)$ for a fixed value of p. However, it is natural to develop a "path following method" in which the value of p is gradually increased in the course of Sinkhorn balancing iterations. In this section we propose such an algorithm. We prove that if the matrix A has support (A has support if it has a positive diagonal), and if the growth of p is moderate enough, then the sequence of matrices produced by the algorithm converges to a point, which belongs to the face generated by optimal permutations.

7.5.1 Definition

Let $A \in \mathbb{R}^{n\times n}$ be a real non-negative matrix. Consider the following iteration, which is a standard Sinkhorn iteration with a deformation of using a sequence p_m, which goes to infinity.

$$\begin{aligned} U_{m+1} &= \mathcal{I}(A^{(p_{m+1})}V_m) \\ V_{m+1} &= \mathcal{I}(A^{(p_{m+1})^T}U_{m+1}) \end{aligned}$$

Let W_{m+1} and Z_m respectively, be column scaled and row scaled matrices defined as the following:

$$\begin{aligned} W_{m+1} &= \operatorname{diag}(U_{m+1})A^{(p_{m+1})}\operatorname{diag}(V_{m+1}) \ , \\ Z_m &= \operatorname{diag}(U_{m+1})A^{(p_{m+1})}\operatorname{diag}(V_m) \ . \end{aligned} \tag{7.8}$$

Proposition 7.5.1. *For a diagonal matrix D, real matrices B, C and the matrices W_m, Z_m in the iteration, the following properties hold.*

1. $\mathcal{R}(C \circ (DB)) = \mathcal{R}(C \circ B)$ *where $\circ$ indicates the Hadamard product*
2. $W_m = \mathcal{C}(Z_{m-1})$
3. $Z_m = \mathcal{R}(W_m \circ A^{(p_{m+1}-p_m)})$

Proof. We only prove the last one since others are straightforward.

$$\begin{aligned} Z_m \quad &= \mathcal{R}(A^{(p_{m+1})}\operatorname{diag}(V_m)) \\ &= \mathcal{R}(A^{(p_m)}\operatorname{diag}(V_m) \circ A^{(p_{m+1}-p_m)}) \\ &= \mathcal{R}((\operatorname{diag}(U_m)A^{(p_m)}\operatorname{diag}(V_m)) \circ A^{(p_{m+1}-p_m)}) \\ &= \mathcal{R}(W_m \circ A^{(p_{m+1}-p_m)}) \end{aligned}$$

□

According to the previous proposition, we define the following iteration, which we refer to as *deformed Sinkhorn iteration*.

$$W_0 = \mathcal{C}(A^{(p_0)});$$
$$W_m = \mathcal{C}(Z_{m-1}), \quad c_m = ({Z_{m-1}}^T)\mathbb{1} \tag{7.9}$$
$$Z_m = \mathcal{R}(W_m \circ A^{(p_{m+1}-p_m)}), \quad r_m = (W_m \circ A^{(p_{m+1}-p_m)})\mathbb{1} \tag{7.10}$$

Here, r_m, c_m respectively are the vectors of row sums and column sums.

7.5.2 Convergence to optimal assignment

For an input matrix, $A = (a_{ij})$, assume that the deformed Sinkhorn iteration converges to a bistochastic matrix. Define the weight of a permutation, σ, with respect to A, to be $\omega_\sigma(A) = \prod_i a_{i\sigma(i)}$. If A has a support, it should have at least one optimal permutation as σ_{opt} with nonzero weight. It is evident that σ_{opt} is the optimal permutation for all the matrices W_m and Z_m produced by each deformed Sinkhorn iteration. Observe that for all permutations σ and π, the ratio $\frac{\omega_\sigma(A)}{\omega_\pi(A)}$ is invariant if we multiply the matrix A by diagonal matrices. So it follows from the Equation 7.8 that

$$\gamma_m = \frac{\omega_\sigma(Z^m)}{\omega_\pi(Z^m)} = \gamma_{m-1}(\frac{\omega_\sigma(A)}{\omega_\pi(A)})^{p_{m+1}-p_m} = (\frac{\omega_\sigma(A)}{\omega_\pi(A)})^{p_{m+1}} \quad .$$

Thus, for all non optimal permutations such as σ, $\frac{\omega_\sigma(Z^m)}{\omega_{\sigma_{opt}}(Z^m)}$ will converge to zero when $p_m \to \infty$. Since in each iteration the weight of optimal permutation, $\omega_{\sigma_{opt}}(Z^m)$, is bounded above by 1, the weight of all non optimal permutations will converge to zero, which yields the following lemma.

Lemma 7.5.2. *Assume that the deformed Sinkhorn iteration converges to a matrix, Z, produced by the deformed Sinkhorn iteration when $p_m \to \infty$. If the original matrix A has a support, then all the permutations of Z have zero weight, except the optimal permutations of the original matrix A.*

Due to the theorem of Birkhoff-von Neumann, a square bistochastic matrix in $\mathbb{R}$ is a convex combination of permutation matrices. Hence, all the nonzero entries of a bistochastic matrix belong to a permutation with nonzero weight. This statement together with the previous lemma yield the following theorem.

Theorem 7.5.3. *For a non-negative matrix A, which has a support, as $p_m \to \infty$, if the deformed Sinkhorn iteration converges to a matrix X, then all the nonzero entries of X belong to an optimal permutation of the original matrix.*

7.5.3 Convergence to bistochastic matrix for positive matrices

Recall that the rate of convergence of classical Sinkhorn iteration is bounded above by $\kappa(A)^2$ where $\kappa(A) = \frac{\theta(A)^{1/2}-1}{\theta(A)^{1/2}+1}$. The following theorem presents the main result of this section:

Theorem 7.5.4. *Let A be a positive matrix. If $p_m = a\log(m+1)$ where $0 < a\log\theta < 2$, then the deformed Sinkhorn iteration will converge to a bistochastic matrix and subsequently to a solution of optimal assignment of the original matrix A.*

The proof relies on the next lemmas. For a matrix A, $\theta(A) = \theta(A^T)$ and for two diagonally equivalent matrices such as A and B, $\theta(A) = \theta(B)$.

Lemma 7.5.5. *For positive matrices A and B and diagonal matrix D and $d(x, x')$ as the Hilbert projective metric, the following properties hold.*

1. $d(Ax, Ax') \leq k(A)d(x, x')$
2. $d((A \circ B)x, x') \leq \log \frac{\max(B)}{\min(B)} + d(Ax, x')$
3. $\kappa(AD \circ B) = \kappa(A \circ BD) = \kappa((A \circ B)D) = \kappa(D(A \circ B)) = \kappa(A \circ B)$

Proof. The proof is straightforward. □

Corollary 7.5.6. *$\kappa(A)$ is invariant under $\mathcal{R}$ or $\mathcal{C}$ operators.*

Lemma 7.5.7. *Let W_m and Z_m be the matrices in equations (7.9,7.10) at iteration m. The following properties hold.*

1. $\kappa(Z_m) = \kappa(A^{(p_{m+1})})$
2. $\kappa(W_m) = \kappa(A^{(p_m)})$

Proof. The proof is straight forward by using the induction on m. □

The next lemma is similar to Lemma 2 in [FL89], where the classical Sinkhorn iteration is considered.

Lemma 7.5.8. *Let r_m, c_m be the vectors defined in equation (7.9,7.10) at iteration m and $M = \frac{\max(A)}{\min(A)}$ then*

$$\begin{aligned} d(r_m, \mathbb{1}) &\leq (p_{m+1} - p_m)\log M + (p_m - p_{m-1})\kappa(A^{(p_m)})\log M \\ &\quad + \kappa(A^{(p_m)})\kappa(A^{(p_{m-1})})d(r_{m-1}, \mathbb{1}) \\ d(c_m, \mathbb{1}) &\leq (p_m - p_{m-1})\log M + (p_m - p_{m-1})\kappa(A^{(p_{m-1})})\log M \\ &\quad + \kappa^2(A^{(p_{m-1})})d(c_{m-1}, \mathbb{1}) \end{aligned}$$

Proof. Let $\mathbb{1}/V$ indicates the entrywise inverse of a given vector, V. We have,

$$\begin{aligned} r_m &= (W_m \circ A^{(p_{m+1}-p_m)})\mathbb{1} = (Z_{m-1}\,\mathrm{diag}(\mathbb{1}/c_m) \circ A^{(p_{m+1}-p_m)})\mathbb{1} \\ &= (Z_{m-1} \circ A^{(p_{m+1}-p_m)})\,\mathrm{diag}(\mathbb{1}/c_m)\mathbb{1} = (Z_{m-1} \circ A^{(p_{m+1}-p_m)})(\mathbb{1}/c_m), \end{aligned}$$

so

$$\begin{aligned} d(r_m, \mathbb{1}) &= d((Z_{m-1} \circ A^{(p_{m+1}-p_m)})(\mathbb{1}/c_m), Z_{m-1}\mathbb{1}) \\ &\leq (p_{m+1} - p_m)\log M + \kappa(Z_{m-1})d(c_m, \mathbb{1}) \\ &= (p_{m+1} - p_m)\log M + \kappa(A^{(p_m)})d(c_m, \mathbb{1}). \end{aligned}$$

Also

$$\begin{aligned} d(c_m, \mathbb{1}) &= d((W_{m-1}{}^T \circ A^{(p_m - p_{m-1})T})(\mathbb{1}/r_{m-1}), W_{m-1}{}^T\mathbb{1}) \\ &\leq (p_m - p_{m-1})\log M + \kappa(W_{m-1}{}^T)d(\mathbb{1}/r_{m-1}, \mathbb{1}) \\ &= (p_m - p_{m-1})\log M + \kappa(W_{m-1})d(r_{m-1}, \mathbb{1}) \\ &= (p_m - p_{m-1})\log M + \kappa(A^{(p_{m-1})})d(r_{m-1}, \mathbb{1}), \end{aligned}$$

then

$$\begin{aligned} d(r_m, \mathbb{1}) \leq\ & (p_{m+1} - p_m)\log M + (p_m - p_{m-1})\kappa(A^{(p_m)})\log M \\ & +\kappa(A^{(p_m)})\kappa(A^{(p_{m-1})})d(r_{m-1}, \mathbb{1}) \ . \end{aligned}$$

The second statement is established in a similar way.

