h^*-POLYNOMIALS OF GRAPHICAL ZONOTOPES.

Master's thesis

Freie Universität Berlin
Fachbereich Mathematik und Informatik

Submitted by	Claudia Mitukiewicz
Matriculation number	5046954
Email	cmitukiewicz@zedat.fu-berlin.de
First supervisor	Prof. Dr. Matthias Beck
Second supervisor	Prof. Dr. Giulia Codenotti

Berlin, 18th June 2024

Contents

1. Introduction

The Ehrhart function, denoted by $\mathrm{ehr}_P(t)$, for a polytope P, records the number of integer lattice points within the t-th positive integer dilation of the polytope. If P is a lattice polytope, meaning that the coordinates of its vertices are integers, Ehrhart demonstrated in [10] that this function agrees with a polynomial, referred to as the Ehrhart polynomial of the polytope. The characterisation of Ehrhart polynomials is still a huge challenge, even in dimension 3. A crucial tool in this endeavor is the h^*-polynomial of a d-dimensional lattice polytope, which encodes its Ehrhart polynomial in the basis $\binom{t}{d}, \binom{t+1}{d}, \ldots, \binom{t+d}{d}$. We discuss Ehrhart's theory in more detail in Section 2.1.

Due to its extraordinary arithmetic properties, the graphical zonotope serves as a famous illustration for the application of Ehrhart theory. Zonotopes, broadly defined as polytopes resulting from the Minkowski sums of a finite number of line segments, were initially explored in 1969 by Bolker [5]. Zonotopes are prevalent across various mathematical domains. Apart from geometry and combinatorics, they contribute to fields such as approximation theory, optimization, and crystallography. The class of zonotopes being the object of interest of this thesis are graphical zonotopes defined as follows

Definition. Given a graph G, the **graphical zonotope** Z_G induced by G is a zonotope generated by all vectors $\mathbf{e}_i - \mathbf{e}_j$, where $\mathbf{e}_i$ and $\mathbf{e}_j$ are unit vectors and ij is an edge in G with $i > j$.

This definition and more background on graphical zonotopes can be found in Section 2.4. The goal of this thesis is to compute the h^*-polynomial of a graphical zonotope directly from the underlying graph.

Inspired by the paper of Matthias Beck, Katharina Jochemko and Emily Maccullough [2], we change the basis of the h^*-polynomial from the standard monomial basis to the (A, j)-Eulerian polynomial basis. The vector whose entries are the coefficients of this representation is called the c-vector. This thesis is not the first work introducing methods of computing the c-vector of a graphical zonotope. In the dissertation of Aaron Dall from 2015 [8] the c-vector of a graphical zonotope for connected graphs is investigated under the name h-vector of a graphical matroid. Dall proves that the coefficients of the c-vector of a graphical zonotope are the coefficients of the arithmetic Tutte polynomial of the graphical matroid evaluated at $y = 1$, i.e.,

$$\mathcal{T}_{\mathcal{A}}(x, 1) = \sum_{i=0}^{d} c_i x^{d-i}, \tag{1.1}$$

where $\mathcal{A}$ is a rank d arithmetic matroid represented by the integer vectors generating this graphical zonotope. A geometrical method of computing the coefficients of (1.1) is described in [13].

Dall's work and this thesis have in common that they are both based on the matroidal aspects of a graphical zonotope, see Section 2.5. There are two ways of representing the graphical zonotope in the world of matroids; first, a matroid whose ground set consists of the generating vectors of a zonotope and sets of linearly independent vectors are the independent sets; second, the graphical matroid of the underlying graph. This discussion

combined with the outcomes of [2] led us to the following theorem, which is the main result of this thesis.

Theorem 1. *The coefficient c_j of a c-vector of a graph G counts the spanning forests of G with $j-1$ internally passive edges.*

The internally passive edges of a spanning forest are those which can be replaced by a smaller edge without creating a cycle.

The role of the spanning trees in computing the c-vector of a graphical zonotope seems reasonable. Indeed, the fact that any zonotope can be decomposed into half-open parallelepipeds is very useful for determining the h^*-polynomial of a zonotope, and hence also its c-vector [2]. Moreover, there is an analogy between the half-open parallelepipeds in the decomposition of the zonotope and the spanning forests of the graph. Specifically, for every spanning forest in a graph, there is a half-open parallelepiped, whose number of missing facets is the number of the internally passive edges in this spanning forest.

Section 3.2 is devoted to proving that to compute the c-vector of a graph G it suffices to consider a simplified version, i.e., connected and without cut-edges, see Corollary 14. We call the result of such a modification of G the reduced graph of G and denote it by $\tilde{G}$. We hoped to come up with a closed formula allowing us to easily compute the number of spanning forests of a graph for every number of internally passive edges. However, this task turned out to be more difficult than expected. In the end, the closest we got to our aim is the following theorem, proved combinatorially in Section 3.3 and algebraically in Section 3.4.

Theorem 2. *Let G be a graph and Z_G a graphical zonotope on G with c-vector $(c_1, c_2, \ldots, c_d)$ and let $\tilde{G}$ be the reduced graph of G. Then*

1. *$c_1 = 1$.*
2. *c_2 is the cyclomatic number of $\tilde{G}$, i.e., $c_2 = |E(\tilde{G})| - |V(\tilde{G})| + 1$.*
3. *$c_3 = c_2 + \binom{c_2}{2}$.*
4. *$c_j > 0$ for $j \in [1, |V(\tilde{G})|]$, and $c_j = 0$ otherwise.*
5. *The sum of all entries of the c-vector equals the number of the spanning trees of $\tilde{G}$, which by Kirchhoff's theorem [12] is the product of the eigenvalues of the Laplacian of $\tilde{G}$ divided by $|V(\tilde{G})|$.*

Alternative proofs of Kirchhoff's theorem can be found in [1] and [8].

A closed formula for computing the j-th coefficient of the c-vector developed in this thesis is

$$c_j = \sum_{i=0}^{j-1} (-1)^{j-1-i} \binom{v-i-1}{j-i-1} b_i(G),$$

coming from the Ehrhart polynomial of a graphical zonotope, see Corollary 17. However, this way of computing the c-vector requires the knowledge of the number of subforests of G with i edges for $i \in \{0, 1, \ldots, j-1\}$, denoted by $b_i(G)$. Since this is not a trivial problem, we did not stop here and continued developing other methods of computing the c-vectors

of different graphs. For instance, if a graph G can be constructed by connecting graphs with known c-vectors, Section 3.5 equips us with Proposition 21, which is a practical tool for computing the c-vector of G.

In Chapter 4 we discuss the c-vectors for three special kinds of graphs. Section 4.1 is devoted to graphs with exactly one cycle. It turns out that all non-zero entries of their c-vectors are 1, see Proposition 23. An interesting aspect of computing the c-vectors of the two remaining classes of graphs is that they are related to two very well-known basic enumeration problems: weak compositions and integer partitions. The first of these are cactus graphs, which also form the simplest class of graphs that Proposition 21 can be used for. In Section 4.2 we then provide an example of an application of Proposition 21, but mainly, as mentioned before, we show in Proposition 24 that computing the c-vector of a cactus graph is equivalent to computing a special kind of restricted weak compositions. Interestingly enough, such restricted weak compositions are topics of many papers in Computer Science, such as [14]. We also propose an algorithm, which can be used for this problem (see Algorithm 4.1). The third kind of graphs, investigated in Section 4.3, are fan graphs. Even though fan graphs do not seem to have complicated structures, computing their c-vectors is quite challenging. This is mostly due to the fact that the simple cycles of fan graphs share edges. In the case of fan graphs, we divide the procedure of computing the c-vector into three parts. Section 4.3.1 introduces the concept of producing spanning trees from clusters, which are special subgraphs of fan graphs. Moreover, we prove Proposition 25, which tells us how many different spanning trees are produced from one cluster. In Section 4.3.2, we distinguish different types of clusters and show how to compute the number of clusters of each type. The most important result of Section 4.3.2 is Proposition 26, applying to arbitrary types of clusters, but we also develop methods which are practical for some special cases. All formulae presented in this section are recursive. Finally, the last thing we need for computing the c-vector of a fan graph is to determine which types of clusters can be its subgraphs. The answer can be found in Section 4.3.3. This is where the aforementioned integer partitions play a major role, see Proposition 27. Similarly to weak compositions, integer partitions have been an object of interest in Computer Science. We provide an algorithm ourselves (see Algorithm 4.2) however, since it is not very efficient, we recommend reaching for some professional sources (see for example [18]). Section 4.3.4 summarises Sections 4.3.1, 4.3.2 and 4.3.3 and explains how the methods developed there can be used in practice. For better understanding, it also contains an example of computing the c-vector of a fan graph with five 3-cycles.

2. Background

This chapter serves as a source of background knowledge for the methods to be developed.

2.1 Lattice polytopes and their Ehrhart polynomials in different bases

We begin with introducing lattice polytopes and basics of the Ehrhart theory. We follow [3] and [4].

Definition. For a finite point set $\{\mathbf{v}_1, \mathbf{v}_2, ..., \mathbf{v}_n\} \subseteq \mathbb{R}^d$, a **polytope** P is the smallest convex set containing these points, i.e.,

$$P = \{\lambda_1 \mathbf{v}_1 + \lambda_2 \mathbf{v}_2 + \cdots + \lambda_n \mathbf{v}_n \,|\lambda_1, \lambda_2, \ldots, \lambda_n \geq 0 \text{ and } \lambda_1 + \lambda_2 + \cdots + \lambda_n = 1\}.$$

For a convex polytope $P \subset \mathbb{R}^d$, we say that the hyperplane $H := \{\mathbf{x} \in \mathbb{R}^d : \mathbf{a} \cdot \mathbf{x} = b\}$ is a **supporting hyperplane** of P if P lies entirely on one side of H, that is,

$$P \subset \{\mathbf{x} \in \mathbb{R}^d : \mathbf{a} \cdot \mathbf{x} \leq b\} \text{ or } P \subset \{\mathbf{x} \in \mathbb{R}^d : \mathbf{a} \cdot \mathbf{x} \geq b\}.$$

A **face** of P is a set of the form $P \cap H$, where H is a supporting hyperplane of P. The **dimension** of a polytope P is the dimension of its **affine hull** $\text{aff}(P)$, defined as the inclusion-minimal affine subspace of $\mathbb{R}^d$ that contains P

$$\text{aff}(P) = \bigcap \Big\{ H \text{ hyperplane in } \mathbb{R}^d : P \subseteq H \Big\}.$$

If P has dimension d, we call P a d-polytope. The 0-dimensional faces of P are called **vertices**. If all vertices of P have only integral coordinates we call P a **lattice polytope**.

Ehrhart proved in [10] that for a lattice d-polytope $P \subseteq \mathbb{R}^d$, the lattice point enumerator

$$\text{ehr}_P(t) := |tP \cap \mathbb{Z}^d|$$

agrees with a polynomial of degree d for positive integers t. We refer to $\text{ehr}_P(t)$ as the **Ehrhart polynomial** of P. Furthermore we call the generating function

$$\text{Ehr}_P(z) := 1 + \sum_{t \geq 1} \text{ehr}_P(t) z^t$$

the **Ehrhart series** of P. We observe that if

$$\text{ehr}_P(t) = a_o + a_1 t + \cdots + a_d t^d$$

then

$$\begin{aligned} \text{Ehr}_P(z) &= a_0 \sum_{t \geq 0} z^t + a_1 \sum_{t \geq 0} t z^t + \cdots + a_d \sum_{t \geq 0} t^d z^t \\ &= \frac{a_0}{1-z} + a_1 z \frac{d}{dz}\Big(\frac{1}{1-z}\Big) + \cdots + a_d \Big(z \frac{d}{dz}\Big)^d \Big(\frac{1}{1-z}\Big) \\ &= \frac{h^*(z)}{(1-z)^{d+1}}, \end{aligned}$$

where $h^*(z)$ is a polynomial in z of degree at most d.

Theorem 3 (Stanley [15]). *Let P be a convex lattice d-polytope with Ehrhart series*

$$\mathrm{Ehr}_P(z) = \frac{h_d^* z^d + h_{d-1}^* z^{d-1} + \cdots + h_0^*}{(1-z)^{d+1}}.$$

Then the coefficients $h_d^*, h_{d-1}^*, \ldots, h_0^*$ are nonnegative.

While Theorem 3 was proven in [15] from a commutative-algebra point of view, an alternative, geometrical proof can be found in [3].
Expanding the Ehrhart series into binomial series yields

$$\mathrm{ehr}_P(t) = \binom{t+d}{d} + h_1^* \binom{t+d-1}{d} + \cdots + h_d^* \binom{t}{d} \tag{2.1}$$

(see the proof of Lemma 3.14 in [3]). Observe that (2.1) is a representation of the Ehrhart polynomial in basis $\binom{t}{d}, \binom{t+1}{d}, \ldots, \binom{t+d}{d}$, called the (h^*)-basis in [4]. In this work however, we focus on the (γ)-basis representation of the Ehrhart polynomial, which is

$$\mathrm{ehr}_P(t) = \sum_{j=0}^{d} c_{j+1} t^j (t+1)^{d-j}. \tag{2.2}$$

Definition. We call $c = (c_1, c_2, \ldots, c_{d+1})$ the **c-vector** of P.

2.2 Descent statistics

In this section we follow [2].
Let $d \in \mathbb{N}$ and let S_d denote the set of all permutations on $[d] := \{1, 2, \ldots, d\}$.

Definition. For a permutation word $\sigma = \sigma_1 \sigma_2 \ldots \sigma_d$ in S_d the **descent set** is defined by

$$\mathrm{Des}(\sigma) := \{i \in [d-1] : \sigma_i > \sigma_{i+1}\}.$$

Definition. The **Eulerian number** $a(d,k)$ counts the number of permutations in S_d with exactly k descents:

$$a(d,k) := |\{\sigma \in S_d : \mathrm{des}(\sigma) = k\}|,$$

where $\mathrm{des}(\sigma) := |\mathrm{Des}(\sigma)|$.

Definition. The **(A,j)-Eulerian number** $a_j(d,k)$ is the number of permutations $\sigma \in S_d$ with last letter $d+1-j$ and exactly k descents:

$$a_j(d,k) := |\{\sigma \in S_d : \sigma_d = d+1-j \text{ and } \mathrm{des}(\sigma) = k\}|.$$

Definition. The **(A,j)-Eulerian polynomial** is

$$A_j(d,t) = \sum_{k=0}^{d-1} a_j(d,k) t^k.$$

Note that by definition $a_j(d,k) = 0$ when $k < 0, j < 1, k > d-1$ or $j > d$. The (A,j)-Eulerian polynomials were most likely first considered by Brenti and Welker [7], though the (A,j)-Eulerian numbers and generalizations of them were considered earlier (see, e.g., [9], [16]).

2.3 Graphs

The definitions in this section are based on [4].

Definition. A **graph** G is an ordered pair $G = (V, E)$ where V is the set of nodes and $E \subseteq \binom{V}{2}$ is the set of edges. We denote the set of nodes of G by $V(G)$ and the set of edges by $E(G)$.

Remark. It is common to define a graph in a way that allows it to have multiple edges and loops and refer to the object defined above as a simple graph. However, since we do not consider any non-simple graphs in this work, for convenience we just call the simple graph a graph.

Definition. A **trail** in G is a sequence $v_1, v_2, \ldots, v_s$ of nodes such that every two element set $\{v_{j-1}, v_j\}$ is a distinct edge in G for all $j = 1, \ldots, s$. If $\{v_s, v_1\}$ is also an edge, then $v_1, v_2, \ldots, v_s, v_{s+1} := v_1$ is called a **cycle**. Moreover, if the nodes v_i for $i \in [s]$ are distinct, we refer to $v_1, v_2, \ldots, v_s, v_{s+1} := v_1$ as a **simple cycle**. A trail that is not a cycle is called a **path**.

Definition. A graph $H = (V', E')$ is called a **subgraph** of a graph $G = (V, E)$ if $V' \subseteq V$ and $E' \subseteq E$. If H is a subgraph of G, we say that G contains H.

Definition. A **connected component** of a graph G is a subgraph H of G, so that for every pair of nodes $v, v' \in V(G)$ there is a trail t with $v, v' \in t$. If G has only one connected component we say it is a **connected graph**. Otherwise we call G a **disconnected graph**.

Definition. A **forest** is a graph that has no cycles. A **tree** is a connected forest. An inclusion-maximal cycle-free subgraph of a graph G is called **spanning forest**, if G is disconnected and **spanning tree** if G is connected.

Definition. A **complete graph** K_v is a graph such that $|V(K_v)| = v$ and $E(G) = \binom{V(K_v)}{2}$.

An example of a complete graph is the graph $G = K_4$ in Figure 3.2.

Definition. The **cyclomatic number** of a graph G is $|E(G)| - |V(G)| + \kappa(G)$, where $\kappa(G)$ is the number of connected components of G.

Definition. We call an edge a **cut-edge** if it is not contained in any cycle.

Definition. Let $V' = \{v_1, v_2, \ldots v_k\} \subseteq V(G)$ and $v \notin V(G)$. The **identification** of V' results in a graph $G^{V'}$ with a node set

$$V(G^{V'}) := V(G) \setminus V' \cup \{v\}$$

and the edge set

$$E(G^{V'}) := \{e \in E(G) : e \cap V' = \varnothing\} \cup \{e \setminus \{v_i\} \cup \{v\} : (\exists i \in [k] : e \cap V' = \{v_i\})\}.$$

Definition. Let G be a graph with connected components $C_1, \ldots, C_k$ and $X = \{x_1, \ldots, x_k\} \subseteq V(G)$ a set of nodes of G, so that $x_i \in C_i$ for all $i \in [k]$. Moreover, let x be a node with $x \notin V(G)$. We call the graph G^X, obtained by identification of X, the **X-connected graph** of G.

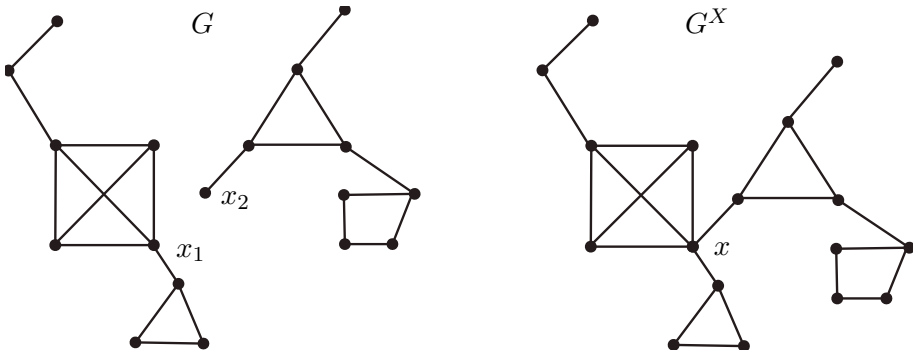

Figure 2.1: Example of an X-connected graph of G.

