BACHELOR'S THESIS

Degree in Mathematics

Title: The cover polynomial of a digraph

Author: Salvi Solà Martinell

Advisor: Anna de Mier Vinué

Department: Departament de Matemàtiques

Academic year: 2017 - 2018

UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Facultat de Matemàtiques i Estadística

Universitat Politècnica de Catalunya
Facultat de Matemàtiques i Estadística

Degree in Mathematics

Bachelor's Degree Thesis

The cover polynomial of a digraph

Salvi Solà Martinell

Supervised by Anna de Mier Vinué,
Departament de Matemàtiques

January, 2018

Abstract

The object of study of this work is the cover polynomial of a digraph, and in particular its power as an invariant. The information of a digraph that can be obtained from this polynomial is collected and used in order to determine families of C-unique digraphs, i.e., digraphs that do not share the cover polynomial with any other one. The work is assisted by computer, in particular to determine the number of C-unique digraphs of small orders for the families of directed acyclic graphs and tournaments.

Keywords

cover polynomial, unique, equivalent, C-unique, C-equivalent, invariant, information, directed graph, digraph, path-cycle cover, cycle-path cover

Contents

1. Introduction

There are many polynomials which can be associated with a graph G, the most well studied perhaps being the Tutte polynomial $T(G; x, y)$. The importance of this polynomial stems from the considerable information it contains about G, connecting it to different parts of graph theory. In particular, the Tutte polynomial evaluated at $y = 0$ gives, essentially, the well-known chromatic polynomial.

For a directed graph D, no clear analogue of the Tutte polynomial is known, but several have been considered. This thesis is focused on the cover polynomial $C(D; x, y)$, introduced by Chung and Graham in [4], which has a number of properties that are comparable to those of $T(G; x, y)$.

A question that arises during the study of these polynomials is to what extent the information they contain can determine a graph, up to isomorphism. Specifically, which graphs have unique polynomials? For the Tutte polynomial, this research was initiated by de Mier and Noy, and pursued by other authors, see [7, 2] and the references therein. This thesis intends to start the same research for the cover polynomial.

The initial goals were the following:

- get acquainted with the cover polynomial, comparing it to the Tutte polynomial;
- list which information of a digraph can be obtained from it and how (number of vertices, number of edges, information about the cycles, etc.);
- explore the most well-known families of digraphs and see if they are determined by the cover polynomial;
- do the same for the digraphs of few vertices/edges (maybe computationally).

This has been organized into the following sections:

- Section 2 presents the cover polynomial and reviews the first results exposed by Chung and Graham, with some added explanations. Later, the geometric the cover polynomial, introduced by D'Antona and Munarini in [5], is also presented. The relations between the two polynomials are studied.
- Section 3 exposes several parameters and relevant information of a digraph that can be obtained from the cover polynomial.
- Section 4 contains new results on the uniqueness of the cover polynomial for different families of digraphs, using information from the previous section.
- Section 5 is a short section on the computer aid used for this work and the numeric results that have been obtained. A selection of the source code with a showcase script is included in Appendix B.
- Section 6 includes conclusions, reflections and some ideas on how the research may be continued.

Through this work, D denotes a *directed graph*, or *digraph*, for short. From now on, digraphs are always considered up to isomorphism. The knowledge of the Discrete Mathematics course of this bachelor degree is assumed; however, the definitions of basic notions of digraph theory that have been used are collected, for completeness, in Appendix A.

Multidigraphs, digraphs with multiple repeated edges, are out of the scope of this thesis, although some of the results are arguably extensible. Also, bear in mind that unless indicated, edges, paths and cycles, always refer to the directed versions, characteristic of directed graph theory.

2. The cover polynomial

This section is a review of the first part of the article [4], introducing the cover polynomial and going over the first results. Definitions, results and proofs have been obtained from the aforementioned article, sometimes reworded and restructured, with explanations and remarks added.

In Subsection 2.4 the very similar geometric cover polynomial is presented. We show a bijection between the two polynomials and derive the analogue properties.

2.1 The recursive definition

We start with the definitions of deletion and contraction, illustrated in Figures 1 and 2.

Definition 2.1. A *deletion* of an edge $e \in E$, denoted by $D\backslash e$, is the digraph with set of vertices V and set of edges $E\backslash\{e\}$.

Definition 2.2. A *contraction* of an edge $e = uv \in E$, denoted by D/e, is the digraph obtained from D by the following process. If $e = uv$ is not a loop, all edges with origin u or end v are deleted, and u and v are replaced in V and in every edge by a new vertex w. If $e = uu$ is a loop, all edges with origin or end u are deleted and vertex u is removed from V.

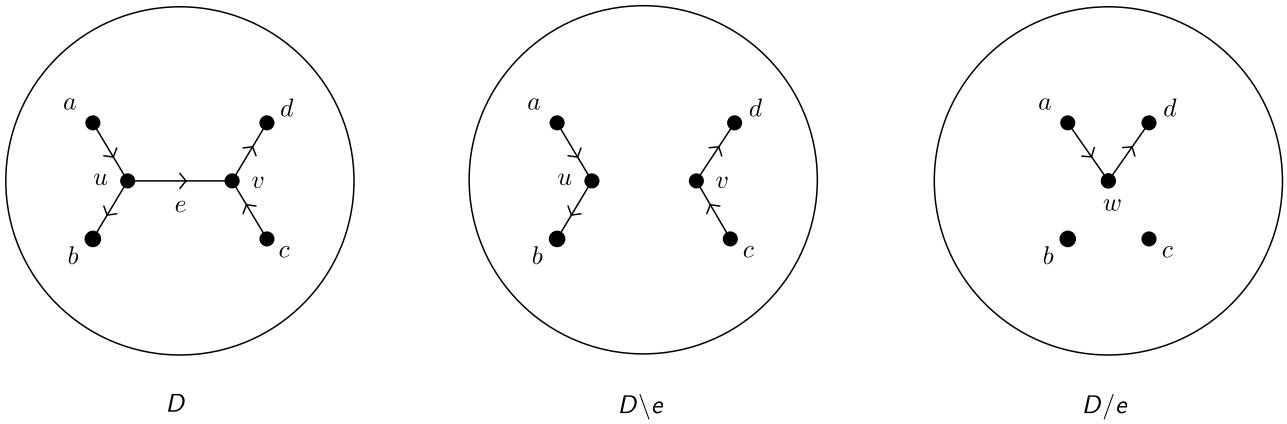


Figure 1: Deleting and contracting a non-loop edge.

Remark. Although the definition of contraction seems to delete more edges than necessary, that is what will allow the construction of the cover polynomial to work.

Definition 2.3. The *falling factorial* is defined by $x^{\underline{n}} := x(x-1)\cdots(x-(n-1))$ and $x^{\underline{0}} := 1$.

Definition 2.4. The *cover polynomial* $C(D) = C(D; x, y)$ is a bivariate polynomial defined recursively as follows:

(i) for I_n, the digraph with n vertices and no edges,

$$C(I_n) = x^{\underline{n}};$$

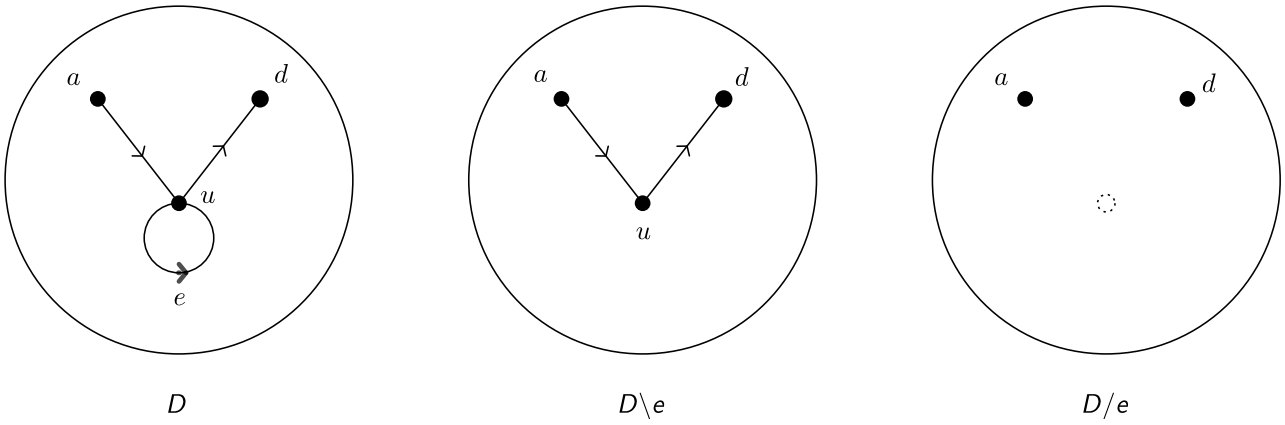


Figure 2: Deleting and contracting a loop.

(ii) if e is an edge of D which is not a loop then

$$C(D) = C(D\backslash e) + C(D/e)\,;$$

(iii) if e is a loop of D then

$$C(D) = C(D\backslash e) + yC(D/e)\,.$$

As both the deletion and contraction always reduce the number of edges, and the base case (i) covers all digraphs without edges, the recurrence can be calculated. However, as the election of e for each step is free, it is not clear yet that $C(D)$ is well-defined. This will be resolved by obtaining an explicit expression in the next section.