□

Lemma 7.5.9. *Assume that* $p_m = a\log(m+1)$*, where* $0 < a\log\theta(A) < 2$*. Then we have* $\lim_{m\to\infty} d(c_m, \mathbb{1}) = 0$*.*

Proof. Since

$$\begin{aligned} d(c_m, \mathbb{1}) &= a\log\frac{m+1}{m}\log M + a\log\frac{m+1}{m}\kappa(A^{(p_{m-1})})\log M \\ &\quad +\kappa^2(A^{(p_{m-1})})d(c_{m-1}, \mathbb{1}) \\ &< \frac{2a\log M}{m} + \kappa^2(A^{(p_{m-1})})d(c_{m-1}, \mathbb{1}) \ . \end{aligned}$$

Let $\beta_1 := d(c_1, \mathbb{1})$, and define the sequence β_m by $\beta_m := f_{m-1}(\beta_{m-1})$, where

$$f_{m-1}(x) = \frac{2a\log M}{m} + \kappa^2(A^{(p_{m-1})})x \ .$$

Since every function f_m is nondecreasing, an immediate induction shows that $d(c_m, \mathbb{1}) \leq \beta_m$, for all $m \geq 1$, and so, it suffices to show that $\lim_m \beta_m = 0$.

Let l_m be the fixed point of f_{m-1}. Setting

$$\alpha := \frac{a\log\theta(A)}{2} \ ,$$

and observing that

$$1 - \kappa^2(A^{(p_{m-1})}) = \frac{4m^{-\alpha}}{(1+m^{-\alpha})^2} \ ,$$

we get

$$l_m = \frac{2a \log M}{m(1-\kappa^2(A^{(p_{m-1})}))} = \frac{a \log M}{2} \frac{\left(1+m^{-\alpha}\right)^2}{m^{1-\alpha}} \ .$$

Since $0 < \alpha < 1$, one readily checks that the sequence l_m decreases with m and converges to zero. If $\beta_{m+1} \leq l_m$ for every m, then $\lim_{m\to\infty} \beta_m \leq \lim_{m\to\infty} l_m = 0$, and the result is established. Assume now that $\beta_{m+1} > l_m$ for some m. Define $\delta_k := \beta_{k+1} - l_k$ for all $k \geq m$. Observe that

$$\delta_{k+1} = f_k(\beta_k) - f_k(l_k) = \kappa^2(A^{(p_k)})(\beta_k - l_k) = \kappa^2(A^{(p_k)})\delta_k + \kappa^2(A^{(p_k)})(l_{k-1} - l_k) \ .$$

Using the fact that $\kappa^2(A^{(p_r)}) \leq 1$ holds for all r, an immediate induction yields

$$\delta_k \leq \big(\prod_{r=m}^{k-1} \kappa^2(A^{(p_r)})\big)\delta_m + l_m - l_k, \qquad \forall k \geq m+1 \ . \tag{7.11}$$

Since $1 - \kappa^2(A^{(p_r)}) \sim 4r^{-\alpha}$, we have

$$\prod_{r=m}^{\infty} \kappa(A^{(p_r)}) = 0 \ .$$

Letting $k \to \infty$ in (7.11), we get $\limsup_{k\to\infty} \delta_k \leq l_m$. Since this holds for all m, it follows that $\limsup_{k\to\infty} \delta_k \leq 0$, and so $\limsup_{k\to\infty} \beta_{k+1} = \limsup_{k\to\infty} \delta_k + l_k \leq \limsup_{k\to\infty} \delta_k + \lim_{k\to\infty} l_k = 0$. Hence, β_k converges to zero.

□

The proof of the Theorem 7.5.4 is achieved since $\lim_{m\to\infty} d(c_m, \mathbb{1}) = 0$ yields $\lim_{m\to\infty} d(r_m, \mathbb{1}) = 0$

7.6 Conclusion

We proposed an algorithm, which can be used as a preprocessing in the solution of large scale optimal assignment problems to reduce the size of the input problem in terms of memory requirements.

Two variants of the algorithm have been implemented. The first variant, which is based on Sinkhorn iteration, shows generally reasonable convergence for dense matrices with the reduction up to 99% of the input problem. However the algorithm works slowly for sparse matrices. This version of the algorithm can be efficiently used as a parallel preprocessing to reduce the size of the input problem in very large dense optimal assignment problems.

Another variant of the algorithm implemented by using the Newton iteration which shows fast convergence for all dense matrices and sparse symmetric matrices. However the convergence speed for sparse nonsymmetric matrices is slow.

The last section of this chapter concerns a new iterative method that we refer to as *deformed-Sinkhorn iteration*. It is proved that the iteration converges to the solution of optimal assignment problem, if the input matrix is positive and if it has only one optimal permutation. For positive matrices with more than one optimal permutation, the iteration converges to a matrix for which all the nonzero entries belong to at least one optimal permutation.

Publications and communications to conferences concerning the present work

[1] M. Akian, S. Gaubert, and M. Sharify. Tropical approximation of matrix eigenvalues. In *16th Conference of the International Linear Algebra Society* (ILAS), Pisa, Italy, 2010. [cited at p. 33]

[2] S. Gaubert, L. Grigori, and M. Sharify. A parallel optimal assignment algorithm based on diagonal scaling, SIAM workshop on combinatorial scientific computing (CSC09). Monterey Bay - Seaside, California, USA, 2009. Extended abstract, http://www.boman.us/CSC09/abstracts/csc09_submission_25.pdf. [cited at p. 67, 79]

[3] S. Gaubert, L. Grigori, and M. Sharify. A parallel preprocessing for the optimal assignment problem based on diagonal scaling, 6th international workshop on parallel matrix algorithms and applications (PMAA10). Basel, Switzerland, 2010. http://www.pmaa10.unibas.ch/programme.php. [cited at p. 67, 79]

[4] S. Gaubert and M. Sharify. Tropical scaling of polynomial eigenvalue problem. In C*ongrès* SMAI, La Colle sur Loup, Alpes Maritimes, France, 2009. Poster representation, http://smai.emath.fr/smai2009/resumesPDF/meisamsharify/Abstract.pdf. [cited at p. 33]

[5] S. Gaubert and M. Sharify. Tropical scaling of polynomial eigenvalue problems, 1st Montreal workshop on idempotent and tropical mathematics. Montreal, Canada, 2009. [cited at p. 33]

[6] S. Gaubert and M. Sharify. Tropical scaling of polynomial eigenvalue problems, SIAM conference on applied linear algebra (LA09). Monterey Bay - Seaside, California, USA, 2009. http://www.siam.org/meetings/la09/LA09abstracts.pdf. [cited at p. 33]

[7] S. Gaubert and M. Sharify. Tropical scaling of polynomial matrices. In Rafael Bru and Sergio Romero-Vivó, editors, *Proceedings of the third Multidisciplinary International Symposium on Positive Systems: Theory and Applications (POSTA 09)*, volume 389

of *LNCIS*, pages 291–303, Valencia, Spain, 2009. Springer. Eprint doi:10.1007/978-3-642-02894-6_28, arXiv:arXiv:0905.0121. [cited at p. 33]

[8] Meisam Sharify, Stéphane Gaubert, and Laura Grigori. A parallel preprocessing for the optimal assignment problem based on diagonal scaling. submitted. Also: arXiv:arXiv:1104.3830, 2011. [cited at p. 67, 79]

Bibliography

[AA09] Sk. Safique Ahmad and Rafikul Alam. Pseudospectra, critical points and multiple eigenvalues of matrix polynomials. *Linear Algebra and its Applications*, 430(4):1171 – 1195, 2009. [cited at p. 3, 34]

[ABG04] Marianne Akian, Ravindra Bapat, and Stéphane Gaubert. Perturbation of eigenvalues of matrix pencils and the optimal assignment problem. *C. R. Math. Acad. Sci. Paris*, 339(2):103–108, 2004. [cited at p. 3, 15, 16, 34, 43, 48, 50, 54]

[ABG05] M. Akian, R. Bapat, and S. Gaubert. Min-plus methods in eigenvalue perturbation theory and generalised Lidskii-Vishik-Ljusternik theorem. arXiv:math.SP/0402090, 2005. [cited at p. 3, 11, 15, 34, 48, 50, 54]

[ABG06] M. Akian, R. Bapat, and S. Gaubert. Max-plus algebras. In L. Hogben, editor, *Handbook of Linear Algebra (Discrete Mathematics and Its Applications)*, volume 39. Chapman & Hall/CRC, 2006. Chapter 25. [cited at p. 10, 14]