Remark.

1. Notice that by the definition of identification

$$V(G^X) = V(G) \setminus X \cup \{x\} \implies |V(G^X)| = |V(G)| - k + 1$$

and

$$E(G^X) = \{e \in E(G) : e \cap X = \varnothing\} \cup \{e \setminus \{x_i\} \cup \{x\} : e \cap X \neq \varnothing\}$$
$$\implies |E(G)| = |E(G^X)|,$$

because G^X has all the edges of G that do not contain any node x_i and for every edge $(v, x_i) \in E(G)$ with $i \in [k]$, so that $v \in V(C_i)$ there is exactly one edge $(v, x) \in E(G^X)$ just as for every edge $(w, x) \in E(G^X)$ there is exactly one edge $(w, x_i) \in E(G)$, since there is exactly one connected component C_i of G with $w \in C_i$. This defines a one-to-one correspondence between the edges of G and G^X. Moreover, in this work, every edge of G^X has the same label as the corresponding edge in G.

2. Analogously, we can connect any two connected graphs G_1 and G_2, by defining a graph G, with two connected components $C_1 = G_1$ and $C_2 = G_2$. We denote such a graph as $G_1 X G_2$, where $X = \{x_1, x_2\}$ for some $x_1 \in V(G_1)$ and $x_2 \in V(G_2)$.

3. Observe that G^X is always connected, since for any vertex $v_i \in C_i$ there is a trail t_i so that $v_i, x_i \in t_i$ and hence for any vertex v of G^X, there is a trail t_v with $v, x \in t_v$. Therefore for any two vertices v, w of G^X there is a trail $t := t_v \cup t_w$ with $v, w \in t$.

Definition. For an edge $e \in E(G)$, the **deletion** of e results in the graph $G \setminus e := (V, E \setminus \{e\})$. The **contraction** G/e is the graph obtained by identification of the two nodes of e and removing e.

Definition. Let G_1 and G_2 be two graphs. The **union** of G_1 and G_2 is a graph $G_1 \cup G_2$ such that

$$V(G_1 \cup G_2) := V(G_1) \cup V(G_2)$$

and

$$E(G_1 \cup G_2) := E(G_1) \cup E(G_2).$$

The **join** of G_1 and G_2 is a graph $G_1 \nabla G_2$ such that

$$V(G_1 \nabla G_2) := V(G_1) \cup V(G_2)$$

and

$$E(G_1 \nabla G_2) := E(G_1) \cup E(G_2) \cup \{\{v, w\} : v \in G_1 \wedge w \in G_2\}.$$

An example of a join of two graphs is presented in Figure 2.2.

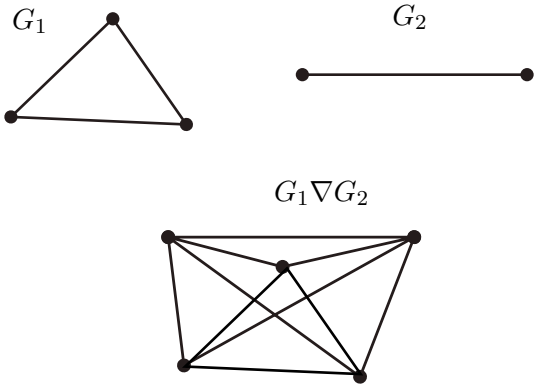

Figure 2.2: Example of a join of two graphs.

2.4 Graphical zonotopes

Just as in the previous section we follow [3] and [4].

Definition. A **Minkowski sum** of two convex sets $K_1, K_2 \subset \mathbb{R}^d$ is

$$K_1 + K_2 = \{\mathbf{p} + \mathbf{q} : \mathbf{p} \in K_1, \mathbf{q} \in K_2\}.$$

Definition. A **Zonotope** Z_G is a Minkowski sum of line segments. Moreover, if for some $n \in \mathbb{N}$ and $\mathbf{w}_i \in \mathbb{Z}^d$, $Z_G = \sum_{i=1}^{n} [0, \mathbf{w}_i]$, we say that it is **generated** by the vectors $\mathbf{w}_i$.

Definition. Given a graph G, the **graphical zonotope** Z_G induced by G is a zonotope generated by all vectors $\mathbf{e}_i - \mathbf{e}_j$, where $\mathbf{e}_i$ and $\mathbf{e}_j$ are unit vectors and ij is an edge in G with $i > j$.

Lemma 4. *Consider a subset $I \subseteq \{\mathbf{e}_i - \mathbf{e}_j : ij \in [d], i > j\}$ and an associated graph G_I with vertex set $[d]$ and edge set $\{ij : \mathbf{e}_j - \mathbf{e}_i \in I\}$. Then I is linearly independent if and only if G_I is a forest.*

Lemma 5. *Suppose $\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_n \in \mathbb{Z}^d$ are linearly independent, let*

$$\Pi := \{\lambda_1 \mathbf{w}_1 + \lambda_2 \mathbf{w}_2 + \cdots + \lambda_n \mathbf{w}_n : 0 \leq \lambda_1, \lambda_2, \ldots, \lambda_n < 1\},$$

and let D be the greatest common divisor of all $n \times n$ minors of the matrix formed by the column vectors $\mathbf{w}_1, \mathbf{w}_2, \ldots, \mathbf{w}_n$. Then for every positive integer t,

$$\#(t\Pi \cap \mathbb{Z}^d) = Dt^n.$$

Consider a case, where every $\mathbf{w}_l$ is a vector of form $\mathbf{e}_i - \mathbf{e}_j$. If they are linearly independent, the minors of the matrix formed by these vectors are 1 and hence, so is their greatest common divisor D. We conclude that Π contains only one lattice point and this point is 0. This leads us to the following corollary.

Corollary 6. *Let I be a linearly independent subset of $\{\mathbf{e}_1 - \mathbf{e}_2, \mathbf{e}_2 - \mathbf{e}_3, \ldots, \mathbf{e}_{d-1} - \mathbf{e}_d\}$. Then the open parallelepiped*

$$\sum_{\mathbf{e}_i - \mathbf{e}_j \in I} (0, \mathbf{e}_i - \mathbf{e}_j)$$

contains no lattice points.

Corollary 7. *Let G be a graph. Then the dimension of the graphical zonotope Z_G is $|V(G)| - \kappa(G)$. In particular, if G is connected, the dimension of $Z(G)$ is $|V(G)| - 1$.*

Proof. First we notice that $|V(G)| - \kappa(G)$ is the number of edges in every spanning forest of G. Hence, by Lemma 4, the largest linearly independent subset of generators of Z_G is of size $|V(G)| - \kappa(G)$. Thus, the affine hull $\mathrm{aff}(Z_G)$ is of dimension $|V(G)| - \kappa(G)$ and consequently, so is Z_G itself. Moreover, if G is a connected graph, then $\kappa(G) = 1$ and the dimension of Z_G is $|V(G)| - 1$. □

2.5 Matroids

The paper [17] by Whitney from 1935 introduces matroids as a way to abstract the concepts of linear algebra and graph theory. Indeed, the theory of matroids has proven very useful for the research in these fields of mathematics, including this thesis. In this section, we present the basic notions and facts concerning matroids, most of which were inspired by [11].

Definition. A **matroid** $M = (S, \mathcal{I})$ is an ordered pair consisting of a finite ground set S and a collection $\mathcal{I}$ of subsets of S that satisfy the following **independent set axioms**:

I1 $\varnothing \in \mathcal{I}$;

I2 $\mathcal{I}$ is closed with respect to taking subsets; and

I3 if $I_1, I_2 \in \mathcal{I}$ with $|I_1| < |I_2|$, then there is some $e \in I_2 \setminus I_1$ such that $I_1 \cup \{e\} \in \mathcal{I}$.

Definition. The maximally independent subsets of $\mathcal{I}$ are called **bases**. We denote the collection of all bases by $\mathcal{B}$.

Corollary 8. *The bases of a matroid have the same cardinality.*

Proof. Let M be a matroid and B_1, B_2 two bases with $|B_1| < |B_2|$. Then by **I3** there is $e \in B_2 \setminus B_1$ such that $B_1 \cup \{e\} \in \mathcal{I}$, which is a contradiction to B_1 being maximally independent. □

Definition. The **rank** of a matroid M is the cardinality of any basis of M.

For this part we use concepts introduced in [2]. Consider a matroid $M = (W, \mathcal{I})$, where $W = \{\mathbf{w}_1, \ldots, \mathbf{w}_m\} \subset \mathbb{Z}^d$ is a set of vectors and $\mathcal{I}$ denotes the set of independent subsets of W, i.e., sets of linearly independent vectors. Moreover, let $\mathbf{w}_1 < \cdots < \mathbf{w}_m$ be a fixed order on the elements of W. For simplicity, we identify $\mathbf{w}_i$ with $i \in [m]$ in the sequel. The order on $[m]$ induces an order on $\mathcal{B}$, namely $B_1 < B_2$ whenever B_1 is lexicographically smaller than B_2.

Definition. An element i in a basis B is called **internally passive** if there is an element $j < i$ with $j \notin B$, such that $\{j\} \cup B \setminus \{i\}$ is a basis. In other words, i can be exchanged with a smaller element j. We denote the set of internally passive elements of B by $\mathrm{IP}(B)$. Note that, in particular, $\mathrm{IP}(B) \subset B$.

Definition. Let G be a graph with node set $V(G)$ and edge set $E(G)$. One defines the **graphic matroid** $M(G)$ as the matroid with ground set $S = E(G)$ and independent sets given by forests. The bases are then spanning forests of G, and hence the rank of $M(G)$ is given by $|V(G)| - \kappa(G)$, where $\kappa(G)$ is the number of connected components of G.

Remark. Notice that we could also represent a graph G as a matrix W with $\mathbf{e}_i - \mathbf{e}_j$ as column vectors for all $ij \in E(G)$. By Lemma 4, $(W, \mathcal{I})$ is a matroid with $I \in \mathcal{I}$ independent if and only if G_I is a forest. Moreover, the bases of $(W, \mathcal{I})$ are all sets B, so that G_B is a spanning forest of G. Furthermore, for every internally passive element $\mathbf{p} \in \mathrm{IP}(B)$ and $\mathbf{q} \notin B$, such that $B' := B \setminus \{\mathbf{p}\} \cup \{\mathbf{q}\}$. Thus, if $e_{\mathbf{p}}$ is the edge corresponding to $\mathbf{p}$, then $e_{\mathbf{p}} \in \mathrm{IP}(G_B)$. We refer to $e_{\mathbf{p}}$ as an **internally passive edge** of G_B.

2.6 h^*- polynomials of zonotopes

The following theorem is stated as Theorem 1.4 in [2] with a slight difference, namely in [2] it was generalised for any translation-invariant valuation while here we only consider the lattice point enumerator.

Theorem 9. *Let Z be a d-dimensional lattice zonotope generated by a set of vectors $W \subset \mathbb{Z}^d$. Then*

$$h^*(Z)(t) = \sum_{I \subseteq \mathcal{I}} l(I) \sum_{\substack{B \in \mathcal{B} \\ B \supseteq I}} A_{|I \cup \mathrm{IP}(B)|+1}(d+1, t), \tag{2.3}$$

where $l(I)$ is the number of lattice points of the open parallelepiped generated by the vectors in I. By convention $l(\varnothing) = 1$.

3. Universal methods

The goal of this thesis is to derive methods of computing c-vector of a graphical zonotope, which can be performed by investigating only the underlying graph rather than the zonotpe itself. We primarly value tools that can be applied to an arbitrary graph, and these are the subject of this chapter.

3.1 Change of basis

In section 2.1 we see that Ehrhart polynomials can be expressed in three different bases. It is not surprising, that there also exist different ways of expressing the h^*-polynomial. One of these is the focus of this thesis. We simplify Theorem 9 for the case of graphical zonotopes.

Corollary 10. *Let G be a graph and Z_G a d-dimensional graphical zonotope on G. Then*

$$h^*(Z_G)(t) = \sum_{T \in \mathcal{T}} A_{|\operatorname{IP}(T)|+1}(d+1, t), \tag{3.1}$$

where $\mathcal{T}$ is the set of all spanning forests of G.

Proof. Let $W \subset \mathbb{Z}^d$ be the set of vectors generating Z_G. First we observe that due to Corollary 6, $l(I) = 0$ unless $I = \varnothing$, in which case $l(I) = 1$. This means that the only independent set of W we need to consider is $I = \varnothing$. Hence, for the graphical zonotope the second sum of (2.3) is taken over all bases of W, since all of them contain $\varnothing$.
By the definition of a graphical zonotope, the vectors generating Z_G are all of the form $\mathbf{e}_i - \mathbf{e}_j$. As described in Section 2.5, the matroid $M = (W, \mathcal{I})$ is a representation of the graphical matroid $M(G)$ and there is a one-to-one correspondence between the bases of M and $M(G)$. Moreover, any order on the vectors of W induces an order on the edges of $E(G)$ and hence the lexicographical order of bases of M also applies to the corresponding spanning forests of G, which means that $|\operatorname{IP}(B)| = |\operatorname{IP}(T_B)|$, for any $B \in \mathcal{B}$ and the corresponding spanning forest T_B.

□

Notice that $|\operatorname{IP}(T)|$ can be at most the number of edges of the spanning forests T of G, which is $|V(G)| - \kappa(G)$. Furthermore, by Corollary 7, the dimension of Z_G is
$d = |V(G)| - \kappa(G)$. From this observation and Corollary 10 it follows that if G is an arbitrary graph, $h^*(Z_G)(t)$ is a linear combination of the (A, j)-Eulerian polynomials $A_j(d+1, t)$ for $j \in [d+1]$.
The following result is taken from [2].

Proposition 11. *Let Z_G be a graphical zonotope with*

$$h^*(Z_G)(t) = c_1 A_1(d+1, t) + c_2 A_2(d+1, t) + \cdots + c_{d+1} A_{d+1}(d+1, t). \tag{3.2}$$

Then $c = (c_1, c_2, \ldots, c_{d+1})$ is the c-vector of Z_G.

For convenience, in this work we often say the c-vector of G instead of Z_G. We are now able to prove Theorem 1, stating that the coefficient c_j of the c-vector of a graph G counts the spanning forests of G with $j-1$ internally passive edges.

Proof of Theorem 1. First we observe that the coefficient c_j of the c-vector of a graph G counts how often $A_j(d+1,t)$ appears in (3.1). Since the sum runs over all spanning forests of G, and every index j is in fact the number of internally passive edges of some spanning forest of G increased by one, Theorem 1 follows. □

3.2 Simplifying a graph

In this section, we show that for computing the c-vector of a graph G, it suffices to consider a simplified version of G, which is connected and devoid of cut-edges.
Recall that connecting a graph means picking an arbitrary vertex in every connected component of the graph and identifying these vertices. See Section 2.3 for more details.

Proposition 12. *Let X be defined as in Section 2.3 and G^X be the X-connected graph of G. Then the c-vectors of G and G^X are the same.*

Proof. We show that there is a bijection between the spanning forests of G and the spanning trees of G^X, preserving the number of internally passive elements. Consider a map sending every spanning forest T of G to T^X, the X-connected graph of T. First we show that any such T^X is a spanning tree of G^X. We observe that since T is a spanning forest, $V(T) = V(G)$. Let k denote the number of connected components of G and C_i the i-th connected component. Since

$$V(T^X) = V(T) \setminus X \cup \{x\} = V(G) \setminus X \cup \{x\} = V(G^X),$$

T^X also contains all nodes of G^X. Moreover, T^X is a tree because it is connected and

$$|E(T^X)| = |E(T)| = |V(T)| - k = |V(T^X)| - 1.$$

Now we prove that this map is bijective. Since the sets of spanning trees of G and G^X are both finite, it suffices to show that this map is injective.
Injectivity:
Assume that T_1 and T_2 are two spanning forests of G with $T_1 \neq T_2$ and $T_1^X = T_2^X$. Then for any edge $e \in E(T_1)$ we have two cases:

1. $$\forall i \in [k] : x_i \notin e \implies e \in E(T_1^X) = E(T_2^X).$$
   Since e is an edge of T_2^X, which does not contain x, it must be an edge of T_2.

2. $$\begin{aligned} \exists i \in [k] : x_i \in e &\implies e = (v, x_i), \text{ for some } v \in V(C_i) \\ &\implies (v, x) \in E(T_2^X) \\ &\implies \exists j : (v, x_j) \in E(T_2), \text{ but } v \in V(C_i) \\ &\implies j = i \\ &\implies E(T_2) \ni (v, x_j) = e. \end{aligned}$$

An analogous argument holds for any $e \in E(T_2)$ and hence

$$E(T_1) = E(T_2) \implies T_1 = T_2,$$

which contradicts our assumption.
Internally passive elements:
It remains to show that the number of internally passive elements is invariant under this map. Let T be again a spanning forest of G and $e \in \mathrm{IP}(T)$. Then there is an edge e_s with a label smaller then the label of e, so that $T_s = T \setminus \{e\} \cup \{e_s\}$ is a spanning forest of G. Hence, T_s^X is a spanning tree of G^X. Let now e^x and e_s^x be the edges of G^X corresponding to e and e_s, respectively. Then $e^x > e_s^x$ and

$$E(T_s^X) = E(T^X) \setminus \{e^x\} \cup \{e_s^x\} \implies e^x \in \mathrm{IP}(T^X).$$

Conversely, let now T^* be a spanning tree of G^X, $e^* \in \mathrm{IP}(T^*)$, $e_s^* < e^*$ and $T_s^* = T^* \setminus \{e\} \cup \{e_s\}$. Since the map is surjective, there is a unique spanning forest T of G, so that $T^* = T^X$ and a unique spanning forest T_s, with $T_s^* = T_s^X$. Then again, if e and e_s are the edges in G corresponding to e^* and e_s^*, respectively, then

$$E(T_s) = E(T) \setminus \{e\} \cup \{e_s\} \implies e \in \mathrm{IP}(T).$$

□

Since Proposition 12 holds independently of the choice of X, throughout this work, we denote by G° a graph connected by any X.