The recursive formulas (ii) and (iii) are of a kind generally known as deletion-contraction formulas. These have been studied deeply for undirected graphs, and can be used to define, for example, the chromatic polynomial and the Tutte polynomial.

In the recursive definition of the Tutte polynomial the general case is that, for any edge e that is neither a loop nor a bridge, $T(G) = T(G\backslash e) + T(G/e)$. This certainly resembles the formula (ii). On the other hand, the base case and the treatment of loops and bridges is different in the definition of each of the polynomials.

2.2 The explicit expression

The deletion-contraction formulas for defining graph polynomials have the general inconvenience of needing a proof that the polynomial is well-defined, i.e, independent of the ordering which edges are contracted and deleted. This will be done in this section by giving an explicit expression that satisfies the recurrence of the cover polynomial.

Definition 2.5. A *path-cycle cover* of a digraph D is a way of disjointly covering all the vertices of D with paths and cycles. Note that a path can consist of a single vertex, and a cycle can consist of a single loop.

Definition 2.6. The *path-cycle cover coefficient* $c_D(i,j)$ is the number of path-cycle covers of D using exactly i paths and j cycles. $c_D(i,j)$ is taken to be 0 for negative values of i or j.

For example, for both digraphs in Figure 3 we have $c_D(1,1) = 1$, $c_D(1,0) = 1$ and $c_D(2,0) = 1$.

Theorem 2.7. ([4], Theorem 1) *The cover polynomial is*

$$C(D; x, y) = \sum_{i,j} c_D(i,j) x^i y^j ,$$

where summation indices range over all integers.

Proof. The proof will proceed by double induction on the number n of vertices of D, and then on the number of edges m of D. The theorem clearly holds for any digraph having no edges (by (i)). Assume it holds for all D with fewer than n vertices, and for all D with n vertices and fewer than m edges, for some fixed $m > 0$, $n > 0$. We will use the recurrence formulas (ii) and (iii). Let e be an edge of D. The set of path-cycle covers of D can be partitioned into those which actually use the edge e in a path or cycle, and those which do not use e. It is clear that $c_{D\setminus e}(i,j)$ counts the number of covers of D by i paths and j cycles which do not use e.

Now, if e is not a loop then $c_{D/e}(i,j)$ counts the number of covers of D by i paths and j cycles which use e (just insert e into the appropriate path or cycle covering the contracted vertex w). In this case the induction step follows by (ii).

On the other hand, if e is a loop then the number of covers of D by i paths and j cycles using e is just $c_{D/e}(i, j-1)$. Namely, each cover of D/e by i paths and $j-1$ cycles augmented by the loop (=cycle) e is such a cover of D. In this case the induction step follows by (iii). This proves the theorem. □

In particular, this shows that $C(D)$ is an invariant of D, and so, is well-defined.

2.3 First results

Theorem 2.7 gives the coefficient of $x^i y^j$ in the cover polynomial. We give next an interpretation of the value $C(D; x, y)$ for positive integer values of x and y. This interpretation will be very useful in the sequel.

Definition 2.8. For fixed positive integers λ and μ, we can assign to each path-cycle cover of D certain colorings of the vertices of D by $\lambda + \mu$ colors as follows: (i) any two vertices in the same path or cycle have the same color; (ii) vertices in different paths have different colors; (iii) vertices in paths have colors from a set of λ colors; (iv) vertices in cycles have colors from a set of μ colors, disjoint from the set of colors in (iii). Note that it is not required that every color is used.

Let us call such an assignment (a path-cycle cover together with a color assignment) a (λ, μ)-*coloring* of D.

Since each of the $c_D(i,j)$ covers of D by i paths and j cycles generates $\lambda^i \mu^j$ such (λ, μ)-colorings, we have:

Corollary 2.9. ([4], Corollary 1) *$C(D; \lambda, \mu)$ is the number of (λ, μ)-colorings of D.*

Corollary 2.10. *Let λ and μ be positive integers. $C(D; \lambda, \mu)$ is null for:*

- *λ lower than the minimum number of paths required to make a path-cycle cover of D;*
- *$\mu = 0$ and λ lower than the minimum number of paths required to make a path cover of D.*

Definition 2.11. The *linked union* $D = D_1 * D_2$ is formed by joining the disjoint digraphs $D_1 = (V_1, E_1)$ and $D_2 = (V_2, E_2)$ with all the edges v_1v_2, $v_1 \in V_1$, $v_2 \in V_2$

The following is the *multiplicative property* of the cover polynomial:

Corollary 2.12. ([4], Corollary 2) *Let $D = D_1 * D_2$ be the linked union of digraphs. Then*

$$C(D) = C(D_1)C(D_2)\,.$$

Proof. For each $\lambda, \mu > 0$, each pair of valid (λ, μ)-colorings of D_1 and D_2 can be extended to a unique (λ, μ)-coloring of D as follows. All the cycles and the paths of a color that appears only in D_1 or D_2 are maintained. For any two paths in D_1 and D_2 that have the same color, one of the new edges of D (the one going from the last vertex of the first path to the first vertex of the second path) is added to form a single path. This extension is clearly injective.

Conversely, each (λ, μ)-coloring of D generates unique (λ, μ)-colorings of D_1 and D_2 as follows. Cycles are necessarily contained entirely in either D_1 or D_2, so they can be maintained, together with paths that do not go from D_1 to D_2. For paths going from vertices of D_1 to vertices of D_2, the edge joining the two subgraphs is deleted forming two paths that go into their corresponding (λ, μ)-coloring. This generation is also clearly injective.

Since this is true for all positive integer choices of λ and μ then this implies the polynomial identity. □

This multiplicative property is presented as analogous to the one of the Tutte polynomial, $T(G) = T(G_1)T(G_2)$, which holds whenever G is the union of two graphs G_1 and G_2 sharing at most one vertex. However, the linked union used for the cover polynomial is different to the disjoint union used for the Tutte polynomial.

The following results are also corollaries of Theorem 2.7.

Corollary 2.13. ([4], Corollary 3) *$C(D)$ is a polynomial in x (i.e., y is absent) if and only if D is acyclic.*

A digraph $\widehat{D}$ is said to be the *reverse* of D if it is formed by reversing all the edges of D (i.e., uv is an edge of $\widehat{D}$ if and only if vu is an edge of D). Observe that for a pair of digraphs D and its reverse $\widehat{D}$, $c_D(i,j)$ and $c_{\widehat{D}}(i,j)$ are equal for all i and j. This yields the following corollary.

Corollary 2.14. ([4], Corollary 6) *Let $\widehat{D}$ be the reverse of D. Then*

$$C(\widehat{D}) = C(D)\,.$$

Remark. If D and $\widehat{D}$ are non-isomorphic, this yields simple examples of different digraphs with the same cover polynomial (see Figure 3).

Figure 3: Two non-isomorphic digraphs having the same cover polynomial, $x^2 + xy$.

2.4 The geometric cover polynomial

The use of the falling factorial in the base case of the definition of the cover polynomial (Definition 2.4) is not required for obtaining a well-defined polynomial. Actually, the falling factorial can be replaced by normal exponentiation, obtaining the geometric cover polynomial introduced by D'Antona and Munarini in [5].

Definition 2.15. The *geometric cover polynomial* $GC(D) = GC(D; x, y)$ is a bivariate polynomial defined by

$$GC(D; x, y) = \sum_{i,j} c_D(i,j) x^i y^j ,$$

where $c_D(i,j)$ are the path-cycle cover coefficients.

This polynomial contains the same information as the cover polynomial. Next we will see how the coefficients of one can be obtained from the coefficients of the other.

Definition 2.16. The (signed) *Stirling numbers of the first kind* $s(n,k)$ are the coefficients in the expansion

$$x^{\underline{n}} = \sum_{k=0}^{n} s(n,k) x^k .$$

Proposition 2.17. *The cover and the geometric cover polynomials determine each other. More concretely, let* $\widetilde{c}_D(i,j)$ *denote the coefficients of the cover polynomial, i.e.:*

$$C(D; x, y) = \sum_{i,j} \widetilde{c}_D(i,j) x^i y^j .$$

Then we have

$$\widetilde{c}_D(i,j) = \sum_{k=i}^{n} s(k,i) c_D(k,j)$$

where n *is the number of vertices of* D*.*

Proof. By Theorem 2.7,

$$\sum_{i,j} \widetilde{c}_D(i,j) x^i y^j = \sum_{i,j} c_D(i,j) x^{\underline{i}} y^j .$$

Fixing j and using the definition of the Stirling numbers:

$$\sum_{i} \widetilde{c}_D(i,j) x^i = \sum_{m} c_D(m,j) \sum_{k=0}^{m} s(m,k) x^k = \sum_{m} \sum_{k=0}^{m} c_D(m,j) s(m,k) x^k .$$

The summation can stop at n because $c_D(i,j) = 0$ for all $i > n$, so

$$\sum_{i} \widetilde{c}_D(i,j) x^i = \sum_{m=0}^{n} \sum_{k=0}^{m} c_D(m,j) s(m,k) x^k = \sum_{k=0}^{n} \sum_{m=k}^{n} c_D(m,j) s(m,k) x^k ,$$

and by equating coefficients:

$$\widetilde{c}_D(i,j) = \sum_{k=i}^{n} s(k,i) c_D(k,j) .$$

A system of linear equations can be formed from the last equation for $i = 0...n$. As the Stirling numbers $s(n,k)$ form a triangular matrix with ones at its diagonal, the system can be solved for the $c_D(i,j)$ as well. □

Now that we have seen how the cover polynomial and its geometric version are related, let us compare their main properties.