[ADRU08] P. Amestoy, I. S. Duff, D. Ruiz, and B. Uçar. A parallel matrix scaling algorithm. In *High Performance Computing for Computational Science - VECPAR 2008*, volume 5336 of *Lecture Notes in Computer Science*, pages 301–313. Springer Berlin / Heidelberg, 2008. [cited at p. 4, 81, 83]

[AGG09] M. Akian, S. Gaubert, and A. Guterman. Linear independence over tropical semirings and beyond. In G.L. Litvinov and S.N. Sergeev, editors, *Proceedings of the International Conference on Tropical and Idempotent Mathematics*, volume 495 of *Contemporary Mathematics*, pages 1–38. American Mathematical Society, 2009. arXiv:0812.3496. [cited at p. 15, 48]

[AGL08] Marianne Akian, Stéphane Gaubert, and Asma Lakhoua. The max-plus finite element method for solving deterministic optimal control problems: basic properties and convergence analysis. *SIAM J. Control Optim.*, 47(2):817–848, 2008. [cited at p. 2]

[AM62] A. R. Amir-Moez. Khayyam's solution of cubic equations. *Mathematics Magazine*, 35:269–271, Nov. 1962. [cited at p. 18]

[BB03] R. E. Burkard and P. Butkovič. Finding all essential terms of a characteristic maxpolynomial. *Discrete Appl. Math.*, 130(3):367–380, 2003. [cited at p. 3, 47, 49, 51, 55]

[BCGG09] P. Butkovič, R. A. Cuninghame-Green, and S. Gaubert. Reducible spectral theory with applications to the robustness of matrices in max-algebra. *SIAM J. Matrix Anal. Appl.*, 31(3):1412–1431, 2009. Eprint doi:10.1137/080731232. [cited at p. 14]

[BCOQ92] François Louis Baccelli, Guy Cohen, Geert Jan Olsder, and Jean-Pierre Quadrat. *Synchronization and linearity*. Wiley Series in Probability and Mathematical Statistics: Probability and Mathematical Statistics. John Wiley & Sons Ltd., Chichester, 1992. An algebra for discrete event systems. [cited at p. 1, 2, 9, 10, 14, 15, 48]

[BDM09] Rainer Burkard, Mauro Dell'Amico, and Silvano Martello. *Assignment problems*. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 2009. [cited at p. 70, 71]

[BEK90] R. Bhatia, L. Elsner, and G. Krause. Bounds for the variation of the roots of a polynomial and the eigenvalues of a matrix. *Linear Algebra Appl.*, 142:195–209, 1990. [cited at p. 39]

[BFH+03] Y. Brenier, U. Frisch, M. Henon, G. Loeper, S. Matarrese, R. Mohayaee, and A. Sobolevskii. Reconstruction of the early universe as a convex optimization problem. *Mon.Not.Roy.Astron.Soc.*, 346:501–524, 2003. [cited at p. 4, 70, 81]

[Bir46] Garrett Birkhoff. Three observations on linear algebra. *Univ. Nac. Tucumán. Revista A.*, 5:147–151, 1946. [cited at p. 85]

[BK76] Truman Bewley and Elon Kohlberg. The asymptotic theory of stochastic games. *Math. Oper. Res.*, 1(3):197–208, 1976. [cited at p. 76]

[BK96] Peter W. Buchen and Michael Kelly. The maximum entropy distribution of an asset inferred from option prices. *Journal of Financial and Quantitative Analysis*, 31(01):143–159, March 1996. [cited at p. 72]

[BLN94] J. M. Borwein, A. S. Lewis, and R. D. Nussbaum. Entropy minimization, *DAD* problems, and doubly stochastic kernels. *J. Funct. Anal.*, 123(2):264–307, 1994. [cited at p. 73]

[BM00] Peter Butkovic and Louise Murfitt. Calculating essential terms of a characteristic maxpolynomial. *CEJOR Cent. Eur. J. Oper. Res.*, 8(3):237–246, 2000. [cited at p. 47, 48]

[BMP02] S. Belongie, J. Malik, and J. Puzicha. Shape matching and object recognition using shape contexts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(4):509–522, April 2002. [cited at p. 70]

[BPR06] Saugata Basu, Richard Pollack, and Marie-Françoise Roy. *Algorithms in real algebraic geometry*, volume 10 of *Algorithms and Computation in Mathematics*. Springer-Verlag, Berlin, second edition, 2006. [cited at p. 75]

[BR97] R. B. Bapat and T. E. S. Raghavan. *Nonnegative matrices and applications*, volume 64 of *Encyclopedia of Mathematics and its Applications*. Cambridge University Press, Cambridge, 1997. [cited at p. 84]

[Bru74] Richard A. Brualdi. The DAD theorem for arbitrary row sums. *Proc. Amer. Math. Soc.*, 45:189–194, 1974. [cited at p. 72]

[BSvdD95] R. B. Bapat, David P. Stanford, and P. van den Driessche. Pattern properties and spectral inequalities in max algebra. *SIAM J. Matrix Anal. Appl.*, 16(3):964–976, 1995. [cited at p. 14]

[BT09] L. Buš and P. Tvrdík. Towards auction algorithms for large dense assignment problems. *Comput. Optim. Appl.*, 43(3):411–436, 2009. [cited at p. 4, 81]

[But10] Peter Butkovič. *Max-linear Systems: Theory and Algorithms*. Springer Monographs in Mathematics, 2010. [cited at p. 2]

[CG60] R.A. Cuninghame-Green. Process synchronization in a steelworks-a problem of feasibility. In *Proceedings of the Second International Conference on Operational Research*, pages 323–328, London, 1960. English University Press. [cited at p. 2]

[CG83] R. A. Cuninghame-Green. The characteristic maxpolynomial of a matrix. *J. Math. Anal. Appl.*, 95(1):110–116, 1983. [cited at p. 48, 51]

[CG94] R.A. Cuninghame-Green. Minimax algebra and applications. volume 90 of *Advances in Imaging and Electron Physics*, pages 1 – 121. Elsevier, 1994. [cited at p. 14]

[CGM80] R. A. Cuninghame-Green and P. F. J. Meijer. An algebra for piecewise-linear minimax problems. *Discrete Appl. Math.*, 2(4):267–294, 1980. [cited at p. 1, 11]

[CGQ01] G. Cohen, S. Gaubert, and J.P. Quadrat. Hahn-Banach separation theorem for max-plus semimodules. In J.L. Menaldi, E. Rofman, and A. Sulem, editors, *Optimal Control and Partial Differential Equations*, pages 325–334. IOS Press, 2001. [cited at p. 2]

[CJMZ01] Jill Cirasella, David S. Johnson, Lyle A. McGeoch, and Weixiong Zhang. The asymmetric traveling salesman problem: Algorithms, instance generators, and tests. In Adam L. Buchsbaum and Jack Snoeyink, editors, *ALENEX*, volume 2153 of *Lecture Notes in Computer Science*, pages 32–59. Springer, 2001. [cited at p. 70]

[CMQV84] G. Cohen, P. Moller, J. P. Quadrat, and M. Viot. Linear system theory for discrete event systems. In *Decision and Control, 1984. The 23rd IEEE Conference on*, volume 23, pages 539 –544, 1984. [cited at p. 2]

[CpQ99] Guy Cohen and Stphane Gaubert Jean pierre Quadrat. Max-plus algebra and system theory: Where we are and where to go now. *Annu. Rev. Control*, pages 207–219, 1999. [cited at p. 1]

[CtCG+98] Jean Cochet-terrasson, Guy Cohen, Stephane Gaubert, Michael Mc Gettrick, and Jean pierre Quadrat. Numerical computation of spectral elements in max-plus algebra, 1998. [cited at p. 14]

[CWC+96] Yong-Qing Cheng, Victor Wu, Robert T. Collins, Allen R. Hanson, and Edward M. Riseman. Maximum-weight bipartite matching technique and its application in image feature matching. In *In Proc. SPIE Visual Comm. and Image Processing*, 1996. [cited at p. 3, 70]

[DG97] Ali Dasdan and Rajesh K. Gupta. Faster maximum and minimum mean cycle algorithms for system performance analysis. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 17:889–899, 1997. [cited at p. 14]

[DK69] E. A. Dinic and M. A. Kronrod. An algorithm for solving the assignment problem. *Dokl. Akad. Nauk SSSR*, 189:23–25, 1969. [cited at p. 71]

[DK00] I. S. Duff and J. Koster. On algorithms for permuting large entries to the diagonal of a sparse matrix. *SIAM J. Matrix Anal. Appl.*, 22(4):973–996, 2000. [cited at p. 3, 71]

[EK70] Jack Edmonds and Richard M. Karp. Theoretical improvements in algorithmic efficiency for network flow problems. In *Combinatorial Structures and their Applications (Proc. Calgary Internat. Conf., Calgary, Alta., 1969)*, pages 93–96. Gordon and Breach, New York, 1970. [cited at p. 71]

[Erl81] Sven Erlander. Entropy in linear programs. *Math. Programming*, 21(2):137–151, 1981. [cited at p. 72]

[FL89] Joel Franklin and Jens Lorenz. On the scaling of multidimensional matrices. *Linear Algebra Appl.*, 114/115:717–735, 1989. [cited at p. 84, 95]

[FLVD04] Hung-Yuan Fan, Wen-Wei Lin, and Paul Van Dooren. Normwise scaling of second order polynomial matrices. *SIAM J. Matrix Anal. Appl.*, 26(1):252–256, 2004. [cited at p. 3, 34, 35, 36, 37, 43]

[FPT00] M. Forsberg, M. Passare, and A. Tsikh. Laurent determinants and arrangements of hyperplane amoebas. *Adv. Math.*, 151(1):45–70, 2000. [cited at p. 2]