Proposition 13. *The c-vector of a graph does not change under the contraction of cut-edges.*

Proof. Let G be a graph and e one of its cut-edges. Then e is contained in every spanning forest of G. Observe that if T is a spanning forest of G then $T/\{e\}$ is a spanning forest of $G/\{e\}$ with $|\mathrm{IP}(T)| = |\mathrm{IP}(T/\{e\})|$, since e cannot be replaced by any smaller edge in T and hence is not an internally passive element of T. Moreover, the collection of $T/\{e\}$ for every spanning forest T of G is the collection of all spanning forests of $G/\{e\}$ and the lemma follows. □

Inspired by Proposition 12 and Proposition 13 we define the **reduced graph** of G.

Definition. We call a graph $\tilde{G}$ obtained by contracting all cut-edges of G° the **reduced graph** of G.

Furthermore, we conclude the following.

Corollary 14. *The c-vector of a graph G is the same as the c-vector of its reduced graph $\tilde{G}$.*

From now on, in order to characterise the c-vector of a graph, we only consider its reduced version, since it is often smaller then the original graph, there are no cut-edges in it and it is always connected. However, we have to keep in mind that the zonotope of the reduced graph can have a different dimension, as it depends on the number of nodes and connected components.

3.3 Determining the c-vector

This section is devoted to presenting a few universal properties of the c-vector, which faciliate its computation. In this work, the order on the edges of any graph is always strict, hence for every graph there exists a unique lexicographically minimal spanning forest $T_{\min}$, constructed from the minimal edges of G, such that no cycle is created.

Definition. A **spare edge** is an edge not contained in the lexicographically minimal spanning forest of a graph.

We observe a very useful fact about the spare edges.

Corollary 15. *If a spanning forest T contains a spare edge s then $s \in \mathrm{IP}(T)$.*

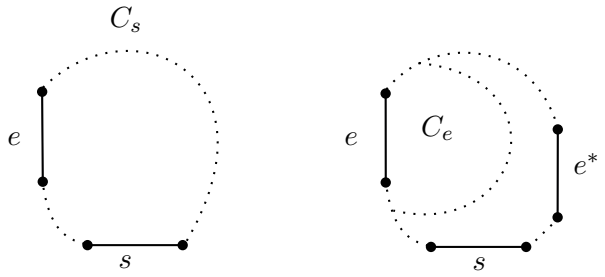

Figure 3.1: This Figure depicts the situation in the proof of Corollary 15. Picture on the left shows cycle C_s, created by adding the spare edge s to the lexicographically minimal spanning forest of graph G. The right side of the figure presents cycle C_e, created by adding e to the spanning forest T. We see that
$C_s \cup C_e \setminus (C_s \cap C_e)$ is indeed a cycle.

Proof. Let T be a forest containing a spare edge s. Furthermore, let C_s be the cycle created by adding s to $T_{\min}$. Then except for s, the cycle C_s consists only of edges of $T_{\min}$. Thus, there must be an edge $e \in E(C_s) \cap E(T_{\min})$, such that $e \notin E(T)$. If the subgraph $T \setminus s \cup e$ is not a forest, and C_e is the cycle of this subgraph, then $C_e \cup C_s \setminus (C_e \cap C_s)$ is a cycle of G and there must be another edge $e' \in E(C_s) \setminus E(C_e)$, such that $e' \notin E(T)$. If $T \setminus s \cup e'$ still has a cycle, we repeat the argument until we find an e^*, such that $T \setminus s \cup e^*$ is a spanning forest of G, see Figure 3.1. Since $T \setminus s \cup e^*$ is lexicographically smaller than T, the edge s is indeed an internally passive edge of T. □

Definition. Let T be a spanning forest of G with $j := |\mathrm{IP}(T)|$. Then we refer to T as an A_{j+1}**-forest**.

Remark. Since we mainly investigate the reduced graphs, which are always connected, we mostly use the term A_{j+1}**-trees**.

To count the number of spare edges of a graph G, we need to subtract the number of edges of a spanning forest from the number of all edges. Therefore, there are

$$|E(G)| - |V(G)| + \kappa(G)$$

spare edges, which brings us to the following corollary.

Corollary 16. *The number of spare edges of a graph is the cyclomatic number of a graph.*

We now have all the tools needed to prove Theorem 2 combinatorially.

Proof of Theorem 2. Let G be a graph and $\tilde{G}$ the reduced graph of G. We show that

1. $c_1 = 1$;
   This is clear, since there is exactly one lexicographically minimal spanning tree of $\tilde{G}$, which does not have internally passive edges. Moreover, by Corollary 15, since the other spanning trees contain at least one spare edge, their sets of internally passive edges are not empty.

2. *c_2 is the cyclomatic number of $\tilde{G}$, i.e., $c_2 = |E(\tilde{G})| - |V(\tilde{G})| + 1$;*
Observe that A_2-trees are those in which exactly one edge can be exchanged in order to get a lexicographically smaller spanning tree. In other words, these are the lexicographically minimal spanning trees containing exactly one spare edge. Since for every spare edge there is exactly one lexicographically minimal spanning tree containing it, the claim follows. We could think about this as adding a spare edge s to the minimal spanning tree and then deleting the maximal edge $e \neq s$, in the cycle produced in this process, see Figure 3.2.

3. *$c_3 = c_2 + \binom{c_2}{2}$;*
Analogously to 2. A_3-trees are those with exactly two internally passive elements. Similarly as before, we add a spare edge s to the lexicographically minimal spanning tree, which results in a cycle C and then remove the second largest edge e of the set $E(C) \setminus \{s\}$, see an example in Figure 3.3. We can always do that, because every cycle has at least three edges. However, if the number of spare edges exceeds one, we also have A_3-trees with exactly two spare edges. In this case, the spare edges must be the only internally passive elements, so all the other edges must be lexicographically minimal, as shown in Figure 3.4. The number of trees of the first type is simply c_2, and the number of trees of the second type is the same as the number of pairs of spare edges, which is $\binom{c_2}{2}$. Hence the claim follows.

4. *$c_j > 0$ for $j \in [1, |V(\tilde{G})|]$, and $c_j = 0$ otherwise*;
We want to show that the largest index j, so that $c_j > 0$, is $j = |V(\tilde{G})| =: v$. First notice that $|V(\tilde{G})| = |E(T)|+1$, where T is any spanning tree of $\tilde{G}$. Hence, A_v-trees are those where every edge can be replaced by an edge with a smaller label, without creating a cycle. We construct such tree by leaving out the minimal edge in every simple cycle and call it $T_{\max}$. We claim that $T_{\max}$ is indeed an A_v-tree. To see this, first note that since there are no cut edges in $\tilde{G}$, for every $e \in T_{\max}$ there is an edge $e^* \notin T_{\max}$, so that $T_{\max} \setminus \{e\} \cup \{e^*\}$ is a spanning tree of $\tilde{G}$. Such e^* must share a simple cycle with e and since while constructing $T_{\max}$, we remove the minimal edges from every simple cycle, the label of e must be larger than the label of e^* and the claim follows.

   We know now that for every graph, there exists an A_v-tree and also that there cannot exist an A_j-tree with $j > v$, since the number of internally passive elements in a tree cannot exceed the number of edges of this tree. However, we have not proved that the construction above is the only way an A_v-tree can be obtained and in fact, there exist graphs where this is not the case, meaning that c_v does not necessarily equal 1, for instance for the complete graph K_4, see Figure 3.2, $c_v = c_4 = 6$, since there are six A_4-trees of K_4, illustrated in Figure 3.5 and Figure 3.6.

   We still need to show that for any $j < v$, the j-th coefficient of the c-vector c_j is a positive integer. Luckily, a slight modification of the approach used to create $T_{\max}$ allows us to construct an A_j-tree T for any $j < n$. We proceed as follows: we first add the maximal edge e_α of $\tilde{G}$ and the maximal edge e_β of $E(\tilde{G}) \setminus \{e_\alpha\}$. Then we add the maximal edge e_γ of $E(\tilde{G}) \setminus \{e_\alpha, e_\beta\}$ if it does not create a cycle, otherwise we skipp e_γ and add the maximal edge e_δ of $E(\tilde{G}) \setminus \{e_\alpha, e_\beta, e_\gamma\}$ instead, unless it creates a cycle and so on. We repeat this procedure until we collect $j - 1$ edges and denote the set containing them $E_{\max}$. For the remaining $v - i$ edges of the tree, we use the analogous procedure, namely, first add the minimal edge f_α of $E(\tilde{G}) \setminus E_{\max}$, provided it does not create a cycle, otherwise we add the minimal

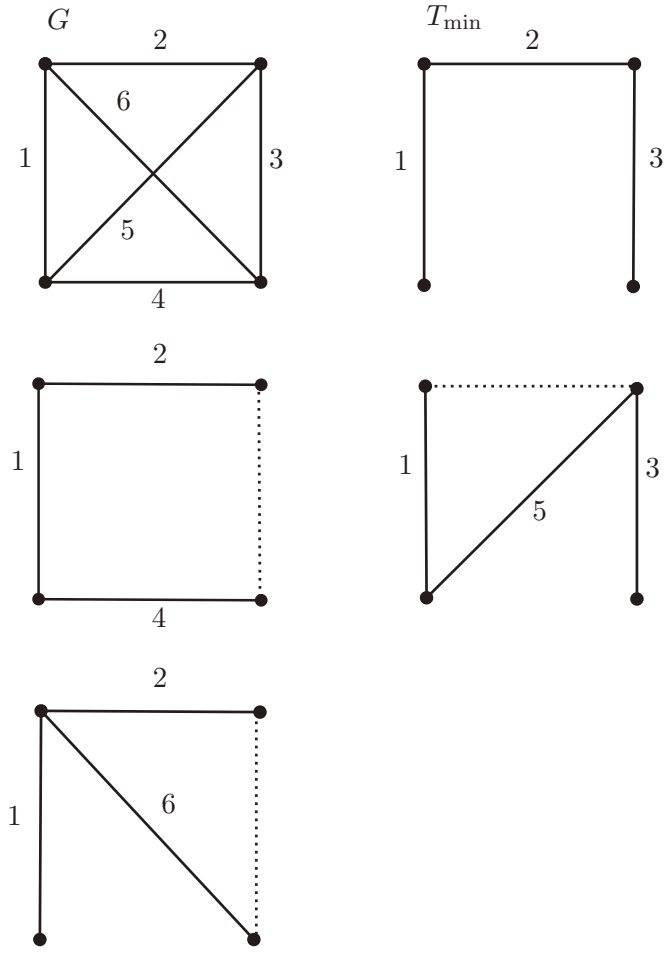

Figure 3.2: Graph G with its minimal spanning tree $T_{\min}$ and all A_2-trees. Dashed lines represent the edges of $T_{\min}$, which have been replaced by spare edges 4,5 and 6 in order to create the A_2-trees of G.

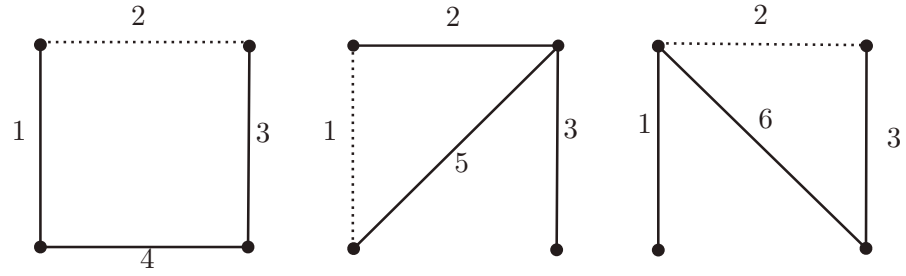

Figure 3.3: A_3-trees of graph G from Figure 3.2 with one spare edge each. Dashed lines again represent the edges of $T_{\min}$, which have been replaced by spare edges 4,5 and 6.

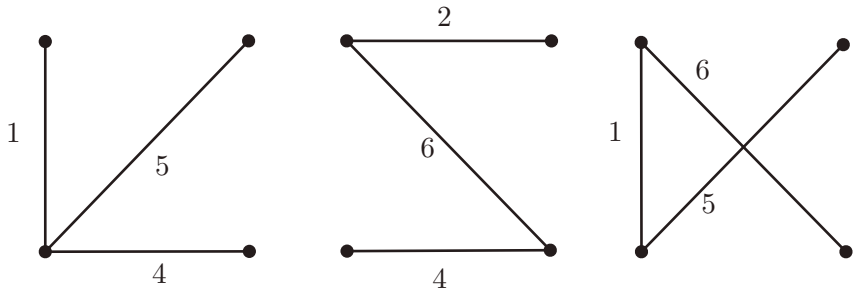

Figure 3.4: A_3-trees of graph G from Figure 3.2 with two spares edges and the remaining edge chosen as small as possible.

edge f_β of $E(\tilde{G}) \setminus E_{\max} \setminus \{f_\alpha\}$ and so on, until we collect $v - i$ of these. By this construction, none of the $v - i$ minimal edges can be replaced by any smaller edge, since either they are the smallest, or replacing them by a smaller edge would lead to a cycle. Thus, these are not internally passive. The only question is, how do we know if the maximal $j - 1$ edges of T are internally passive? Let $e \in E_{\max}$. Since $\tilde{G}$ has no cut-edges, there must be an edge $e' \notin E(T)$ so that $T \setminus e \cup e'$ is a spanning tree. By construction $e' < e$, because otherwise e' would form a cycle with another edges from $E_{\max}$. Hence e is internally passive.

5. *The sum of all entries of the c-vector equals the number of the spanning trees of* $\tilde{G}$; Since every c_j counts the number of A_j-trees, by (3.2), the sum of all coefficients of every c-vector must be the number of all spanning trees of $\tilde{G}$.

□

Remark.

1. One might guess that the number of A_j-trees is $c_2 + \binom{c_2}{2} + \cdots + \binom{c_2}{j-1}$. However, this fails already for $j = 4$. For example let s be a spare edge, so that there is a 3-cycle C containing no spare edge other than s. Then we can only obtain a spanning tree containing s and no other spare edge, if we exchange one of the edges belonging to both C and the minimal spanning tree for s. Since we have only two edges which could be replaced by s, we can only construct two spanning trees this way. In particular, we get an A_2-tree if we remove the maximal edge and an A_3-tree if we remove the minimal edge.

2. Since we are able to compute c_1, c_2 and c_3 easily, knowing how many nonzero coefficient there are and what their sum is could help us to compute the remaining entries of the c-vector.

The remaining question is: how to compute the coefficient c_j for $j > 3$? In this section we do not provide closed formula answering this question, but we show an example of how this can be done. We finish computing the c-vector of the graph $G = K_4$.

Example. (Computing the coefficients c_j for $j > 3$ of the c-vector of K_4.) By Theorem 2.4 the maximal j so that $c_j > 0$ is $j = 4$, which is the number of A_4-trees, i.e., spanning trees of K_4 with three internally passive edges. By similar logic as before, we can obtain such trees by adding a spare edge to $T_{\min}$ which creates a cycle C and removing the third largest edge of $E(C) \setminus \{s\}$. This is only possible if $|E(C)| \geq 4$, which for K_4 happens exactly once, i.e., the cycle C, such that $E(C) = \{1, 2, 3, 4\}$ and the edges of the resulting A_3-tree are $\{2, 3, 4\}$ as presented in Figure 3.5.

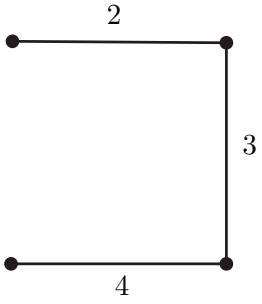

Figure 3.5: The A_4-tree of K_4 with exactly one spare edge.

Now we consider the trees with exactly two spare edges. These are obtained, if to every unordered pair of spare edges, we add a nonminimal edge. Precisely, for $\{4, 5\}$ the only choice is 2, the pair $\{4, 6\}$ must be completed with the edge 3 and to $\{5, 6\}$ we either add 2 or 3, see Figure 3.6.

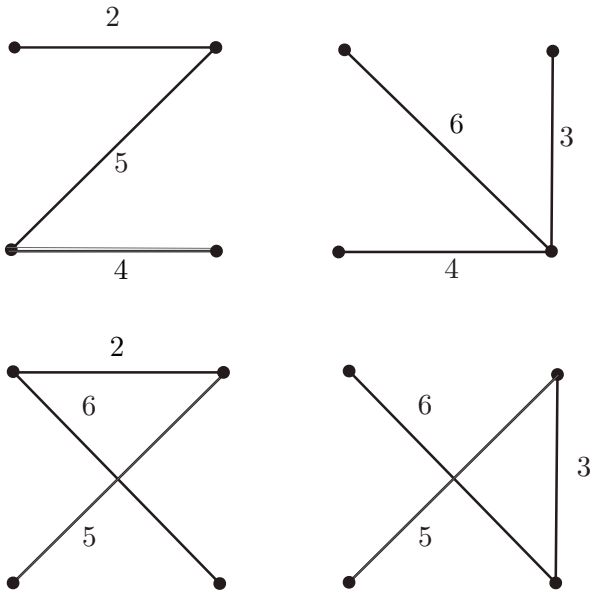

Figure 3.6: The A_4-trees of K_4 with exactly two spare edges.

Finally, we get exactly one spanning tree consisting of all three spare edges, i.e., T with $E(T) = \{4, 5, 6\}$. Altogether, these are six A_4-trees, which means that $c_4 = 6$. By Theorem 2 we also know that $c_1 = 1, c_2 = 3$ and $c_3 = 6$. Thus, the c-vector of K_4 is $c = (1, 3, 6, 6)$.

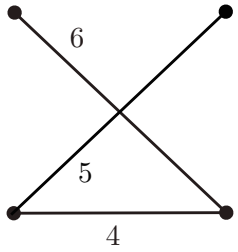

Figure 3.7: The A_4-tree of K_4 with exactly three spare edges.

3.4 Computing the c-vector from the Ehrhart polynomial

In this section we show how to compute the c-vector of a graphical zonotope from its Ehrhart polynomial. We recall that we defined the c-vector of a lattice polytope P as the coefficients of its Ehrhart polynomial written in the following form

$$\mathrm{ehr}_P(t) = \sum_{j=0}^{d} c_{j+1} t^j (t+1)^{d-j}. \tag{3.3}$$

If P is a graphical zonotope generated by a graph G, we can use Corollary 14 which says that the c-vector of P is the same as the c-vector of $Z_{\tilde{G}}$, the zonotope generated by the reduced graph $\tilde{G}$ of G. Since $\tilde{G}$ is a connected graph, by Corollary 7, the dimension of $Z_{\tilde{G}}$ is $d = |V(\tilde{G})| - 1$. The only thing we need to know now is how to compute the Ehrhart polynomial of $Z_{\tilde{G}}$. For this we use the following corollary, stated in [4].

