Combinatorics, Probability and Computing (2010) 19, 133–157. © Cambridge University Press 2009
doi:10.1017/S0963548309990381

Weighted Interlace Polynomials

LORENZO TRALDI

Lafayette College, Easton, Pennsylvania 18042, USA
(e-mail: traldil@lafayette.edu)

Received 21 May 2008; revised 26 June 2009; first published online 10 August 2009

The interlace polynomials introduced by Arratia, Bollobás and Sorkin extend to invariants of graphs with vertex weights, and these weighted interlace polynomials have several novel properties. One novel property is a version of the fundamental three-term formula

$$q(G) = q(G - a) + q(G^{ab} - b) + ((x-1)^2 - 1)q(G^{ab} - a - b)$$

that lacks the last term. It follows that interlace polynomial computations can be represented by binary trees rather than mixed binary–ternary trees. Binary computation trees provide a description of $q(G)$ that is analogous to the activities description of the Tutte polynomial. If G is a tree or forest then these 'algorithmic activities' are associated with a certain kind of independent set in G. Three other novel properties are weighted pendant-twin reductions, which involve removing certain kinds of vertices from a graph and adjusting the weights of the remaining vertices in such a way that the interlace polynomials are unchanged. These reductions allow for smaller computation trees as they eliminate some branches. If a graph can be completely analysed using pendant-twin reductions, then its interlace polynomial can be calculated in polynomial time. An intuitively pleasing property is that graphs which can be constructed through graph substitutions have vertex-weighted interlace polynomials which can be obtained through algebraic substitutions.

1. Introduction

Motivated by problems arising from DNA sequencing and the properties of circle graphs of 2-in, 2-out digraphs, Arratia, Bollobás and Sorkin introduced a new family of graph invariants, the interlace polynomials, in [3, 4, 5]. These invariants may be defined either through recursive elimination of vertices or as sums indexed by subsets of vertices [1, 5], much as the Tutte polynomial may be defined either through recursive elimination of edges or as a sum indexed by subsets of edges. Ellis-Monaghan and Sarmiento [20] have shown (among other results) that one of the one-variable interlace polynomials, q_N, can be computed in polynomial time for bipartite distance hereditary graphs. Their proof depends on the corresponding result for the Tutte polynomials of series-parallel graphs, first proved in [26], and the fact that bipartite distance hereditary graphs are circle graphs of Euler circuits in medial graphs of series-parallel graphs.

https://doi.org/10.1017/S0963548309990381 Published online by Cambridge University Press

In this paper we discuss several useful features of interlace polynomials that have been modified to incorporate vertex weights. After defining the weighted interlace polynomials in Section 2, we observe that a simple adjustment of weights makes it unnecessary to have the third term in the fundamental recursion $q(G) = q(G-a) + q(G^{ab}-b) + ((x-1)^2-1)q(G^{ab}-a-b)$ of [5]. In Section 3 we present reduction formulas that can be used to eliminate a vertex that is a twin of another, or pendant on another. These pendant-twin reductions are analogous to the series-parallel reductions of electrical circuit theory, in which two resistors wired in series (resp. parallel) are replaced by one resistor with $R = R_1 + R_2$ (resp. $R^{-1} = R_1^{-1} + R_2^{-1}$). The pendant-twin reductions are used to extend the result of Ellis-Monaghan and Sarmiento mentioned above to the two-variable interlace polynomials of looped, non-bipartite distance hereditary graphs. In Section 4 we show that if $G = H * K$ is a looped graph obtained using the composition construction of Cunningham [19], then $q(G)$ is equal to the interlace polynomial of a suitably re-weighted version of K; this generalizes results of [4] that describe the q_N polynomials of simple graphs constructed through substitution. Composition has proved useful in the study of circle graphs (see, for instance, [11], [12], [17]), so it is not surprising to see it appear in the theory of the interlace polynomials. In Section 5 we discuss some elementary properties of the unweighted q_N polynomial, focusing on simple (unlooped) graphs. In Section 6 we sketch a combinatorial description of the interlace polynomials of trees and forests introduced by Anderson, Cutler, Radcliffe and the present author in [2]. This combinatorial description bears a striking resemblance to the activities description of the Tutte polynomial, and in Section 7 we extend it to arbitrary graphs using 'activities' defined with respect to recursive interlace polynomial calculations. We do not know whether or not these activities have a convenient combinatorial description in general.

We should observe that the idea of using vertex weights for interlace polynomials has appeared before, though our implementation of the idea is different from those we have seen elsewhere. In [16], Courcelle introduced a multivariate interlace polynomial that is more complicated than the polynomials we consider here, which involves assigning indeterminates to the vertices of a graph. He used monadic second-order logic to show that it is possible to compute bounded portions of this polynomial (and the entire unweighted interlace polynomial q) in polynomial time for graphs of bounded clique-width. This technique is quite general but involves large built-in constants, so for pendant-twin reductions and compositions the formulas presented here are considerably more practical. Also, Bläser and Hoffmann [8] use the idea of assigning indeterminates to vertices, along with the adjunction of two types of pendant-twin vertices, to show that evaluating interlace polynomials is generally $\#P$-hard for almost all values of the variables.

In an early version of [4] available on the arxiv, Arratia, Bollobás and Sorkin observe that there is a natural (so natural it is 'practically a tautology') correspondence between the Kauffman bracket of an alternating link diagram and an interlace polynomial of an interlace graph of an associated 2-in, 2-out digraph. The situation is clear enough that we do not discuss it in detail, but it is worth mentioning that this correspondence may be extended to arbitrary link diagrams using vertex weights. The well-known relationship between the Jones and Tutte polynomials is similar, in that an edge-weighted or edge-signed version of the Tutte polynomial conveniently incorporates crossing information when dealing with non-alternating links [23, 25, 27].

https://doi.org/10.1017/S0963548309990381 Published online by Cambridge University Press

2. Expansions and recursions

We recall terminology and notation of [5]. *Graphs* may have loops but not multiple edges or multiple loops. The *rank* $r(G)$ and *nullity* $n(G)$ of a graph G are those of its adjacency matrix considered over $GF(2)$. If $S \subseteq V(G)$ then $G[S]$ is the subgraph of G induced by S. In addition, we say a graph is (*vertex-*) *weighted* by functions α and β mapping $V(G)$ into some commutative ring with unity R. For ease of notation we prefer to denote a weighted graph G rather than using the triple (G, α, β); even when two graphs differ only in their weights we will denote them G and G' rather than (G, α, β) and (G, α', β'). Also, if a graph is modified then, unless otherwise stated, we presume that the weight functions α and β are modified in the most natural way. For instance, if $a \in V(G)$ then the other vertices of G have the same vertex weights in $G - a$ as they have in G. If G is a weighted graph then the *unweighted version* of G is denoted G^u; it has the same underlying graph and the trivial weights $\alpha \equiv \beta \equiv 1 \in \mathbb{Z}$.

Definition 1. If G is a vertex-weighted graph, then the *weighted interlace polynomial* of G is

$$q(G) = \sum_{S \subseteq V(G)} \left(\prod_{s \in S} \alpha(s) \right) \left(\prod_{v \notin S} \beta(v) \right) (x-1)^{r(G[S])} (y-1)^{n(G[S])}.$$

Definition 2. If G is a vertex-weighted graph, then the *weighted vertex-nullity interlace polynomial* of G is

$$q_N(G) = \sum_{S \subseteq V(G)} \left(\prod_{s \in S} \alpha(s) \right) \left(\prod_{v \notin S} \beta(v) \right) (y-1)^{n(G[S])}.$$

Definition 3. If G is a vertex-weighted graph, then the *weighted vertex-rank interlace polynomial* of G is

$$q_R(G) = \sum_{S \subseteq V(G)} \left(\prod_{s \in S} \alpha(s) \right) \left(\prod_{v \notin S} \beta(v) \right) (x-1)^{r(G[S])}.$$

The original, unweighted interlace polynomials of G are recovered by using G^u.

The three weighted interlace polynomials of a particular weighted graph G may be substantially different from each other. Considered as functions defined on the class of all weighted graphs, however, the three polynomials are essentially equivalent: if $\tilde{G}$ is obtained from G by re-weighting $V(G) = \{v_1, \ldots, v_n\}$ using the indeterminates in the polynomial ring $\mathbb{Z}[\alpha_1, \ldots, \alpha_n, \beta_1, \ldots, \beta_n]$, then the value of any one weighted interlace polynomial on $\tilde{G}$ determines the values of all three polynomials on all weighted versions of G. For instance, $q(G)$ can be obtained from $q_N(\tilde{G})$ by substituting $(x-1) \cdot \alpha(v_i)$ for α_i, $\beta(v_i)$ for β_i and $1 + \frac{y-1}{x-1}$ for y. (In contrast, the functions defined by the three unweighted interlace polynomials are rather different from one another, as there are many pairs of graphs distinguished by q_R but not by q_N [5].) We mention all three weighted polynomials simply because one of them may be more convenient for some purposes than the

https://doi.org/10.1017/S0963548309990381 Published online by Cambridge University Press

others. Most of our results are stated for q because it specializes to the others most readily.

