Evaluating a Weighted Graph Polynomial for Graphs of Bounded Tree-Width

S. D. Noble

Department of Mathematical Sciences
Brunel University
Kingston Lane, Uxbridge
UB8 3PH, U.K.

steven.noble@brunel.ac.uk

Submitted: Oct 30, 2006; Accepted: May 18, 2009; Published: May 29, 2009
Mathematics Subject Classification: 05C85, 05C15, 68R10

Abstract

We show that for any k there is a polynomial time algorithm to evaluate the weighted graph polynomial U of any graph with tree-width at most k at any point. For a graph with n vertices, the algorithm requires $O(a_k n^{2k+3})$ arithmetical operations, where a_k depends only on k.

1 Introduction

Motivated by a series of papers [9, 10, 11], the weighted graph polynomial U was introduced in [22]. Chmutov, Duzhin and Lando [9, 10, 11] introduce a graph polynomial derived from Vassiliev invariants of knots and note that this polynomial does not include the Tutte polynomial as a special case. With a slight generalisation of their definition we obtain the weighted graph polynomial U that does include the Tutte polynomial.

The attraction of U is that it contains many other graph invariants as specialisations, for instance the 2-polymatroid rank generating function of Oxley and Whittle [23], and as a consequence the matching polynomial, the stable set polynomial [13] and the symmetric function generalisation of the chromatic polynomial [27]. Note however that there are non-isomorphic graphs with the same U polynomial. This is a corollary of a result of Sarmiento [26], showing that the coefficients of U and the polychromate determine one another. It remains an open problem to determine whether or not there are two non-isomorphic trees with the same U polynomial. We introduce U in Section 2 and review some of these results in more detail.

The notion of tree-width was introduced by Robertson and Seymour as a key tool in their work on the graph minors project [24, 25]. An equivalent notion, studied extensively by Arnborg and Proskurowski, (see for instance [3, 4]), is that of a partial k-tree.

Many well-studied classes of graphs have bounded tree-width: for instance, series-parallel networks are the graphs with tree-width at most two. A large class of graph problems, which are thought to be intractable, can be solved when the input is restricted to graphs with tree-width at most a fixed constant k. For example, the NP-complete problems, 3-Colouring and Hamiltonian Circuit can be solved in linear time for graphs of bounded tree-width [4]. For a good survey of tree-width see [5].

When the underlying graph is obvious, we let n be its number of vertices, m be its number of edges and p be the largest size of a set of mutually parallel edges.

Theorem 1.1. *For any $k \in \mathbb{N}$, there exists an algorithm $\mathcal{A}_k$ with the following properties. The input is any graph G, with tree-width at most k, and rationals $x_1 = p_1/q_1, \ldots, x_n = p_n/q_n$ and $y = p_0/q_0$ such that for all i, p_i and q_i are coprime. The output is $U_G(x_1, \ldots, x_n, y)$; the running time is*

$$O(a_k n^{2k+3}(n^2 + m)r \log p \log(r(n + m)) \log(\log(r(n + m)))),$$

where $r = \log(\max\{|p_0|, \ldots, |p_n|, |q_0|, \ldots, |q_n|\})$ and a_k depends only on k.

The result extends that of [20] and independently [2] where an algorithm to evaluate the Tutte polynomial of a graph having tree-width at most k is presented. In [20], the algorithm given requires only a linear (in n) number of multiplications. Despite using the same basic idea as in [20], we are unable to reduce the amount of computational effort required to evaluate U down to $O(n^\alpha)$ operations, where α is independent of k.

More recently Hliněný [15] has shown that the Tutte polynomial is computable in polynomial time when the input is restricted to matroids with bounded branchwidth representable over a finite field. Furthermore Makowsky [17] and Makowsky and Mariño [19] have shown that there are polynomial time algorithms to evaluate a wide range of graph polynomials that are definable in monadic second order logic when the input graph has bounded tree-width. Examples include the Tutte polynomial for coloured graphs due to Bollobás and Riordan [7] and certain instances of the very general graph polynomials introduced by Farrell [14]. However it has been shown that U is not even definable in second order logic [18], so none of these results applies. For a recent survey covering the complexity of evaluating many of these polynomials, see [21].