Corollary 17. *Let G be a graph and $v := |V(G)|$. Then*

$$ehr_{Z_G}(t) = \sum_{i=0}^{v-1} b_i(G) t^i, \tag{3.4}$$

where $b_i(G)$ is the number of forests in G with i edges.

Remark. In Corollary 17 and later in this thesis by forests in G we actually mean forests which are also subgraphs of G.

Notice that $b_0(G) = 1$, since there is exactly one way to construct a forest with no edges. Furthermore, $b_1(G)$ is the number of edges of G and $b_2(G)$ is the number of pairs of edges. However, $b_3(G)$ is not the number of tripples of edges, since some tripples of edges can form a 3-cycle, so for $i \geq 3$, from all possible sets of i edges of G we need to subtract those containing some cycles. Combining (3.3) and (3.4) together and applying it on $\tilde{G}$ yields

$$\sum_{j=0}^{d} c_{j+1} t^j (t+1)^{d-j} = \sum_{i=0}^{d} b_i(\tilde{G}) t^i. \tag{3.5}$$

We replaced $v - 1$ in (3.4) by d, since $v - 1$ is the dimension of the graphical zonotope $Z_{\tilde{G}}$. We rewrite the summands on the left side of (3.5) using the binomial expansion of $(t+1)^{d-j}$ as follows,

$$c_{j+1} t^j (t+1)^{d-j} = c_{j+1} \sum_{k=0}^{d-j} \binom{d-j}{k} t^{k+j}. \tag{3.6}$$

A careful examination of (3.5) and (3.6) brings us to the conclusion that while the coefficient of t^i on the right side of Equation (3.5) is $b_i(\tilde{G})$, the coefficient of t^i on the left side is

$$\sum_{j=1}^{i+1} c_j \binom{d-j+1}{i+1-j}.$$

Thus,

$$b_i(\tilde{G}) = \sum_{j=1}^{i+1} c_j \binom{d-j+1}{i+1-j} = \sum_{j=1}^{i+1} c_j \binom{v-j}{i+1-j} \tag{3.7}$$

for $i \in [0, v-1]$.
Now we have all the tools needed to prove Theorem 2 algebraically.

Alternative proof of Theorem 2. Let Z_G be the graphical zonotope generated by graph G with a reduced graph $\tilde{G}$. Moreover, let $e := |E(\tilde{G})|$ and $v := |V(\tilde{G})|$. Then

1. Setting $i = 0$ in (3.7) yields
$$1 = b_0(\tilde{G}) = c_1.$$

2. Since $c_1 = 1$, if we set $i = 1$ in (3.7), then
$$e = b_1(\tilde{G}) = c_1(v-1) + c_2 = v - 1 + c_2 \implies c_2 = e - v + 1,$$
which is the number of spare edges of G.

3. Knowing what c_1, c_2 and $b_2(\tilde{G})$ are, we can compute c_3
$$\begin{aligned}
\binom{e}{2} = b_2(\tilde{G}) &= c_1\binom{v-1}{2} + c_2(v-2) + c_3 = \binom{v-1}{2} + (e-v+1)(v-2) + c_3 \\
\implies c_3 &= \binom{e}{2} - \binom{v-1}{2} - (e-v+1)(v-2) \\
&= \frac{e(e-1) - (v-1)(v-2) - 2(ev - v^2 + v - 2e + 2v - 2)}{2} \\
&= \frac{e^2 - e - v^2 + 3v - 2 - 2ev + 2v^2 - 6v + 4e + 4}{2} \\
&= \frac{e^2 + 3e + v^2 - 3v - 2ev + 2}{2} \\
&= \frac{(e-v+1)(e-v+2)}{2} \\
&= \binom{e-v+2}{2} \\
&= \binom{c_2+1}{2} \\
&= c_2 + \binom{c_2}{2}.
\end{aligned}$$

4. By (3.3), if $j > d + 1 = v$, the j-th coefficient of the c-vector of $\tilde{G}$ must be $c_j = 0$. Recall that we also showed combinatorially that $c_j > 0$ for $j \in [d+1]$, however, we did not come up with a fully algebraic way of proving this.

5. We observe that $b_{v-1}(\tilde{G})$ is the number of spanning trees of $\tilde{G}$. Setting $i = v-1$ in (3.7),

$$b_{v-1}(\tilde{G}) = \sum_{j=1}^{v} c_j \binom{v-j}{v-j} = \sum_{j=1}^{v} c_j$$

which proves the claim.

□

Remark. In order to compute any c_j for $j > 3$ we need to know what $b_{j-1}(\tilde{G})$ is. As mentioned before, this gets more complicated, since counting the forests with more than 2 edges requires knowing which of the edges form cycles.

Our next goal is to express every coefficient c_j as a linear combination of $b_i(\tilde{G})$ using (3.7). First we see how this can be done for $j \in [4]$. For sake of clarity, we write b_i instead of $b_i(\tilde{G})$ in the following computations.

$$\begin{aligned}
c_1 &= b_0,\\
c_2 &= b_1 - \binom{v-1}{1} c_1 = b_1 - \binom{v-1}{1} b_0,\\
c_3 &= b_2 - \binom{v-2}{1} c_2 - \binom{v-1}{2} c_1\\
&= b_2 - \binom{v-2}{1}\Big(b_1 - \binom{v-1}{1} b_0\Big) - \binom{v-1}{2} b_0\\
&= b_2 - \binom{v-2}{1} b_1 + \Big(\binom{v-2}{1}\binom{v-1}{1} - \binom{v-1}{2}\Big) b_0\\
&= b_2 - \binom{v-2}{1} b_1 + \Big((v-2)(v-1) - \frac{(v-2)(v-1)}{2}\Big) b_0\\
&= b_2 - \binom{v-2}{1} b_1 + \binom{v-1}{2} b_0,\\
c_4 &= b_3 - \binom{v-3}{1} c_3 - \binom{v-2}{2} c_2 - \binom{v-1}{3} c_1\\
&= b_3 - \binom{v-3}{1}\Big(b_2 - \binom{v-2}{1} b_1 - \binom{v-1}{2} b_0\Big) - \binom{v-2}{2}\Big(b_1 + \binom{v-1}{1} b_0\Big) - \binom{v-1}{3} b_0\\
&= b_3 - \binom{v-3}{1} b_2 + \Big(\binom{v-3}{1}\binom{v-2}{1} - \binom{v-2}{2}\Big) b_1\\
&+ \Big(\binom{v-3}{1}\binom{v-1}{2} - \binom{v-2}{2}\binom{v-1}{1} - \binom{v-1}{3}\Big) b_0\\
&= b_3 - \binom{v-3}{1} b_2 + \Big((v-3)(v-2) - \frac{(v-3)(v-2)}{2}\Big) b_1\\
&+ \Big(\frac{(v-3)(v-2)(v-1)}{2} - \frac{(v-3)(v-2)(v-1)}{2} - \binom{v-1}{3}\Big) b_0\\
&= b_3 - \binom{v-3}{1} b_2 + \binom{v-2}{2} b_1 - \binom{v-1}{3} b_0.
\end{aligned}$$

Based on these results, we come up with the following proposition.

Proposition 18. *The coefficient c_i of a c-vector of a graph G can be expressed as the following linear combination of $b_i(\tilde{G})$*

$$c_j = \sum_{i=0}^{j-1} (-1)^{j-1-i} \binom{v-i-1}{j-i-1} b_i(\tilde{G}). \tag{3.8}$$

Proof. We prove Proposition 18 by induction on j. We already showed that (3.8) holds for $j = 1, 2, 3, 4$, so we might treat these as base cases. Now we show that if for all coefficients of the c-vector up to c_j the equation (3.8) is true, it is also true for c_{j+1}. By (3.7),

$$c_{j+1} = b_j(\tilde{G}) - \sum_{k=0}^{j} c_k \binom{v-k}{j+1-k}.$$

Since $k \leq j$, we can plug in our induction hypotesis

$$c_{j+1} = b_j(\tilde{G}) - \sum_{k=1}^{j} \binom{v-k}{j+1-k} \sum_{i=0}^{k-1} (-1)^{k-1-i} \binom{v-i-1}{k-i-1} b_i(\tilde{G}). \tag{3.9}$$

Observe that

$$\begin{aligned}
\binom{v-k}{j+1-k}\binom{v-i-1}{k-i-1} &= \frac{(v-k)!}{(j+1-k)!(v-j-1)!} \cdot \frac{(v-i-1)!}{(k-i-1)!(v-k)!} \\
&= \frac{(v-i-1)!}{(j+1-k)!(v-j-1)!(k-i-1)!} \\
&= \frac{(v-i-1)!}{(j-i)!(v-j-1)!} \cdot \frac{(j-i)!}{(j+1-k)!(k-i-1)!} \\
&= \binom{v-i-1}{j-i}\binom{j-i}{j+1-k}.
\end{aligned}$$

Applying this result to (3.9) yields

$$c_{j+1} = b_j(\tilde{G}) - \sum_{k=1}^{j} \sum_{i=0}^{k-1} (-1)^{k-1-i} \binom{v-i-1}{j-i} \binom{j-i}{j+1-k} b_i(\tilde{G}). \tag{3.10}$$

We rewrite the right side of (3.10), so that the outer sum runs over the indices of $b_i(\tilde{G})$.

$$\begin{aligned}
c_{j+1} &= b_j(\tilde{G}) - \sum_{i=0}^{j-1} b_i(\tilde{G}) \sum_{k=i+1}^{j} (-1)^{k-1-i} \binom{v-i-1}{j-i} \binom{j-i}{j+1-k} \\
&= b_j(\tilde{G}) - \sum_{i=0}^{j-1} \binom{v-i-1}{j-i} b_i(\tilde{G}) \sum_{k=i+1}^{j} (-1)^{k-1-i} \binom{j-i}{j+1-k} \\
&= b_j(\tilde{G}) - \sum_{i=0}^{j-1} \binom{v-i-1}{j-i} b_i(\tilde{G}) \sum_{k=i+1}^{j} (-1)^{k-1-i} \binom{j-i}{k-i-1} \\
&= b_j(\tilde{G}) - \sum_{i=0}^{j-1} \binom{v-i-1}{j-i} b_i(\tilde{G}) \sum_{k=0}^{j-i-1} (-1)^{k} \binom{j-i}{k} \\
&= b_j(\tilde{G}) - \sum_{i=0}^{j-1} \binom{v-i-1}{j-i} b_i(\tilde{G}) \big((1-1)^{j-i} - (-1)^{j-i}\big) \\
&= b_j(\tilde{G}) - \sum_{i=0}^{j-1} \binom{v-i-1}{j-i} b_i(\tilde{G}) (-1)^{j-i+1} \\
&= (-1)^{j-j} \binom{v-j-1}{j-j} b_j(\tilde{G}) + \sum_{i=0}^{j-1} (-1)^{j-i} \binom{v-i-1}{j-i} b_i(\tilde{G}) \\
&= \sum_{i=0}^{j} (-1)^{j-i} \binom{v-i-1}{j-i} b_i(\tilde{G}).
\end{aligned}$$

Thus, Proposition 18 follows. □

We can also prove Proposition 18 in a combinatorial manner, i.e., by showing that (3.8) indeed counts the A_j-forests of a graph G. Since it is quite a complicated proof, it might be helpful to see the example at the end of this section, where we compute the coefficient c_4 of the graph presented in Figure 3.8, in the spirit of our combinatorial interpretation of (3.8).
Recall that computing the A_j-forests of a graph G is the same as computing the A_j-trees of a graph $\tilde{G}$. We begin with introducing a few observations which are crucial for understanding the equation (3.8).

1. For any forest F in $\tilde{G}$, there is a unique lexicographically minimal spanning tree containing F, which we denote by T_F.

2. It is not neccessarily true that the set of edges of F equals the set of internally passive edges of T_F, for instance, consider any forest F being a subgraph of $T_{\min}$, the lexicographically minimal spanning tree of $\tilde{G}$. Then $T_F = T_{\min}$ and $\mathrm{IP}(T_{\min}) = \varnothing$.

3. It is always true that $\mathrm{IP}(T_F) \subseteq E(F)$, since the edges in the set $E(T_F) \setminus E(F)$ are as minimal as possible.

Lemma 19. *Let F be a forest in a graph $\tilde{G}$. Then $E(F)$ is the set of internally passive elements for a spanning tree T of $\tilde{G}$ if and only if $T_F = T$ and there is no $F' \subsetneq F$, such that $T = T_{F'}$.*

Proof. If $E(F) = \mathrm{IP}(T)$, then the edges in $E(T) \setminus E(F)$ must be minimal, so that there are no more internally passive edges in T. Thus, $T_F = T$. Furthermore, if there is a forest

$F' \subsetneq F$, such that $T = T_{F'}$, then, as observed in 3, $\mathrm{IP}(T) \subseteq E(F')$. Since $E(F)$ contains more elements than $E(F')$, this is a contradiction to $E(F) = \mathrm{IP}(T)$. Conversely, consider an arbitrary spanning tree T of $\tilde{G}$. Then $T = T_{\mathrm{IP}(T)}$, because the non-internally passive edges must be as minimal as possible. Now let F be a forest in $\tilde{G}$, such that $T_F = T$. Then again, by observation 3, we know that $\mathrm{IP}(T) \subseteq E(F)$. If additionally there is no $F' \subsetneq F$, such that $T = T_{F'}$, the edges of the forest F must indeed be the internally passive edges of the spanning tree T. □

The idea of proving Proposition 18 is the following: we show that the right side of the equation (3.8) counts the A_j-trees of $\tilde{G}$ by counting the sets of $j-1$ edges of $\tilde{G}$, which are exactly the internally passive elements for a unique spanning tree of $\tilde{G}$. Precisely, from all forests of $\tilde{G}$ with $j-1$ edges, we subtract those, for which this is not the case, i.e., by Lemma 19, we remove every set of $j-1$ edges E, if for the forest F with $E = E(F)$ there is a subforest $F' \subsetneq F$ such that $E(F') = \mathrm{IP}(T_F)$.
Notice that if we reverse the order of summation in (3.8), we first subtract $b_{j-2}(\tilde{G})\binom{v-j-1}{1}$ from $b_{j-1}(\tilde{G})$, which is the number of all forests of $\tilde{G}$ with $j-1$ edges. It seems reasonable to interpret $b_{j-2}(\tilde{G})\binom{v-j-1}{1}$ as the number of forests of $\tilde{G}$ with $j-1$ edges, constructed in the following way: to every forest F_{j-2} with $j-2$ edges, we add an edge e, chosen from the remaining edges of the corresponding lexicographically minimal spanning tree $T_{F_{j-2}}$. Let F_e be the forest in $\tilde{G}$, such that $E(F_e) = E(F_{j-2}) \cup \{e\}$. Then $T_{F_e} = T_{F_{j-2}}$ and by Lemma 19 and observation 3, e is not internally passive in T_{F_e}. Since we construct such forests for all possible forests of $\tilde{G}$ on $j-2$ edges, we remove all forest with at least one edge that cannot be internally passive. However, the forests constructed this way are not unique, i.e., some forests are subtracted more then once. This problem is solved by the other summands of (3.8). Let us have a glimpse at the technical details.
Let F_i be a forest of $\tilde{G}$ with i edges and let T_{F_i} be the lexicographically minimal spanning tree of $\tilde{G}$ containing the edges of F_i. Consider a multiset $\mathcal{M}_i$ of sets of edges of $\tilde{G}$, defined as follows

$$\mathcal{M}_i := \bigcup_{F_i} \{\{E(F_i) \cup R\} | R \subseteq E(T_{F_i}) \setminus E(F_i) \wedge |R| = j-1-i\}.$$

In particular, the sets in $\mathcal{M}_0$ contain only the edges of the lexicographically minimal spanning tree $T_{\min}$ and the elements of $\mathcal{M}_{j-1}$ are sets of edges of all possible forests in $\tilde{G}$ with $j-1$ edges. First we observe that the cardinality of $\mathcal{M}_i$ is

$$|\mathcal{M}_i| = b_i(\tilde{G})\binom{v-i-1}{j-i-1},$$

since there are $b_i(\tilde{G})$ forests with i edges in $\tilde{G}$ and for every choice of such forest F_i, there are $\binom{v-i-1}{j-i-1}$ ways of choosing the $j-1-i$ edges of R from the $v-i-1$ remaining edges of $E(T_{F_i}) \setminus E(F_i)$. This means that we can write (3.8) as

$$c_j = \sum_{i=0}^{j-1} (-1)^{j-1-i} |\mathcal{M}_i|. \tag{3.11}$$

Moreover, the multiplicity of a set $\{E(F_i) \cup R\}$ in $\mathcal{M}_i$ is

$$\binom{j-i-1+|E(F_i) \setminus E(F')|}{j-i-1}, \tag{3.12}$$

where F' is a forest, such that $E(F') = \mathrm{IP}(T_{F_i})$. Indeed, we can express $\{E(F_i) \cup R\}$ as a disjoint union of sets $E(F')$, R and $E(F_i) \setminus E(F')$. Then for a forest F_i^* and $R^* \subseteq E(T_{F_i^*})$, the set $\{E(F_i^*) \cup R^*\}$ is equal to $\{E(F_i) \cup R\}$ if and only if $T_{F_i} = T_{F_i^*}$ and for $F' := \mathrm{IP}(T_{F_i}) = \mathrm{IP}(T_{F_i^*})$,

$$R^* \cup (E(F_i^*) \setminus E(F')) = R \cup (E(F_i) \setminus E(F')).$$

Since the number of possible choices for $j-1-i$ edges of R from $j-i-1+|E(F_i) \setminus E(F')|$ edges of $R \cup (E(F_i) \setminus E(F'))$ is (3.12), so is the multiplicity of $\{E(F_i) \cup R\}$ in $\mathcal{M}_i$. Notice that each set of $\mathcal{M}_{j-1}$ appears exactly once.
Furthermore, we observe that a set $\{E(F_i) \cup R\}$ can also appear in $\mathcal{M}_k$ for $k \neq i$. The next step needed for proving Proposition 18 combinatorially is checking how often a set E of $j-1$ acyclic edges of $\tilde{G}$, such that there is no spanning tree T with $\mathrm{IP}(T) = E$, is counted in (3.8) or equivalently in (3.11). Alternatively, by Lemma 19, if F_E is the forest such that $E = E(F_E)$, then these are exactly the sets for which there is a subset $E' \subsetneq E$ with $\mathrm{IP}(T_{F_E}) = E'$. Consequently, each set that we want to count is a disjoint union of a set of all internally passive edges for some spanning tree with at most $j-2$ internally passive edges and a subset of non-spare edges of this spanning tree. A more mathematical description of these sets can be found in the following lemma.