In fact any one of $q(\tilde{G})$, $q_N(\tilde{G})$, $q_R(\tilde{G})$ determines G up to isomorphism, because the vertex weights identify the contribution of each $S \subseteq V(G)$. The looped vertices of G appear in the 1-element subsets S of rank 1 (nullity 0), some pairs of adjacent vertices of G appear in the 2-element subsets S that contain at least one unlooped vertex and have rank 2 (nullity 0), and the other pairs of adjacent vertices of G appear in the 2-element subsets S that contain two looped vertices and have rank 1 (nullity 1). Moreover, the same comment applies even if only one of α, β is non-trivial, *i.e.*, if $\beta \equiv 1$ or $\alpha \equiv 1$. This might make it seem possible to simplify our discussion by considering only α or only β, but we prefer to use both weights because they produce especially easy-to-read formulas, as noted in the discussions of Theorems 2.2 and 3.1 below. Setting one weight identically equal to 1 would not simply leave us with the other weight; in essence, the remaining one would replace the ratio of the original two. Consequently, using only α or only β would entail unnecessary algebraic complications and losses of generality, because many formulas would require division. For instance, with $\beta \equiv 1$, Proposition 2.1 below would state that replacing $\alpha(a)$ with $\alpha'(a) = r_1\alpha(a)/(r_1 + r_2)$ results in a graph G' with $(r_1 + r_2)q(G') = r_1 q(G) + r_2 q(G - a)$. The special case $r_1 = -r_2$ would require a separate statement, and of course $\beta(a) = 0$ would be ruled out.

Our first proposition follows immediately from Definition 1.

Proposition 2.1. *Suppose $a \in V(G)$ and $r_1, r_2 \in R$. Let G' be obtained from G by changing the weights of a to $\alpha'(a) = r_1\alpha(a)$ and $\beta'(a) = r_1\beta(a) + r_2$. Then $q(G') = r_1 q(G) + r_2 q(G - a)$.*

A fundamental property of the interlace polynomials is that they can be calculated recursively with the local complementation and pivot operations used by Kotzig [24], Bouchet [13, 14] and Arratia, Bollobás and Sorkin [3, 4, 5].

Definition 4 (Local complementation). If a is a vertex of G, then G^a is obtained from G by toggling adjacencies $\{x, y\}$ involving neighbours of a that are distinct from a.

Definition 5 (Pivot). If a and b are distinct vertices of G, then the graph G^{ab} is obtained from G by toggling adjacencies $\{x, y\}$ such that $x, y \notin \{a, b\}$, x is adjacent to a in G, y is adjacent to b in G, and either x is not adjacent to b or y is not adjacent to a.

Note that local complementation includes loop-toggling (when $x = y$ is a neighbour of a distinct from a) but pivoting does not.

Theorem 2.2. *If G is a weighted graph, then $q(G)$ can be calculated recursively using the following properties.*

(a) *If a is a looped vertex, then*

$$q(G) = \beta(a)q(G - a) + \alpha(a)(x - 1)q(G^a - a).$$

https://doi.org/10.1017/S0963548309990381 Published online by Cambridge University Press

(b) *If a and b are loopless neighbours, then*

$$q(G) = \beta(a)q(G-a) + \beta(b)q(G^{ab}-b) + (\alpha(a)\alpha(b)(x-1)^2 - \beta(a)\beta(b))q(G^{ab}-a-b).$$

(c) *If G has no non-loop edges, then $q(G)$ is*

$$\left(\prod_{\text{unlooped } v\in V(G)} (\alpha(v)(y-1)+\beta(v))\right)\cdot\left(\prod_{\text{looped } v\in V(G)} (\alpha(v)(x-1)+\beta(v))\right).$$

Proof. The proofs of parts (a) and (b) of Theorem 2.2 are essentially the same as the proofs of the corresponding formulas for the unweighted two-variable interlace polynomial [5]. Indeed, if we read α as 'includes' and β as 'excludes', then the weighted formulas serve as mnemonic devices to recall the proofs.

For instance part (a) is proved as follows. The term $\beta(a)q(G-a)$ reflects the fact that if $a \notin S \subseteq V(G)$ then $G[S] = (G-a)[S]$, so the contributions of S to $q(G)$ and $q(G-a)$ differ only by a factor $\beta(a)$. The term $\alpha(a)(x-1)q(G^a-a)$ reflects the fact that if $a \in S \subseteq V(G)$ then

$$r(G[S]) = r\begin{pmatrix} 1 & \mathbf{1} & \mathbf{0} \\ \mathbf{1} & M_{11} & M_{12} \\ \mathbf{0} & M_{21} & M_{22} \end{pmatrix} = 1 + r\begin{pmatrix} M_{11}^c & M_{12} \\ M_{21} & M_{22} \end{pmatrix} = 1 + r((G^a-a)[S-a]),$$

where bold numerals represent rows and columns and M_{11}^c differs from M_{11} in every entry. Hence the contributions of S to $q(G)$ and $S-\{a\}$ to $q(G^a-a)$ differ by a factor $\alpha(a)(x-1)$.

The more complicated formula of part (b) reflects, among other things, the fact that each subset $S \subseteq V(G)$ with $a, b \notin S$ contributes to all three terms; the last two contributions cancel each other.

Part (c) follows directly from Definition 1. It is not technically necessary to mention graphs with loops in (c), as loops can always be removed using part (a). However, applying part (a) to completely disconnected graphs is obviously inefficient. □

Other properties of the unweighted interlace polynomials also extend naturally to the weighted polynomials. For instance, parts (b) and (c) of Theorem 2.3 below extend two properties discussed in Section 3 of [5]. Part (a) will be useful in Section 4, where we discuss substituted graphs.

Theorem 2.3.

(a) *If $a \in V(G)$ is loopless, then*

$$q(G) - \beta(a)q(G-a) = q(G^a) - \beta(a)q(G^a-a).$$

(b) *If $a, b \in V(G)$ are loopless neighbours, then*

$$q(G-a) - \beta(a)q(G-a-b) = q(G^{ab}-a) - \beta(a)q(G^{ab}-a-b).$$

(c) *If G is the union of disjoint subgraphs G_1 and G_2, then $q(G) = q(G_1)q(G_2)$.*

https://doi.org/10.1017/S0963548309990381 Published online by Cambridge University Press

Proof. To prove (a), note that if $a \in S \subseteq V(G)$ then the adjacency matrix of $G[S]$ may be represented by

$$\begin{pmatrix} 0 & \mathbf{1} & \mathbf{0} \\ \mathbf{1} & M_{11} & M_{12} \\ \mathbf{0} & M_{21} & M_{22} \end{pmatrix}.$$

Adding the first row to each of those in the second group results in

$$\begin{pmatrix} 0 & \mathbf{1} & \mathbf{0} \\ \mathbf{1} & M_{11}^c & M_{12} \\ \mathbf{0} & M_{21} & M_{22} \end{pmatrix},$$

the adjacency matrix of $G^a[S]$. Definition 1 then tells us that

$$\begin{aligned} q(G) - \beta(a)q(G-a) &= \sum_{a \in S \subseteq V(G)} \left(\prod_{s \in S} \alpha(s)\right)\left(\prod_{v \notin S} \beta(v)\right)(x-1)^{r(G[S])}(y-1)^{n(G[S])} \\ &= \sum_{a \in S \subseteq V(G)} \left(\prod_{s \in S} \alpha(s)\right)\left(\prod_{v \notin S} \beta(v)\right)(x-1)^{r(G^a[S])}(y-1)^{n(G^a[S])} \\ &= q(G^a) - \beta(a)q(G^a - a). \end{aligned}$$

The proofs of (b) and (c) are essentially the same as the proofs of the corresponding results in [5]. □

Many properties of the unweighted interlace polynomials extend directly to the weighted polynomials, as we see in Theorems 2.2 and 2.3. It may be a surprise that some properties of the unweighted interlace polynomials can be significantly simplified using vertex weights. For instance, consider a computation tree representing a recursive implementation of Theorem 2.2. The tree has two branches for each application of part (a), three branches for each application of part (b), and a leaf for each application of part (c). As noted in the proof of Theorem 2.2, the three terms of part (b) all incorporate contributions from the same subsets $S \subseteq V(G)$. The recursive computation calculates these same contributions three times, on separate branches. This inefficiency can be eliminated by rephrasing part (b) of Theorem 2.2 so that no three-fold branches are necessary.

Corollary 2.4. *If a and b are loopless neighbours in G, then*

$$q(G) = \beta(a)q(G-a) + \alpha(a)q((G^{ab} - b)'),$$

where $(G^{ab} - b)'$ is obtained from $G^{ab} - b$ by changing the weights of a to $\alpha'(a) = \beta(b)$ and $\beta'(a) = \alpha(b)(x-1)^2$.

Proof. Let $(G^{ab} - b)''$ be the graph obtained from $G^{ab} - b$ by changing the weights of a as in Proposition 2.1, with $r_1 = \beta(b)$ and $r_2 = \alpha(a)\alpha(b)(x-1)^2 - \beta(a)\beta(b)$. Then $\alpha''(a) = a(a)\beta(b)$ and $\beta''(a) = \alpha(a)\alpha(b)(x-1)^2$, so $q((G^{ab} - b)'') = \alpha(a)q((G^{ab} - b)')$. On the other hand, Proposition 2.1 tells us that

$$q((G^{ab} - b)'') = \beta(b)q(G^{ab} - b) + (\alpha(a)\alpha(b)(x-1)^2 - \beta(a)\beta(b))q(G^{ab} - b - a). \qquad \square$$

https://doi.org/10.1017/S0963548309990381 Published online by Cambridge University Press

If $\alpha'(a) = \beta(b)/(x-1)$ and $\beta'(a) = \alpha(b)(x-1)$ are used instead of the weights given in Corollary 2.4, then the resulting formula $q(G) = \beta(a)q(G-a) + \alpha(a)(x-1)q((G^{ab}-b)')$ still has only two branches, and bears an interesting resemblance to part (a) of Theorem 2.2. However, using division to define $\alpha'(a)$ may occasionally cause some algebraic difficulties, *e.g.*, it complicates the evaluation at $x = 1$, and it prohibits the use of rings in which $x-1$ is a divisor of zero.