2 A weighted graph polynomial

We begin with a few definitions and then define U, the weighted graph polynomial. We then state some of the key results about U for which the proofs may be found in [22]. Most of our definitions are standard. Our graphs are allowed to have loops and multiple edges. By a *simple* graph we mean one with no loops or multiple edges. If G is a graph and $A \subseteq E(G)$ then $G|A$ is the graph with vertex set $V(G)$ and edge set A. However, in general, our subgraphs do not have to be spanning, that is if H is a subgraph of G then we do not require that $V(H) = V(G)$. The number of connected components of a graph G is denoted by $k(G)$. The *rank* of a set $A \subseteq E$ is denoted by $r(A)$ and defined by $r(A) = |V(G)| - k(G|A)$.

The original definition of U involved a recurrence relation using deletion and contraction, but for the purposes of this paper it is more useful to define U using the "states model expansion" from Proposition 5.1 in [22].

$$U_G(\mathbf{x}, y) = \sum_{A \subseteq E} x_{n_1} x_{n_2} \cdots x_{n_{k(G|A)}} (y-1)^{|A|-r(A)}, \tag{2.1}$$

where $n_1, \ldots, n_{k(G|A)}$ are the numbers of vertices in the connected components of $G|A$. For example, if G is a triangle then

$$U_G(\mathbf{x}, y) = x_1^3 + 3x_1x_2 + 2x_3 + yx_3.$$

We now state some of the results from [22] concerning specialisations of U. The Tutte polynomial $T_G(x, y)$ is an extremely well-studied two-variable graph polynomial which is defined as follows:

$$T_G(x, y) = \sum_{A \subseteq E} (x-1)^{r(E)-r(A)} (y-1)^{|A|-r(A)}.$$

Evaluations of T include the number of spanning trees, number of spanning forests, the chromatic polynomial and the reliability polynomial as well as applications in statistical mechanics and knot theory. See for instance [8, 29].

Proposition 2.1. *For any graph G,*

$$T_G(x, y) = (x-1)^{-k(G)} U_G(x_i = x-1, y).$$

Note that we have abused notation somewhat by writing $U_G(x_i = x-1, y)$ where we mean for all i setting $x_i = x - 1$. It is well-known that if the class of input graphs is not restricted, then apart from along one specific curve and at a small number of other specific points, it is $\#P$-hard to evaluate the Tutte polynomial [16]. Except for the addition of one extra exceptional curve this result may be extended to bipartite planar graphs [28]. These results combined with Proposition 2.1 show that if we do not restrict the class of input graphs then the problem of evaluating U at a point specified in the input is $\#P$-hard.

The 2-polymatroid rank generating function $S_G(u, v)$ was introduced by Oxley and Whittle in [23] and is defined as follows. Given a graph $G = (V, E)$ and $A \subseteq E$ let $f(A)$ denote the number of vertices of G that are an endpoint of an edge in A. Then

$$S_G(u, v) = \sum_{A \subseteq E} u^{|V(G)|-f(A)} v^{2|A|-f(A)}.$$

S contains the matching polynomial as a specialisation.

Proposition 2.2. *Let G be a loopless graph with no isolated vertices. Then*

$$S_G(u, v) = U_G(x_1 = u,\ x_2 = 1,\ x_j = v^{j-2} \text{ for } j > 2,\ y = v^2 + 1).$$

A *stable* set in a graph $G = (V, E)$ is a set S of vertices for which G has no edge with both endpoints in S. The stability polynomial $A_G(p)$ was introduced by Farr in [13] and is given by

$$A_G(p) = \sum_{U \in \mathcal{S}(G)} p^{|U|}(1-p)^{|V \setminus U|},$$

where $\mathcal{S}(G)$ is the set of all stable sets of G.

Proposition 2.3. *If G is loopless then $A(G;p)$ is given by*

$$A(G;p) = U_G(x_1 = 1,\, x_j = -(-p)^j \text{ for } j \geq 2,\, y = 0).$$

The symmetric function generalisation of the chromatic polynomial was developed by Stanley in [27]. Let G be a graph with vertex set $V = \{v_1, \dots, v_n\}$. Then X_G is a homogeneous symmetric function of degree n defined by

$$X_G(x_1, x_2, \dots) = \sum_{\chi} x_{\chi(v_1)} x_{\chi(v_2)} \cdots x_{\chi(v_n)}$$

where the sum ranges over all proper colourings $\chi : V \to \mathbb{Z}^+$. Let $p_0 = 1$ and for $r \geq 1$ let

$$p_r(x_1, x_2, \dots) = \sum_{i=1}^{\infty} x_i^r.$$

Then we have the following.