[FRT97] S.-C. Fang, J. R. Rajasekera, and H.-S. J. Tsao. *Entropy optimization and mathematical programming*. International Series in Operations Research & Management Science, 8. Kluwer Academic Publishers, Boston, MA, 1997. [cited at p. 72]

[FT87] Michael L. Fredman and Robert Endre Tarjan. Fibonacci heaps and their uses in improved network optimization algorithms. *J. Assoc. Comput. Mach.*, 34(3):596–615, 1987. [cited at p. 71]

[Gau92] S. Gaubert. *Théorie des systèmes linéaires dans les dioïdes*. Thèse, École des Mines de Paris, July 1992. [cited at p. 14]

[Gau98] Stéphane Gaubert. Two lectures on max-plus algebra. In *In Proceedings of the 26th Spring School on Theoretical Computer Science and Automatic Control, Noirmoutier*, 1998. [cited at p. 15]

[Gau09] Stephane Gaubert. Max-plus and tropical convexity unified: Some unexpected results. Seminar on Applications of Tropical Algebra, UC Berkeley, 2009. [cited at p. 1]

[GH08] A. Galántai and C. J. Hegedűs. Perturbation bounds for polynomials. *Numer. Math.*, 109(1):77–100, 2008. [cited at p. 38]

[GK10] Elisabeth Gassner and Bettina Klinz. A fast parametric assignment algorithm with applications in max-algebra. *Networks*, 55(2):61–77, 2010. [cited at p. 47, 49]

[GM84] M. Gondran and M. Minoux. Linear algebra in dioids: a survey of recent results. In *Algebraic and combinatorial methods in operations research*, volume 95 of *North-Holland Math. Stud.*, pages 147–163. North-Holland, Amsterdam, 1984. [cited at p. 2, 15, 48]

[Gol97] Andrew V. Goldberg. An efficient implementation of a scaling minimum-cost flow algorithm. *J. Algorithms*, 22(1):1–29, 1997. [cited at p. 125]

[Gra72] R. L. Graham. An efficient algorithm for determining the convex hull of a finite planar set. *Inf. Proc. Lett.*, 1(4):132–133, 1972. [cited at p. 11]

[Had93] Jacques Hadamard. Étude sur les propriétés des fonctions entières et en particulier d'une fonction considéré par Riemann. *Journal de Mathématiques Pures et Appliquées*, 58:171215, 1893. [cited at p. 2, 19]

[Hal35] P. Hall. On representatives of subsets. *J. London Math. Soc.*, 10(37):26–30, 1935. [cited at p. 68]

[HCLH90] Chu-Yi Huang, Yen-Shen Chen, Yan-Long Lin, and Yu-Chin Hsu. Data path allocation based on bipartite weighted matching. *Design Automation Conference*, pages 499–504, 1990. [cited at p. 3, 70]

[HLT07] Nicholas J. Higham, Ren-Cang Li, and Françoise Tisseur. Backward error of polynomial eigenproblems solved by linearization. *SIAM J. Matrix Anal. Appl.*, 29(4):1218–1241, 2007. [cited at p. 3, 34]

[HMT06] Nicholas J. Higham, D. Steven Mackey, and Françoise Tisseur. The conditioning of linearizations of matrix polynomials. *SIAM J. Matrix Anal. Appl.*, 28(4):1005–1028, 2006. [cited at p. 3, 34]

[Hol93] L. Holm. Protein Structure Comparison by Alignment of Distance Matrices. *Journal of Molecular Biology*, 233(1):123–138, September 1993. [cited at p. 3, 70]

[HT03] Nicholas J. Higham and Françoise Tisseur. Bounds for eigenvalues of matrix polynomials. *Linear Algebra Appl.*, 358:5–22, 2003. Special issue on accurate solution of eigenvalue problems (Hagen, 2000). [cited at p. 41]

[IMS07] I. Itenberg, G. Mikhalkin, and E. Shustin. *Tropical algebraic geometry.* Oberwolfach seminars. Birkhäuser, 2007. [cited at p. 11, 34]

[Jay57] E. T. Jaynes. Information theory and statistical mechanics. II. *Phys. Rev. (2)*, 108:171–190, 1957. [cited at p. 72]

[JvdWJ06] Olsder G. J. and van der Woude J. *Max Plus at Work - Modeling and Analysis of Synchronized Systems.* Princeton Series in Applied Mathematics. Princeton University Press, 2006. [cited at p. 1]

[Kar72] Richard M. Karp. Reducibility among combinatorial problems. In *Complexity of computer computations (Proc. Sympos., IBM Thomas J. Watson Res. Center, Yorktown Heights, N.Y., 1972)*, pages 85–103. Plenum, New York, 1972. [cited at p. 70]

[Kar78] Richard M. Karp. A characterization of the minimum cycle mean in a digraph. *Discrete Math.*, 23(3):309–311, 1978. [cited at p. 14]

[KB57] Tjalling C. Koopmans and Martin Beckmann. Assignment problems and the location of economic activities. *Econometrica*, 25:53–76, 1957. [cited at p. 70]

[Kir09] B. Kh. Kirshtein. Complex roots of systems of tropical equations and stability of electrical power networks. In *Tropical and idempotent mathematics*, volume 495 of *Contemp. Math.*, pages 213–238. Amer. Math. Soc., Providence, RI, 2009. [cited at p. 20]

[KK90] Ikeda Kiyohiro and Murota Kazuo. Critical initial imperfection of structures. *International Journal of Solids and Structures*, 26(8):865 – 886, 1990. [cited at p. 16]

[KK92a] J. N. Kapur and H. K. Kesavan. *Entropy optimization principles with applications.* Academic Press Inc., Boston, MA, 1992. [cited at p. 71]

[KK92b] L. Khachiyan and B. Kalantari. Diagonal matrix scaling and linear programming. *SIAM J. Optim.*, 2(4):668–672, 1992. [cited at p. 89]

[KM97] V. N. Kolokoltsov and V. P. Maslov. *Idempotent analysis and its applications*, volume 401 of *Mathematics and its Applications.* Kluwer Academic Publishers Group, Dordrecht, 1997. [cited at p. 1, 2, 10]

[Kni08] Philip A. Knight. The Sinkhorn-Knopp algorithm: convergence and applications. *SIAM J. Matrix Anal. Appl.*, 30(1):261–275, 2008. [cited at p. 84]

[KR07] P. A. Knight and D. Ruiz. A fast algorithm for matrix balancing. In Andreas Frommer, Michael W. Mahoney, and Daniel B. Szyld, editors, *Web Information Retrieval and Linear Algebra Algorithms*, number 07071 in Dagstuhl Seminar Proceedings, Dagstuhl, Germany, 2007. Internationales Begegnungs- und Forschungszentrum für Informatik (IBFI), Schloss Dagstuhl, Germany. [cited at p. 4, 5, 82, 89, 91, 129]

[Kuh55] H. W. Kuhn. The Hungarian method for the assignment problem. *Naval Res. Logist. Quart.*, 2:83–97, 1955. [cited at p. 71]

[LCL04] Ying-Hung Lin, Hsun-Chang Chang, and Yaw-Ling Lin. A study on tools and algorithms for 3-d protein structures alignment and comparison. In *International Computer Symposium*, 2004. [cited at p. 3, 70]

[LD03] X. S. Li and J. W. Demmel. SuperLU_DIST: A Scalable Distributed-memory Sparse Direct Solver for Unsymmetric linear systems. *ACM Transactions on Mathematical Software*, 29(2), 2003. [cited at p. 3, 71]

[LMS01] G. L. Litvinov, V. P. Maslov, and G. B. Shpiz. Idempotent functional analysis: an algebraic approach. *Math. Notes*, 69(5):696–729, 2001. [cited at p. 1]

[LO94] Y. Lee and J. B. Orlin. On very large scale assignment problems. In *Large scale optimization (Gainesville, FL, 1993)*, pages 206–244. Kluwer Acad. Publ., Dordrecht, 1994. [cited at p. 4, 81]

[LSW00] N. Linial, A. Samorodnitsky, and A. Wigderson. A deterministic strongly polynomial algorithm for matrix scaling and approximate permanents. *Combinatorica*, 20(4):545–568, 2000. [cited at p. 81]

[Mar] Thomas Markwig. A field of generalised puiseux series for tropical geometry. to appear in Rend. Semin. Mat. Torino (2009), see also arXiv:0709.3784. [cited at p. 76]

[Mar66] Morris Marden. *Geometry of polynomials*. Second edition. Mathematical Surveys, No. 3. American Mathematical Society, Providence, R.I., 1966. [cited at p. 2]

[McE06] William M. McEneaney. *Max-plus methods for nonlinear control and estimation*. Systems & Control: Foundations & Applications. Birkhäuser Boston Inc., Boston, MA, 2006. [cited at p. 1]

[Mik05] G. Mikhalkin. Enumerative tropical algebraic geometry in $\mathbb{R}^2$. *J. Amer. Math. Soc.*, 18(2):313–377 (electronic), 2005. [cited at p. 2]

[Mir60] L. Mirsky. Symmetric gauge functions and unitarily invariant norms. *Quart. J. Math. Oxford Ser. (2)*, 11:50–59, 1960. [cited at p. 84]

[MMMM06] D. Steven Mackey, Niloufer Mackey, Christian Mehl, and Volker Mehrmann. Vector spaces of linearizations for matrix polynomials. *SIAM J. Matrix Anal. Appl.*, 28(4):971–1004, 2006. [cited at p. 35]