Corollary 2.4 is one of several results in which vertex weights allow us to extend properties of the unweighted version of q_N, seemingly the simplest kind of interlace polynomial, to the other interlace polynomials. In this instance the extended property is that of possessing a recursive description represented by a binary computation tree. Another result of this type is Corollary 2.5, which extends Remark 18 of [4]: if a and b are loopless neighbours in G, then the unweighted vertex-nullity interlace polynomials of G and G^{ab} are the same.

Corollary 2.5. *Let a and b be unlooped neighbours in G, and let $(G^{ab})'$ be the weighted graph obtained from G^{ab} by changing the weights of a and b to*

$$\alpha'(a) = \beta(b), \quad \beta'(a) = \alpha(b)(x-1)^2, \quad \alpha'(b) = \beta(a) \quad \textit{and} \quad \beta'(b) = \alpha(a)(x-1)^2.$$

Then $(x-1)^2 q(G) = q((G^{ab})')$.

Proof. Applying Corollary 2.4 to $(G^{ab})'$, with the roles of a and b reversed, tells us that $q((G^{ab})') = \beta'(b)q((G^{ab})' - b) + \alpha'(b)q((((G^{ab})')^{ab} - a)')$. Observe that $(((G^{ab})')^{ab} - a)'$ has the underlying graph $(G^{ab})^{ab} - a = G - a$, and differs from $G-a$ only in the weights of b, which are given by $\alpha''(b) = \beta'(a) = \alpha(b)(x-1)^2$ and $\beta''(b) = \alpha'(a)(x-1)^2 = \beta(b)(x-1)^2$. Consequently

$$\alpha'(b)q((((G^{ab})')^{ab} - a)') = \alpha'(b)(x-1)^2 q(G-a) = \beta(a)(x-1)^2 q(G-a).$$

As $\beta'(b)q((G^{ab})' - b) = \alpha(a)(x-1)^2 q((G^{ab}-b)')$, the result follows directly from Corollary 2.4. □

If we are confident that division by $x-1$ will cause no trouble, Corollary 2.5 may be restated with a simpler conclusion: using

$$\alpha'(a) = \beta(b)/(x-1), \quad \beta'(a) = \alpha(b)(x-1), \quad \alpha'(b) = \beta(a)/(x-1), \quad \beta'(b) = \alpha(a)(x-1)$$

yields $q(G) = q((G^{ab})')$.

3. Pendant-twin reductions

Other novel properties of the weighted interlace polynomials arise from a general observation. Suppose $a, b \in V(G)$ happen to have the property that for $S \subseteq V(G)$, the rank and nullity of S are determined by the rank and nullity of $S - \{b\}$, perhaps in different ways according to which of a, b is contained in S. Then it may be possible to adjust $\alpha(a)$ and $\beta(a)$ in such a way that the weighted interlace polynomial of the weight-adjusted version of $G-b$ incorporates all the information in $q(G)$. The simplest

https://doi.org/10.1017/S0963548309990381 Published online by Cambridge University Press

instance of this observation occurs when a and b give rise to identical rows and columns in the adjacency matrix of G.

Definition 6. Two vertices a, b of G are *identical twins* if (i) either they are looped and adjacent or they are unlooped and not adjacent, and (ii) they have the same neighbours outside $\{a, b\}$.

Identical twins are called *clones* in [8], and unlooped identical twins are called *duplicates* in [4] and *false twins* in [20]. We prefer the present terminology because the adjective *false* seems inappropriate, and because we do not know what we would call non-identical duplicates or clones in Definition 7 below. The following result extends Proposition 40 of [4], Proposition 4.14 of [20] and Section 3.1 of [8] to vertex-weighted graphs.

Theorem 3.1. *Suppose a and b are identical twins in G. Let G' be the graph obtained from $G - b$ by changing the weights of a:*

$$\beta'(a) = \beta(a)\beta(b) \quad \text{and} \quad \alpha'(a) = \alpha(a)\beta(b) + \alpha(a)\alpha(b)(y-1) + \beta(a)\alpha(b).$$

Then $q(G) = q(G')$.

Proof. If $a \notin S \subseteq V(G')$ then $r(G'[S]) = r(G[S])$, because the adjacency matrices are the same. If $a \in S \subseteq V(G')$ then $r(G'[S]) = r(G[S]) = r(G[S \cup \{b\}]) = r(G[(S - \{a\}) \cup \{b\}])$, because the only difference among the adjacency matrices is that the matrix of $G[S \cup \{b\}]$ has two identical rows and columns, corresponding to a and b. As mentioned above, it is helpful to read α as 'includes' and β as 'excludes', so that (for instance) the appearance of $\beta(a)\alpha(b)$ in $\alpha'(a)$ indicates that if $a \in S \subseteq V(G')$ then the summand of $q(G')$ corresponding to S includes the summand of $q(G)$ corresponding to $(S - \{a\}) \cup \{b\}$. □

Theorem 3.1 has the following inductive generalization. Suppose $k \geqslant 1$ and $a = b_0, b_1, \ldots, b_k$ are identical twins in G. If G' is the graph obtained from $G - b_1 - \cdots - b_k$ by changing the weights of a to

$$\alpha'(a) = \sum_{\emptyset \neq S \subseteq \{b_0,\ldots,b_k\}} \left(\prod_{b_i \in S} \alpha(b_i)\right)\left(\prod_{b_j \notin S} \beta(b_j)\right)(y-1)^{|S|-1}$$

$$\text{and} \quad \beta'(a) = \prod_{i=0}^{k} \beta(b_i),$$

then $q(G) = q(G')$. We refer to the process of combining several identical twins into a single vertex as an *identical twin reduction*, no matter how many identical twins are combined.

Theorem 3.2. *Suppose b is an unlooped degree-one vertex pendant on a. Let G' be the graph obtained from $G - b$ by changing the weights of a:*

$$\alpha'(a) = \alpha(a)\beta(b) \quad \text{and} \quad \beta'(a) = \alpha(a)\alpha(b)(x-1)^2 + \beta(a)\alpha(b)(y-1) + \beta(a)\beta(b).$$

Then $q(G) = q(G')$.

https://doi.org/10.1017/S0963548309990381 Published online by Cambridge University Press

Proof. If $a \in S \subseteq V(G')$ then $r(G'[S]) = r(G[S])$, and if $a \notin S \subseteq V(G')$ then

$$r(G'[S]) = r(G[S \cup \{a, b\}]) - 2 = r(G[S \cup \{b\}]) = r(G[S]). \qquad \square$$

If $b = b_0, b_1, \ldots, b_k$ are unlooped and pendant on a then $b_0, b_1, \ldots, b_k$ are identical twins, so we can combine them into a single re-weighted vertex b using Theorem 3.1 and then remove b using Theorem 3.2. We refer to the removal of any number of unlooped vertices pendant on the same vertex as an *unlooped pendant vertex reduction*. Theorem 3.2 extends Proposition 4.12 of [20] and Section 3.2 of [8], where collections of pendant vertices are called *combs*.

A recursive calculation that depends solely on Theorem 2.2 and Corollary 2.4 is represented by a mixed binary–ternary computation tree; if Corollary 2.4 is always used in place of part (b) of Theorem 2.2 then the tree will be binary. Using identical twin and unlooped pendant vertex reductions makes the computation tree smaller, because each time one of these reductions is used, we avoid splitting the resulting portion of the calculation into branches. Similarly, a Tutte polynomial computation that incorporates series-parallel reductions and deletion–contraction operations will generally result in a smaller formula than a computation that involves only deletion–contraction operations can provide. This latter observation was made precise in [28]: computing the Tutte polynomial of a matroid M using series-parallel reductions and deletion–contraction operations will result in an expression with at least $\beta(M)$ terms, where $\beta(M)$ is Crapo's β invariant [18]; moreover, if $\beta(M) > 0$ the lower bound is attainable. (This result is merely the extension to the Tutte polynomial of the important theory of reliability domination; see [9, 15] for expositions.) For interlace polynomials, analogous lower bounds are derived from the unweighted vertex-nullity polynomial $q_N(G^u)$. The coefficient of y in $q_N(G^u)$ is denoted $\gamma(G)$ as in [20], and the evaluation $q_N(G^u)(0)$ is denoted $\varepsilon(G)$.

Theorem 3.3. *If part* (c) *of Theorem 2.2 is applied only to loopless graphs, then a computation of* $q(G)$ *using Corollary 2.4 and Theorems 2.2, 3.1 and 3.2 is represented by a computation tree with no fewer than* $\frac{1}{2}\varepsilon(G)$ *leaves. If* G *is a simple graph then the computation tree has no fewer than* $\frac{1}{2}\gamma(G)$ *leaves.*

Proof. We quickly review some elementary properties of $q_N(G^u)$ from [3, 4, 5, 20]. This polynomial is described recursively as follows. If G has a loop at a then $q_N(G^u) = q_N((G^u)^a - a) + q_N(G^u - a)$, if a and b are loopless neighbours then $q_N(G^u) = q_N(G^u - a) + q_N((G^u)^{ab} - b)$, and if E_n is the edgeless n-vertex graph then $q_N(E_n^u) = y^n$. The latter includes the empty graph E_0 with $q_N(E_0^u) = 1$. It follows by induction on the number of vertices that no graph has any negative coefficient in $q_N(G^u)$, and hence every graph has $\gamma(G) \geqslant 0$ and $\varepsilon(G) \geqslant 0$. Moreover, every non-empty simple graph has $\varepsilon(G) = 0$ and every disconnected simple graph has $\gamma(G) = 0$.

Theorem 3.3 is certainly true for E_n, as $\frac{1}{2}\gamma(E_n), \frac{1}{2}\varepsilon(E_n) \leqslant 1$.