Lemma 20. *Let F_k be a forest of $\tilde{G}$ with $k \in \{0, 1, \ldots, j-2\}$ edges, such that $E(F_k) = \mathrm{IP}(T_{F_k})$. Then the count of the set of edges $\{E(F_k) \cup R\}$, such that $R \subseteq E(T_{F_k}) \setminus E(F_k)$ with $|R| = j-1-k$ in* (3.11) *sums up to zero.*

Proof. Notice that for $i \geq k$ a set $\{E(F_k) \cup R\}$ is an element of the multiset $\mathcal{M}_i$, since we can express it as $\{E(F_i) \cup R^*\}$, where $R^* \subset R$ with $|R^*| = j-1-i$ and $E(F_i) = E(F_k) \cup (R \setminus R^*)$. Then by (3.12), the multiplicity of $\{E(F_k) \cup R\}$ in $\mathcal{M}_i$ is

$$\binom{j-1-i+|E(F_i) \setminus E(F_k)|}{j-i-1} = \binom{j-i-1+i-k}{j-i-1} = \binom{j-1-k}{j-i-1}.$$

Summing over all $i \in \{k, k+1, \ldots, j-1\}$, the number of times that $\{E(F_k) \cup R\}$ is counted in (3.11) is

$$\sum_{i=k}^{j-1} (-1)^{j-1-i} \binom{j-1-k}{j-1-i} = \sum_{i=0}^{j-1-k} (-1)^i \binom{j-1-k}{i} = 0.$$

□

Now we are ready to prove Proposition 18.

Combinatorial proof of Proposition 18. Observe that all sets E of $j-1$ edges, such that there is a spanning tree T of $\tilde{G}$, with $\mathrm{IP} = E$, are elements of $\mathcal{M}_i$ if and only if $i = j-1$. Indeed, if $i < j-1$, then for every set $E := \{E(F_i) \cup R\} \in \mathcal{M}_i$, the set R, which does not contain internally passive edges of T_{F_i}, is non-empty. Hence, if F is a forest of $\tilde{G}$ with $E(F) = E$, then $T_F = T_{F_i}$ and by Lemma 19, the set E cannot be a set of internally passive edges for any spanning tree of $\tilde{G}$. Moreover, as mentioned before, the multiplicity of all elements of $\mathcal{M}_{j-1}$ is 1. Given a set $\{E(F_{j-1})\} \in \mathcal{M}_{j-1}$, either $E(F_{j-1}) = \mathrm{IP}(T_{F_{j-1}})$ or there is a subforest $F' \subseteq F$, such that $E(F') = \mathrm{IP}(T_{F_{j-1}})$. By Lemma 20, the sets from the second case cancel out in (3.11) and consequently also in (3.8). Hence, (3.8) only counts the sets of the first kind. For every such set $\{E(F_{j-1})\}$ there is a unique spanning tree $T_{F_{j-1}}$, with $|\mathrm{IP}(T_{F_{j-1}})| = j-1$. Conversely, by construction of $\mathcal{M}_{j-1}$, for any spanning tree T with $j-1$ internally passive edges, the set $\mathrm{IP}(T) \in \mathcal{M}_{j-1}$ with multiplicity 1. Thus, (3.8) indeed counts the A_j-trees of $\tilde{G}$. □

Example. (Computing the c-vector from the Ehrhart polynomial) Let G be a graph as illustrated in Figure 3.8. Then $G = \tilde{G}$. We compute the c-vector of G using (3.8). First

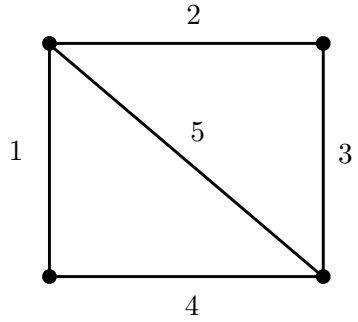

Figure 3.8: Graph G with four vertices and five edges. Since G is connected and has no cut-edges, the reduced graph of G is G itself. The graph G contains three cycles, a 4-cycle with edges $\{1, 2, 3, 4\}$ and two 3-cycles with edges $\{1, 4, 5\}$ and $\{2, 3, 5\}$.

we need to know the value of $b_i(G)$ for $i \in \{0, 1, 2, 3\}$. As mentioned before,

$$b_0(G) = 1, b_1(G) = |E(G)| = 5, b_2(G) = \binom{|E(G)|}{2} = 10$$

and if $\chi_3(G)$ is the number of 3-cycles of G, then

$$b_3(G) = \binom{|E(G)|}{3} - \chi_3(G) = \binom{5}{3} - 2 = 8.$$

Applying these results into (3.8), we get

$$\begin{aligned} c_1 &= 1, \\ c_2 &= 5 - \binom{3}{1} = 2, \\ c_3 &= 10 - \binom{2}{1}5 + \binom{3}{2} = 3 \quad \text{and} \\ c_4 &= 8 - \binom{1}{1}10 + \binom{2}{2}5 - \binom{3}{3} = 2. \end{aligned}$$

Notice that the results for c_1, c_2 and c_3 agree with Theorem 2. Let us see how the interpretation of (3.8) presented in the combinatorial proof of Proposition 18 works in the case of c_4. We determine the multisets $\mathcal{M}_i$. For $i = 3$, the elements of $\mathcal{M}_i$ are simply the sets of edges of all forests of G with 3 edges, which in this case are the spanning trees of G. Explicitly,

$$\mathcal{M}_3 = \{\{1, 2, 3\}, \{1, 2, 4\}, \{1, 2, 5\}, \{1, 3, 4\}, \{1, 3, 5\}, \{2, 3, 4\}, \{2, 4, 5\}, \{3, 4, 5\}\}.$$

We construct the elements of $\mathcal{M}_2$. First, we pair every subforest of $T_{\min}$ consisting of two edges with the remaining edge of $T_{\min}$. This yields three sets $\{1, 2, 3\}$. Now we consider the forests containing exactly one edge of $T_{\min}$. The corresponding sets of two edges are

$$\{1, 4\}, \{1, 5\}, \{2, 4\}, \{2, 5\}, \{3, 4\}, \{3, 5\}.$$

We complete each of these sets with the remaining edge of the lexicographically minimal spanning tree containing them and obtain

$$\{1, 2, 4\}, \{1, 2, 5\}, \{1, 2, 4\}, \{1, 2, 5\}, \{1, 3, 4\}, \{1, 3, 5\},$$

respectively. We observe that the sets $\{1,2,4\},\{1,2,5\}$ repeat each twice. The set of edges of the only remaining forest with 2 edges is $\{4,5\}$ and the corresponding lexicographically minimal spanning tree is $\{2,4,5\}$. This yields

$$\mathcal{M}_2 = \{\{1,2,3\},\{1,2,3\},\{1,2,3\},\{1,2,4\},\{1,2,4\},\{1,2,5\},\{1,2,5\},\{1,3,4\},\{1,3,5\},\{2,4,5\}\}.$$

Analogously,

$$\begin{aligned}\mathcal{M}_1 &= \{\{\{1\}\cup\{2,3\}\},\{\{2\}\cup\{1,3\}\},\{\{3\}\cup\{1,2\}\},\{\{4\}\cup\{1,2\}\},\{\{5\}\cup\{1,2\}\}\}\\ &= \{\{1,2,3\},\{1,2,3\},\{1,2,3\},\{1,2,4\},\{1,2,5\}\}.\end{aligned}$$

It is easy to see that

$$\mathcal{M}_0 = \{\{1,2,3\}\}.$$

To compute the number of occurences of each set of three acyclic edges of G in (3.8), we need to add their multplicity in $\mathcal{M}_3$, subtract their multplicity in $\mathcal{M}_2$, add their multplicity in $\mathcal{M}_1$ and subtract their multplicity in $\mathcal{M}_0$. The results are the following

$$\begin{aligned}\#\{1,2,3\} &= 1-3+3-1 = 0,\\ \#\{1,2,4\} &= 1-2+1 = 0,\\ \#\{1,2,5\} &= 1-2+1 = 0,\\ \#\{1,3,4\} &= 1-1 = 0,\\ \#\{1,3,5\} &= 1-1 = 0,\\ \#\{2,3,4\} &= 1,\\ \#\{2,4,5\} &= 1-1 = 0,\\ \#\{3,4,5\} &= 1.\end{aligned}$$

These results agree with our primar computation of c_4. Moreover, the sets of edges $\{2,3,4\}$ and $\{3,4,5\}$ are indeed the sets of edges of the only two A_4-trees of G, since all of their edges are internally passive, which is not true for any other spanning tree of G.

3.5 Combining the c-vectors

Recall that any two connected graphs G and G' can be X-connected for $X = \{x_1, x_2\}$, where $x_1 \in V(G)$ and $x_2 \in V(G')$, resulting in a graph GXG' with

$$E(GXG') = \{e \in E(G)\cup E(G') : e\cap X = \varnothing\}\cup\{e\backslash\{x_i\}\cup\{x\} : e \in E(G)\cup E(G')\wedge e\cap X = \{x_i\}\}.$$

As mentioned before the c-vector of GXG' is the same for any choice of X and hence in this chapter, for convenience, we denote a graph obtained by X-connecting G and G' for any set X by $G\circ G'$. Moreover, if T and T' are spanning trees of G and G', respectively, then in this thesis, $T\circ T'$ means that T and T' are X-connected with the same set X as $G\circ G'$.

Proposition 21. *If c is the c-vector of G and c' the c-vector of G', then the entry j of the c-vector of $G\circ G'$ is*

$$c^\circ_j = \sum_{i=1}^{j} c_i \cdot c'_{j-i+1}.$$

Before we prove this proposition, we prove the following lemma.

Lemma 22. *Let G and G' be graphs with spanning trees T and T', respectively. Then if T is an A_i-tree and T' an A_j-tree, $T \circ T'$ is an A_{i+j-1}-tree of $G \circ G'$.*

Example. Consider two graphs G and G' as presented in Figure 3.9 and a spanning trees T of G and T' of G', as in Figure 3.10. Then T is an A_3-tree, since $\mathrm{IP}(T) = \{4, 5\}$ and T' is an A_2-tree, since $\mathrm{IP}(T') = \{3'\}$. Then $T \circ T'$ is an A_4-tree of $G \circ G'$ with $\mathrm{IP}(T \circ T') = \{4, 5, 3'\}$ as depicted in Figure 3.11.

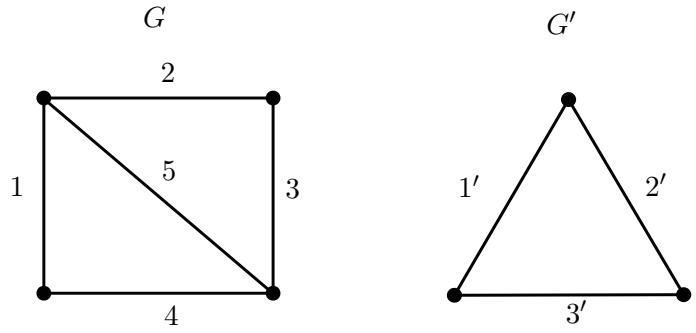

Figure 3.9: Graphs G and G'.

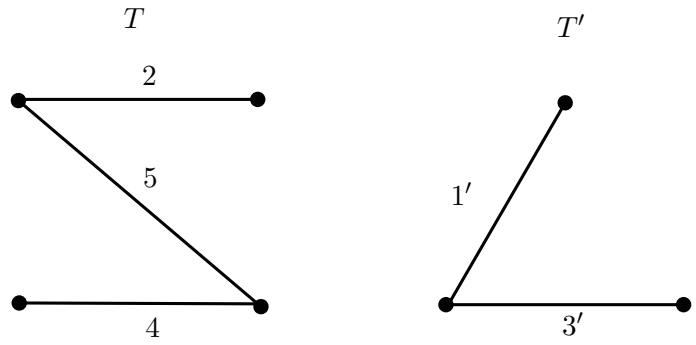

Figure 3.10: An A_3-tree T of graph G and an A_2-tree T' of graph G'.

Proof. Let T be an A_i-tree of G and T' an A_j-tree of G'. In the proof of Proposition 12 we showed that $T \circ T'$ is a spanning tree of $G \circ G'$. We still need to show that it is an A_{i+j-1}-tree. For this, notice that $|\mathrm{IP}(T \circ T')| = |\mathrm{IP}(T)| + |\mathrm{IP}(T')| = i-1+j-1 = i+j-2$ and hence $T \circ T'$ is indeed an A_{i+j-1}-tree of $G \circ G'$. □

Proof of Proposition 21. We recall that the spanning trees of a graph $G \circ G'$ are exactly all possible results of connecting spanning trees of G and G'. Since c_j° counts the spanning trees with $j-1$ internally passive elements, by Lemma 22, we need to see how many combinations of spanning trees of G with spanning trees of G' give us an A_j-tree. To compute this we sum up all products $c_{a_1} \cdot c'_{a_2}$ where a_1, a_2 are nonegative integers, such that $a_1 + a_2 = j+1$. All products satisfying these conditions are $c_1 \cdot c'_j$, $c_2 \cdot c'_{j-1}, \ldots, c_j \cdot c'_1$ and the claim follows.

□

Remark. Observe that we can use Proposition 21 to compute the c-vector of any disconnected graph, if we know the c-vectors of each of its components.

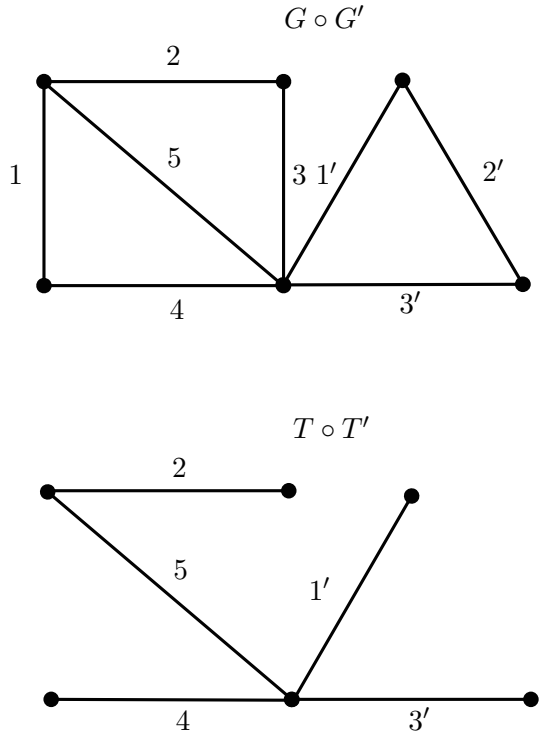

Figure 3.11: $T \circ T'$ is an A_4-tree of $G \circ G'$.

The simplest class of a graphs on which Proposition 21 can be applied are cactus graphs, introduced in Section 4.2.

4. Special cases

In Chapter 3, we conclude that computing the coefficient c_j of the c-vector of a graph G for $j > 4$ is challenging without knowledge of the number of forests of G with i edges for $i \in \{0, 1, \ldots, j-1\}$. However, there are certain kinds of graphs, for which this task is simpler, than for the others. In this chapter we introduce three of them and derive methods for computing all entries of their c-vectors.

4.1 Graphs with a single cycle

First we take a look at the easiest case: graphs with a single cycle. Notice that the cycle in such graph must be simple, because if there was a node repeating itself, we could tile this cycle into at least two cycles.

Proposition 23. *Let G be a graph containing exactly one cycle. Then the c-vector of Z_G is $\mathbf{1} \in \mathbb{N}^v$, where v is the number of nodes in this cycle.*

Proof. Notice that the reduced graph $\tilde{G}$ of G is a simple cycle. Hence, for every edge e of $\tilde{G}$ there is exactly one spanning tree T_e which does not contain e and conversely, for any spanning tree T there is exactly one edge e_T not contained in T. Let T_e be a spanning tree not containg the edge e and m_e the number of edges with higher labels than e. Then T_e has m_e internally passive elements, since all the edges with labels bigger than e can be replaced with e in order to get a lexicographically smaller spanning tree. Furthermore, consider the largest edge smaller than e, say e^*. Then there are exactly $m_e + 1$ edges with labels larger than e^*. We can repeat this until we arrive at the smallest edge $e_{\min}$, which has $|E(\tilde{G})| - 1 = v - 1$ edges larger than itself. Since the largest edge of $\tilde{G}$ has no edges bigger than itself, we conclude that for every number of internally passive elements $m \in [0, v-1]$ there is a unique edge e_m, so that there are exactly m edges with larger label than e_m and automatically exactly one spanning tree T_{e_m}, which is an A_{m+1}-tree. Hence, $c_1 = c_2 = \cdots = c_v = 1$, which proves the proposition. □

4.2 Cactus graphs

We now progress to the next level of difficulty: **cactus forests** defined as follows.

Definition. A **cactus forest** is a graph where any two cycles have at most one node in common. If the graph is connected, we call it simply a **cactus graph**. We refer to the simple cycles of such graphs as **sections**.

Since the reduced graph of any cactus forest is a cactus graph, we consider only the latter. An example of a cactus graph with sections s_1, s_2 and s_3 is depicted in Figure 4.1.
It turns out that determining the c-vector of a cactus graph can be interpreted as a slight modification of a famous distribution problem **weak composition**, described with more detail in [6].

Definition. A sequence $(a_1, a_2, \ldots, a_t)$ of integers fulfilling $a_j \geq 0$ for all j and $a_1 + a_2 + \cdots + a_t = b$ is called a **weak composition** of b into t parts.

Proposition 24. *Let G be a cactus graph with sections $s_1, s_2, \ldots, s_n$. Then c_j is the number of weak compositions of $j-1$ into n parts $m_1, m_2, \ldots, m_n$, so that $m_i \leq |E(s_i)-1)|$.*

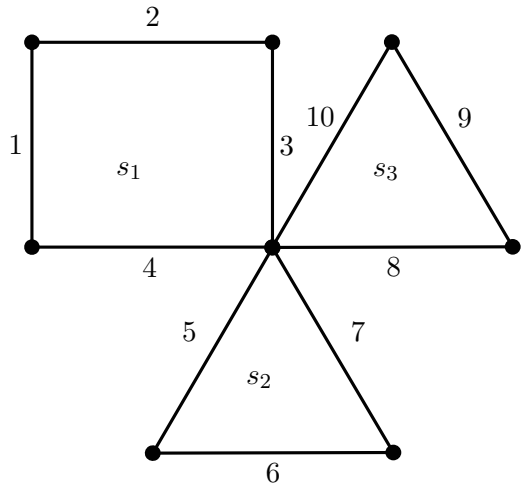

Figure 4.1: Cactus graph with three sections.