[MPV87] M. Mezard, G. Parisi, and M. Virasoro. *Spin Glass Theory and Beyond (World Scientific Lecture Notes in Physics, Vol 9)*. World Scientific Publishing Company, 1987. [cited at p. 4, 70, 81]

[MS69] M. V. Menon and Hans Schneider. The spectrum of a nonlinear operator associated with a matrix. *Linear Algebra and Appl.*, 2:321–334, 1969. [cited at p. 72]

[MS73] C. B. Moler and G. W. Stewart. An algorithm for generalized matrix eigenvalue problems. *SIAM J. Numer. Anal.*, 10:241–256, 1973. [cited at p. 2, 34]

[MS92] V. P. Maslov and S. N. Samborskiĭ, editors. *Idempotent analysis*, volume 13 of *Advances in Soviet Mathematics*. Amer. Math. Soc., Providence, RI, 1992. [cited at p. 1]

[Mur90] Kazuo Murota. Computing puiseux-series solutions to determinantal equations via combinatorial relaxation. *SIAM J. Comput.*, 19(6):1132–1161, 1990. [cited at p. 16]

[MZ01] G. Malajovich and J. P. Zubelli. Tangent Graeffe iteration. *Numer. Math.*, 89(4):749–782, 2001. [cited at p. 86]

[NN94] Y. Nesterov and A. Nemirovskii. *Interior-point polynomial algorithms in convex programming*, volume 13 of *SIAM Studies in Applied Mathematics*. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 1994. [cited at p. 77]

[ON96] Markus Olschowka and Arnold Neumaier. A new pivoting strategy for Gaussian elimination. *Linear Algebra Appl.*, 240:131–151, 1996. [cited at p. 3, 71]

[Ost40a] Alexandre Ostrowski. Recherches sur la méthode de Graeffe et les zéros des polynomes et des séries de Laurent. *Acta Math.*, 72:99–155, 1940. [cited at p. 2, 19]

[Ost40b] Alexandre Ostrowski. Recherches sur la méthode de Graeffe et les zéros des polynomes et des séries de Laurent. Chapitres III et IV. *Acta Math.*, 72:157–257, 1940. [cited at p. 2, 19]

[Pan97] Victor Y. Pan. Solving a polynomial equation: Some history and recent progress. *SIAM Rev.*, 39:187–220, June 1997. [cited at p. 18]

[Par02] G. Parisi. Euclidean random matrices, the glass transition and the boson peak. *The European Physical Journal E: Soft Matter and Biological Physics*, 9(3):213–218, November 2002. [cited at p. 88]

[Pin98] Jean-Éric Pin. Tropical semirings. In Jeremy Gunawardena, editor, *Idempotency*, pages 50–69. Cambridge University Press, 1998. [cited at p. 1]

[PT05] Mikael Passare and August Tsikh. Amoebas: their spines and their contours. In *Idempotent mathematics and mathematical physics*, volume 377 of *Contemp. Math.*, pages 275–288. Amer. Math. Soc., Providence, RI, 2005. [cited at p. 34]

[RGST05] Jürgen Richter-Gebert, Bernd Sturmfels, and Thorsten Theobald. First steps in tropical geometry. In *Idempotent mathematics and mathematical physics*, volume 377 of *Contemp. Math.*, pages 289–317. Amer. Math. Soc., Providence, RI, 2005. [cited at p. 2]

[RW98] R.T. Rockafellar and R.J.B. Wets. *Variational analysis*. Number v. 317 in Grundlehren der mathematischen Wissenschaften. Springer, 1998. [cited at p. 56]

[Sch82] A. Schönhage. The fundamental theorem of algebra in terms of computational complexity. Technical report, Univ. Tübingen, 1982. 73 pages. [cited at p. 18]

[Sch89] Michael H. Schneider. Matrix scaling, entropy minimization, and conjugate duality. I. Existence conditions. *Linear Algebra Appl.*, 114/115:785–813, 1989. [cited at p. 73]

[SG76] Sartaj Sahni and Teofilo Gonzalez. P-complete approximation problems. *J. Assoc. Comput. Mach.*, 23(3):555–565, 1976. [cited at p. 70]

[Sim78] Imre Simon. Limited subsets of a free monoid. In *Proceedings of the 19th Annual Symposium on Foundations of Computer Science*, pages 143–150, Washington, DC, USA, 1978. IEEE Computer Society. [cited at p. 1]

[SK67] Richard Sinkhorn and Paul Knopp. Concerning nonnegative matrices and doubly stochastic matrices. *Pacific J. Math.*, 21:343–348, 1967. [cited at p. 4, 5, 72, 76, 80, 82, 83]

[Spe38] W. Specht. Zur theorie der algebraischen gleichungen. *Jahr. DMV*, 48:142–145, 1938. [cited at p. 2, 19]

[SW49] Claude E. Shannon and Warren Weaver. *The Mathematical Theory of Communication*. The University of Illinois Press, Urbana, Ill., 1949. [cited at p. 72]

[SZ90] Michael H. Schneider and Stavros A. Zenios. A comparative study of algorithms for matrix balancing. *Oper. Res.*, 38:439–455, May 1990. [cited at p. 72]

[Tar51] Alfred Tarski. *A decision method for elementary algebra and geometry*. University of California Press, Berkeley and Los Angeles, Calif., 1951. 2nd ed. [cited at p. 76]

[TDM10] Fernando De Teran, Froilan M. Dopico, and D. Steven Mackey. Fiedler companion linearizations and the recovery of minimal indices. *Siam Journal on Matrix Analysis and Applications*, 31, 2010. [cited at p. 3]

[Tis00] Françoise Tisseur. Backward error and condition of polynomial eigenvalue problems. *Linear Algebra Appl.*, 309(1-3):339–361, 2000. Proceedings of the International Workshop on Accurate Solution of Eigenvalue Problems (University Park,PA, 1998). [cited at p. 3, 34, 35]

[VAL14] G. VALIRON. *Sur les fonctions entières d'ordre nul et d'ordre fini et en particulier les fonctions à correspondance régulière*. PhD thesis, Thèse de Paris, 1914. Réimprimé dans les annales de la faculté des sciences de Toulous. [cited at p. 20]

[Vir01] O. Viro. Dequantization of real algebraic geometry on logarithmic paper. In *European Congress of Mathematics, Vol. I (Barcelona, 2000)*, volume 201 of *Progr. Math.*, pages 135–146. Birkhäuser, Basel, 2001. [cited at p. 2, 77]

[Vor67] N.N. Vorobyev. Extremal algebra of positive matrices. *Elektron. Informationsverarbeitung und Kybernetik*, 3:39–71, 1967. in Russian. [cited at p. 2]

[Zim77] K. Zimmermann. A general separation theorem in extremal algebras. *Ekonom.-Mat. Obzor*, 13(2):179–201, 1977. [cited at p. 2]

Appendices

APPENDIX A

Computing the tropical roots in linear time

The following code is a Scilab implementation of an algorithm, which computes the tropical roots of a given polynomial, $p(x) = \sum_{i=0}^{n} a_i x^i$, in $O(n)$.