Proceeding inductively, observe that if Corollary 2.4 or part (a) or (b) of Theorem 2.2 is applied to a graph H represented by a certain node of the computation tree, and

https://doi.org/10.1017/S0963548309990381 Published online by Cambridge University Press

H_1 and H_2 are the graphs represented by the (first two) resulting branch nodes, then $q_N(H^u) = q_N(H_1^u) + q_N(H_2^u)$; certainly then $\frac{1}{2}\varepsilon(H) = \frac{1}{2}\varepsilon(H_1) + \frac{1}{2}\varepsilon(H_2)$. (By the way, we call vertices of the computation tree *nodes* in order to distinguish them from vertices of G.) If Theorem 3.1 or 3.2 is applied to remove a vertex b from a graph H, then $\frac{1}{2}q_N(H^u)(0) = \frac{1}{2}q_N(H^u - b)(0)$, because the formulas of Theorems 3.1 and 3.2 yield $\alpha'(a) = \beta'(a) = 1$ when $\alpha(a) = \beta(a) = \alpha(b) = \beta(b) = 1$, $x = 2$ and $y = 0$.

It remains to consider the special case involving simple graphs. We actually prove a slightly different result, as follows. If G is simple, then the portion of the computation tree involving only nodes corresponding to connected graphs has no fewer than $\frac{1}{2}\gamma(G)$ leaves. If G is disconnected, then $\gamma(G) = 0$ so G satisfies the result trivially. If G is a connected, simple graph with $n \leqslant 2$, then G satisfies the result because $\frac{1}{2}\gamma(G) \leqslant 1$. Proceeding inductively, observe that if Corollary 2.4 or Theorem 2.2(b) is applied to a connected graph H represented by a certain node of the computation tree, and H_1 and H_2 are the graphs represented by the (first two) resulting branch nodes, then $q_N(H^u) = q_N(H_1^u) + q_N(H_2^u)$. It follows that $\frac{1}{2}\gamma(H) = \frac{1}{2}\gamma(H_1) + \frac{1}{2}\gamma(H_2)$. If Theorem 3.1 or Theorem 3.2 is used to remove a vertex b from a connected, simple graph H with 3 or more vertices, then $\gamma(H) = \gamma(H - b)$, by Corollary 4.17 of [20]. □

Theorem 3.3 is of limited value because the computations discussed are not optimal. The restriction that part (c) of Theorem 2.2 is only applied to loopless graphs is an obvious inefficiency. In addition, if some (combinations of) weights are 0, then it would be natural to simply ignore the corresponding parts of the computation. There are also other useful twin reductions that do not fall under Theorem 3.3.

Definition 7. Two vertices a, b of G are *fraternal twins* if (i) either they are looped and non-adjacent or they are unlooped and adjacent, and (ii) they have the same neighbours outside $\{a, b\}$.

Unlooped fraternal twins are called *true twins* in [20], but we prefer the present terminology because the rows and columns of the adjacency matrix corresponding to fraternal twins are not quite the same. Here is an extension of Proposition 4.15 of [20] to weighted graphs.

Theorem 3.4. *Suppose a and b are fraternal twins in G. Let G' be the graph obtained from $G - b$ by changing the weights of a:*

$$\alpha'(a) = \alpha(a)\beta(b) + \beta(a)\alpha(b) \quad \text{and} \quad \beta'(a) = \beta(a)\beta(b) + \alpha(a)\alpha(b)(x-1)^2.$$

Then $q(G) = q(G')$.

Proof. If $a \in S \subseteq V(G')$ then $r(G'[S]) = r(G[S]) = r(G[(S \cup \{b\}) - \{a\}])$, because the adjacency matrices are the same. If $a \notin S \subseteq V(G')$ then

$$r(G'[S]) = r(G[S]) = r(G[S \cup \{a, b\}]) - 2.$$ □

https://doi.org/10.1017/S0963548309990381 Published online by Cambridge University Press

Theorem 3.4 has the following inductive generalization. Suppose $k \geqslant 1$ and $a = b_0, b_1, \ldots, b_k$ are fraternal twins in G. If G' is the graph obtained from $G - b_1 - \cdots - b_k$ by changing the weights of a to

$$\alpha'(a) = \sum_{\substack{S \subseteq \{b_0,\ldots,b_k\} \\ |S| \text{ odd}}} \left(\prod_{b_i \in S} \alpha(b_i)\right)\left(\prod_{b_j \notin S} \beta(b_j)\right)(x-1)^{|S|-1}$$

$$\text{and} \quad \beta'(a) = \sum_{\substack{S \subseteq \{b_0,\ldots,b_k\} \\ |S| \text{ even}}} \left(\prod_{b_i \in S} \alpha(b_i)\right)\left(\prod_{b_j \notin S} \beta(b_j)\right)(x-1)^{|S|},$$

then $q(G) = q(G')$. We refer to the process of combining several fraternal twins into a single vertex as a *fraternal twin reduction*, no matter how many fraternal twins are combined.

In general, fraternal twin reductions are just as useful as identical twin reductions. However, as noted in Proposition 38 of [4] and Corollary 4.16 of [20] they have the effect of multiplying the unweighted vertex-nullity polynomial by powers of 2. (Observe that if $\alpha(a) = \beta(a) = \alpha(b) = \beta(b) = 1$ and $x = 2$, then Theorem 3.4 gives $\alpha'(a) = \beta'(a) = 2$.) Consequently the lower bounds of Theorem 3.3 are not valid for computations that utilize Theorem 3.4 along with Corollary 2.4 and Theorems 2.2, 3.1 and 3.2.

If we are given a reduction of a graph G to disconnected vertices using twin reductions and unlooped pendant vertex reductions, then Theorems 3.1, 3.2 and 3.4 describe $q(G)$ in linear time: simply update the vertex weights at each step, and at the end refer to part (c) of Theorem 2.2. If we are not given such a reduction, then determining whether or not any such reduction exists, and finding one if possible, can be accomplished in polynomial time: as in Corollary 5.3 of [20], simply search $V(G)$ repeatedly for unlooped degree-one vertices and pairs of vertices a, b with the same neighbours outside $\{a, b\}$. Hence, if G has such a reduction then, in polynomial time, Theorems 3.1, 3.2 and 3.4 provide a description of $q(G)$ that completely avoids the branching formulas of Corollary 2.4 and parts (a) and (b) of Theorem 2.2. This observation extends Theorem 6.4 of [20] from q_N to q and from simple graphs that can be analysed without fraternal twin reductions to looped graphs that can be analysed using all three types of reductions.

Theorem 3.5. *If a graph G can be reduced to a collection of disconnected vertices using unlooped pendant vertex reductions and the two types of twin reductions, then Theorems 3.1, 3.2 and 3.4 provide a polynomial-time description of $q(G)$.*

The theorem refers to a *description* rather than a *computation* because we have ignored the cost of arithmetic operations in the ring R. If $R = \mathbb{Q}$, arithmetic operations have low cost, and describing a weighted 'polynomial' $q(G)$ is the same as computing an evaluation of the unweighted polynomial $q(G)$. The full unweighted polynomial may be recovered from several evaluations by interpolation, so the theorem provides a polynomial-time computation of the unweighted polynomial. In more complicated rings such as $\mathbb{Z}[x, y, \alpha_1, \ldots, \alpha_n, \beta_1, \ldots, \beta_n]$, arithmetic operations may be so expensive as to prohibit

https://doi.org/10.1017/S0963548309990381 Published online by Cambridge University Press

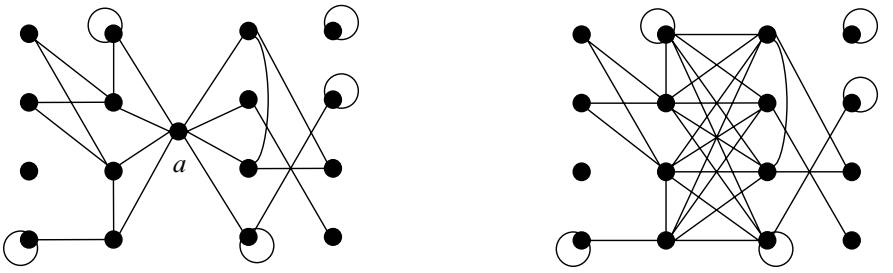


Figure 1. Composition of graphs.

polynomial-time computation of entire weighted polynomials. A thorough discussion of these matters is given by Courcelle [16].

4. Composition

In this section we reformulate and extend some results of Arratia, Bollobás and Sorkin [4] regarding substituted graphs, using the following version of a construction introduced by Cunningham [19].

Definition 8. A vertex-weighted graph G is the *composition* of vertex-weighted graphs H and K, $G = H * K$, if the following conditions hold.

(a) $V(H) \cap V(K)$ consists of a single unlooped vertex a.
(b) The vertex a is unweighted in both H and K, *i.e.*, $\alpha(a) = 1 = \beta(a)$ in H and K.
(c) $V(G) = (V(H) \cup V(K)) - a$, and the vertices of G inherit their weights from H and K.
(d) $E(G) = E(H - a) \cup E(K - a) \cup \{vw | va \in E(H) \text{ and } aw \in E(K)\}$.

Requiring a to be unlooped and unweighted in both H and K guarantees that no information is lost when we remove a in constructing G.

Definition 8 includes several other familiar notions. If a is isolated in H or K, then $H * K$ is the disjoint union of $H - a$ and $K - a$. If a is adjacent to every other vertex of H and K, then $H * K$ is also denoted $(H - a) + (K - a)$. This is traditionally called a 'join', but that term has recently been used for general compositions [17]. If a is adjacent to every other vertex of H, then $H * K$ is the graph obtained by substituting $H - a$ for a in K. If a is adjacent to every other vertex of H and $H - a$ is edgeless or complete, then the (un)looped vertices of $H - a$ are twins in $H * K$.

The following observation will be useful.