Example. Let G be a cactus graph as depicted in Figure 4.1. Table 4.1 presents all A_4-trees of G and corresponding weak compositions (m_1, m_2, m_3), such that $m_1+m_2+m_3 = 3$ and $0 \leq m_i \leq |E(s_i) - 1)|$.

A_4-tree of G	internally passive edges from s_1	internally passive edges from s_2	internally passive edges from s_3	corresponding weak composition
1 2 3 5 7 9 10	-	7	9,10	(0,1,2)
1 2 3 6 7 8 10	-	6,7	10	(0,2,1)
1 2 4 5 6 9 10	4	-	9,10	(1,0,2)
1 2 4 5 7 8 10	4	7	10	(1,1,1)
1 2 4 6 7 8 9	4	6,7	-	(1,2,0)
1 3 4 5 6 8 10	3,4	-	10	(2,0,1)
1 3 4 5 7 8 9	3,4	7	-	(2,1,0)
2 3 4 5 6 8 9	2,3,4	-	-	(3,0,0)

Table 4.1: The A_4-trees of the cactus graph G and the corresponding weak compositions.

Remark. The weak compositions satisfying the conditions of Proposition 24 are a special case of second-order restricted weak compositions described in [14].

Proof. When we construct a tree in a cactus graph, for every section s_i we remove exactly one edge e_i. Similarly as in the proof of Proposition 23, for every edge $e_i \in s_i$, if there are m_i edges in s_i with larger label then e_i, all these edges can be exchanged for e_i in order to get a lexicographically smaller spanning tree and hence the number of internally passive elements in a spanning tree is $m_1 + m_2 + \cdots + m_n$. Now if we choose a natural number j and want to know how many A_j-trees there are, we have to count how many possible solutions there are for

$$m_1 + m_2 + \cdots + m_n = j - 1,$$

with $0 \leq m_i \leq |E(s_i)| - 1$. □

Applying the algorithm presented in [14] to our case, we obtain the list of all weak compositions of $j-1$ restricted in terms of Proposition 24. Then c_j is the length of the list. According to [14], the worst-case time-complexity of the algorithm is $O((j-1)\binom{n+j-2}{n-1})$. The coefficient c_j can also be computed recursively using depth-first-search, for example by implementing the pseudocode Algorithm 4.1. This approach is not as efficient as the algorithm from [14], but provides intuitive insights. It can be optimized using dynamic programming, i.e. by memorizing solutions to sub-problems.

```
Input:  an integer j and an integer list s.
Function weakCompositions(j-1,s)
  Set n = len(s), c=0
  Function dfs(p,currentSum)
    If currentSum==j-1 Then
      Increment c
      Return
    Endif
    If p==n Or currentSum>j-1 Then
      Return
    Endif
    For i=0 To s[p]-1 Do
      Call dfs(p+1,currentSum+i)
    Endfor
  Endfunction
  Call dfs(0,0)
  Output: c
Endfunction
```

Algorithm 4.1: A pseudo code which computes the coefficient c_j of the c-vector of a cactus graph.

Here is a step-by-step explanation of how the code in the Algorithm 4.1 works:

1. $weakCompositions(j-1, s)$ is the main function that calculates the number of valid weak compositions of $j-1$ using non-negative integers $m_1, m_2, \ldots, m_n$, so that m_k is smaller than the entry k of s. In the cactus graph world, s is a list of the sizes of sections of the cactus graph G and the function $weakCompositions(j-1, s)$ counts the A_j-trees of G.

2. c is initialized to 0. This variable stores the count of valid compositions. At the end c is the value of c_j.

3. The dfs function is defined within $weakCompositions$. It takes two arguments: p and $currentSum$. The variable $p \in [n]$ represents the current position in the list s, and $currentSum$ represents the running sum of integers chosen so far.

4. Inside the dfs function: "If $currentSum == j-1$": checks if the current running sum equals the target $j-1$. If it does, it means we have found a valid composition, so we increment the result by 1."If $p == n$ Or $currentSum > j-1$:" checks if we have reached the end of the list s or if the current running sum has exceeded $j-1$. If either condition is met, we return from the function, as further exploration of this branch cannot lead to a valid composition. The loop "For $i \in [0, s[p]-1]$:" iterates through all possible values of m_p from 0 to $s[p]-1$.

5. Inside the loop, $dfs(j+1, currentSum+i)$ is called recursively to explore compositions that include the current value i from the list s. The index j is incremented to consider the next integer from s, and the $currentSum$ is updated by adding i.

6. Finally, the main function *weakCompositions* initiates the depth-first-search by calling $dfs(0,0)$ with the initial index j and *currentSum* set to zero. The function returns the value of result, which represents the total number of valid compositions.

It is also possible to compute the c-vector of a cactus graph without linking this problem to restricted weak compositions. Namely, we can use the method described in Section 3.5. We present an example illustrating how combining the c-vectors can be used to compute the c-vector of a cactus graph.

Example. Let G be a graph, as presented in Figure 4.1 . Then G is a possible outcome of connecting three cycle graphs, C^*, C^{**}, C^{***}, such that C^* has four edges, and C^{**} and C^{***} have three edges each, in an arbitrary order, for example: $(C^* \circ C^{**}) \circ C^{***}$ By Proposition 23, the c-vector of C^*, denoted by c^*, is $\mathbf{1} \in \mathbb{N}^4$, and the c-vectors of C^{**} and C^{***}, denoted by c^{**} and c^{***}, respectively, are both $\mathbf{1} \in \mathbb{N}^3$. First we compute the c-vector of $C^* \circ C^{**}$, denoted by c°, using Proposition 21.

$$
\begin{aligned}
c_1^\circ &= 1,\\
c_2^\circ &= \sum_{i=1}^{2} c_i^* \cdot c_{j-i+1}^{**} = c_1^* \cdot c_2^{**} + c_2^* \cdot c_1^{**} = 1+1 = 2,\\
c_3^\circ &= \sum_{i=1}^{3} c_i^* \cdot c_{j-i+1}^{**} = c_1^* \cdot c_3^{**} + c_2^* \cdot c_2^{**} + c_3^* \cdot c_1^{**} = 1+1+1 = 3,\\
c_4^\circ &= \sum_{i=1}^{4} c_i^* \cdot c_{j-i+1}^{**} = c_1^* \cdot c_4^{**} + c_2^* \cdot c_3^{**} + c_3^* \cdot c_2^{**} + c_4^* \cdot c_1^{**} = 0+1+1+1 = 3,\\
c_5^\circ &= \sum_{i=1}^{5} c_i^* \cdot c_{j-i+1}^{**} = c_3^* \cdot c_3^{**} + c_4^* \cdot c_2^{**} = 1+1 = 2,\\
c_6^\circ &= \sum_{i=1}^{6} c_i^* \cdot c_{j-i+1}^{**} = c_4^* \cdot c_3^{**} = 1.
\end{aligned}
$$

Now we compute the c-vextor of $(C^* \circ C^{**}) \circ C^{***}$, denoted by $c^{\circ\circ}$,

$$
\begin{aligned}
c_1^{\circ\circ} &= 1,\\
c_2^{\circ\circ} &= c_1^\circ \cdot c_2^{***} + c_2^\circ \cdot c_1^{***} = 1+2 = 3,\\
c_3^{\circ\circ} &= c_1^\circ \cdot c_3^{***} + c_2^\circ \cdot c_2^{***} + c_3^\circ \cdot c_1^{***} = 1+2+3 = 6,\\
c_4^{\circ\circ} &= c_1^\circ \cdot c_4^{***} + c_2^\circ \cdot c_3^{***} + c_3^\circ \cdot c_2^{***} + c_4^\circ \cdot c_1^{***} = 0+2+3+3 = 8,\\
c_5^{\circ\circ} &= c_3^\circ \cdot c_3^{***} + c_4^\circ \cdot c_2^{***} + c_5^\circ \cdot c_1^{***} = 3+3+2 = 8,\\
c_6^{\circ\circ} &= c_4^\circ \cdot c_3^{***} + c_5^\circ \cdot c_2^{***} + c_6^\circ \cdot c_1^{***} = 3+2+1 = 6,\\
c_7^{\circ\circ} &= c_5^\circ \cdot c_3^{***} + c_6^\circ \cdot c_2^{***} = 2+1 = 3,\\
c_8^{\circ\circ} &= c_6^\circ \cdot c_3^{***} = 1.
\end{aligned}
$$

4.3 Fan graphs

In this section we present **fan graphs**, the simplest class of graphs among the graphs with edges belonging to more than one cycle.

Definition. A **fan graph** F_k is the join of a single vertex o and a path with $k+1$ vertices where $k \geq 1$.

Remark. In the literature we can find a more general definition of a fan graph where the single vertex o is replaced by an empty graph on $n \in \mathbb{N}$ nodes.

For this work we fix an embedding of a fan graph where the path lies horizontally above the vertex o as presented in Figure 4.2.

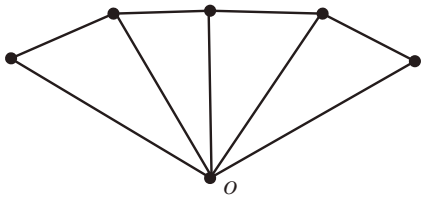

Figure 4.2: Fan graph F_4.

Since the c-vector does not depend on the labeling of the edges of a graph, we label the edges containing o with numbers from 1 to $k+1$ from left to right and the edges not containing o with numbers from $k+2$ to $2k+2$ also from left to right as presented in Figure 4.3. With this labeling, the edges from 1 to $k+1$ form the lexicographically minimal spanning tree of a fan graph F_k, denoted as before by $T_{\min}$, and the edges from $k+2$ to $2k+2$ are the spare edges of F_k.

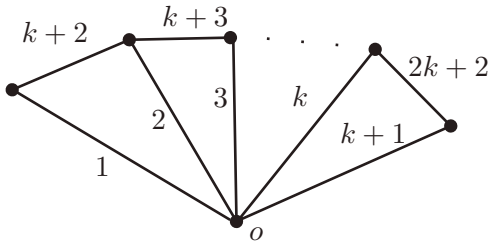

Figure 4.3: Labeling of F_k.

Definition. For $j \in [k]$, we call the 3-cycle $\Delta_j \subseteq F_k$ with edges $j, j+1, k+j+1$ a **triangle** of F_k. Moreover, if $j+1 \in [k]$, we say that Δ_j and Δ_{j+1} are **neighbouring**.

Definition. We call a subgraph $B \subseteq F_k$ a **block** of F_k of **size** i, if there exist i consecutive triangles $\Delta_j, \ldots, \Delta_{j+i-1}$ of F_k such that $B = \bigcup_{\ell=0}^{i-1} \Delta_{j+\ell}$.

See Figure 4.4 for an example of a block.

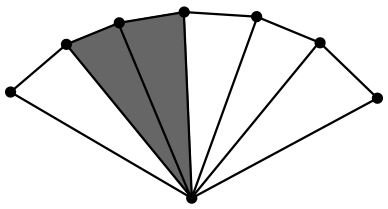

Figure 4.4: The shaded area represents a block of F_6 which is the union of triangles Δ_2 and Δ_3.

If the triangles are not necessarily consecutive arrive at the following more general definition.

Definition. We call a subgraph $K \subseteq F_k$ a **cluster** of F_k, if K is a union of triangles of F_k. Moreover, we call a block $B \subseteq K$ of F_k a **maximal block** of K if there is no block $B' \subseteq K$ such that $B \subseteq B'$.

Remark. In the following subsections we think of clusters rather as unions of their maximal blocks. For convenience we refer to the maxmimal blocks simply as blocks.

See Figure 4.5 to understand the concept of a cluster and the difference between a block and a maximal block. Other examples of clusters are presented in Figure 4.6 and Figures 4.11-4.16.

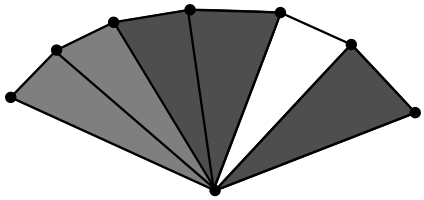

Figure 4.5: The shaded area represents a cluster K of F_6. There are two maximal blocks of K, namely B_1, which is the union of triangles $\Delta_1, \Delta_2, \Delta_3$ and Δ_4 and B_2, which is just the triangle Δ_6. The union of triangles Δ_1 and Δ_2, shaded with a brighter colour, is a block of F_6, but not a maximal block of the cluster K, since it is contained in the block $B_1 \subset K$.

We recall the definition of an A_j-tree, which is a spanning tree T of a graph G such that T has $i-1$ internally passive edges, where an edge e is called internally passive if there exists an edge $e' \notin T$ so that replacing e by e' results in a lexicographically smaller spanning tree. Moreover the entry c_j of the c-vector of any graph is the number of A_j-trees of this graph. Since the goal of this section is to determine the c-vector of a fan graph F_k, we need to investigate its spanning trees. First we consider an interesting relation between the clusters and the spanning trees of F_k.

4.3.1 Trees from a cluster

Observe that any cluster K of F_k is uniquely determined by a subset of the spare edges and vice versa. Indeed, for any subset E_s of spare edges we can construct a cluster K containing exactly these spare edges by taking the join of the graph induced by E_s and the vertex o. Conversely, any cluster K of F_k is the union of triangles, each of which uniquely determines the corresponding spare edge, i.e., the spare edge $k+j+1$ corresponds to the the triangle Δ_j. Thus, there is a bijection between the subsets of spare edges and the clusters of F_k.

Definition. We say that a spanning tree T of F_k is **produced** from a cluster K if it contains all spare edges belonging to K and no other spare edge of F_k.

Remark. It is important to notice that a spanning tree produced from a cluster K is not unique. Moreover, every edge $e \in T_{\min} \setminus K$ must be in all spanning trees produced from K, since this is the only way to reach the vertices of $F_k \setminus K$ without using the spare edges that are not in K.

Before we show an example ilustrating the above definition, we state Proposition 25.

Proposition 25. *Let K be a cluster of a fan F_k with blocks $B_1, B_2, \ldots, B_m$. Let t_ℓ denote the number of triangles of B_ℓ and $t := \sum_{\ell=1}^{m} t_i$ the total number of triangles in K. Then, for $0 \leq x \leq m$, the number of A_{t+x+1}-trees produced from K is*

$$\sum_{M \in \binom{[m]}{x}} \prod_{j \in M} t_j. \tag{4.1}$$

Moreover, the number of A_{t+x+1}-trees produced from K is zero if $x > m$.

In the following example we consider a cluster K of F_6, construct all possible spanning trees produced from K and show that Proposition 25 holds for K.

Example. Let K be a cluster of F_6 with two blocks B_1 and B_2 as presented in Figure 4.6. Note that here $t_1 = 2$ and $t_2 = 1$.

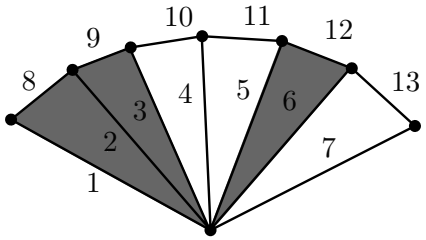

Figure 4.6: The shaded triangles represent cluster K with two blocks: B_1 consisting of triangles Δ_1 and Δ_2 and B_2 consisting of a single triangle Δ_5. The set of spare edges corresponding to this cluster is $\{8, 9, 12\}$.

By definition, every spanning tree produced from K contains the edges 8,9 and 12. Hence, every spanning tree produced from K has at least three internally passive elements. This means that the lexicographically minimal spanning tree of F_6 that can be produced from K is an A_4-tree and besides 8,9 and 12 contains only minimal edges of the blocks of K, which are 1 for B_1 and 5 for B_2, and the edges of $T_{\min} \setminus K$, which are 4 and 7. This tree is presented in Figure 4.7.

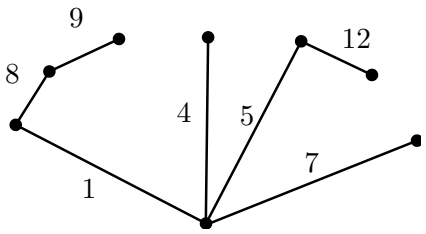

Figure 4.7: The only A_4-tree produced from K.

An A_5-tree must have four internally passive edges. Since three of them are the spare edges 8,9 and 12, we must replace either 1 or 5 by some larger edge from the same block. There are three possibilities to do this, i.e., we can replace 1 with 2 or 3, or 5 with 6. This gives us three different A_5-trees as ilustrated in Figure 4.8.
Finally, we consider all A_6-trees that can be produced from K. These cannot contain the minimal edge of neither of blocks. This leaves edge 2 or 3 as the choice for block B_1 and edge 6 as the only choice for block B_2. The two resulting A_6-trees are depicted in Figure 4.9.
Notice that the trees described in this example are the only spanning trees of F_6 containing exactly the spare edges 8,9, and 12 and hence for $i > 6$ there are no A_j-trees produced from K.

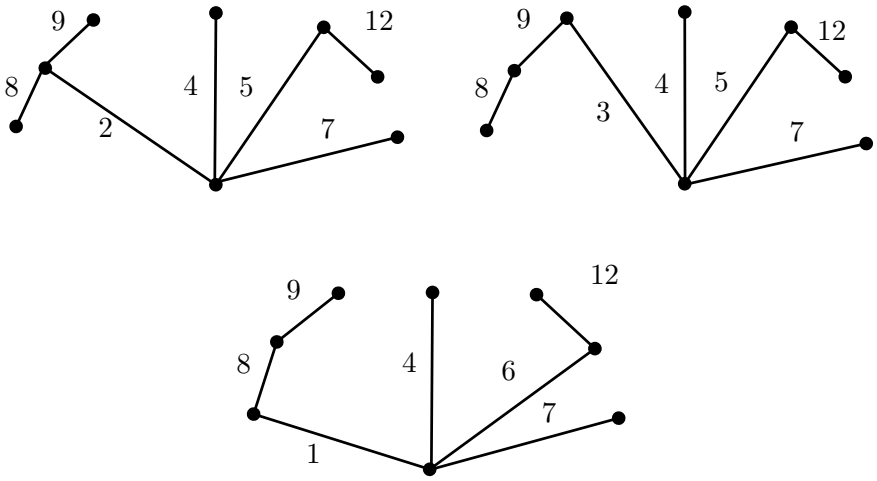

Figure 4.8: Three A_5-trees produced from K.