The *tropical_root* function gets a polynomial p as an input and call *newton_polygon* function to compute the Newton polygon of the input. The latter function, calls *calcpoints* function to provide a list of points for which the convex hull should be computed. Next, this list will be sent to *graham_scan* function, which computes the convex polygon of the polynomial. Since the input set of points are sorted, the computation will be done in $O(n)$ instead of $O(n \log(n))$. There is as well the *newton_polygon_demo* function in the code, which can be used to plot the Newton polygon for a given polynomial.

```
function y=tropical_roots(p)
//This function computes the tropical roots in linear time.
  points_list=newton_polygon(p,1);
  if size(points_list,:)<2 then
    error('The number of monomials is less than two!');
  end,
    coefl=abs(coeff(p));
  for i= 1:(size(points_list,:)-1) do
     tr=(coefl(points_list(i,1)+1)/coefl(points_list((i+1),1)+1))
         ^(-1/(points_list(i,1)-points_list(i+1,1)));
```

```
      templist1(i,:)=[points_list(i+1,1)-points_list(i,1),tr];
  end,
  y=templist1;
endfunction

function y=newton_polygon(p,upper_lower)
  pointlist=calcpoints(p);
  if upper_lower==0 | upper_lower==1 then
     y=graham_scan(pointlist,upper_lower);
  else
     error('Invalid Input Data');
  end,
endfunction

function y=calcpoints(p)
    j=1;
    templist1=coeff(p);
    if length(templist1) < 2 then
    error('The number of monomials is less than two!');
  end,
    for i= 1:length(templist1) do
        if  templist1(i) <> 0 then
           templog= log(abs(templist1(i)));
           templist2(j,:)=[i-1,templog];
           j=j+1;
        end,
    end,
    y=templist2;
endfunction


function y=graham_scan(listofpoints,upper_lower)
  //if upper_lower=1 then the upper boundary of the convex hull will
      be computed,
  //if it is zero then the the lower boundary of the convex hull will
       be computed
   n=size(listofpoints,:);
   if n<2 then
    error('This Polynomial has less than two monomials');
   elseif n==2 then
      y=listofpoints;
      return;
   else
      if upper_lower==0 | upper_lower==1 then
         if upper_lower==1 then
           direction=-1;
         else
           direction=1;
         end,
         ptlist=upperlowerline(listofpoints,upper_lower);
         if size(ptlist,:) == 2 then
```

```
        y=ptlist;
        return;
      end,
      stackpointer=1;
      points_stack(stackpointer,:)=ptlist(1,:);
      firstp=ptlist(2,:);
      secondp=ptlist(1,:);
      for i= 3:size(ptlist,:) do
         turnp=((ptlist(i,1)-secondp(1))*(firstp(2)-secondp(2)))
             -((firstp(1)-secondp(1))*(ptlist(i,2)-secondp(2)));
         while (turnp*direction)>0 & stackpointer>1 do
              points_stack(stackpointer,:)=[-1,-1];
              stackpointer=stackpointer-1;
              firstp=secondp;
              secondp=points_stack(stackpointer,:);
              turnp=(ptlist(i,1)-secondp(1))*(firstp(2)-secondp(2))
                  -(firstp(1)-secondp(1))*(ptlist(i,2)-secondp(2));
         end,
         if (turnp*direction)>0 & stackpointer==1 then
              firstp=ptlist(i,:);
         else
              stackpointer=stackpointer+1;
              points_stack(stackpointer,:)=firstp;
              secondp=firstp;
              firstp=ptlist(i,:);
         end,
      end,
      stackpointer=stackpointer+1;
      points_stack(stackpointer,:)=firstp;
      j=stackpointer;
      while stackpointer<>0 do
        tlist1(j,:)=points_stack(stackpointer,:);
        j=j-1;
        points_stack(stackpointer,:)=[-1,-1];
        stackpointer=stackpointer-1;
      end,
      finallist2(1,:)=tlist1(1,:);
j=2;
k=size(tlist1,:);
      for i= 2:(k-1) do
         if ((tlist1(i,2)-tlist1((i-1),2))*(tlist1((i+1),1)-tlist1
             (i,1)))<>((tlist1((i+1),2)-tlist1(i,2))*(tlist1(i,1)-
             tlist1((i-1),1))) then
              finallist2(j,:)=tlist1(i,:);
              j=j+1;
         end,
      end,
      finallist2(j,:)=tlist1(k,:);
      y=finallist2;
  else
      error('Invalid Input Data');
```

```
      end,
  end,
endfunction

function y=upperlowerline(pointslist,upperlower)
  i=2;
  n=size(pointslist,:);
  a=pointslist(1,:);
  b=pointslist(n,:);
  templist(1,:)=a;
  m=(b(2)-a(2))/(b(1)-a(1));
  for j= 2:(n-1) do
     yc=a(2)+ (m*(pointslist(j,1)-a(1)));
     if yc <= pointslist(j,2) & upperlower==1 then
        templist(i,:)=pointslist(j,:);
        i=i+1;
     elseif yc >= pointslist(j,2) & upperlower == 0 then
        templist(i,:)=pointslist(j,:);
        i=i+1;
     end,
   end,
   templist(i,:)=b;
   y=templist;
endfunction;

function newton_polygon_demo(p,upper_lower)
  pointlist=newton_polygon(p,upper_lower);
  all_points=calcpoints(p);
  plot(all_points(:,1),all_points(:,2),'r.');
  plot(pointlist(:,1),pointlist(:,2),'b');
  return;
endfunction
```

APPENDIX B

The implementation of the tropical scaling for the matrix eigenvalue problem

Here we present a Matlab implementation of the tropical scaling for a quadratic matrix eigenvalue problem and a Scilab implementation of the tropical scaling in the general case i.e. a matrix polynomial with an arbitrary degree. Table B.1 demonstrates the list of all Matlab functions. In the Matlab code, $a0, a1, a2$ respectively denote A_0, A_1 and A_2 and for a quadratic matrix polynomial, $p(\lambda) = A_0 + A_1\lambda + A_2\lambda^2$. Also, $gama0, gama1, gama2$ respectively denote the Euclidean norm of A_0, A_1, A_2.

Table B.2 presents the name of functions, input and output variables and a short description of each function, which is used in the Scilab implementation. The data format that we use to store a matrix polynomial, $p(\lambda) = A_0 + \ldots + A_d\lambda^d$, is an $n \times n(d+1)$ matrix, $a = [A_0, A_1, A_2, ..., A_d]$, where n is the dimension of any matrix, A_i. The format of other variables are explained in Table B.2

Listing B.1: Matlab code for the tropical scaling of a quadratic matrix polynomial

```
function [elistp,vlist1p,vlist2p,backerr1p,backerr2p]=alphaplus(a2,a1
    ,a0,gama2,gama1,gama0)
```

```
  if gama1^2>gama0*gama2
    alpha1=max(gama1/gama2,(gama0/gama2)^(1/2));
    d2=gama2/(gama1^2);
    ahatp0=d2*a0;
    ahatp1=d2*alpha1*a1;
    ahatp2=d2*(alpha1^2)*a2;
    [A,E]=constpen(ahatp2,ahatp1,ahatp0);
    [elistp,vlist1p,vlist2p]=eigens(A,E);
    elistp=alpha1*elistp;
    backerr1p=backwarderr(a2,a1,a0,elistp,vlist1p,gama2,gama1,gama0);
    backerr2p=backwarderr(a2,a1,a0,elistp,vlist2p,gama2,gama1,gama0);
  else
    elistp=0;
    vlist1p=0;
    vlist2p=0;
    backerr1p=0;
    backerr2p=0;
  end;
end

function [elistm,vlist1m,vlist2m,backerr1m,backerr2m]=alphaminus(a2,
    a1,a0,gama2,gama1,gama0)
  if gama1^2>gama0*gama2
    alpha2=min(gama0/gama1,(gama0/gama2)^(1/2));
    d2=1/gama0;
    ahatm0=d2*a0;
    ahatm1=d2*alpha2*a1;
    ahatm2=d2*(alpha2^2)*a2;
    [A,E]=constpen(ahatm2,ahatm1,ahatm0);
    [elistm,vlist1m,vlist2m]=eigens(A,E);
    elistm=alpha2*elistm;
    backerr1m=backwarderr(a2,a1,a0,elistm,vlist1m,gama2,gama1,gama0);
    backerr2m=backwarderr(a2,a1,a0,elistm,vlist2m,gama2,gama1,gama0);
  else
    elistm=0;
    vlist1m=0;
    vlist2m=0;
    backerr1m=0;
    backerr2m=0;
  end;
end

function [A,E]=constpen(a2,a1,a0)
   tmpdima=size(a2);
   dima=tmpdima(1);
   A=[zeros(dima,dima),eye(dima,dima);-a0,-a1];
   E=[eye(dima,dima),zeros(dima,dima);zeros(dima,dima),a2];
end


function [eign1,vect1,vect2]=eigens(A,E)
```

```
  n=size(A,2)/2;
  [v,d]=eig(A,E);
  eign1=eig(A,E);
  vect1=v(1:size(v,1)/2,1:size(v,2));
  vect2=v(size(v,1)/2+1:size(v,1),1:size(v,2));
end

function y=backwarderr(a2,a1,a0,e,v,gama2,gama1,gama0)
  for i=1:length(e)
    p=(e(i)^2)*a2+e(i)*a1+a0;
    egvet=p*v(:,i);
    egval=abs(e(i));
    backerr1(i,1)=norm(egvet,2)/(((egval^2)*gama2+egval*gama1+gama0)*
        norm(v(:,i),2));
  end
  y=backerr1;
end
```