Proposition 4.1. *Suppose two graphs* Γ_1 *and* Γ_2 *are identical except for the weights of a single vertex* a*, and let* Γ *be the graph that is identical to both* Γ_1 *and* Γ_2 *except for*

$$\alpha_\Gamma(a) = \alpha_{\Gamma_1}(a) + \alpha_{\Gamma_2}(a) \quad \text{and} \quad \beta_\Gamma(a) = \beta_{\Gamma_1}(a) + \beta_{\Gamma_2}(a).$$

Then $q(\Gamma_1) + q(\Gamma_2) = q(\Gamma)$.

https://doi.org/10.1017/S0963548309990381 Published online by Cambridge University Press

Let a be an unweighted vertex of a simple vertex-weighted graph H, and let H_1 and H_2 be the full subgraphs of $H - a$ induced by the neighbours (resp. non-neighbours) of a. Any weighted interlace polynomial $q(H * K)$ may be analysed in the following way.

Step 1. Eliminate all edges between vertices of $H - a$ using pivots and weight-changes as in Corollary 2.4. (In general there will be many different sequences of pivots that may be used; this lack of uniqueness is not important in the analysis.) The assignments of individual vertices of H to H_1 and H_2 may change during this process, and vertex weights may also change; but these reassignments and weight changes will be the same for all graphs K. As in the proof of Proposition 39 of [4], these pivots will not affect the internal structure of K, because no two vertices of K have distinct, non-empty sets of neighbours in H. Here the phrase 'internal structure' refers to vertex weights, the positions of loops, and adjacencies, including adjacencies between a and other vertices of K.

Step 2. When Step 1 is complete, $q(H * K)$ is expressed as a sum in which each summand is the product of an initial multiplying factor and the weighted interlace polynomial $q(H' * K)$ of a graph in which every edge is incident on a vertex of $K - a$. If $v \in V(H_2')$ then v is isolated, so the only effect of v is to multiply that summand by $q(\{v\}) = \alpha(v)(y-1) + \beta(v)$. This same effect is realized by removing v and incorporating $\alpha(v)(y-1) + \beta(v)$ into the initial multiplying factor, so we may assume that every summand has $H_2' = \emptyset$. In a summand with $|V(H_1')| > 1$, the vertices of H_1' are all identical twins, and may be consolidated into a single vertex a using Theorem 3.1. In a summand with $|V(H_1')| = 0$, a single vertex $a \in V(H_1')$ may be introduced with $\alpha(a) = 0$ and $\beta(a) = 1$; this will not affect the value of the corresponding summand. As in Step 1, these manipulations are the same for all K.

Step 3. The weighted interlace polynomial $q(H * K)$ is now expressed as a sum in which each summand is the product of an initial multiplying factor and a weighted interlace polynomial $q(K')$ in which K' differs from K only in the weights of a. In each summand we multiply the α and β weights of a by the initial multiplying factor. This has the effect of multiplying $q(K')$ by that factor, so the summand is now simply $q(K')$. Proposition 4.1 tells us that the sum is equal to a single weighted interlace polynomial $q(K')$, where $\alpha(a)$ and $\beta(a)$ are obtained by adding together the α and β weights of a in the various summands.

We deduce that there are weights $\alpha(a)$ and $\beta(a)$ that depend only on H and a, and have the following property. In every instance of Definition 8 involving H, the interlace polynomial $q(H * K)$ equals $q(K')$, where K' is obtained from K by using $\alpha(a)$ and $\beta(a)$ as weights for a. Finding explicit formulas for these weights is not difficult.

Theorem 4.2. *Let H be a vertex-weighted simple graph with an unweighted vertex a. Then every composition $H * K$ has $q(H * K) = q(K')$, where K' is obtained from K by using the following weights for a:*

$$\alpha(a) = \frac{q(H) - yq(H-a)}{(x-1)^2 - (y-1)^2},$$

$$\beta(a) = \frac{((x-1)^2 + y - 1)q(H-a) - (y-1)q(H)}{(x-1)^2 - (y-1)^2}.$$

https://doi.org/10.1017/S0963548309990381 Published online by Cambridge University Press

Proof. With $V(K) = \{a\}$, we have

$$q(H - a) = q(H * K) = q(K') = \beta(a) + (y-1)\alpha(a).$$

With K consisting of two adjacent, unlooped, unweighted vertices a and v, we have

$$q(H) = q(H * K) = q(K') = ((x-1)^2 + y - 1)\alpha(a) + y\beta(a).$$

The stated formulas for $\alpha(a)$ and $\beta(a)$ follow. □

If $(x-1)^2 - (y-1)^2$ might be a zero divisor in the ring R, one can avoid any difficulty with the formulas of Theorem 4.2 by first evaluating them in the polynomial ring $\mathbb{Z}[x, y, \alpha_1, \ldots, \alpha_n, \beta_1, \ldots, \beta_n]$, and then evaluating the resulting division-free formulas in R.

Theorem 4.2 concerns compositions $H * K$ in which H is simple. If H has looped vertices, a similar analysis requires two new steps.

Step 0. Begin by removing all loops in $H - a$ using local complementation as in part (a) of Theorem 2.2. The result is a description of $q(H * K)$ as a sum in which each summand is the product of an initial factor and an interlace polynomial $q(H' * K)$ or $q(H' * K^a)$, with no loops in H'.

Step 4. After applying Steps 1–3 to each of these summands, collect terms to obtain a formula

$$q(H * K) = q(K') + q((K^a)').$$

In order to distinguish the two terms on the right-hand side we denote by a_c the copy of a in K^a. It might seem that we now have to determine four unknowns, namely the vertex weights $\alpha(a)$, $\alpha(a_c)$, $\beta(a)$ and $\beta(a_c)$. It turns out, though, that these unknowns are not independent. There is an obvious isomorphism between $(K')^a$ and K^a, and consequently

$$q((K')^a) - \beta(a)q((K')^a - a)$$

may be obtained from

$$q((K^a)') - \beta(a_c)q((K^a)' - a_c)$$

simply by replacing $\alpha(a_c)$ with $\alpha(a)$. Theorem 2.3 tells us that

$$q(K') - \beta(a)q(K' - a) = q((K')^a) - \beta(a)q((K')^a - a),$$

so

$$q(K') - \beta(a)q(K' - a)$$

may be obtained from

$$q((K^a)') - \beta(a_c)q((K^a)' - a_c)$$

by replacing $\alpha(a_c)$ with $\alpha(a)$. That is, the coefficient of $\alpha(a)$ in $q(K')$ is precisely the same as the coefficient of $\alpha(a_c)$ in $q((K^a)')$. It follows that the sum $q(K') + q((K^a)')$ is unchanged if we replace $\alpha(a)$ by $\alpha(a) + \alpha(a_c)$ and replace $\alpha(a_c)$ by 0.

https://doi.org/10.1017/S0963548309990381 Published online by Cambridge University Press

Theorem 4.3. *Let H be a vertex-weighted graph with an unweighted, unlooped vertex a. Then H and a determine weights $\alpha(a)$, $\beta(a)$ and $\beta(a_c)$ such that every composition $H * K$ has $q(H * K) = q(K') + q((K^a)')$, where K' is obtained from K by using $\alpha(a)$ and $\beta(a)$ as weights for a and $(K^a)'$ is obtained from K^a by using $\alpha(a_c) = 0$ and $\beta(a_c)$ as weights for a_c, the copy of a in $(K^a)'$.*

Formulas for the three weights mentioned in Theorem 4.3 may be derived from three instances of the theorem. We use $H - a$ and H as in Theorem 4.2, and also H^ℓ, the graph obtained from H by attaching a loop at a. These correspond, respectively, to compositions of H with graphs K_1, K_2, K_3 such that $V(K_1) = \{a\}$; $V(K_2) = \{a, v\}$ with v an unweighted, unlooped neighbour of a; and $V(K_3) = \{a, \ell\}$ with ℓ an unweighted, looped neighbour of a. Definition 1 gives the following values:

$$\begin{aligned} q(H - a) &= q(K_1') + q((K_1^a)') = (y-1)\alpha(a) + \beta(a) + \beta(a_c), \\ q(H) &= q(K_2') + q((K_2^a)') = ((x-1)^2 + y - 1)\alpha(a) + y\beta(a) + x\beta(a_c), \\ q(H^\ell) &= q(K_3') + q((K_3^a)') = ((x-1)^2 + y - 1)\alpha(a) + x\beta(a) + y\beta(a_c). \end{aligned}$$

We deduce these formulas:

$$\begin{aligned} \alpha(a) &= \frac{(x+y)q(H-a) - q(H) - q(H^\ell)}{(x+y)(y-1) - 2((x-1)^2 + y - 1)}, \\ \beta(a) - \beta(a_c) &= \frac{q(H) - q(H^\ell)}{y - x}, \\ \beta(a) + \beta(a_c) &= \frac{2((x-1)^2 + y - 1)q(H-a) - (y-1)\big(q(H) + q(H^\ell)\big)}{2((x-1)^2 + y - 1) - (y-1)(x+y)}. \end{aligned}$$

Separate formulas for $\beta(a)$ and $\beta(a_c)$ are derived in the obvious ways by adding and subtracting the last two. As before, possible problems with denominators may be avoided by first evaluating the formulas in $\mathbb{Z}[x, y, \alpha_1, \ldots, \alpha_n, \beta_1, \ldots, \beta_n]$.