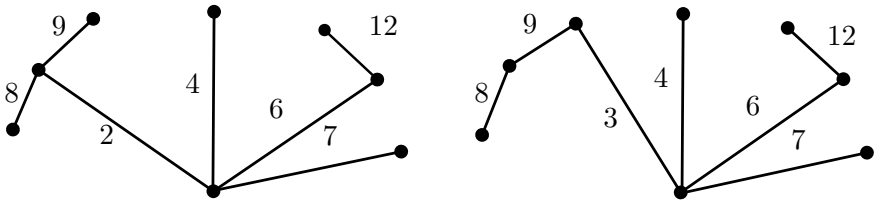

Figure 4.9: Two A_6-trees produced from K.

Now we are ready to prove Proposition 25.

Proof. We count the number of A_{t+x+1}-trees produced from a cluster K. By definition an A_{t+x+1}-tree contains $t+x$ internally passive edges. Notice that the spare edges are always internally passive and there are t of them. Hence, the remaining x internally passive edges are non-spare edges. Note that any spanning tree produced from K contains exactly one non-spare edge from each block of K. Otherwise there would be a cycle. Thus, to obtain a total of x internally passive non-spare edges, we choose exactly x blocks from each of which we choose one edge. Since any block B_i contains t_i triangles, it has t_i+1 non-spare edges, one of which is minimal and hence cannot be internally passive. Replacing any of the other t_i edges by the minimal edge results in a lexicographically smaller spanning tree of F_k and therefore yields a valid choice for an internally passive edge. Consequently, for any choice of x blocks $B_{j_1}, \ldots, B_{j_x}$, we have $t_{j_1} \cdots t_{j_x}$ possible A_{l+x+1}-trees. Summing this up for any x-element subset of blocks, we obtain (4.1). □

Knowing Proposition 25, a natural question is how many such clusters there are.

4.3.2 Types of clusters

This subsection is devoted to categorising the clusters of a given fan graph. For this we define **types** of clusters.

Definition. Let K be a cluster with blocks $B_1, B_2, \ldots, B_m$ of sizes $t_1, t_2, \ldots, t_m$, respectively. We say that K is of **type** $t_{\sigma(1)}t_{\sigma(2)}\ldots t_{\sigma(m)}$, where σ is a permutation of $[m]$, such that $t_{\sigma(1)} \geq t_{\sigma(2)} \geq \ldots \geq t_{\sigma(m)}$.

See an example of two different clusters of the same type in Figure 4.10.

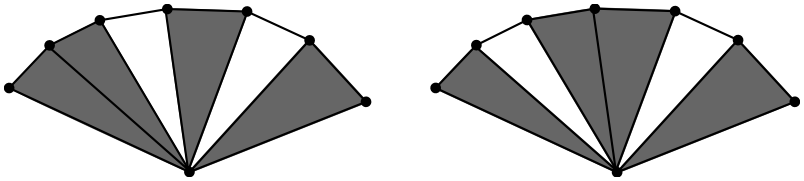

Figure 4.10: Both of these clusters are of type 2 1 1.

Remark. By this definition, if two distinct clusters K and K' are of the same type, they consist of the same number of triangles t and the same number of blocks m. Moreover, there is a bijection f between K and K', such that

$$B'_j = f(B_i) \implies t'_j = t_i,$$

where B_i is a block of K of size t_i and B'_j is a block of K' of size t'_j for $i, j \in [m]$. Thus, by Proposition 25, K and K' produce the same amount of A_{t+x+1} trees, for $0 \leq x \leq m$. Since K and K' are distinct, they must contain different triangles and consequently different spare edges. Hence, there is no spanning tree which can be produced from both of them.

We want to know how many clusters of a given type there are in a fan F_k. To address this question it seems natural to think about recursions: whenever we choose the leftmost block of a cluster K, a smaller fan remains, from which we pick the other blocks of K. First we consider the special cases, which are easier to compute and at the end of this subsection we derive a formula for all possible clusters.

1. **One-block clusters**

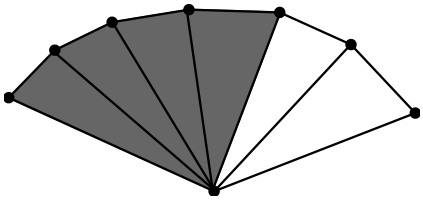

Figure 4.11: The shaded area represents a one-block cluster of F_6 of size 4.

Firstly, we consider the one-block clusters of size t in F_k, as presented in Figure 4.11. To investigate how many such clusters there are, we simply count how many choices of the leftmost triangle, i.e., the triangle with the lowest index, of the block there are. It is easy to see that this number is $k - t + 1$, since all but last $t - 1$ triangles can be chosen for this purpose.

2. **Same-sized blocks**

   (a) **One-triangle blocks**
   Let S_m^k be the number of clusters of a fan F_k with m one-triangle blocks, i.e., m non-neighbouring triangles, as shown in Figure 4.12. We claim that

$$S_m^k = \sum_{j=1}^{k-2} S_{m-1}^j. \tag{4.2}$$

   Indeed, if a cluster contains Δ_1, we can choose the remaining $m - 1$ triangles from F_{k-2}, since Δ_1 is already taken and choosing Δ_2 results in a block of size

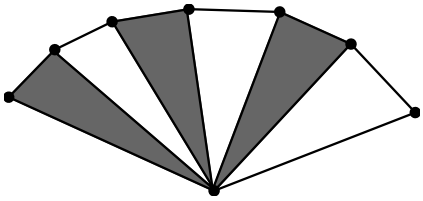

Figure 4.12: The shaded area represents a cluster of F_6 with three one-triangle blocks.

2. This yields S_{m-1}^{k-2} different clusters. By the same logic, if the first block is Δ_2 we have S_{m-1}^{k-3} possibilities of choosing the other triangles, see an example in Figure 4.13. We repeat this procedure until the leftmost triangle of the cluster is Δ_{k-2m+2} and we have exactly one possibility of placing the remaining $m-1$ triangles in F_{2m-3}. Since $S_{m-1}^{j} = 0$ for $j < 2m-3$, the equation (4.2) holds. Note that for $m = 1$ and $m = 2$ the computations are quite simple, namely, case 1 implies $S_1^k = k$ and consequently, S_2^k is the triangular number T_{k-2}:

$$S_2^k = \sum_{j=1}^{k-2} S_1^j = 1 + 2 + \cdots + (k-2) = \binom{k-1}{2}.$$

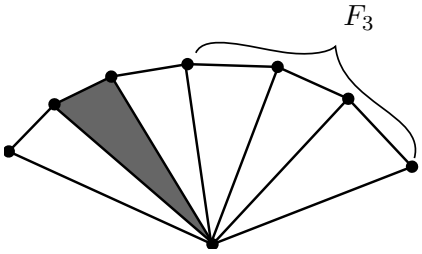

Figure 4.13: The shaded triangle is the already chosen triangle of the cluster. The remaining triangles must be chosen from the fan F_3.

(b) **Multiple-triangle blocks**

We can derive an analogous recursion for clusters with blocks containing more than one triangle. Let $S_{t,m}^k$ be the number of clusters with m blocks of size t. Thus, the case 2a is a special case where $t = 1$. Then

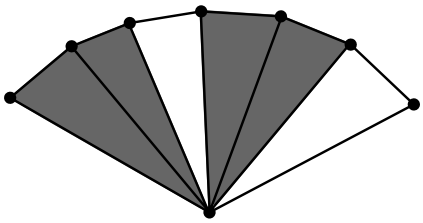

Figure 4.14: The shaded area represents a cluster of F_6 with two blocks of size 2 each.

$$S_{t,m}^k = \sum_{j=1}^{k-t-1} S_{t,m-1}^j. \tag{4.3}$$

In contrary to the case 2a, now the leftmost block is of size t. Consequently, the largest fan graph, from which we choose the remaining blocks is F_{k-t-1} and therefore the largest index j in the sum of (4.3) is $k-t-1$. Notice that in order to construct a cluster with $m-1$ blocks each of size t, the fan graph must consist of at least $t(m-1)+m-2$ triangles. Thus, the summands of (4.3)
for $j < t(m-1)+m-2$ are 0. Besides (4.3), we can express $S^k_{t,m}$ in terms of the case 2a. Namely

$$S^k_{t,m} = S^{k-m(t-1)}_m. \tag{4.4}$$

Indeed, from every one-triangle-block cluster we can build exactly one t-triangle block cluster by adding $t-1$ triangles to each of its blocks. This increases the number of triangles of the fan by $m(t-1)$. Conversely, from every cluster of a fan F_k with m blocks of size t each, we can obtain a unique one-triangle-block cluster of the fan $F_{k-m(t-1)}$, by removing $t-1$ triangles from each block as presented in Figure 4.15.

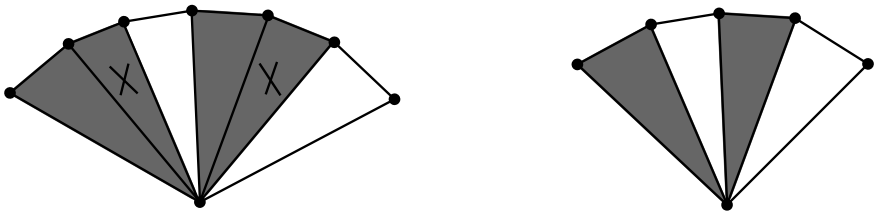

Figure 4.15: The shaded area of the left figure represents a cluster of type 2 2 of F_6, with triangles Δ_1 and Δ_2 forming one block and triangles Δ_4 and Δ_5 forming the other. The triangles Δ_2 and Δ_5 are marked with X, since they are being removed in order to obtain the 1 1 cluster of F_4 with triangles Δ_1 and Δ_3 shaded in the right figure.

3. **Two blocks of different sizes**
Let $M^k_{t_1,t_2}$ be the number of two-block clusters, where one block is of size t_1 and the

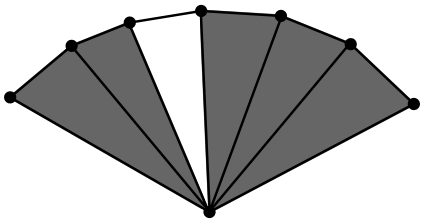

Figure 4.16: The shaded area represents a cluster of F_6 of type 3 2.

other is of size t_2, so that $t_1 \neq t_2$. First we count the clusters where the block with t_1 triangles is on the left and the one with t_2 triangles is on the right. If the left block consists of the triangles $\Delta_1, \ldots, \Delta_{t_1}$, there are $k-(t_1+1)-(t_2-1) = k-t_1-t_2$ possible ways of picking the leftmost triangle of the right block, since we cannot take the triangles $\Delta_1, \ldots, \Delta_{t_1+1}$, nor the triangles $\Delta_{k-t_2+1}, \ldots, \Delta_k$. Every shift of the left block by one triangle to the right, decreases the number of possibilities of building the right block by one. Hence, altogether we get $\binom{k-t_1-t_2+1}{2}$ clusters with left block of size t_1 and right block of size t_2. Analogously, there are $k-t_2-t_1$

clusters with the block of size t_2 on the left and the block of size t_1 on the right. Thus,

$$M^k_{t_1,t_2} = 2\binom{k-t_1-t_2+1}{2}.$$

Observe that $M^k_{t_1,t_2} = M^k_{t_2,t_1}$.

While the above methods might be convenient to apply for some cases, we need a general formula which can be used for an arbitrary cluster.

Proposition 26. *Let $M^k_{(t_1,m_1),\dots,(t_p,m_p)}$ be the number of clusters of F_k with m_i blocks of size t_i for $i \in [p]$, so that $t_1 > \cdots > t_p$. Then, for $m := m_1 + \cdots + m_p$,*

$$M^k_{(t_1,m_1),\dots,(t_p,m_p)} = \binom{m}{m_1,\dots,m_p} S_m^{k-\sum_{j=1}^p m_i(t_i-1)}. \tag{4.5}$$

Proof. Note that similarly to (4.4), we reduce $M^k_{(t_1,m_1),\dots,(t_p,m_p)}$ to the case 2a of clusters consisting of m one-triangle blocks. To that end, let K be a cluster of $F_{k-\sum_{j=1}^p m_i(t_i-1)}$ consisting of m one-triangle blocks. Analogously to case 2a, we choose m_1 of these blocks and add $t_1 - 1$ triangles to each of them. Then we choose m_2 of the remaining $m - m_1$ one-triangle blocks and add $t_2 - 1$ triangles to each of them and so on, until we obtain a cluster of F_k with m_i blocks of size t_i for all $i \in [p]$. Thereby, we construct $\binom{m}{m_1,\dots,m_p}$ different clusters.
Moreover, every cluster of F_k with m_i blocks of size t_i is constructed this way and (4.5) follows.

□

4.3.3 Which cluster types for which k?

The main concept of this subsection is the concept of **integer partition**. For more details and examples see [6].

Definition. Let $t_1 \geq t_2 \geq \cdots \geq t_m \geq 1$ be integers so that $t_1 + t_2 + \cdots + t_m = t$. Then the sequence $(t_1, t_2, \dots, t_m)$ is called a **partition** of the integer t.

Given a fan F_k we would like to know which types of clusters can be constructed within F_k, since with this information, using the methods introduced before, we can compute how many clusters of each of these types there are and which trees can be produced from them which is necessary to determine the c-vector of F_k.

Proposition 27. *The types of clusters that can be built within a fan F_k are in bijection with the partitions of integers from m to $k - m + 1$ into m parts for $m \in [\lceil \frac{k}{2} \rceil]$.*

Proof. We can think of the cluster types as integer partitions: Consider the cluster type $t_1\, t_2\, \dots\, t_m$ and let $t := t_1 + t_2 + \cdots + t_m$ be the total number of triangles in any cluster of this type. Then $t_1 \geq t_2 \geq \cdots \geq t_m \geq 1$ and hence $(t_1, t_2, \dots, t_m)$ is a partition of t into m parts. Analogously, any partition $(t_1, t_2, \dots, t_m)$ corresponds to a unique cluster type $t_1\, t_2\, \dots\, t_m$. Furthermore we observe that a cluster of F_k with m blocks can have at most $k - (m-1) = k - m + 1$ triangles, since between every two blocks there must be at least one triangle separating them. We conclude that the maximal number of blocks a cluster of F_k can have is $\lceil \frac{k}{2} \rceil$. Also it is clear that t must be at least as large as m, since we need to have at least one triangle in each block. □

This means that given a fan graph F_k, we need to list all the partitions satisfying the conditions from Proposition 27 in order to compute its c-vector. As in Section 4.2 we can use a recursive approach to determine the desired restricted integer partitions, which again is not the optimal solution. Faster algorithms are the subject of [18].

```
Input:  an integer t and an integer m.
Function generate_partitions(t, m)
  Set partitions = []
  Function dfs(remaining, parts)
    If remaining==0 And length of parts==m Then
      Append parts To partitions
      Return
    Endif
    If remaining<0 Or length of parts==m Then
      Return
    Endif
    If parts Is Empty
      Set start=1
    Else
      Return parts[-1]
    Endif
    For i=start To remaining Do
      Call dfs(remaining - i, parts + [i])
    Endfor
  Endfunction
  Call dfs(t,[])
  Output: partitions
Endfunction
```

Algorithm 4.2: Partitions of t into m parts.

Here is a step by step description of the algorithm.

1. Initialization: We initialize an empty list called *partitions*. This list will store all valid partitions of the integer t into exactly m parts.

2. dfs Function: We define a recursive function called dfs that takes two parameters: *remaining* and *parts*. *remaining* represents the remaining value of the integer to be partitioned. *parts* represents the current partition being constructed.

3. Base Cases: If *remaining* equals 0 and the length of *parts* equals m, it means we have found a valid *partition* of t into m parts. We add this *partition* to the *partitions* list and return. If *remaining* becomes negative or the length of *parts* exceeds m, we terminate the current path of the recursion as it leads to invalid partitions. We return without making any changes.

4. Recursive Exploration: We iterate over possible values for the next part of the partition. We start the iteration from either 1 (if *parts* is empty) or the last element of *parts*. This ensures that the next part is always greater than or equal to the previous part, avoiding duplicate partitions. For each possible next part value, we recursively call the dfs function with the updated remaining value (reduced by the chosen part) and parts list (appended with the chosen part).

5. Generating Partitions: By recursively exploring all possible paths, the dfs function generates all valid partitions of t into m parts.

6. Return: Once all valid partitions have been generated, we return the *partitions* list containing all partitions.

Overall, this algorithm uses a recursive approach to systematically explore all possible partitions of the integer t into exactly m parts, ensuring that each generated partition is valid and appears exactly once.

4.3.4 The c-vector of a fan graph

In this subsection we put the methods from subsections 4.3.1, 4.3.2 and 4.3.3 together and show how these can be used to compute the c-vector of a fan graph.

Proposition 28. *The collection of the spanning trees produced by all clusters of F_k together with $T_{\min}$ is the collection of all spanning trees of F_k.*

Proof. Let $\mathcal{C}$ be the collection of the spanning trees produced by all clusters of F_k and $\mathcal{S}$ the collection of all spanning trees of F_k. We want to show that

$$\mathcal{C} \cup \{T_{\min}\} = \mathcal{S}.$$

It is clear that

$$\mathcal{C} \cup \{T_{\min}\} \subseteq \mathcal{S}.$$

It remains to show that

$$T \in \mathcal{S} \implies T \in \mathcal{C} \cup \{T_{\min}\}.$$

For this purpose let $T \in S$. Then either $T = T_{\min}$ or the set $E_s(T)$ of all spare edges of T is nonempty. At the beginning of the subsection 4.3.1 we argued that there is a bijection between the subsets of spare edges and the clusters of F_k. Thus, there is a cluster K containing exactly the spare edges of T. Then, by the definition, T is produced from K and we are done. □

We compute the c-vector of a fan graph F_k in 4 steps:

1. We list all integer partitions satisfying conditions of Proposition 27, in order to get all cluster types that can be built within F_k.

2. For each cluster type, we compute how many spanning trees of each kind are produced from a cluster of this type.

3. We count how many clusters of each type there are.

4. We multiply the number of spanning trees produced from one cluster by the number of clusters of the same type, to obtain the total number of trees produced from every cluster. We determine every entry c_j of the c-vector of F_k, which is the total number of the A_i trees produced from all clusters of F_k.