If we redefine a (λ,μ)-coloring (Definition 2.8) so that vertices in different paths do not need to have different colors (that is, removing condition (ii)), then following an analog reasoning, the Corollary 2.9 applies for the geometric cover polynomial: $GC(D;\lambda,\mu)$ is the number of (λ,μ)-colorings of D.

Corollaries 2.10, 2.13 and 2.14 also apply for the geometric cover polynomial. Regarding the multiplicative property (Corollary 2.12), an analogous and even simpler argument shows that the geometric cover polynomial is multiplicative for the disjoint union of digraphs. The *multiplicative property* of the geometric cover polynomial is presented next.

Proposition 2.18. *Let $D = D_1 \cup D_2$ be the disjoint union of digraphs. Then*

$$GC(D) = GC(D_1)GC(D_2)\,.$$

This multiplicative property uses the disjoint union, just like that of the Tutte polynomial, so this is a clearer analogue than the one we had with the cover polynomial. However, this equality still does not hold for unions of digraphs sharing one vertex, as the Tutte multiplicative property does.

3. Invariants determined by the cover polynomial

The purpose of this section is to summarize the parameters that can be deduced from the cover polynomial and can be useful to reconstruct the digraph. This information will be of use in the next section to find digraphs uniquely determined by their cover polynomial.

As with the Tutte polynomial, the evaluation of the cover polynomial at some points has a special meaning.

The value $C(D;0,0)$ can be interpreted directly from the meaning of $c_D(0,0)$:

Lemma 3.1. $C(D;0,0) = c_D(0,0) = \begin{cases} 1, & \text{if } D \text{ is the empty digraph} \\ 0, & \text{otherwise} \end{cases}$

Lemma 3.2. ([1], Lemma 1) *For any non-empty digraph D:*

- **(i)** $C(D;0,0) = 0$
- **(ii)** $C(D;1,0) = c_D(1,0)$, *the number of Hamiltonian paths*
- **(iii)** $C(D;0,1) = \sum_j c_D(0,j) = \mathrm{perm}(D)$, *the number cycle covers of D*
- **(iv)** $C(D;0,-1) = \sum_j (-1)^j c_D(0,j) = (-1)^n \det(D)$

where perm(D) *and* det(D) *are the* permanent *and the* determinant *of the adjacency matrix of D.*

Proof. (partial) The first equality on the 4 cases follows from the expression of the cover polynomial with $c_D(0,0) = 0$. The permanent is defined as

$$\text{perm}(D) = \sum_{\sigma \in S_n} \prod_{i=1}^{n} a_{i,\sigma(i)} \,,$$

so it is counting the number of permutations of the vertices where for which every edge is defined (the coefficients $a_{i,\sigma(i)}$ are the values of the adjacency matrix, which are 1 if $(i, \sigma(i))$ is an edge and 0 otherwise). These permutations define every possible cycle cover.

The equality with the determinant is just included for completeness, a proof sketch can be found in the reference. □

Meanings and expressions for some other points are collected in [4] and [1]. However, the information that seems most useful for deducing structural properties of the digraph can be directly deduced from the $c_D(i,j)$ path-cycle cover coefficients.

Observe that non-zero coefficients $c_D(i,j)$ are limited by the triangle $i \geq 0$, $j \geq 0$ and $i + j \leq n$, because each path-cycle cover contains at most one path or one cycle for each vertex.

Lemma 3.3. *From the cover polynomial of a digraph D, the following information can be obtained:*

- **(i)** *the number of vertices, $n = max\{i \in \mathbb{N} \mid c_D(i,0) \neq 0\}$;*
- **(ii)** *the number of edges, $c_D(n-1,0)$;*
- **(iii)** *the number of loops, $c_D(n-1,1)$;*
- **(iv)** *the length of the shortest cycle (known as girth), $g = min\{i \in \mathbb{N} \mid c_D(n-i,1) \neq 0\}$;*
- **(v)** *the number of shortest cycles, $c_D(n-g,1)$;*
- **(vi)** *the combined length of the minimal disjoint set of k cycles, $\ell_k = min\{i \in \mathbb{N} \mid c_D(n-i,k) \neq 0\}$;*
- **(vii)** *the number of minimal disjoint sets of k cycles, $c_D(n-\ell_k,k)$;*
- **(viii)** *the maximum number of disjoint cycles, $max\{j \in \mathbb{N} \mid \exists i \in \mathbb{N} : c_D(i,j) \neq 0\}$;*
- **(ix)** *the number of Hamiltonian cycles, $c_D(0,1)$;*
- **(x)** *the number of path covers (with disjoint paths), $\sum_i c_D(i,0)$.*

Proof. These claims can be deduced from the meaning of $c_D(i,j)$.

(i) There is always a cover by n paths, where each path is a single vertex. There is not any cover by more than n paths, because each path must contain at least one vertex.

(ii) Covers by $n-1$ paths must contain $n-2$ paths of length 0 and one path of length 1. Paths of length 1 are equivalent to edges. The choice of the edge determines the cover.

(iii) Covers by $n-1$ paths and 1 cycle must contain $n-1$ paths of length 0 and one loop. The choice of the loop determines the cover.

(iv) There is always a cover by $n-g$ paths and 1 cycle, where each path is a single vertex and the cycle has length g. There is not any cover by more than $n-g$ paths and 1 cycle, because each path must contain at least one vertex and the cycle must contain at least g vertices.

(v) Covers by $n-g$ paths and 1 cycle must contain $n-g$ paths of length 0 and one cycle of length g. The choice of the cycle determines the cover.

(vi) There is always a cover by $n-\ell_k$ paths and k cycles, where each path is a single vertex and the cycles have combined length ℓ_k. There is not any cover by more than $n-\ell_k$ paths and k cycles, because each path must contain at least one vertex and the cycles must contain at least ℓ_k vertices.

(vii) Covers by $n-\ell_k$ paths and k cycles must contain $n-\ell_k$ paths of length 0 and k cycles with combined length ℓ_k. The choice of the set of k cycles determines the cover.

(viii) There is always a cover with the maximum number of disjoint cycles, covering the remaining vertexes with paths. No covers can exist with more cycles, for these cycles could not be disjoint.

(ix) Directly from the meaning of $c_D(0,1)$.

(x) Directly from the meaning of $c_D(i,0)$.

□

By Corollary 2.13, the cover polynomial tells us whether a digraph is acyclic. We next present another family of digraphs for which this also happens.

Definition 3.4. A *tournament* is a loopless digraph D such that for every pair of vertices $u, v \in V$ such that $u \neq v$, either $(u,v) \in E$ or $(v,u) \in E$, but not both.

Lemma 3.5. *Whether a digraph is a tournament can be deduced from the cover polynomial.*

Proof. A tournament has as many edges as pairs of vertices, $n(n-1)/2$ edges, and by definition it has no loops and no cycles of length 2, so its girth is $g \geq 3$.

If a digraph D has girth $g \geq 3$, then it has no loops, and for every pair of vertices at most one of the edges may exist. That makes a total of $n(n-1)/2$ possible edges. If additionally this is exactly the number of edges of D, then an edge must exist for every pair of vertices, and so the digraph is a tournament.

Thus, a digraph is a tournament if and only it has $n(n-1)/2$ edges and girth $g \geq 3$. The three parameters considered here, that is, the number of vertices n, the number of edges, and the girth, can all be obtained from the cover polynomial by Lemma 3.3. □

4. C-equivalent and C-unique digraphs

In this section we consider digraphs of different types and group them by their cover polynomial, intending to find C-unique examples. First of all, however, we define the notions of C-equivalent, C-closed and C-unique:

Definition 4.1. Two digraphs D_1 and D_2 such that $C(D_1;x,y) = C(D_2;x,y)$ are called *C-equivalent*. A class of digraphs $\mathcal{D}$ is called *C-closed* if for every digraph $D \in \mathcal{D}$, all digraphs C-equivalent to D belong to $\mathcal{D}$. A digraph D is called *C-unique* if every digraph C-equivalent to D is isomorphic to D.

For example, the digraphs in Figure 3 are C-equivalent, and the families of acyclic digraphs and tournaments are both C-closed.

The cover polynomial of a digraph does not change when replacing a weak component or a union of weak components with a C-equivalent subdigraph, as stated in the next lemma:

Lemma 4.2. *Let D, D_1, D_2 be disjoint digraphs. If D_1 and D_2 are C-equivalent, then $D \cup D_1$ and $D \cup D_2$ are C-equivalent.*

Proof. Using the bijection of Proposition 2.17, the C-equivalence can be transferred to the geometric cover polynomial, yielding $GC(D_1) = GC(D_2)$. Using the multiplicative property of the geometric cover polynomial (Proposition 2.18),

$$GC(D \cup D_1) = GC(D)GC(D_1) = GC(D)GC(D_2) = GC(D \cup D_2)\,.$$

Then, using the bijection backwards, $C(D \cup D_1) = C(D \cup D_2)\,.$ □

Let be $\mathcal{D}(n, p, K)$ the class of digraphs of order n that are the disjoint union of any p paths and a fixed collection of cycles K, possibly empty.