We close this section with different formulas for the weights $\alpha(a)$, $\beta(a)$ and $\beta(a_c)$ that appear in Theorems 4.2 and 4.3. Suppose H has an unweighted, unlooped vertex a. Let $N(a)$ denote the open neighbourhood of a, *i.e.*, the set containing the vertices $v \neq a \in V(H)$ that neighbour a. Given a subset $S \subseteq V(H - a)$ let $\rho = \rho_{S,a}$ (resp. $\kappa = \kappa_{S,a}$) be the row (resp. column) vector with entries indexed by $\{i : v_i \in S\}$ whose ith entry is 1 or 0 according to whether $v_i \in N(A)$ or $v_i \notin N(a)$. Also let $M = M_S$ be the adjacency matrix of $H[S]$. Note that $r(M) \leqslant r(M \quad \kappa) \leqslant r(M) + 1$, and

$$r(M) \leqslant r\begin{pmatrix} M & \kappa \\ \rho & 0 \end{pmatrix}, \quad r\begin{pmatrix} M & \kappa \\ \rho & 1 \end{pmatrix} \leqslant r(M) + 2,$$

because adjoining a single row or column to a matrix raises the rank by 0 or 1.

https://doi.org/10.1017/S0963548309990381 Published online by Cambridge University Press

Definition 9. The *type* of S with respect to a is defined as follows:

$$S \text{ is of type 1 if } r(M) = r\begin{pmatrix} M & \kappa \\ \rho & 0 \end{pmatrix} = r\begin{pmatrix} M & \kappa \\ \rho & 1 \end{pmatrix} - 1,$$

$$S \text{ is of type 2 if } r(M) + 2 = r\begin{pmatrix} M & \kappa \\ \rho & 0 \end{pmatrix} = r\begin{pmatrix} M & \kappa \\ \rho & 1 \end{pmatrix},$$

$$S \text{ is of type 3 if } r(M) = r\begin{pmatrix} M & \kappa \\ \rho & 1 \end{pmatrix} = r\begin{pmatrix} M & \kappa \\ \rho & 0 \end{pmatrix} - 1.$$

Lemma 4.4. *Every $S \subseteq V(H-a)$ is of one of these types. Moreover, if a has no looped neighbour then there is no $S \subseteq V(H-a)$ of type 3.*

Proof. The fact that every $S \subseteq V(H-a)$ is of type 1, 2 or 3 appears in Lemma 2 of [6].

Suppose S is of type 3, so

$$r(M) + 1 = r\begin{pmatrix} M & \kappa \\ \rho & 0 \end{pmatrix}.$$

The row vector $(\rho \quad 0)$ cannot be a sum of rows of $(M \quad \kappa)$. For if it were, then ρ would be the corresponding sum of rows of M, and by symmetry κ would be the corresponding sum of columns of M. Consequently, it would follow that

$$r(M) = r\begin{pmatrix} M & \kappa \end{pmatrix} = r\begin{pmatrix} M & \kappa \\ \rho & 0 \end{pmatrix}.$$

On the other hand, ρ must be the sum of the rows of M corresponding to the elements of some subset $T \subseteq S$. For if it were not, then it would follow that

$$r(M) + 1 = r\begin{pmatrix} M \\ \rho \end{pmatrix} = r\begin{pmatrix} M & \kappa \end{pmatrix} = r\begin{pmatrix} M & \kappa \\ \rho & 0 \end{pmatrix} - 1.$$

Every such T must contain an odd number of neighbours of a, to avoid giving a sum of rows of $(M \quad \kappa)$ equal to $(\rho \quad 0)$.

Choose such a T, and consider the induced subgraph $H[T \cap N(a)]$. As the sum of the rows of M corresponding to elements of T is ρ, the sum of the rows of the adjacency matrix of $H[T \cap N(a)]$ is the row vector $(1 \cdots 1)$. $|T \cap N(a)|$ is odd, so the adjacency matrix of $H[T \cap N(a)]$ has an odd number of entries equal to 1. The matrix is symmetric, so at least one of these entries must appear on the diagonal. That is, at least one vertex of $H[T \cap N(a)]$ is looped. □

For $S \subseteq V(H-a)$, let $M^a = M^a_S$ be the matrix obtained from M by toggling every entry m_{ij} that has $v_i, v_j \in N(a)$. Then $r(M \quad \kappa) = r(M^a \quad \kappa)$ because the first matrix is transformed into the second by adding the last column to every column corresponding to a neighbour of a. Similarly, adding the last column of the first matrix below to every column corresponding to a neighbour of a tells us that

$$r\begin{pmatrix} M & \kappa \\ \rho & 1 \end{pmatrix} = r\begin{pmatrix} M^a & \kappa \\ 0 & 1 \end{pmatrix} = 1 + r(M^a).$$

https://doi.org/10.1017/S0963548309990381 Published online by Cambridge University Press

Consequently, Definition 9 may be restated using the relationship between $r(M)$ and $r(M^a)$: if S is of type 1 then $r(M) = r(M^a)$, if S is of type 2 then $r(M) = r(M^a) - 1$, and if S is of type 3 then $r(M) = r(M^a) + 1$.

Proposition 4.5. *For $i \in \{1, 2, 3\}$ let*

$$q_i(H - a) = \sum_{\substack{S \subseteq V(H-a) \\ \text{of type } i}} \left(\prod_{s \in S} \alpha(s) \right) \left(\prod_{v \notin S} \beta(v) \right) (x-1)^{r((H-a)[S])} (y-1)^{n((H-a)[S])}.$$

Then the following equations hold:

$$q(H - a) = q_1(H - a) + q_2(H - a) + q_3(H - a),$$

$$q(H) = y q_1(H - a) + \left(1 + \frac{(x-1)^2}{y-1}\right) q_2(H - a) + x q_3(H - a),$$

$$q(H^{\ell}) = x q_1(H - a) + \left(1 + \frac{(x-1)^2}{y-1}\right) q_2(H - a) + y q_3(H - a).$$

Also,

$$q(H^a - a) = q_1(H - a) + \left(\frac{x-1}{y-1}\right) q_2(H - a) + \left(\frac{y-1}{x-1}\right) q_3(H - a).$$

Proof. The first equality is obvious. For the second, note that each $S \subseteq V(H - a)$ gives rise to two subsets of $V(H)$, namely S and $S \cup \{a\}$; these correspond to the adjacency matrices

$$M \quad \text{and} \quad \begin{pmatrix} M & \kappa \\ \rho & 0 \end{pmatrix}.$$

Similarly, the third equality is derived by considering the contributions of two adjacency matrices for each $S \subseteq V(H - a)$, namely

$$M \quad \text{and} \quad \begin{pmatrix} M & \kappa \\ \rho & 1 \end{pmatrix}.$$

The last equality follows from the discussion preceding the proposition. □

Corollary 4.6. *The weights mentioned in Theorems 4.2 and 4.3 are*

$$\beta(a) = q_1(H - a), \quad \alpha(a) = q_2(H - a)/(y - 1) \quad \text{and} \quad \beta(a_c) = q_3(H - a).$$

In particular, $\beta(a_c) = 0$ if a has no looped neighbour.

Proof. The corollary follows from Proposition 4.5 and the formulas given in Theorem 4.2 and immediately after Theorem 4.3. □

5. A characterization of simple graphs

In this section we focus our attention on the unweighted vertex-nullity polynomial.

https://doi.org/10.1017/S0963548309990381 Published online by Cambridge University Press

Proposition 5.1. *If G is a connected, unweighted graph with at least one looped vertex, then $\varepsilon(G) = q_N(G)(0) > 1$.*

Proof. The proposition is certainly true if G has $n \leqslant 2$ vertices, as all three such graphs have $\varepsilon(G) = 2$. The argument proceeds by induction on $n \geqslant 3$. Recall that $\varepsilon(G) \geqslant 0$ for every graph G, and let a be a looped vertex of G. Then $q_N(G) = q_N(G-a) + q_N(G^a - a)$, so $\varepsilon(G) \geqslant \max\{\varepsilon(G-a), \varepsilon(G^a-a)\}$.

Suppose a is not a cutpoint of G. If G has some looped vertex other than a then the inductive hypothesis implies that $\varepsilon(G-a) > 1$. If G has no looped vertex other than a then every neighbour of a is looped in $G^a - a$. Every component of $G^a - a$ contains at least one neighbour of a, so the inductive hypothesis implies that $\varepsilon(C) > 1$ for every component C of $G^a - a$. Hence $\varepsilon(G^a - a) = \prod_C \varepsilon(C) > 1$.

Suppose now that G has a looped cutpoint a. For each component C of $G-a$ and each vertex $v \in N(a)$ that lies in some other component of $G-a$, no edge connecting v to an element of $N(a) \cap V(C)$ appears in $G-a$. Consequently every such edge appears in $G^a - a$, so all of $N(a)$ is contained in a single component of $G^a - a$. As G is connected, this implies that $G^a - a$ is also connected. Hence, if any neighbour of a is unlooped in G, $G^a - a$ is a connected graph with a looped vertex and the inductive hypothesis implies that $\varepsilon(G^a - a) > 1$. If instead every neighbour of a is looped in G, then every component C of $G-a$ has a looped vertex and the inductive hypothesis implies $\varepsilon(G-a) = \prod_C \varepsilon(C) > 1$. □

Corollary 5.2. *An unweighted graph G has $\varepsilon(G) > 0$ if and only if G has no non-empty simple component.*

Proof. As noted in Remark 20 of [4], every non-empty simple graph has $\varepsilon(G) = 0$. It follows that every graph with a non-empty simple component also has $\varepsilon(G) = 0$. On the other hand, Proposition 5.1 and the fact that $\varepsilon(E_0) = 1$ together imply that every graph with no non-empty simple component has $\varepsilon(G) > 0$. □

Corollary 5.3. *Let G be an unweighted graph, and let G^c be its* complement*, i.e., the graph with $V(G^c) = V(G)$ whose edges (including loops) are precisely the edges absent in G. Then the following are equivalent: G is simple, $q_N(G + E_1^u) = yq_N(G^c)$ and $y | q_N(G + E_1^u)$.*