Listing B.2: Scilab code of the tropical scaling for a given matrix polynomial with an arbitrary degree

```
function [eign1_sorted,seign1,backwithoutscal_sorted,backscal]=gexamp
    (a)
  d=(size(a,2)/size(a,1))-1;
  [seign1,seigenv,infin]=tropscal(a);
  [A,E]=gconstpen(a);
  [eign1,vect]=geigens(A,E,d);
  [e,eign1_sortedindex]=gsort(abs(eign1),'r','d');
  backwithoutscal=gpencilbackerror(a,eign1,vect);
  for i=1:size(eign1,1)
    backwithoutscal_sorted(i)=backwithoutscal(eign1_sortedindex(i));
    eign1_sorted(i)=eign1(eign1_sortedindex(i));
  end;
  backscal=gpencilbackerror(a,seign1,seigenv);
endfunction

function [eign1,eigenv]=tropscal(a)
  infin=%F;
  ind=0;
  n=size(a,1);
  pdegree=(size(a,2)/n)-1;
  aj=zeros(n,(pdegree+1)*n);
  y=zeros(n*pdegree,1);
  troplist=troppencilroots(a);
  numtroproots=size(troplist,1);
  for i=1:numtroproots
    tropicalroot=troplist(numtroproots-i+1,2);
    multipl_troot=troplist(numtroproots-i+1,1);
    scalingalphai=1/troplist(numtroproots-i+1,3);
    for j=0:pdegree
```

```
      aj(:,j*n+1:(j+1)*n)=scalingalphai*(tropicalroot^j)*a(:,j*n+1:(j
          +1)*n);
    end;
    [As,Es]=gconstpen(aj);
    [seigenv,seigenvect]=geigens(As,Es,pdegree);
    seigenv=seigenv*tropicalroot;

    if isnan(mean(abs(seigenv))) then
      cleanlistv=cleannaninf(seigenv);
      [esorted,index1]=gsort(abs(cleanlistv(:,1)),'r','d');
      flist=eye(size(seigenv,1)-size(cleanlistv,1),1)*%nan;
      evectorlist=zeros(size(seigenv,1),size(seigenv,1)-size(
          cleanlistv,1));
      for tmr=1: size(cleanlistv,1)
         indindex=abs(cleanlistv(index1(tmr),2));
         flist(size(seigenv,1)-size(cleanlistv,1)+tmr)= seigenv(
             indindex);
         evectorlist(:,size(seigenv,1)-size(cleanlistv,1)+tmr)=
             seigenvect(:,indindex);
      end;
      eign1(ind*n+1:(ind+multipl_troot)*n)=flist(ind*n+1:(ind+
          multipl_troot)*n);
      eigenv(1:n*pdegree,ind*n+1:(ind+multipl_troot)*n)=seigenvect(:,
          ind*n+1:(ind+multipl_troot)*n);
      infin=%T;
    else
      [seigenv_abs,sortl]=gsort(abs(seigenv),'r','d');
      eign1(ind*n+1:(ind+multipl_troot)*n)=seigenv(sortl(ind*n+1:(ind
          +multipl_troot)*n));
      eigenv(1:n*pdegree,ind*n+1:(ind+multipl_troot)*n)=seigenvect(:,
          sortl(ind*n+1:(ind+multipl_troot)*n));
    end;
    ind=ind+multipl_troot;
  end;
endfunction;

function [A,E]=gconstpen(a)
   t=size(a,2);
   n=size(a,1);
   pdegree=(size(a,2)/n)-1;
   a2=zeros(a);
   for i=0:pdegree
    a2(:,i*n+1:i*n+n)=a(:,(pdegree-i)*n+1:(pdegree-i+1)*n);
   end;
   A=diag(ones((pdegree-1)*n,1),-n);
   A(1:n,:)=-a2(:,n+1:(pdegree+1)*n);
   E=eye(n*pdegree,n*pdegree);
   E(1:n,1:n)=a(:,pdegree*n+1:(pdegree+1)*n);
endfunction

function [eign1,vect]=geigens(A,E,d)
```

```
  n=size(A,2)/d;
  [temp1,temp2,temp3,temp4]=spec(A,E);
  if min(abs(temp2))==0 then
    eign1=spec(A,E);
  else
    for i=1:size(temp1,1)
      eign1(i)=temp1(i)/temp2(i);
    end;
  end;
  vect=temp4;
endfunction

function y=valp(a,l)
  n=size(a,1);
  y=zeros(n,n);
  pdegree=(size(a,2)/n)-1;
  for i=1: pdegree+1
    mtemp=a(1:n,1+(i-1)*n:(i*n));
    y=y+l^(i-1)*mtemp;
  end;
endfunction;

function y=gpencilbackerror(a,e,v)
  dim=size(e,1);
  n=size(a,1);
  pdegree=(size(a,2)/n)-1;
  for i=1:pdegree+1
    mtemp=a(1:n,1+(i-1)*n:(i*n));
    vectornorms(i)=norm(mtemp,2);
  end;
  for i=1:size(e,1)
    pencilval=valp(a,e(i));
    eigenvector=v(1:n,i);
    r=norm(pencilval*eigenvector,2);
    eigenvalabs=abs(e(i));
    al=0;
    for j=1:pdegree+1
      al=al+vectornorms(j)*eigenvalabs^(j-1);
    end;
    if al==0 then
      error('al is zero');
    elseif norm(eigenvector,2)==0 then
      warning('eigenvector is zero');
      y(i)=%inf;
    else
      y(i)=r/(al*norm(eigenvector,2));
    end;
  end;
endfunction;

function y=troppencilroots(a)
```

```
  n=size(a,1);
  pdegree=(size(a,2)/n)-1;
  for i=1: pdegree+1
    mtemp=a(1:n,1+(i-1)*n:(i*n));
    coefl(i)=norm(mtemp,2);
  end;
  p=poly(coefl,"x","coeff");
  y=tropical_roots2(p);
endfunction;

function y=tropical_roots2(p)
  points_list=newton_polygon(p,1);
  if size(points_list,:)<2 then
    error('Numebr of monomials is less than two!');
  end,
    coefl=abs(coeff(p));
  for i= 1:(size(points_list,:)-1) do
     tr=(coefl(points_list(i,1)+1)/coefl(points_list((i+1),1)+1))
         ^(-1/(points_list(i,1)-points_list(i+1,1)));
     p_alphai=coefl(points_list(i,1)+1)*tr^points_list(i,1);
     templist1(i,:)=[points_list(i+1,1)-points_list(i,1),tr,p_alphai
         ];
  end,
  y=templist1;
endfunction

function y=cleannaninf(v)
  n=size(v,1);
  j=1;
  for i=1:n
    if ~isnan(v(i)) & ~isinf(v(i)) then
      y(j,:)=[v(i),i];
      j=j+1;
    end;
  end;
endfunction;
```

Table B.1: The List of Matlab functions for the tropical scaling of a quadratic matrix eigenvalue problem.

Function name	**Description**	**Input para.**	**Output para.**
alphaplus	Computes the eigenvalues, eigenvectors and backward error by using the largest tropical root	**a2,a1,a0:** The input quadratic matrix polynomial **gama2, gama1, gama0:** Corresponding norm 2 of a0,a1,a2	**elistp:** List of the eigenvalues **vlist1p:** Matrix of the eigenvectors **vlist2p:** Matrix of the eigenvectors **backerr1p:** List of the backward errors corresponding to the eigenvalues and vlist1p **backerr2p:** List of the backward errors corresponding to the eigenvalues and vlist2p
alphaminus	Computes the eigenvalues, eigenvectors and backward error by using the smallest tropical root	**a2,a1,a0:** The input quadratic matrix polynomial **gama2, gama1, gama0:** Corresponding norm 2 of a0,a1,a2	**elistm:** List of the eigenvalues **vlist1m:** Matrix of the eigenvectors **vlist2m:** Matrix of the eigenvectors **backerr1m:** List of the backward errors corresponding to eigenvalues and vlist1m **backerr2m:** List of the backward errors corresponding to eigenvalues and vlistm
constpen	Converts a quadratic matrix polynomial to a pencil	**a2,a1,a0:** The input quadratic matrix polynomial	**A,E:** The pencil matrices
eigens	Returns the list of the eigenvalues, and two matrices of eigenvectors	**A,E:** The pencil input matrices	**eign1:** List of the eigenvalues **vect1:** Matrix of the eigenvectors **vect2:** Matrix of the eigenvectors
backwarderr	Computes the backward error for a given quadratic matrix polynomial	**a2,a1,a0:** The input quadratic matrix polynomial **gama2, gama1, gama0:** Corresponding norm 2 of a0,a1,a2	**y:** List of the backward errors corresponding to the all eigenvalues

Table B.2: The List of Scilab functions for the tropical scaling of a matrix eigenvalue problem.

Function name	Description	Input para.	Output para.
gexamp	For a matrix polynomial, computes the list of the eigenvalues and backward errors by using the scaling and without using the scaling	**a:** polynomial matrix	**eign1_sorted:** list of the eigenvalues without using the scaling **seign1:** list of the eigenvalues by using the scaling **backwithoutscal_sorted:** list of the backward errors for the eigenvalues without using the scaling **backscal:** list of the backward errors by using the scaling
tropscal	Computes the eigenvalues and eigenvectors of a given polynomial matrix by using the tropical scaling	**a:** polynomial matrix	**eign1:** list of the eigenvalues by using the tropical scaling **eigenv:** the matrix of the eigenvectors for the corresponding eigenvalues
gconstpen	Converts the polynomial eigenvalue to a pencil	**a:** matrix polynomial	**A,E:** matrices of dimension nd where n is dimension of A_i and d is the degree of polynomial
geigens	Returns the eigenvalues and eigenvectors of a given pencil	**A,E:** the pencil matrices **d:**degree of polynomial matrix	**eign1:** list of the eigenvalues **vect:** right eigenvector of the pencil
valp	Computes the value of a pencil for a given λ	**a:** matrix polynomial	**y:** a matrix of dimension n
gpencilbackerror	Computes the eigenvalues, the eigenvectors and the backward errors for a given matrix polynomial	**a:** matrix polynomial **e:** the list of eigenvalues **v:** matrix of the eigenvectors	**y:** list of the backward errors computed for each eigenvalue
troppencilroots	Computes the corresponding tropical roots of a matrix polynomial	**a:** matrix polynomial	**y:** a matrix of three columns for which the first column presents the multiplicity of tropical roots, the second column present the value of these roots and the third column presents the value of max-times polynomial for the corresponding tropical root
tropical_roots2	Computes the tropical roots of a given polynomial by calling Newton_polygon function presented in A	**p:** a given polynomial	**y:** list of the tropical roots, multiplicities and values of p for each tropical root
cleannaninf	Delete all non or inf values for a given vector	**v:** input vector	**y:** output vector

APPENDIX C

Computing the tropical eigenvalues of a max-plus matrix polynomial

The following is a Scilab implementation to compute the tropical eigenvalues of a max-plus matrix polynomial. Here, the main function is *tropical_eigenvalues* which takes a max-plus matrix polynomial , $\mathsf{t}p(\lambda) = A_0 \oplus \lambda A_1 \oplus \ldots \oplus \lambda^d A_d$ in the form of $a = [A_0, A_1, \ldots, A_d]$ as an input, and returns the tropical eigenvalues. Recall that the tropical eigenvalues are the tropical roots of the function $\mathrm{maxper}(\mathsf{t}p(\lambda))$ defined in Equations 5.2 and 5.2. We use the max-plus toolbox of Scilab which is developed by M. McGettrick, G. Cohen, S. Gaubert, and J.-P. Quadrat*. We also use an external function, *perm5*, to solve an optimal assignment problem. This function, takes a max-plus matrix and returns the Hungarian-pairs, the value of the optimal assignment, and the optimal permutation. perm5 is a Scilab interface of a C function, *CS2*, which is developed by A. Goldberg and B. Cherkassky [Gol97] to solve Minimum-Cost Flow problem.