Proposition 4.3. *The class $\mathcal{D}(n, p, K)$ is C-closed and its elements are C-equivalent.*

Proof. Firstly, we will prove that $\mathcal{D}(n, p, K)$ is C-closed. Let $D \in \mathcal{D}(n, p, K)$ and let $\widetilde{D}$ be a digraph C-equivalent to D, so that $C(\widetilde{D}) = C(D)$. This implies that the number of vertices of $\widetilde{D}$ is n. As $c_{\widetilde{D}}(p, |K|) = 1$, there is a cover of $\widetilde{D}$ with p paths and $|K|$ cycles. This cover uses $n - p$ edges, because each path has one edge less than vertices, and the cycles have the same number of both. But the number of edges of $\widetilde{D}$ is also known, from its cover polynomial, to be $n - p$. Since all edges of $\widetilde{D}$ are used in this cover, $\widetilde{D}$ is equal to the cover itself, i.e., $\widetilde{D}$ is the union of p paths and $|K|$ cycles.

The girth $k_1 = min\{i \in \mathbb{N} \mid c_D(n - i, 1) \neq 0\}$ and number of shortest cycles $a_1 = c_D(n - k_1, 1)$ of $\widetilde{D}$ can be determined from the cover polynomial. In this particular case, the cycles of every length can be determined as well: the q-th shortest length is $k_q = min\{i \in \mathbb{N} \mid c_D(n - i - \sum_{j=1}^{q-1} k_j a_j, \sum_{j=1}^{q-1} a_j + 1) \neq 0\}$ and the number of q-th shortest cycles is $a_q = c_D(n - \sum_{j=1}^{q} k_j a_j, \sum_{j=1}^{q-1} a_j + 1)$. In order to see this, the idea is that using covers with the maximum numbers of paths for a given number of cycles will yield the covers with the shortest cycles. This way, the cycles of $\widetilde{D}$ are determined to be the same as those of D, that is, K. Therefore, $\widetilde{D} \in \mathcal{D}(n, p, K)$ and this class is C-closed.

Now, we have to prove the C-equivalence of any pair of digraphs $D_1, D_2 \in \mathcal{D}(n, p, K)$. The cycles of D_1 and D_2 are the same, so, by Lemma 4.2, we can assume $K = \emptyset$ without loss of generality. The proof will follow by induction on the number of edges, which is $n - p$. The digraphs without edges, I_m, are unique up to isomorphism, so the statement clearly holds. Suppose the statement is true for digraphs with fewer than $n - p$ edges. By the recursive definition, $C(D_1) = C(D_1 \backslash e) + C(D_1/e)$ and $C(D_2) = C(D_2 \backslash e') + C(D_2/e')$. Both deletions belong to $\mathcal{D}(n, p + 1, \emptyset)$ and both contractions belong to $\mathcal{D}(n - 1, p, \emptyset)$, so their cover polynomials are the same. Consequently, $C(D_1) = C(D_2)$. □

Digraphs that are the only element of their C-closed class are C-unique. For $\mathcal{D}(n, p, K)$, these digraphs are listed in the next corollary.

Corollary 4.4. *The following digraphs are C-unique:*

- *digraphs with 0 or 1 edges;*

- *paths;*
- *cycles or union of multiple cycles, possibly united to one path or one digraph with 0 or 1 edges.*

Remark. In general, it is not true that the union of C-unique digraphs is C-unique. Proposition 4.3 provides counterexamples: digraphs of order n that are the disjoint union of any p paths are C-equivalent although paths are C-unique (see Figure 4).

Figure 4: A pair of C-equivalent digraphs, formed as the disjoint union of 2 paths with a total of 4 vertices.

Let $\mathcal{Q}(n)$ be the class of digraphs of order n that can be formed as a path $(u_1, \dots, u_n)$ and one additional edge (u_i, u_j), with $j - i > 1$. We call such an edge a *leap edge* of the digraph.

Proposition 4.5. *The class $\mathcal{Q}(n)$ is C-closed and its elements are C-equivalent.*

Proof. Firstly, we will prove that $\mathcal{Q}(n)$ is C-closed. Let $D \in \mathcal{Q}(n)$ and let $\widetilde{D}$ be a digraph C-equivalent to D, so that $C(\widetilde{D}) = C(D)$. From the cover polynomial, we know that $\widetilde{D}$ is acyclic, has order n, contains n edges and has a Hamiltonian path $u_1, \dots, u_n$. The remaining edge (u_i, u_j) cannot form a cycle nor be repeated so $j - i > 1$. Therefore, $\widetilde{D} \in \mathcal{Q}(n)$, and the class is C-closed.

Now, we prove the C-equivalence of any pair of digraphs $D_1, D_2 \in \mathcal{Q}(n)$. Consider the recurrence formula $C(D) = C(D\backslash e) + C(D/e)$ applied to D_1 and D_2 and their respective leap edges e_1 and e_2. Both deletions $D_1\backslash e_1$ and $D_2\backslash e_2$ form a path of length n, so $C(D_1\backslash e_1) = C(D_2\backslash e_2)$. Both contractions can be seen to belong to $\mathcal{P}(n-1, 2, \emptyset)$, so $C(D_1/e_1) = C(D_2/e_2)$. Therefore, $C(D_1) = C(D_2)$ as required. □

For $n > 3$, the classes $\mathcal{Q}(n)$ contain more than one element, so the digraphs in these classes are not C-unique.

Let $\mathcal{R}(n, k)$ be the class of digraphs of order n that can be formed as a path $(u_1, \dots, u_n)$ and one additional edge (u_i, u_j), with $k = j - i \leq 0$, $j > 1$ and $i < n$. This additional edge will effectively create a cycle of length $k + 1$.

Proposition 4.6. *The class $\mathcal{R}(n, k)$ is C-closed and its elements are C-equivalent.*

Proof. Firstly, we will prove that $\mathcal{R}(n, k)$ is C-closed. Let $D \in \mathcal{R}(n, k)$ and let $\widetilde{D}$ be a digraph C-equivalent to D, so that $C(\widetilde{D}) = C(D)$. From the path-cycle cover coefficients we know that $\widetilde{D}$ has a Hamiltonian path and exactly one cycle of length $k + 1$. Let the Hamiltonian path be $u_1, \dots, u_n$. The remaining edge (u_i, u_j) must form the cycle of length $k + 1$, so $j - i \leq 0$. In addition, we know that $\widetilde{D}$ cannot be covered with 1 cycle and 1 path, and $\widetilde{D}$ is not a cycle, so we can deduce that $j > 1$ and $i < n$. Therefore, $\widetilde{D} \in \mathcal{R}(n, k)$, and the class is C-closed.

The proof of the C-equivalence is analogue to that of Proposition 4.5, observing that in this case the contraction of the additional edge yields a digraph belonging to $\mathcal{P}(n-1, 2, C_{k+1})$. □

The classes $\mathcal{R}(n, n-2)$ contain only one element: the digraph formed of a path $(u_1, \dots, u_n)$ and one additional edge (u_{n-1}, u_2). Hence the following corollary.

Corollary 4.7. *Digraphs belonging to $\mathcal{R}(n, n-2)$ are C-unique.*

Proposition 4.8. *Complete digraphs and complete digraphs with one edge removed are C-unique. This is also true for loopless digraphs.*

Proof. These digraphs can be determined by the number of vertices n, the number of non-loop edges $n(n-1)$ (or $n(n-1)-1$) and the number of loops, which is all information of the cover polynomial. □

Definition 4.9. Let *directed stars* be the pairs of digraphs of order n and number of edges $n-1$ formed by a vertex of indegree $n-1$ or a vertex of outdegree $n-1$.

Proposition 4.10. *The pair of directed stars of order n (united optionally with m disjoint vertices) are C-equivalent digraphs and form a C-closed class.*

Proof. The C-equivalence is a consequence of Corollary 2.14.

To prove that the class is C-closed, let S be one directed star of order n with m disjoint vertices and let $\widehat{S}$ be a digraph C-equivalent to S, so that $C(\widehat{S}) = C(S)$. The cover polynomial indicates that $\widehat{S}$ is acyclic, so loops and back-and-forth pairs of edges can be discarded. Moreover, As $c_{\widehat{S}}(n+m-2, 0) = 0$, there are neither paths of length 2 nor pairs of disjoint edges in $\widehat{S}$. Therefore, every pair of edges must have the same origin or the same end.

Pick one pair of vertices, supposing at first that they share the origin: $(u, v), (u, w) \in E(\widehat{S})$, with $v \neq w$. Then, for any other edge of $e \in \widehat{S}$, having the same origin or the same end with each of the previous edges implies that e origins at u, because it cannot end both at v and w. So $e = (u, x)$. There are a total of $n-1$ edges of this form. Therefore, $\widehat{S}$ will be a directed star of order n with m disjoint vertices.

The same conclusion can be obtained, analogously, for a pair of vertices sharing the end, so the class is C-closed.