Proof. Suppose first that G is simple, and let $H = G + E_1^u$, with a the vertex of E_1^u. Let K be the two-vertex graph with $V(K) = \{a, \ell\}$ in which ℓ is an unweighted, looped neighbour of a. Theorem 4.2 tells us that

$$q(H * K) = q(K') = x\beta(a) + ((x-1)^2 + y - 1)\alpha(a).$$

On the other hand, the recursive description of q [5] tells us that

$$q(H * K) = q((H * K) - \ell) + (x-1)q((H * K)^{\ell} - \ell) = q(G) + (x-1)q(G^c).$$

Recalling that $q(G) = q(H - a) = \beta(a) + (y-1)\alpha(a)$, we see that

$$0 = q(H * K) - q(H * K) = (x-1)\beta(a) + (x-1)^2\alpha(a) - (x-1)q(G^c),$$

https://doi.org/10.1017/S0963548309990381 Published online by Cambridge University Press

and consequently $q(G^c) = \beta(a) + (x-1)\alpha(a)$. Recalling that

$$q(G + E_1^u) = q(H) = ((x-1)^2 + y - 1)\alpha(a) + y\beta(a),$$

we see that

$$q_N\big(G + E_1^u\big) = q\big(G + E_1^u\big)|_{x=2} = y\big(\alpha(a) + \beta(a)\big)|_{x=2} = yq(G^c)|_{x=2} = yq_N(G^c).$$

Now suppose that G is not simple. Then G has a looped vertex, and so does $G + E_1^u$. As $G + E_1^u$ is connected, Corollary 5.2 states that $\varepsilon(G + E_1^u) > 0$; consequently $y \nmid q_N(G + E_1^u)$. □

6. Trees

A combinatorial description of the interlace polynomials of trees and forests is given in [2]. In order to motivate the next section we sketch this description briefly here, omitting details and proofs. Recall that a tree T is *rooted* by specifying a root vertex $r \in V(T)$. Each non-root vertex $v \in V(T)$ then has a unique *parent* $p(v)$, a neighbour whose distance from r is less than the distance from r to v. The elements of $p^{-1}(\{p(v)\})$ are the *children* of $p(v)$, and the children of $p(v)$ other than v itself are *siblings* of v. An *ordered* tree is a rooted tree given with an order on the set of children of each parent vertex; non-root vertices may then have *earlier siblings* and *later siblings*. A set of vertices that contains no adjacent pair is *independent*, and a set of vertices *dominates* a vertex v if it contains v or contains some neighbour of v.

Definition 10. An *earlier sibling cover* (or *es-cover*) in an ordered tree T is an independent set I that dominates r and has the property that for every non-root vertex $v \in I$, every earlier sibling of v is dominated by I.

Definition 11. *For integers s and t, the* es-number $c_{s,t}(T)$ *is the number of s-element es-covers in T whose non-root elements have t different parents.*

If T' is a subtree of T and $r \in V(T')$ then we presume that the children of each parent vertex v in T' are ordered by restricting the order of the children of v in T. With this convention, it is easy to verify that the earlier sibling covers in large trees arise from earlier sibling covers in subtrees.

Lemma 6.1. *Let T be an ordered tree with a leaf ℓ such that $p(\ell) \neq r \neq \ell$, all the siblings of ℓ are leaves, and ℓ has no later siblings. Then*

$$\{\text{es-covers } I \text{ in } T \text{ with } \ell \notin I\} = \{\text{es-covers in } T - \ell\}$$

and

$$\{\text{es-covers } I \text{ in } T \text{ with } \ell \in I\} = \{\text{unions } p^{-1}(\{p(\ell)\}) \cup I \text{ with } I \text{ an es-cover in } T - p(\ell) - p^{-1}(\{p(\ell)\})\}.$$

https://doi.org/10.1017/S0963548309990381 Published online by Cambridge University Press

Lemma 6.1 is the key to an inductive proof that the terms in the definition of $q(T)$ can be collected into sub-totals corresponding to earlier sibling covers.

Definition 12. Let T be an ordered tree with vertex weights, and let I be an es-cover in T. Let

$$I_r = \{v \in I : \text{either } v = r \text{ or } I \text{ contains a later sibling of } v\},$$
$$I_l = \{v \in I : v \neq r \text{ and } I \text{ contains no later sibling of } v\}, \text{ and}$$
$$I'_c = \{v \notin I : I \text{ contains a child of } v\}.$$

For each vertex v define the *I-weight* $w_I(v)$ as follows:

$$w_I(v) = \begin{cases} \beta(v) + \alpha(v)(y-1) & \text{if } v \in I_r \\ \alpha(v) \cdot \big((y-1)\beta(p(v)) + (x-1)^2\alpha(p(v))\big) & \text{if } v \in I_l \\ 1 & \text{if } v \in I'_c \\ \beta(v) & \text{if } v \notin I \text{ and } v \notin I'_c. \end{cases}$$

The product

$$\prod_{v \in V(T)} w_I(v)$$

is the *total weight* of I in T, denoted $w_T(I)$.

Theorem 6.2. *If T is an ordered tree with vertex weights then*

$$q(T) = \sum_{I \text{ an es-cover}} w_T(I).$$

Proof. If T has no more than one vertex of degree $\geqslant 2$, it is not difficult to verify the theorem directly. If T has more than one vertex of degree $\geqslant 2$ then the theorem follows inductively from Theorem 2.2 and Corollary 2.4 using Lemma 6.1, as detailed in [2]. □

Setting $\alpha \equiv 1$ and $\beta \equiv 1$, we deduce that the unweighted interlace polynomial of a tree is determined by a simple formula involving earlier sibling covers.

Corollary 6.3. *If T is a tree, then the unweighted interlace polynomial of T is*

$$\sum_{s,t} c_{s,t}(T) \cdot y^{s-t}(y - 1 + (x-1)^2)^t.$$

As the interlace polynomials are multiplicative on disjoint unions, these results extend directly to disconnected forests.

7. Algorithmic activities

The Tutte polynomial is a very useful invariant of graphs and matroids, which incorporates a great deal of information and can be defined in several different ways; see [10], [29] and

https://doi.org/10.1017/S0963548309990381 Published online by Cambridge University Press

[30] for more detailed discussions than we provide here. One of its definitions is given in Definitions 13 and 14.

Definition 13. Suppose G is an unweighted graph with $E(G) = \{e_1, \ldots, e_m\}$, and T is a maximal spanning forest of G. An element $e_i \notin E(T)$ is *externally active* with respect to T if i is the least index of an element of the unique circuit contained in $E(T) \cup \{e_i\}$. An element $e_i \in E(T)$ is *internally active* with respect to T if i is the least index of an element of the unique cutset contained in $(E(G) - E(T)) \cup \{e_i\}$. The numbers of edges that are externally and internally active with respect to T are denoted $e(T)$ and $i(T)$, respectively.

Definition 14. If G is an unweighted graph with $E(G) = \{e_1, \ldots, e_m\}$, then the Tutte polynomial of G is

$$t(G) = \sum_T x^{i(T)} y^{e(T)}.$$

Theorem 6.2 and Corollary 6.3 bear a strong resemblance to this formula, with earlier sibling covers in rooted trees replacing maximal spanning forests in arbitrary graphs. We do not know whether there is a combinatorial analogue of Definition 14 that describes the interlace polynomials of an arbitrary graph, but there is an algorithmic analogue. Before presenting it, we recall another definition of the Tutte polynomial.

Definition 15. If G is an unweighted graph, then the Tutte polynomial $t(G)$ is an element of the polynomial ring $\mathbb{Z}[x, y]$ determined recursively by these properties.

(a) If $e \in E(G)$ is neither a loop nor an isthmus, then $t(G) = t(G - e) + t(G/e)$.
(b) If $\lambda \in E(G)$ is a loop, then $t(G) = yt(G - \lambda)$.
(c) If $\beta \in E(G)$ is an isthmus of G, then $t(G) = xt(G/\beta)$.
(d) For any positive integer n, $t(E_n) = 1$.

To recursively calculate $t(G)$ we simply choose an arbitrary edge of G, and apply the appropriate part of Definition 15; this process is repeated as many times as necessary. Such a computation is represented by a computation tree in which a node that represents an instance of part (a) has two children and a node that represents an instance of part (b) or (c) has only one.

Proposition 7.1. *Let G be a graph with $E(G) = \{e_1, \ldots, e_m\}$, and consider the recursive implementation of Definition 15 in which e_m is removed first, then e_{m-1} is removed in all branches, then e_{m-2} is removed in all branches, and so on. The leaves of the computation tree representing this implementation correspond to the maximal spanning forests of G, with the leaf corresponding to T resulting from the portion of the computation in which edges of T are contracted and elements of $E(G) - E(T)$ are deleted. A node of this portion of the computation tree represents the removal of an active edge if and only if it has precisely one child.*

https://doi.org/10.1017/S0963548309990381 Published online by Cambridge University Press

Proof. The proposition is implicit in the fact that Definition 15 and Definition 14 both yield $t(G)$, so it appears implicitly in just about every presentation of the Tutte polynomial. See Theorem IX.65 of [29] or Theorem X.10 of [10], for instance. Explicit discussions of the connection between activities and computation are less common in the literature, though there are some [7, 21, 22].

The proof is a direct induction on $|E(G)|$. If e_m is a loop, then the maximal spanning forests of G and $G - e_m$ coincide, and the computation tree for $t(G)$ is obtained from the computation tree for $t(G - e_m)$ by attaching a new root node of degree 1, representing the removal of e_m. If e_m is an isthmus, then the maximal spanning forests of G and G/e_m correspond, and the computation tree for $t(G)$ is obtained from the computation tree for $t(G/e_m)$ by attaching a new root node of degree 1. Otherwise, the maximal spanning forests of G that contain e_m correspond to the maximal spanning forests of G/e_m, and the maximal spanning forests of G that do not contain e_m are the maximal spanning forests of $G - e_m$. The computation tree for $t(G)$ consists of the root and two disjoint subtrees that are the computation trees for $t(G/e_m)$ and $t(G - e_m)$. □

Definition 13 defines active edges using the structure of G, and this leads to Definition 14's description of $t(G)$ as a generating function for maximal spanning forests. Proposition 7.1 shows that we may also see activity from an algorithmic viewpoint: the external activity of a particular $e \notin E(T)$ is revealed in the fact that one step of a calculation of $t(G)$ involves removing e using part (b) of Definition 15 rather than part (a). This distinction affects the result of the computation, so it would be important even if activity could not be conveniently described using the structure of G, or did not contribute to a convenient closed form for $t(G)$.