Example. (The c-vector of F_5)

1. **Valid partitions**
   Inserting $k = 5$ into Proposition 27, we see that in order to obtain all types of clusters of F_5, we need to consider partitions of the integers from m to $5 - m + 1$ into m parts for $m \in [\lceil \frac{5}{2} \rceil] = [3]$. These are precisely the partitions of the integers 1, 2, 3, 4, 5 into one part, of 2,3,4 into two parts and of 3 into three parts. All these are presented in Table 4.2.

# parts / integer	1	2	3
1	(1)	-	-
2	(2)	(1,1)	-
3	(3)	(2,1)	(1,1,1)
4	(4)	(2,2),(3,1)	-
5	(5)	-	-

Table 4.2: The partitions corresponding to the cluster types of F_5.

2. **Trees produced from each type of cluster**
   By Proposition 27, every partition from Table 4.2 corresponds to exactly one cluster type. For example partition (3,1) corresponds to cluster type 3 1. Now, for all cluster types of F_5, we use Proposition 25 to compute how many spanning trees of each kind can be produced from clusters of this type. We observe that the maximal number of blocks in the clusters of F_5 is 3. To avoid unneccessary computations, we observe that for $x = 0$, $x = 1$ and $x = m$, (4.1) can be simplified.

   (a) Inserting $x = 0$ into (4.1) yields

   $$\sum_{M \in \binom{[m]}{0}} \prod_{j \in M} t_j = \prod_{j \in \varnothing} t_j = 1, \tag{4.6}$$

   since this is the empty product. Consequently, for every cluster with t triangles there is exactly one A_{t+1}-tree. We apply (4.6) to the cluster types of F_5. The results are listed in Table 4.3.

cluster type	t	A_{t+1}-trees produced per cluster
1	1	$1A_2$
2	2	$1A_3$
3	3	$1A_4$
4	4	$1A_5$
5	5	$1A_6$
1 1	1+1=2	$1A_3$
2 1	2+1=3	$1A_4$
2 2	2+2=4	$1A_5$
3 1	3+1=4	$1A_5$
1 1 1	1+1+1=3	$1A_4$

Table 4.3: The table illustrates the results of inserting $x = 0$ into (4.1), i.e., the number of A_{t+1}-trees produced by a single cluster of each cluster type.

   (b) We insert $x = 1$ into (4.1):

   $$\sum_{M \in \binom{[m]}{1}} \prod_{j \in M} t_j = \prod_{j \in \{1\}} t_j + \prod_{j \in \{2\}} t_j + \cdots + \prod_{j \in \{m\}} t_j = \sum_{j=1}^{m} t_j = t. \tag{4.7}$$

   Thus, every cluster with t triangles produces t A_{t+2}-trees. See Table 4.4 for the results of applying (4.7) to the cluster types of F_5.

cluster type	t	A_{t+2}-trees produced per cluster
1	1	$1A_3$
2	2	$2A_4$
3	3	$3A_5$
4	4	$4A_6$
5	5	$5A_7$
1 1	2	$2A_4$
2 1	3	$3A_5$
2 2	4	$4A_6$
3 1	4	$4A_6$
1 1 1	3	$3A_5$

Table 4.4: The table illustrates the results of inserting $x = 1$ into (4.1), i.e., the number of A_{t+2}-trees produced by one cluster of each cluster type.

(c) For $x = m$, we get

$$\sum_{M \in \binom{[m]}{m}} \prod_{j \in M} t_j = \prod_{j \in [m]} t_j. \tag{4.8}$$

Notice that if $m = 1$, there is only one factor in the above product, namely t_1. Since in this case $t_1 = t$, we get the same result as in (b). Applying (4.8) on the cluster types of F_5 yields trees as presented in Table 4.5.

cluster type	t	m	$\prod_{j=1}^{m} t_j$	A_{t+m+1}-trees produced per cluster
1	1	1	1	$1A_3$
2	2	1	2	$2A_4$
3	3	1	3	$3A_5$
4	4	1	4	$4A_6$
5	5	1	5	$5A_7$
1 1	2	2	$1 \cdot 1 = 1$	$1A_5$
2 1	3	2	$2 \cdot 1 = 2$	$2A_6$
2 2	4	2	$2 \cdot 2 = 4$	$4A_7$
3 1	4	2	$3 \cdot 1 = 3$	$3A_7$
1 1 1	3	3	$1 \cdot 1 \cdot 1 = 1$	$1A_7$

Table 4.5: The table illustrates the results of inserting $x = m$ into (4.1), i.e., the number of A_{t+m+1}-trees produced by a cluster of each cluster type.

The only cluster type of F_5 for which the above cases are not sufficient is 1 1 1, since we also have to consider $x = 2 \neq m$. We compute how many A_6-trees are produced by any cluster of type 1 1 1 :

$$\sum_{M \in \binom{[3]}{2}} \prod_{j \in M} t_j = \prod_{j \in \{1,2\}} t_j + \prod_{j \in \{1,3\}} t_j + \prod_{j \in \{2,3\}} t_j = t_1t_2 + t_1t_3 + t_2t_3 = 1 + 1 + 1 = 3.$$

The final results of the computations of step 2 are presented in Table 4.6.

cluster type	all A_j-trees produced per cluster
1	$1A_2$, $1A_3$
2	$1A_3$, $2A_4$
3	$1A_4$, $3A_5$
4	$1A_5$, $4A_6$
5	$1A_6$, $5A_7$
1 1	$1A_3$, $2A_4$, $1A_5$
2 1	$1A_4$, $3A_5$, $2A_6$
2 2	$1A_5$, $4A_6$, $4A_7$
3 1	$1A_5$, $4A_6$, $3A_7$
1 1 1	$1A_4$, $3A_5$, $3A_6$, $1A_7$

Table 4.6: All spanning trees produced by each cluster type.

cluster type	special case	# clusters
1	1	$S_1^5 = 5 - 1 + 1 = 5$
2	1	$S_{2,1}^5 = 5 - 2 + 1 = 4$
3	1	$S_{3,1}^5 = 5 - 3 + 1 = 3$
4	1	$S_{4,1}^5 = 5 - 4 + 1 = 2$
5	1	$S_{5,1}^5 = 5 - 5 + 1 = 1$
1 1	2a	$S_2^5 = \binom{5-1}{2} = 6$
2 1	3	$M_{2,1}^5 = 2\binom{5-2-1+1}{2} = 6$
2 2	2b	$S_{2,2}^5 = S_2^3 = \binom{2}{2}$
3 1	3	$M_{3,1}^5 = 2\binom{5-3-1+1}{2} = 2$
1 1 1	2a	$S_3^5 = \sum_{j=1}^3 B_2^j = 0 + 0 + \binom{2}{2} = 1$

Table 4.7: The table lists the number of clusters in each of the cluster types and the method used to compute this number.

3. **Counting how many clusters of each type there are**

   To compute how many clusters of each type there are we use the methods from subsection 4.3.2. Notice that all cluster types of F_5 are special cases from subsection 4.3.2. Table 4.7 shows for every cluster type of F_5 the special case that concerns it and how to compute the number of clusters of this type. For any of this cluster types we can also use Proposition 26 instead. As an example we apply (4.5) to the cluster type 2 1:

$$M_{(2,1),(1,1)}^5 = \binom{2}{1,1} S_2^{5-(2-1)-(1-1)} = 2S_2^4 = 2\binom{3}{2} = 6.$$

4. **Summary**

   We summarise the results from steps 1-3 in Table 4.8 and compute the entries c_j of the c-vector of F_5. Notice that every fan graph has exactly one A_1-tree, namely the lexicographically minimal spanning tree $T_{\min}$, which cannot be produced from a cluster, since it does not contain any spare edges. Thus $c_1 = 1$. We compute the remaining entries of the c-vector of F_5 by adding all the coefficients of the corresponding A_j-trees from the last column of Table 4.8.

cluster type	trees produced per cluster	# clusters	total # trees
1	$1A_2$, $1A_3$	5	$5A_2$, $5A_3$
2	$1A_3$, $2A_4$	4	$4A_3$, $8A_4$
3	$1A_4$, $3A_5$	3	$3A_4$, $9A_5$
4	$1A_5$, $4A_6$	2	$2A_5$, $8A_6$
5	$1A_6$, $5A_7$	1	$1A_6$, $5A_7$
1 1	$1A_3$, $2A_4$, $1A_5$	6	$6A_3$, $12A_4$, $6A_5$
2 1	$1A_4$, $3A_5$, $2A_6$	6	$6A_4$, $18A_5$, $12A_6$
2 2	$1A_5$, $4A_6$, $4A_7$	1	$1A_5$, $4A_6$, $4A_7$
3 1	$1A_5$, $4A_6$, $3A_7$	2	$2A_5$, $8A_6$, $6A_7$
1 1 1	$1A_4$, $3A_5$, $3A_6$, $1A_7$	1	$1A_4$, $3A_5$, $3A_6$, $1A_7$

Table 4.8: The table shows the total number of spanning trees produced from every cluster type.

The entries of the c-vector of F_5 are

$$\begin{aligned}
c_1 &= 1,\\
c_2 &= 5,\\
c_3 &= 5+4+6 = 15,\\
c_4 &= 8+3+12+6+1 = 30,\\
c_5 &= 9+2+6+18+1+2+3 = 41,\\
c_6 &= 8+1+12+4+8+3 = 36,\\
c_7 &= 5+4+6+1 = 16.
\end{aligned}$$

We see that already for a very small k we need many computations to determine the c-vector of a fan graph F_k. For large k these become much more, since, as mentioned before, computing the partitions of large integers is a difficult problem. Moreover, solving the recurrence relations from subsection 4.3.2 is challenging for clusters consisting of many blocks of different, partially large, sizes.

5. Open problems

In this thesis we developed a closed formula for computing the j-th coefficient of the c-vector in dependence of the $b_i(G)$, which themselves are difficult to determine. Thus, the answer to the following question would be desirable.

Problem 29. *Is there a closed formula for computing the c-vector directly from the graph which does not depend on the $b_i(G)$?*

Seeing how complicated computing the c-vector is for the simplest types of graphs it seems unlikely that this problem can be solved with our methods. Maybe a good approach would be to answer the following question first.

Problem 30. *Is there a closed formula for computing the coefficients c_4 and c_v directly from the graph which does not depend on the $b_i(G)$?*

Here v denotes again the number of vertices of the reduced graph of G. The reason for choosing these two coefficients is simple: c_4 is the coefficient with the lowest index, for which we still do not have a closed formula and c_v is special, since its index is the largest. A good start of analysing c_v could be to determine when $c_v = 1$ and when it is larger. Our guess is that $c_v = 1$, if there are no cycles with common edges, i.e., for graphs whose reduced graph is a cactus graph.

For the next problem consider the following example.

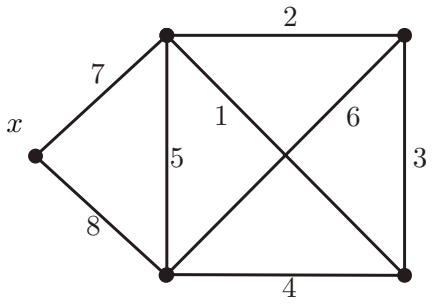

Figure 5.1: A graph G constructed by adding a vertex x to the complete graph K_4 and connecting x with the two vertices of the edge 5 of K_4. Notice that every spanning tree of G contains at least one of the edges 7 and 8.

Example. Let G be a graph as presented in Figure 5.1. Recall that the c-vector of K_4 is $c = (1, 3, 6, 6)$. Since all spanning trees of G must contain at least one of the edges 7 and 8, we consider the following three cases.

1. The spanning trees of G containing the edge 7 but not 8.
2. The spanning trees of G containing the edge 8 but not 7.
3. The spanning trees of G containing both the edge 7 and 8.

The spanning trees from the first two cases are simply the results of adding 7 or 8 to all spanning trees of K_4. Specifically, every A_j-tree of K_4 becomes an A_j-tree of K_4 if we add the edge 7 to it and an A_{j+1}-tree if we add the edge 8. The spanning trees from the

third case can be obtained from any spanning tree of K_4 which contains the edge 5, by replacing 5 by 7 and 8. In the labelling presented in Figure 5.1, the edge 5 is an internally passive edge for all spanning trees of K_4 and hence every A_j-tree containing it becomes an A_{j+1}-tree of G after replacing 5 by 7 and 8.

This example brings us to the following problem.

Problem 31. *Let G_1 be a graph, c_1 its c-vector, e an arbitrary edge of G_1 and x a vertex with $x \notin V(G_1)$. Consider the graph $G_2 = G_1 \cup (e\nabla x)$. Is there a method of computing the c-vector of G_2 using c_1?*

By the expression $(e\nabla x)$, we mean the join of the graph containing only edge e and its vertices with the empty graph containing only vertex x. The difficulty of this problem might be determining the number of the spanning trees of G_1 containing e and for each such tree, the number of its internally passive edges.

To understand the next problem we need the following definition.

Definition. A **wheel graph** W_k is the join of a cycle graph C_k with a one vertex graph.

See an example of a wheel graph in Figure 5.2.

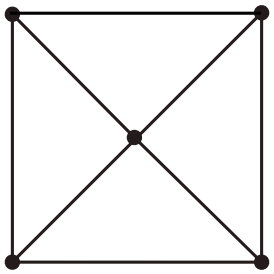

Figure 5.2: Wheel graph W_4.

Problem 32. *Is it possible to compute the c-vector of a wheel graph in a similar manner as for the fan graph?*

We observe that the c-vector of a fan graph F_k and the c-vector of a wheel graph W_k are very similar for $k = 3$ and $k = 4$. Indeed, the c-vector of F_3 is $(1, 3, 6, 7, 4)$ and the c-vector of W_3 is $(1, 3, 6, 6)$, while the c-vector of F_4 is $(1, 4, 10, 16, 16, 6)$ and the c-vector of W_4 is $(1, 4, 10, 16, 14)$.
This is not surprising, since W_k and F_k contain the same number of triangles, but the wheel graph has one vertex and one edge less. Consequently, the coefficients c_1, c_2 and c_3 are the same for both F_k and W_k, and W_k has $k + 1$ non-zero coefficients, while for F_k this number is $k + 2$. Even though W_k has the same number of triangles as F_k, not all of the clusters of F_k are clusters of W_k and the number of clusters is not always the same. However, we believe that it is possible to derive a similar method of determining the types of clusters that are subsets of W_k and count how many of each types are there, so that Proposition 25 can be applied.
The next question concerns the impact of modifications of the graph on its c-vector.

Problem 33. *How do deletion and contraction of an edge change the c-vector?*

To solve this problem we would definitely need a case distinction, since these two operations on graphs have different results depending on the kind of edge they are applied on. Proposition 13 already tells us that the contraction of a cut-edge does not change the c-vector and a similar argument can be used for proving that neither does the deletion. Next case could be an edge from a simple cycle of a graph, such that this cycle does not share an edge with any other cycle. Deleting any edge of such a cycle breaks the cycle, and we might be able to compute the c-vector if we could somehow reverse the method from Proposition 21. On the other hand, contracting such an edge yields the same result only if the cycle contains exactly three edges. If this is not the case, the cycle does not disappear in the reduced graph; it only gets smaller. We again believe that studying Proposition 21 could result in an elegant solution. Another case which seems worth analysing is deleting or contracting the edges containing x from Problem 31. It is quite likely that there are other cases for which we could obtain a closed formula. At the moment we do not see any pattern that could lead us to a general solution to this problem.

Bibliography

[1] Martin Aigner and Günter M. Ziegler. *Proofs from THE BOOK. Including illustrations by Karl H. Hofmann.* Berlin: Springer, revised and enlarged 6th edition, 2018.

[2] Matthias Beck, Katharina Jochemko, and Emily McCullough. h^*-polynomials of zonotopes. *Trans. Am. Math. Soc.*, 371(3):2021–2042, 2019.

[3] Matthias Beck and Sinai Robins. *Computing the continuous discretely. Integer-point enumeration in polyhedra. With illustrations by David Austin.* Undergraduate Texts Math. New York, NY: Springer, 2nd edition, 2015.

[4] Matthias Beck and Raman Sanyal. *Combinatorial reciprocity theorems. An invitation to enumerative geometric combinatorics*, volume 195 of *Grad. Stud. Math.* Providence, RI: American Mathematical Society (AMS), 2018.

[5] E. D. Bolker. A class of convex bodies. *Trans. Am. Math. Soc.*, 145:323–345, 1969.

[6] Miklós Bóna. *A walk through combinatorics. An introduction to enumeration and graph theory.* Hackensack, NJ: World Scientific, 2nd edition, 2006.

[7] Francesco Brenti and Volkmar Welker. f-vectors of barycentric subdivisions. *Math. Z.*, 259(4):849–865, 2008.

[8] Aaron Matthew Dall. *Matroids : h-vectors, zonotopes, and Lawrence polytopes.* PhD thesis, Universitat Politècnica de Catalunya, 2015.

[9] R. Ehrenborg and M. Readdy. Mixed volumes and slices of the cube. *J. Comb. Theory, Ser. A*, 81(1):121–126, 1998.

[10] Eugène Ehrhart. Sur les polyèdres rationnels homothétiques à n dimensions. *C. R. Acad. Sci., Paris*, 254:616–618, 1962.

[11] Gary Gordon and Jennifer McNulty. *Matroids. A geometric introduction.* Cambridge: Cambridge University Press, 2012.

[12] Gustav R. Kirchhoff. Über die auflösung der gleichungen, auf welche man bei der untersuchung der linearen vertheilung galvanischer ströme geführt wird. *Annalen der Physik*, 148:497–508.

[13] Luca Moci. A Tutte polynomial for toric arrangements. *Trans. Am. Math. Soc.*, 364(2):1067–1088, 2012.

[14] Daniel R. Page. Generalized algorithm for restricted weak composition generation. *J. Math. Model. Algorithms Oper. Res.*, 12(4):345–372, 2013.

[15] Richard P. Stanley. Decompositions of rational convex polytopes. Ann. Discrete Math. 6, 333-342 (1980)., 1980.

[16] Richard P. Stanley. Two combinatorial applications of the Aleksandrov-Fenchel inequalities. *J. Comb. Theory, Ser. A*, 31:56–65, 1981.

[17] Hassler Whitney. On the abstract properties of linear dependence. *Am. J. Math.*, 57:509–533, 1935.

[18] Antoine Zoghbi and Ivan Stojmenović. Fast algorithms for generating integer partitions. *Int. J. Comput. Math.*, 70(2):319–332, 1998.