□

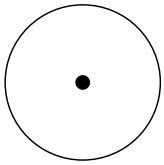
4.1 Small acyclic digraphs

In Figure 4, directed acyclic graphs (DAGs) up to order $n = 4$ have been grouped by C-equivalence using computer assistance (see Section 5). The digraphs are named with an A after *acyclic*, a superindex indicating the order, and a subindex indicating their arbitrary position within the DAGs of the same order. Acyclic digraphs induce a partial order of the vertices, so it has been possible to draw the edges from left to right (A^4_{20} has not been represented this way). Digraphs that are not grouped with any other are C-unique. Next we offer a theoretical justification for the C-equivalent digraphs.

The first C-equivalent DAGs are A^3_4 and A^3_5. The C-equivalence can be explained by Corollary 2.14, because these digraphs are reverses.

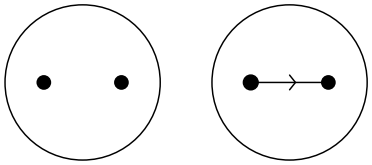
The C-equivalence of digraphs A^4_3 and A^4_4 is explained by Proposition 4.3, because they are both the union of 2 paths with a total of 4 vertices. A^4_6 and A^4_7 are reverses. The C-equivalence of A^4_8 and A^4_9 can be explained by the recursive definition (Definition 2.4): selecting the right edge for A^4_8 and the lower edge for A^4_9, both deletions produce A^4_6, and both contractions produce A^3_2, so both digraphs will have the same cover polynomial.

$n = 1$



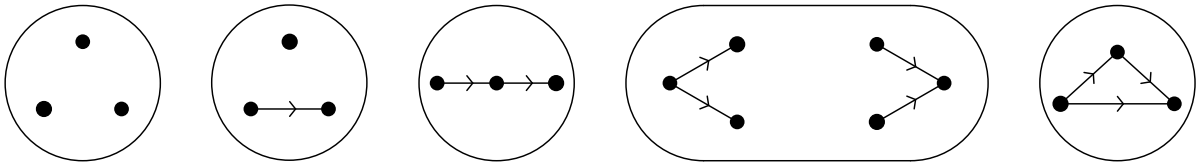
A_1^1

$n = 2$



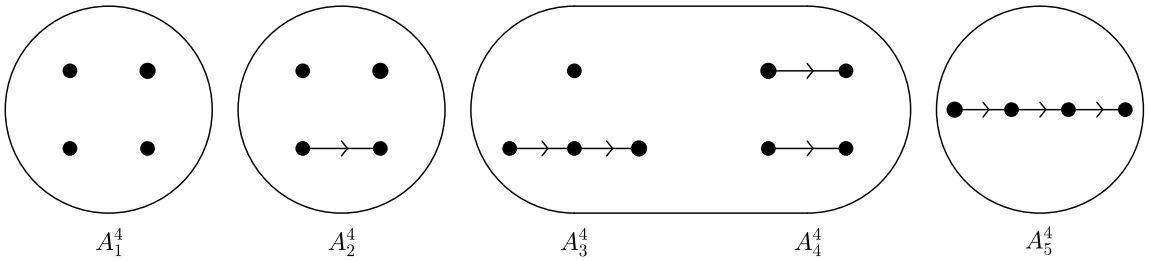
A_1^2 A_2^2

$n = 3$

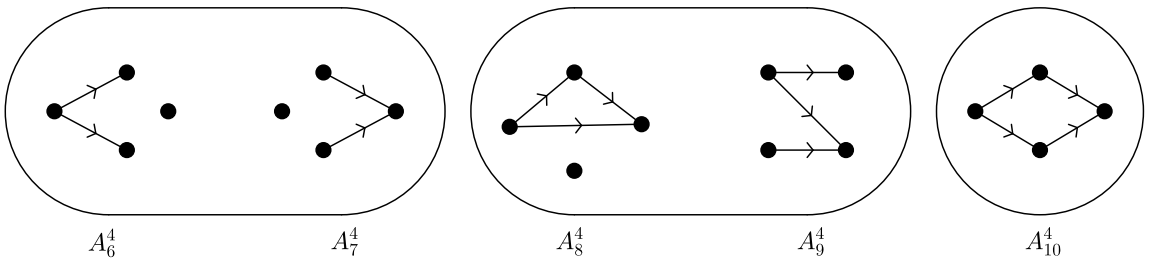


A_1^3 A_2^3 A_3^3 A_4^3 A_5^3 A_6^3

$n = 4$



A_1^4 A_2^4 A_3^4 A_4^4 A_5^4



A_6^4 A_7^4 A_8^4 A_9^4 A_{10}^4

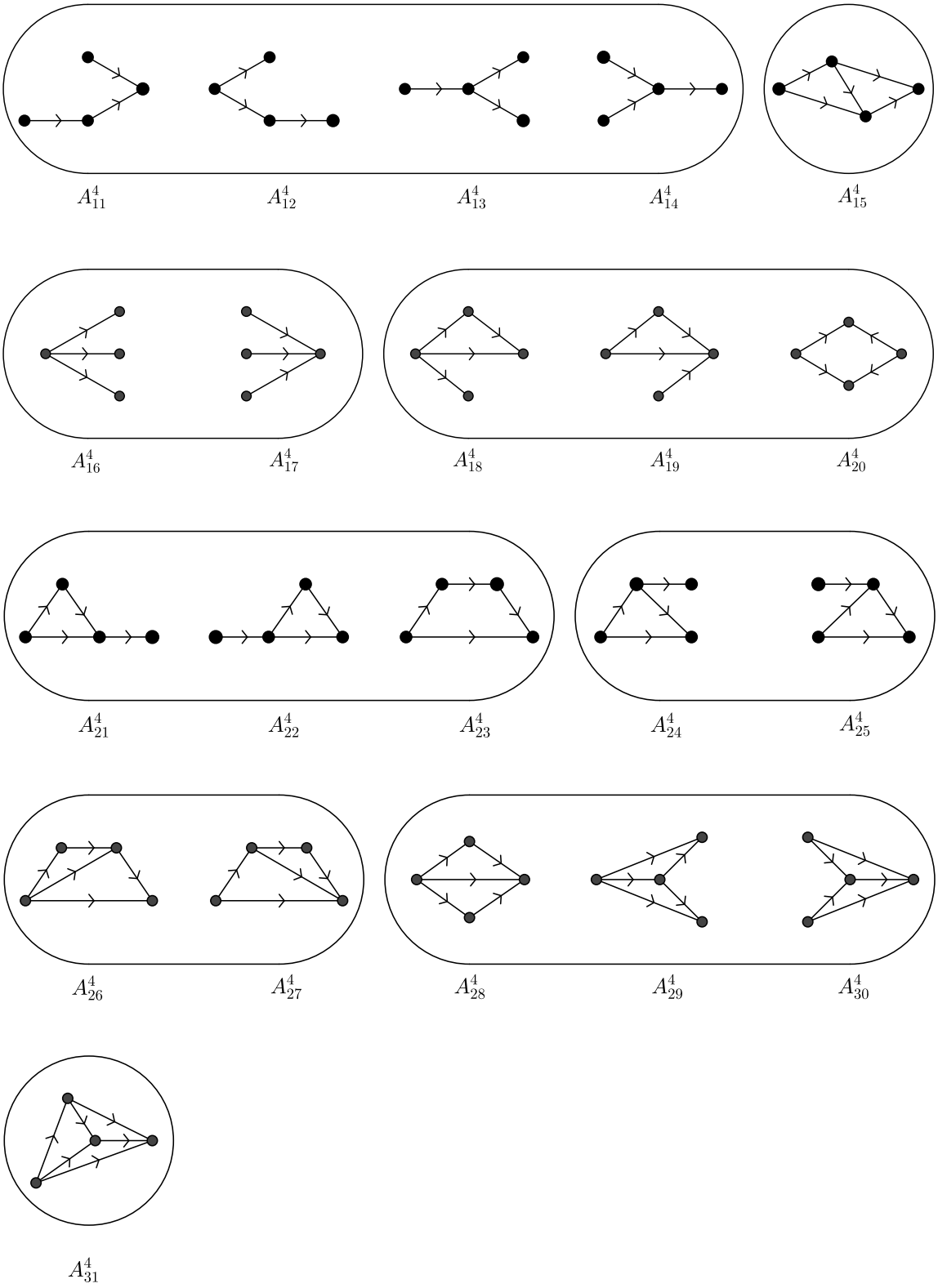


Figure 4: Acyclic digraphs up to order $n = 4$, grouped by C-equivalence.

A^4_{11} and A^4_{12} are reverses, as well as A^4_{13} and A^4_{14}. The C-equivalence between the four can also be explained by the recursive definition: selecting the upper edge of any of the digraphs, the deletion will be A^4_3 and the contraction will be A^3_2.

A^4_{16} and A^4_{17} are reverses. The C-equivalence between A^4_{18}, A^4_{19} and A^4_{20} is also explained by the recursive definition: selecting the upper-left edge of A^4_{18}, the upper-right edge of A^4_{19} and the lower-left edge of A^4_{20}, every deletion produces A^4_{12}, and every contraction produces A^3_2.