Here is an analogue of Definition 15 for the weighted interlace polynomial.

Definition 16. If G is a weighted graph, then $q(G)$ is determined recursively by the following properties.

(a) If a is a looped vertex, then

$$q(G) = \beta(a)q(G - a) + \alpha(a)(x - 1)q(G^a - a).$$

(b) If a and b are loopless neighbours in G, then

$$q(G) = \beta(a)q(G - a) + \alpha(a)q((G^{ab} - b)'),$$

where $(G^{ab} - b)'$ is obtained from $G^{ab} - b$ by changing the weights of a to $\alpha'(a) = \beta(b)$ and $\beta'(a) = \alpha(b)(x - 1)^2$.

(c) If a is isolated and looped, then $q(G) = (\alpha(a)(x - 1) + \beta(a))q(G - a)$.

(d) If a is isolated and unlooped, then $q(G) = (\alpha(a)(y - 1) + \beta(a))q(G - a)$.

(e) The empty graph $\emptyset$ has $q(\emptyset) = 1$.

The preceding discussion of activities and the Tutte polynomial suggests the following.

https://doi.org/10.1017/S0963548309990381 Published online by Cambridge University Press

Definition 17. A node of a computation tree representing a recursive implementation of Definition 16 is *active* if it has precisely one child, *i.e.*, if it represents an application of part (c) or part (d).

An 'activities formula' for $q(G)$ arises directly from a computation tree representing an implementation of Definition 16. The formula has one summand for each leaf of the computation tree (each call to part (e) of Definition 16), representing the product of the coefficients contributed by the nodes in the portion of the computation tree that gives rise to that leaf. Active and non-active nodes contribute different coefficients.

If G is a rooted tree, the formulas of Theorem 6.2 and Corollary 6.3 are activities formulas.

Proposition 7.2. *Let T be a rooted tree with root r, and consider a recursive implementation of Definition 16 structured as follows. If possible, apply part* (d) *of Definition 16; if not and there is a parent vertex other than r, then apply part* (b) *with a leaf of the type denoted ℓ in Lemma 6.1 as a; otherwise apply part* (b) *with the last child of r as a. The leaves of the computation tree representing this implementation correspond to the earlier sibling covers of T, with the es-cover I corresponding to a given leaf constructed from the portion of the computation that gives rise to that leaf as follows: an occurrence of part* (d) *of Definition 16 contributes its a to I, and an occurrence of the $q((G^{ab} - b)')$ branch of part* (b) *contributes its a to I.*

Proof. If T has no vertex other than r, then $\{r\}$ is the only es-cover in T, and the computation consists simply of a single call to part (e) of Definition 16. If T contains no parent vertex other than r and a is the last child of r, then the es-covers in T include $V(T) - \{r\}$ and the es-covers of $T - a$. The first step of the computation is an application of part (b) of Definition 16 with $b = r$. The inductive hypothesis applies to $T - a$, and the branch of the computation corresponding to $(T^{ab} - b)' = (T - r)'$ consists solely of calls to part (d), because every vertex of $T - r$ is isolated; consequently the latter part of the computation tree contains only one leaf, corresponding to $V(T) - \{r\}$. If T contains a parent vertex other than r, then the first step of the computation is an application of part (b) of Definition 16 with $a = \ell$ as in Lemma 6.1, and $b = p(\ell)$. The computation tree contains a single node representing this first step and also two subtrees, one corresponding to $T - a = T - \ell$ and the other corresponding to $(T^{ab} - b)' = (T - p(\ell))'$. The proposition follows inductively from Lemma 6.1. □

We do not know whether or not it is possible to reformulate Definition 17 so that it always refers to G instead of a computation tree. Such a reformulation would certainly be valuable, as the resulting activities formulas would shed light on the combinatorial significance of the interlace polynomials.

Definition 17 extends directly to computation trees representing implementations of other recursions. For instance, if Definition 16 is augmented by incorporating pendant-twin reductions then the resulting computation trees will have active nodes representing these reductions, in addition to active nodes representing parts (c) and (d) of Definition 16.

https://doi.org/10.1017/S0963548309990381 Published online by Cambridge University Press

Acknowledgements

We are grateful to M. Bläser, B. Courcelle, J. A. Ellis-Monaghan, G. Gordon, C. Hoffmann and an anonymous referee for advice and encouragement. We also appreciate the support of Lafayette College.

References

[1] Aigner, M. and van der Holst, H. (2004) Interlace polynomials. *Linear Alg. Appl.* **377** 11–30.
[2] Anderson, C., Cutler, J. D., Radcliffe, A. J., and Traldi, L. On the interlace polynomials of forests. *Discrete Math.*, to appear.
[3] Arratia, R., Bollobás, B., and Sorkin, G. B. (2000) The interlace polynomial: A new graph polynomial. In *Proc. 11th Annual ACM–SIAM Symposium on Discrete Algorithms* (San Francisco 2000), ACM, New York, pp. 237–245.
[4] Arratia, R., Bollobás, B., and Sorkin, G. B. (2004) The interlace polynomial of a graph. *J. Combin. Theory Ser. B* **92** 199–233.
[5] Arratia, R., Bollobás, B., and Sorkin, G. B. (2004) A two-variable interlace polynomial. *Combinatorica* **24** 567–584.
[6] Balister, P. N., Bollobás, B., Cutler, J. and Pebody, L. (2002) The interlace polynomial of graphs at −1. *Europ. J. Combin.* **23** 761–767.
[7] Bari, R. A. (1979) Chromatic polynomials and the internal and external activities of Tutte. In *Graph Theory and Related Topics* (Waterloo 1977), Academic Press, New York, pp. 41–52.
[8] Bläser, M. and Hoffmann, C. (2008) On the complexity of the interlace polynomial. In *STACS 2008: 25th International Symposium on Theoretical Aspects of Computer Science* (Bordeaux 2008), pp. 97–108. Available at `http://www.stacs-conf.org/`.
[9] Boesch, F. T., Satyanarayana, A. and Suffel, C. L. (1988) Some recent advances in reliability analysis using graph theory: A tutorial. *Congr. Numer.* **64** 253–276.
[10] Bollobás, B. (1998) *Modern Graph Theory*, Springer, New York.
[11] Bouchet, A. (1987) Digraph decompositions and Eulerian systems. *SIAM J. Alg. Disc. Meth.* **8** 323–337.
[12] Bouchet, A. (1987) Reducing prime graphs and recognizing circle graphs. *Combinatorica* **7** 243–254.
[13] Bouchet, A. (1994) Circle graph obstructions. *J. Combin. Theory Ser. B* **60** 107–144.
[14] Bouchet, A. (2001) Multimatroids III: Tightness and fundamental graphs. *Europ. J. Combin.* **22** 657–677.
[15] Colbourn, C. J. (1987) *The Combinatorics of Network Reliability*, Oxford University Press, Oxford.
[16] Courcelle, B. (2008) A multivariate interlace polynomial and its computation for graphs of bounded clique-width. *Electron. J. Combin.* **15** #R69.
[17] Courcelle, B. (2008) Circle graphs and monadic second-order logic. *J. Appl. Logic* **6** 416–442.
[18] Crapo, H. (1967) A higher invariant for matroids. *J. Combin. Theory* **2** 406–417.
[19] Cunningham, W. H. (1982) Decomposition of directed graphs. *SIAM J. Alg. Disc. Meth.* **3** 214–228.
[20] Ellis-Monaghan, J. A. and Sarmiento, I. (2007) Distance hereditary graphs and the interlace polynomial. *Combin. Probab. Comput.* **16** 947–973.
[21] Gordon, G. and McMahon, E. (1997) Interval partitions and activities for the greedoid Tutte polynomial. *Adv. Appl. Math.* **18** 33–49.
[22] Gordon, G. and Traldi, L. (1990) Generalized activities and the Tutte polynomial. *Discrete Math.* **85** 167–176.
[23] Kauffman, L. H. (1989) A Tutte polynomial for signed graphs. *Discrete Appl. Math.* **25** 105–127.

https://doi.org/10.1017/S0963548309990381 Published online by Cambridge University Press

[24] Kotzig, A. (1968) Eulerian lines in finite 4-valent graphs and their transformations. In *Theory of Graphs* (Tihany 1966), Academic Press, New York, pp. 219–230.
[25] Murasugi, K. (1989) On invariants of graphs with applications to knot theory. *Trans. Amer. Math. Soc.* **314** 1–49.
[26] Oxley, J. G. and Welsh, D. J. A. (1992) Tutte polynomials computable in polynomial time. *Discrete Math.* **109** 185–192.
[27] Traldi, L. (1989) A dichromatic polynomial for weighted graphs and link polynomials. *Proc. Amer. Math. Soc.* **106** 279–286.
[28] Traldi, L. (2000) Series and parallel reductions for the Tutte polynomial. *Discrete Math.* **220** 291–297.
[29] Tutte, W. T. (1984) *Graph Theory*, Cambridge University Press, Cambridge.
[30] White, N., ed. (1992) *Matroid Applications*, Cambridge University Press, Cambridge.

https://doi.org/10.1017/S0963548309990381 Published online by Cambridge University Press