In the code, there are three other subfunctions which are called by the function *tropical_eigenvalues*. The following is a short description of these functions.

* This toolbox is downloadable from http://amadeus.inria.fr/gaubert/MaxplusToolbox.html

- *value_matpol()*: Takes a max-plus matrix polynomial, a, and a scalar value l and computes the value of a at the point, l.

- *pencil_val_degree()*: For a max-plus matrix polynomial $\mathrm{t}p(\lambda) = A_0 \oplus \ldots \oplus \lambda^d A_d$, it computes the degree, v_t and the valuation, v_0, of the function $f(\lambda) = \mathrm{maxper}(\mathrm{t}p(\lambda)) = \delta_0 \otimes \lambda^{v_0} \oplus \ldots \oplus \delta_t \otimes \lambda^{v_t}$ which is defined in Equations 5.2 and 5.2. It also computes δ_0 and δ_t, the coefficients of the terms with minimum and maximum degree, respectively.

- *compute_right_left_driv()*: Takes a max-plus polynomial matrix a, and a scalar, l, and computes the right and the left derivatives for the given point, l. For more details see Section 5.3.

```
function pl=value_matpol(a,l)
  n=size(a,1);
  pdegree=(size(a,2)/n)-1;
  l=#(l);
  for i=1: pdegree+1
    t=a(1:n,1+(i-1)*n:(i*n));
    if i==1
      pl=t;
    else
      t=((l)^(i-1))*t;
      pl=pl+t;
    end
  end
endfunction

function [pen_deg,pen_val,dcoff,vcoff]=pencil_val_degree(a)
  n=size(a,1);
  pdegree=(size(a,2)/n)-1;
  for i=1: pdegree+1
    t=a(1:n,1+(i-1)*n:(i*n));
    if i>1
      t(t>%0)=#(i-1);
      pt_deg=pt_deg+t;
      pt_val=pt_val+(-t);
    else
      t(t>%0)=%1;
      pt_deg=t;
      pt_val=t;
    end
  end;
  clear t;
  [pen_deg,perm,u,v]=perm5(pt_deg);
  t=(pt_deg==#(diag(u)*ones(n,n)+ones(n,n)*diag(v)));
  for i=1:n
    for j=1:n
      if t(i,j)
```

```
        coef_deg(i,j)=a(i,plustimes(pt_deg(i,j))*n+j);
      else
        coef_deg(i,j)=%0;
      end
    end
  end
  [pen_val,perm,u,v]=perm5(pt_val);
  t=(pt_val==#(diag(u)*ones(n,n)+ones(n,n)*diag(v)));
  for i=1:n
    for j=1:n
      if t(i,j)
        coef_val(i,j)=#(a(i,-plustimes(pt_val(i,j))*n+j));
      else
        coef_val(i,j)=%0;
      end
    end
  end
  dcoff=perm5(coef_deg);   vcoff=perm5(coef_val);
  pen_deg=round(pen_deg);   pen_val=-round(pen_val);
endfunction;

function [rdriv,ldriv]=compute_right_left_driv(a,l)
  precision=10^(-12)
  l=#(l);
  n=size(a,1);
  d=(size(a,2)/n)-1;
  pl=value_matpol(a,l);
  [val,perm,u,v]=perm5(pl);
  t=abs(plustimes(pl)-(diag(u)*ones(n,n)+ones(n,n)*diag(v)))<
      precision
  for i=1:n
    for j=1:n
      if t(i,j)
        maxk=d;
        while ((a(i,maxk*n+j)==%0) | (a(i,maxk*n+j)*(l)^maxk~=pl(i,j)
            ))
          maxk=maxk-1;
        end
        mink=0;
        while ((a(i,mink*n+j)==%0)   | (a(i,mink*n+j)*(l)^mink~=pl(i,j
            )))
          mink=mink+1;
        end
        maxm(i,j)=maxk;
        minm(i,j)=mink;
      else
        maxm(i,j)=%0;
        minm(i,j)=%0;
      end
    end
  end
```

```
  rdriv=round(perm5(#(maxm)));
  ldriv=round(-perm5(#(-minm)));
endfunction

function [pval,pdeg,trop_list]=tropical_eigenvalues(a)
  precision=10^(-8);
  k=1;
  [pdeg,pval,dcoff,vcoff]=pencil_val_degree(a);
  l=(vcoff-dcoff)/(pdeg-pval);
  list1(1,:)=[l,pval,vcoff,pdeg,dcoff];
  while size(list1,1)>0
    l=list1(1,1);
    vl=list1(1,2);
    cl=list1(1,3);
    vr=list1(1,4);
    cr=list1(1,5);
    list1(1,:)=[];
    lsize=size(list1,1);
    if vr==vl+1
      if abs(l-round(l))<precision
        l=round(l);
      end;
      trop_list(k,:)=[l,1];
      k=k+1;
    else
      pl=value_matpol(a,l);
      fl=perm5(pl);
      if abs(fl-(cl+vl*l))<precision
        if abs(l-round(l))<precision
          l=round(l);
        end;
        trop_list(k,:)=[l,vr-vl];
        k=k+1;
      else
        [rdriv,ldriv]=compute_right_left_driv(a,l);
        if ldriv<rdriv
          trop_list(k,:)=[l,rdriv-ldriv];
          k=k+1;
        end
        cl2=fl-ldriv*l;
        cr2=fl-rdriv*l;
        list1(lsize+1,:)=[(cl-cl2)/(ldriv-vl),vl,cl,ldriv,cl2];
        list1(lsize+2,:)=[(cr-cr2)/(rdriv-vr),rdriv,cr2,vr,cr];
      end
    end
  end
endfunction
```

APPENDIX D

Newton Algorithm to compute the diagonal scaling matrices

The following code is the Matlab implementation of the Newton method, which is appeared in the work of Knight and Ruiz [KR07]. The input matrix is a non-negative symmetric matrix and in each Newton step a linear system of equations should be solved for which the Conjugate Gradient method is used. If the original matrix, A, is nonsymmetric then the input matrix can be computed as

$$S = \begin{pmatrix} 0 & A \\ A^T & 0 \end{pmatrix} .$$

In this case, the linear system, which should be solved in each Newton step is singular, however it is proved that the system is consistent whenever A has support ($A \geq 0$ has support if it has a positive diagonal).

```
function [x,res] = bnewt(A,tol,x0,delta,fl)
% BNEWT A balancing algorithm for symmetric matrices
%
% X = BNEWT(A) attempts to find a vector X such that
% diag(X)*A*diag(X) is close to doubly stochastic. A must
% be symmetric and nonnegative.
%
% X0: initial guess. TOL: error tolerance.
```

```
% DEL: how close balancing vectors can get to the edge of the
% positive cone. We use a relative measure on the size of elements.
% FL: intermediate convergence statistics on/off.
% RES: residual error, measured by norm(diag(x)*A*x - e).

% Initialise
[n,n]=size(A); e = ones(n,1); res=[];
if nargin < 5, fl = 0; end
if nargin < 4, delta = 0.1; end
if nargin < 3, x0 = e; end
if nargin < 2, tol = 1e-6; end
g=0.9; etamax = 0.1; % Parameters used in inner stopping criterion.
eta = etamax;

x = x0; rt = tol^2; v = x.*(A*x); rk = 1 - v;
rho_km1 = rk *rk; rout = rho_km1; rold = rout;

MVP = 0; % Well count matrix vector products.
i = 0; % Outer iteration count.

if fl == 1, fprintf( it in. it res\n ), end
while rout > rt % Outer iteration
  i = i + 1; k = 0; y = e;
  innertol = max([eta^2*rout,rt]);
  while rho_km1 > innertol %Inner iteration by CG
    k = k + 1;
    if k == 1
      Z = rk./v; p=Z; rho_km1 = rk *Z;
    else
      beta=rho_km1/rho_km2;
      p=Z + beta*p;
    end
    % Update search direction efficiently.
    w = x.*(A*(x.*p)) + v.*p;
    alpha = rho_km1/(p *w);
    ap = alpha*p;
    % Test distance to boundary of cone.
    ynew = y + ap;
    if min(ynew) <= delta
      if delta == 0, break, end
      ind = find(ap < 0);
      gamma = min((delta - y(ind))./ap(ind));
      y = y + gamma*ap;
      break
    end
    y = ynew;
    rk = rk - alpha*w; rho_km2 = rho_km1;
    Z = rk./v; rho_km1 = rk *Z;
  end
  x = x.*y; v = x.*(A*x);
  rk = 1 - v; rho_km1 = rk *rk; rout = rho_km1;
```

```
  MVP = MVP + k + 1;

  % Update inner iteration stopping criterion.
  rat = rout/rold; rold = rout; r_norm = sqrt(rout);
  eta_o = eta; eta = g*rat;
  if g*eta_o^2 > 0.1
    eta = max([eta,g*eta_o^2]);
  end
  eta = max([min([eta,etamax]),0.5*tol/r_norm]);

  if fl == 1
    fprintf( %3d %6d %.3e %.3e %.3e \n , i,k, r_norm,min(y),min(x));
    res=[res; r_norm];
  end
end
fprintf(Matrix-vector products = %6d\n , MVP)
```

List of Figures

List of Tables