The same explanation is valid for A^4_{21}, A^4_{22} and A^4_{23}, selecting the lower edge that forms an underlying cycle. A^4_{24} and A^4_{25} are reverses, like A^4_{26} and A^4_{27}.

For A^4_{28}, A^4_{29} and A^4_{30} there is not a single edge whose deletion produces the same digraph for the three (observe that deleting the middle edge of A^4_{28} does not work, and deleting any other edge leaves a vertex with in-degree or out-degree 3, which the other two digraphs do not both have). However, we can choose edges to make the three deletions C-equivalent (selecting the lower-left, lower-right and lower-left, for instance). Therefore the cover polynomial can be expressed as the sum of the same two polynomials, one for a deletion and one for a contraction, for the three digraphs.

Another way of explaining this and other cases of C-equivalence is through the multiplicative formula and the linked union. Let u be a single vertex, then $A^4_{28} = u * A^3_5$, $A^4_{29} = u * A^3_4$ and $A^4_{30} = A^3_5 * u$. Since $C(A^3_4) = C(A^3_5)$, the three are C-equivalent. It is the case that $A^3_4 * u$ is the same as $u * A^3_5$, so this combination does not produce a fourth C-equivalent digraph.

4.2 Tournaments

Following we review some known properties about tournaments.

Lemma 4.11. *([3] Theorem 5.14) Every tournament contains a Hamiltonian path.*

Proof. Let T be a tournament of order n, and let $P : v_1, v_2, \dots, v_k$ be a longest path of T. If P is not a Hamiltonian path of T, then $1 \leq k < n$ and there is a vertex v of T not on P. Since P is a longest path, $(v, v_1), (v_k, v) \notin E(T)$, and so $(v_1, v), (v, v_k) \in E(T)$. This implies that there is an integer i $(1 \leq i < k)$ such that $(v_i, v) \in E(T)$ and $(v, v_{i+1}) \in E(T)$. But then

$$v_1, v_2, \dots, v_i, v, v_{i+1}, \dots, v_k$$

is a path whose length exceeds that of P, producing a contradiction. □

Lemma 4.12. *([3] Corollary 5.6) For every positive integer n, there is exactly one acyclic tournament of order n.*

Proof. This proof is different than the one given by the source. Let T_1, T_2 be 2 acyclic tournaments of order n. By Lemma 4.11, we can take a Hamiltonian path for both and define the natural bijection associating the vertices occupying the same position in the path. As it is a tournament, for every pair of vertices of T_1 and their images on T_2, one edge in one of the directions will exist. The edge will follow the direction of the ordering induced by the path in both cases, as otherwise a cycle would be formed. Therefore, the bijection is an isomorphism. □

By Corollary 2.13 and Lemma 3.5, a graph can be determined by its cover polynomial to be an acyclic tournament of order n, and by Lemma 4.12 it will be the only one. Therefore:

Proposition 4.13. *The acyclic tournaments are C-unique.*

Remark 4.14. For acyclic tournaments, the cover polynomial has a simple form that can be determined by the multiplicative property of the cover polynomial (Corollary 2.12). The linked union, as required for this property, of n single vertices (each having cover polynomial x) produces the acyclic tournament T_n of order n, and therefore

$$C(T_n; x, y) = x^n .$$

Following we give, without proof, some more results that can be found in [3].

Proposition 4.15. *([3], Theorem 5.3) A tournament is transitive if and only if it is acyclic.*

Transitive tournaments are, therefore, the same as acyclic tournaments, and there is one for every positive integer n.

The strong components of a tournament are also tournaments (called *strong tournaments*). The partition of a tournament into strong components is often considered for its characteristic structure:

Theorem 4.16. *([3], Theorem 5.7) Let T be a tournament with exactly k strong components, and let $\widetilde{T}$ be the digraph of order k obtained by identifying the vertices in every strong component of T. Then, $\widetilde{T}$ is the transitive tournament of order k.*

We saw in Lemma 4.11 that every tournament contains a Hamiltonian path, but only strong tournaments contain Hamiltonian cycles:

Theorem 4.17. *(Camion's theorem) Every non-trivial strong tournament contains a Hamiltonian cycle.*

A stronger result is proven in [3]: strong tournaments contain cycles of every possible length ≥ 3. However, the former will suffice.

There is one strong tournament of order 1 (the trivial one, a vertex), one of order 3 (a 3-cycle), one of order 4 (see [3]), and more than one for higher orders. Assembling the previous results we can affirm:

Proposition 4.18. *Tournaments $T = C_3 * \cdots * C_3$ formed by the linked union of cycles of length 3 are C-unique.*

Proof. Let T be such a tournament and D be any digraph such that $C(D) = C(T)$. From the cover polynomial, it can be deduced that D is a tournament, and that D does not have cycles of length ≥ 4. Then, by Camion's theorem, every cycle will be of length ≤ 3. However, D has $n/3$ cycles of length 3 and each one must go into a different strong component, so every strong component will be C_3. Using Theorem 4.16, we can see that D and T are the same. □

Proposition 4.19. *Let S be the strong tournament of order 4. Then, tournaments $T = S * \cdots * S$ formed by the linked union of instances of S are C-unique.*

Proof. The strong tournament of order 4 happens to have only one Hamiltonian cycle, so the proof is analogue to the previous proposition. □

5. Computer assistance

Using computer assistance in order to find C-unique digraphs with few vertices or edges was an idea we had right from the beginning of this work. This has lead us to the development of a piece of software able to compute the $c_D(i,j)$ values and the cover polynomial of digraphs, and compare them looking for C-equivalence. The developed tool has been an aid to find some of the results presented in this thesis, and for classifying digraphs such as in Figure 4.

For the ease of development, we have not used any preexisting software that would require learning; and the performance, although careful programming is needed, has not been the focus. We have obtained comprehensive relations of C-unique digraphs for some types of digraphs up to 7 vertices. Pushing for obtaining results for larger digraphs could only have a limited effect, as the number of digraphs grows exponentially respect to the number of vertices or edges.

The source code, in Java, is currently available at `https://github.com/salvisolamartinell/cover-polynomial`. The most relevant Java classes are reproduced in Appendix B.

The recurrence of Definition 2.4 provides for an exponential time algorithm for computing the cover polynomial. Exponential is a fine time complexity considering that the problem of evaluating the cover polynomial is #P-hard for most points, as shown by Bläser-Dell in [1].

However, the approach taken has been slightly different. Using the same recurrence but a different base case that corresponds to the geometric version, we have computed the $c_D(i,j)$ values first, and have compared these directly for deducing C-equivalence. When the need to compute the proper cover polynomial coefficients arose, we used Proposition 2.17.

The paradigm of dynamic programming has been considered, as it is natural for this kind of recurrences. This would require to compute a canonical isomorphism for each digraph and to store graphs of all sizes at once. We have not developed this but it could be an improvement.

Being a directed acyclic graph or a directed tournament is determined by the cover polynomial, as seen in Corollary 2.13 and Lemma 3.5, respectively. These families can be considered apart in respect to their C-uniqueness, as they will never share the cover polynomial with graphs that are not in the family. We have used lists of digraphs by number of vertices in these two families that have been made publicly available online by Brendan McKay in [6]. The number of C-unique digraphs found using this data is shown at Tables 1 and 2. An observation made during these computations is that, when considering the classes of C-equivalent digraphs by size, the most common size is 2 (see the output of the sample script in Appendix B.3 for an example).

# of vertices	# of C-unique DAGs	# of total DAGs [8]
2	2	2
3	4	6
4	6	31
5	11	302
6	24	5984
7	94	243668

Table 1: Results of the computation run on DAGs

# of vertices	# of C-unique tournaments	# of total tournaments [9]
2	1	1
3	2	2
4	2	4
5	7	12
6	11	56
7	73	456
8	157	6880

Table 2: Results of the computation run on tournaments

6. Conclusions

The cover polynomial contains much information about a digraph, specifically anything that can be derived from the number of covers with i paths and j cycles. There is a transformation of the coefficients that yields the geometric cover polynomial, which contains the same information and has properties that can be useful when studying C-equivalence.

The information the cover polynomial contains is different to that of the Tutte polynomial. However, there is a parallelism on the definition and methods used to work with them.

The goals of this thesis stated in the introduction have been accomplished:

- we have got acquainted with the cover polynomial, and compared it to the Tutte polynomial;
- we have listed information that can be obtained from the cover polynomial and indicated how;
- we have explored some families of digraphs and found a few that are determined by the cover polynomial;
- we have used computer help to determine the number of C-unique digraphs in some families.

The approach to determine C-unique digraphs has been to consider families that are extremal for some of the invariants of the cover polynomial, and are therefore very symmetric. Some families of digraphs that remain to be studied this way are bidirectional cycles, general orientations of star graphs, some kind of bipartite digraphs, n-cubes, wheels and toroidal digraphs. This approach, however, ignores that most digraphs are not symmetric or extremal in this sense, and they may still be C-unique.

It is not clear yet whether C-unique digraphs are common or uncommon among all digraphs. The rapid growth of the number of digraphs, together with the need of generating them efficiently, has left computing the general case out of reach of this thesis. We can tell that C-unique digraphs seem to be rare in the families of DAGs and tournaments. Their number would increase significantly if we considered digraphs up to edge reversal, because most digraphs are probably not isomorphic to their reverse, and thus cannot be C-unique.

We have not found operations that leave the cover polynomial invariant, apart from the edge reversal and the reorder of linked unions. If there are not more general operations that preserve the cover polynomial, it may be possible for most digraphs, up to edge reversal, to be C-unique. This may not be the case for

DAGs and tournaments because of their high symmetry, or may not be the case only for the small orders that we have computed.

We saw in Section 4.1 that the C-equivalence of the small acyclic digraphs could be explained by different means: by the reversion of edges, by the recursive definition with equivalent deletion and contraction or by a linked union with equivalent components. The explanation using the recursive definition seems to apply for every case, i.e., C-equivalent digraphs seem to have a loop or a non-loop edge such that the result of the deletion of this edge has the same cover polynomial for all of them, and the result of the contraction of this edge also has the same cover polynomial for all of them.

An open question is, therefore, can we find a pair of C-equivalent digraphs that cannot be explained using this method? It would be enough to see that they don't have C-equivalent deletions. Whether the answer is yes or no, it will help to better understand the structure of C-equivalent digraphs.

The subject of digraphs that are determined by their cover polynomial can be further researched. This work may be a good basis for anyone who wishes to do so.

7. References

[1] M. Bläser and H. Dell, "Complexity of the cover polynomial", *ICALP* (2007), 801-812.

[2] J.E. Bonin and A. de Mier, "Tutte uniqueness and Tutte equivalence", chapter in the forthcoming *Handbook on the Tutte polynomial and Related Topics* (J. Ellis-Monaghan and I. Moffatt, eds.), CRC Press, available from the authors by request.

[3] G. Chartrand and L. Lesniak, *Graphs & Digraphs: Fourth Edition*, CRC Press (2004), 113-126. ISBN 978-1-58488-390-6.

[4] F.R.K. Chung and R.L. Graham, "On the cover polynomial of a digraph", *Journal of combinatorial theory, Series B* **65**(2) (1995), 273-290.

[5] O.M. D'Antona and E. Munarini, "The cycle-path indicator polynomial of a digraph", *Advances in Applied Mathematics* **25**(1) (2000), 41-56.

[6] B. McKay, "Digraphs", personal web page, available at `http://users.cecs.anu.edu.au/~bdm/data/digraphs.html`, retrieved on 26 November 2017.

[7] A. de Mier, "Graphs and matroids determined by their Tutte polynomials", Ph.D. thesis directed by M. Noy, Universitat Politècnica de Catalunya, 2003.

[8] N.J.A. Sloane, "A003087 Number of acyclic digraphs with n unlabeled nodes", *On-Line Encyclopedia of Integer Sequences*, available at `https://oeis.org/A003087`, retrieved on 4 January 2018.

[9] N.J.A. Sloane, "A000568 Number of outcomes of unlabeled n-team round-robin tournaments", *On-Line Encyclopedia of Integer Sequences*, available at `https://oeis.org/A000568`, retrieved on 4 January 2018.

A. Basic notions of digraph theory

Digraph theory is the branch of graph theory concerning directed graphs, which appear when graph edges are replaced by ordered pairs of vertices. Digraph theory has directed analogues for many graph concepts including edges, paths and cycles.

A *directed graph*, or *digraph*, is a pair $D = (V, E)$, where V is a finite set of *vertices*, and $E \subseteq V \times V$ is a set of *edges*. Notations include $V = V(D)$ for the vertex set; $E = E(D)$ for the edge set; and $(u, v) = uv$ for edges, with *origin* u and *end* v. The *order* of a digraph is the number of vertices.

An *orientation* of a graph is an assignation of a direction to each edge, effectively forming a digraph. The *underlying graph* of a digraph is the graph obtained when directed edges are replaced by undirected edges.

An *isomorphism of digraphs* is a bijection between the vertex sets of two digraphs D_1 and D_2, $f\colon V(D_1) \to V(D_2)$, such that any two vertices u and v of D_1, uv is an edge of D_1 if and only if $f(u)f(v)$ is an edge of D_2.

The *indegree* $deg^-(u)$ and *outdegree* $deg^+(u)$ are the number of edges with end u and origin u, respectively. The *indegree (outdegree) sequence* of a digraph is the sequence obtained by ordering the indegrees (outdegrees) of all vertices in increasing order.

A *path* is an alternating sequence of vertices and edges, starting and ending at a vertex, without vertex nor edge repetition. A *cycle* is the set of elements of an alternating sequence of vertices and edges, starting and ending at the same vertex, without any further vertex nor edge repetition. Paths and cycles are often considered digraphs on their own. The *length* of a path or cycle is its number of edges. Paths of length n are denoted P_n and cycles of length n are denoted C_n.

A vertex u is said to be *reachable* from another vertex v if there exists a path from v to u. A *Hamiltonian path* is a path that goes through every vertex. A *Hamiltonian cycle* is a cycle that goes through every vertex.

A *loop* is a cycle of length 1. The *girth* of a digraph is the length of its shortest cycle. A digraph is called *acyclic* (or *directed acyclic graph*, DAG) if it does not contain cycles.

A *vertex cover* or *covering* of a digraph is a subset of its edges that use every vertex, either as origin or as end.

A *subdigraph* is generated from a subset of the vertices of a digraph by taking all the existing edges between vertices of this subset. Subdigraphs are digraphs on their own.

A digraph is said to be *weakly connected* if its underlying graph is connected. It is said to be *strongly connected* if every vertex is reachable from every other vertex. Maximal weakly connected subdigraphs are called *weak components* and maximal strongly connected subdigraphs are called *strong components*.

Two digraphs are said to be *disjoint* if their vertex sets are disjoint. The *union* of two digraphs $D_1 \cup D_2$ is a digraph with vertex set $V(D_1) \cup V(D_2)$ and edge set $E(D_1) \cup E(D_2)$. It is called *disjoint union* if the graphs are disjoint.

The *complete digraph* is a digraph that, for a given vertex set, contains all the possible edges.

B. Java source code

A selection of the Java classes that have been written to provide computer assistance during this work. A sample script and output is provided at the end.

B.1 Digraph package

The `digraph` Java package provides the core functionality for computing the path-cycle covers and the cover polynomial. It can be used as a library by other scripts. It includes the Java classes `Digraph`, `PathCycleCovers` and `CoverPolynomial`.

Digraph.java

```
package digraph;

import java.util.BitSet;

/**
 * Directed graph
 */
public class Digraph {
   private final int n;
   BitSet adjMatrix;
   public Digraph (int n) {
      this.n = n;
      adjMatrix = new BitSet(n*n);
   }

   public Digraph (int n, BitSet adjMatrix) {
      this.n = n;
      this.adjMatrix = adjMatrix;
   }

   public void addArc(int u, int v) {
      if (u > n || v > n || u < 0 || v < 0) throw new IllegalArgumentException();
      adjMatrix.set(u*n + v);
   }

   /**
    * Computes the path-cycle covers of the digraph by using the deletion-contraction
        recursion
    * @return array with the number of path-cycle covers with i paths and j cycles
    */
   public PathCycleCovers pathCycleCovers() {
      int[][] coefs = new int[n+1][n+1];
      if (adjMatrix.isEmpty()) {
         coefs[n][0] = 1;
      } else {
         int arc = adjMatrix.previousSetBit(n*n - 1);
```

```
      int u = arc/n;
      int v = arc%n;
      int[][] delCoefs = deletion(u,v).pathCycleCovers().getCoefs();
      int[][] conCoefs = contraction(u,v).pathCycleCovers().getCoefs();
      for (int i = 0; i <= n; ++i) {
         for (int j = 0; j <= n; j++) {
            coefs[i][j] += delCoefs[i][j];
            if (u != v) {
               if (i < n && j < n) coefs[i][j] += conCoefs[i][j];
            } else {
               if (i < n && j > 0) coefs[i][j] += conCoefs[i][j-1];
            }
         }
      }

   }
   return new PathCycleCovers(coefs);
}

/**
 * Deletes an arc
 * @param u origin of the arc to delete
 * @param v destination of the arc to delete
 * @return a new {@link Digraph} resulting of the (u,v) deeletion
 */
public Digraph deletion(int u, int v) {
   BitSet delAdjMatrix = (BitSet) adjMatrix.clone();
   delAdjMatrix.clear(u*n + v);
   Digraph d_del = new Digraph(n, delAdjMatrix);
   return d_del;
}

/**
 * Contracts an arc using the Chung-Graham method
 * @param u origin of the arc to contract
 * @param v destination of the arc to contract
 * @return a new {@link Digraph} resulting of the (u,v) contraction
 */
public Digraph contraction(int u, int v) {
   BitSet conAdjMatrix = new BitSet((n-1)*(n-1));
   for (int i = 0; i < n; ++i) {
      for (int j = 0; j < n; ++j) {
         if (i == u || j == v);
         else {
            if (adjMatrix.get(i*n + j)) conAdjMatrix.set(newVertex(u,v,i)*(n-1) +
                newVertex(u,v,j));
         }
      }
   }
   Digraph d_con = new Digraph(n-1, conAdjMatrix);
   return d_con;
}
```

```
   // Computes the new index of a vertex i after a (u,v) contraction
   private static int newVertex(int u, int v, int i) {
      if (i < v) return i;
      if (i > v) return i - 1;
      if (u < v) return u;
      if (u > v) return u - 1;
      throw new RuntimeException();
   }

   /**
    * For a visual representation of the digraph, paste the output to <a
        href="http://www.webgraphviz.com/" target="_parent">www.webgraphviz.com</a>
    */
   public void print() {
      System.out.println("digraph D {");
      for (int i = 0; i < n; ++i) {
         System.out.println(" "+'"'+i+'"');
         for (int j = 0; j < n; ++j) {
            if (adjMatrix.get(i*n + j)) System.out.println(" "+'"'+i+'"'+" -> "+'"'+j+'"');
         }
      }
      System.out.println("}");
   }

   /**
    * Prints the adjacency matrix
    */
   public void printAdjMatrix() {
      for (int i = 0; i < n; ++i) {
         for (int j = 0; j < n; ++j) {
            System.out.print(adjMatrix.get(i*n + j) ? '1' : '0');
         }
          System.out.println();
      }
   }
}
```

PathCycleCovers.java

```
package digraph;

import java.util.Arrays;

/**
 * A class for the array that contains the number of path-cycle covers of a digraph
 * for every given number of paths and cycles. Also used to compute the coefficients
 * of the cover polynomial. Two instances of this class are equal if and only if
 * the cover polynomials of the original graphs are equal.
 * Warning: ints used, may overflow for big digraphs (many edges)
```

```
 */
public class PathCycleCovers {

   private static final int[][] signedStirlingFirstKind = {
      {1},
      {0,1},
      {0,-1,1},
      {0,2,-3,1},
      {0,-6,11,-6,1},
      {0,24,-50,35,-10,1},
      {0,-120,274,-225,85,-15,1},
      {0,720,-1764,1624,-735,175,-21,1},
      {0,-5040,13068,-13132,6769,-1960,322,-28,1},
      {0,40320,-109584,118124,-67284,22449,-4536,546,-36,1}
   };

   private final int[][] coefs;

   public PathCycleCovers(int[][] coefs) {
      this.coefs = coefs;
   }

   public int[][] getCoefs() {
      return coefs;
   }

   public void print() {
      System.out.println("Path-cycle covers coefs:");
      int size = coefs.length;
      for (int i = 0; i < size; i++) {
         for (int j = 0; j < size; j++) {
            System.out.print(coefs[i][j] + "\t");
         }
         System.out.println();
      }
   }

   public CoverPolynomial getCoverPolynomial() {
      int size = coefs.length;
      int[][] coverPolCoefs = new int[size][size];
      for (int i = 0; i < size; ++i) {
         for (int j = 0; j < size; ++j) {
            for (int k = i; k < size; ++k) {
               coverPolCoefs[i][j] += signedStirlingFirstKind[k][i]*coefs[k][j];
            }
         }
      }
      return new CoverPolynomial(coverPolCoefs);
   }

   @Override
   public int hashCode() {
```

```
      final int prime = 31;
      int result = 1;
      result = prime * result + Arrays.deepHashCode(coefs);
      return result;
   }

   @Override
   public boolean equals(Object obj) {
      if (this == obj)
         return true;
      if (obj == null)
         return false;
      if (getClass() != obj.getClass())
         return false;
      PathCycleCovers other = (PathCycleCovers) obj;
      if (!Arrays.deepEquals(coefs, other.coefs))
         return false;
      return true;
   }
}
```

CoverPolynomial.java

```
package digraph;

import java.util.Arrays;

/**
 * Class to represent cover polynomial instances.
 * Provides methods for printing and comparing.
 */
public class CoverPolynomial {

   private final int[][] coefs;

   public CoverPolynomial(int[][] coefs) {
      this.coefs = coefs;
   }

   public void print() {
      System.out.println("Cover polynomial coefs:");
      int size = coefs.length;
      for (int i = 0; i < size; i++) {
         for (int j = 0; j < size; j++) {
            System.out.print(coefs[i][j] + "\t");
         }
         System.out.println();
      }
   }
```

```
   @Override
   public int hashCode() {
      final int prime = 37;
      int result = 1;
      result = prime * result + Arrays.deepHashCode(coefs);
      return result;
   }

   @Override
   public boolean equals(Object obj) {
      if (this == obj)
         return true;
      if (obj == null)
         return false;
      if (getClass() != obj.getClass())
         return false;
      CoverPolynomial other = (CoverPolynomial) obj;
      if (!Arrays.deepEquals(coefs, other.coefs))
         return false;
      return true;
   }
}
```

B.2 Sample generator

A generator (a Java `Iterator`) for the DAGs obtained from [6].

AcyclicDigraphGenerator.java

```
package generator;

import java.io.BufferedReader;
import java.io.FileNotFoundException;
import java.io.FileReader;
import java.io.IOException;
import java.util.BitSet;
import java.util.Iterator;
import java.util.NoSuchElementException;

import digraph.Digraph;

/**
 * Iterator to generate directed acyclic graphs (DAGs) for a given number of vertices
 * reading from txt files such as the ones that can be found in the Acyclic graphs section
     at
 * <a href="http://users.cecs.anu.edu.au/~bdm/data/digraphs.html"
     target="_parent">http://users.cecs.anu.edu.au/~bdm/data/digraphs.html</a>.
 * Create such files in the resources folder of this project for this iterator to work.
 */
```

```
public class AcyclicDigraphGenerator implements Iterator<Digraph> {

   private final int n;
    private final BufferedReader in;
   private String nextLine;
   private boolean nextLinePrepared = false;

   /**
    * @param n the number of vertices that the generated DAGs will have
    * @throws FileNotFoundException if the dag{n}.txt file doesn't exist in the resources
        folder
    */
   public AcyclicDigraphGenerator(int n) throws FileNotFoundException {
      this.n = n;
      in = new BufferedReader(new FileReader("resources\\dag"+n+".txt"));
   }

   @Override
   public boolean hasNext() {
      prepareNextLine();
      return nextLine != null;
   }

   @Override
   public Digraph next() {
      prepareNextLine();
      if (nextLine == null) throw new NoSuchElementException();
      BitSet adjMatrix = new BitSet(n*n);
      int k = 0;
      for (int i = 0; i < n-1; ++i) {
         for (int j = i + 1; j < n; ++j) {
            if (nextLine.charAt(k) == '1') adjMatrix.set(i*n+j);
            ++k;
         }
      }
      nextLinePrepared = false;
      return new Digraph(n, adjMatrix);
   }

   private void prepareNextLine() {
      if (!nextLinePrepared) {
         try {
            nextLine = in.readLine();
         } catch (IOException e) {
            nextLine = null;
            e.printStackTrace();
         } finally {
            nextLinePrepared = true;
         }
      }
   }
}
```

B.3 Sample script

A showcase script that counts the classes of C-equivalent DAGs of order 5 by size of the class. The output is shown below.

DAGs.java

```
package scripts;

import java.io.FileNotFoundException;
import java.util.HashMap;
import java.util.HashSet;
import java.util.Map;
import java.util.Set;
import java.util.TreeMap;

import digraph.Digraph;
import digraph.PathCycleCovers;
import generator.AcyclicDigraphGenerator;

public class DAGs {
   public static void main(String[] args) {

      Map<PathCycleCovers, Set<Digraph>> pccMap = new HashMap<>();
      int n = 5;
      System.out.println("DAGs of order " + n);
      try {
         AcyclicDigraphGenerator generator = new AcyclicDigraphGenerator(n);
         while (generator.hasNext()) {
            Digraph d = generator.next();
            PathCycleCovers pcc = d.pathCycleCovers();
            pccMap.computeIfAbsent(pcc, k -> new HashSet<>()).add(d);
         }
      } catch (FileNotFoundException e) {
         e.printStackTrace();
      }

      Map<Integer, Integer> classesBySize = new TreeMap<>();

      pccMap.forEach((k,v) -> {
         classesBySize.putIfAbsent(v.size(), 0);
         classesBySize.put(v.size(), classesBySize.get(v.size()) + 1);
         /* For printing the digraphs grouped by C-equivalence:
         System.out.println("Number of C-equivalent digraphs: " + v.size());
         v.forEach(g -> {
            g.printAdjMatrix();
            System.out.println("----");
            //g.print();
         });
         System.out.println();*/
      });
```

```
      System.out.println("Classes of C-equivalent digraphs by size of the class");
      System.out.println();
      System.out.println("Size\t# of classes");
      System.out.println("--------------------");
      classesBySize.forEach((k,v) -> {
         System.out.print(""+k+"\t"+v);
         if (k==1) System.out.print("\t  <-- # of C-unique digraphs");
         System.out.println();
      });
      System.out.println("-----------");
      int classTotal = classesBySize.values().stream().mapToInt(i -> i).sum();
      int digraphTotal = classesBySize.entrySet().stream()
            .mapToInt(e -> e.getKey()*e.getValue()).sum();
      System.out.println("Total number of classes: " + classTotal);
      System.out.println("Total number of digraphs: " + digraphTotal);
   }
}
```

Output

```
DAGs of order 5
Classes of C-equivalent digraphs by size of the class

Size    # of classes
--------------------
1       11            <-- # of C-unique digraphs
2       19
3       4
4       8
5       6
6       12
8       7
10      4
11      1
-----------
Total number of classes: 72
Total number of digraphs: 302
```