Proposition 2.4. *For any graph G,*

$$X_G(x_1, x_2, \dots) = (-1)^{|V|} U_G(x_j = -p_j, y = 0).$$

3 Preliminary results

We begin this section with a few definitions that are needed in the algorithm. Although the ideas behind the algorithm are quite simple, they do involve introducing a lot of notation. A *weighted partition* of a set A consists of a partition π of A into non-empty blocks, together with the assignment to each block of a non-negative integer label. If B is a block in a weighted partition π, we write $B \in \pi$ and we denote the label of B by $w_\pi(B)$. The number of blocks in π is denoted by $\#\pi$.

Given two weighted partitions π_1 and π_2, of the same set, we define their *join* $\pi = \pi_1 \vee \pi_2$ as follows. The blocks are minimal sets such that if two elements are in the same block of either π_1 or π_2 then they are in the same block of π. In other words, before considering weights, the join operation corresponds to join in the partition lattice. If B is a block of π then for $i = 1, 2$, B is the disjoint union of a collection of blocks from π_i. We define $w_\pi(B)$ by

$$w_\pi(B) = \sum_{\substack{B' \in \pi_1: \\ B' \subseteq B}} w_{\pi_1}(B') + \sum_{\substack{B' \in \pi_2: \\ B' \subseteq B}} w_{\pi_2}(B').$$

Let $G = (V, E)$ be a graph and $A \subseteq E$. Let $\pi(A)$ be the partition of V given by the connected components of $G|A$. We use $\pi(A)$ to make two definitions. The first definition is the *weighted partition induced by A on S* and we denote it by $\pi_G(S, A)$, often omitting G when it is obvious from the context. The weighted partition $\pi_G(S, A)$ is formed from $\pi(A)$ by labelling each block B with $|B - S|$ and deleting all the elements of $V - S$ together with any empty blocks that are created in the deletion process.

Now let $c(S, A, i)$ denote the number of blocks of $\pi(A)$ contained entirely in S and having i vertices. The *component type of A on S* is the monomial

$$\mathbf{x}(S, A) = \prod_{i=1}^{\infty} x_i^{c(S,A,i)}.$$

Finally we let $c(S, A) = \sum_{i=1}^{n} c(S, A, i)$.

Let G be the graph in Figure 1, let $S = \{v_1, v_2, v_4, v_8\}$ and $A = \{e, f, g, h\}$. Then $\pi(A)$ has blocks $\{v_1, v_2, v_4, v_7\}$, $\{v_3, v_5\}$, $\{v_6\}$, $\{v_8\}$. So $\pi_G(S, A)$ has blocks $\{v_1, v_2, v_4\}$ and $\{v_8\}$ with weights one and zero respectively. Furthermore $\mathbf{x}(V - S, A) = x_1 x_2$ corresponding to the blocks $\{v_3, v_5\}$ and $\{v_6\}$.

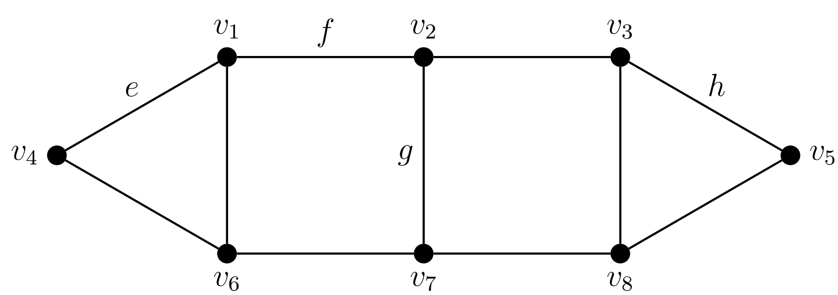

Figure 1: Weighted partition example

Note that

$$U_G(\mathbf{x}, y) = \sum_{A \subseteq E} \mathbf{x}(V, A)(y - 1)^{|A| - r(A)}$$

and also for any $S \subseteq V$

$$U_G(\mathbf{x}, y) = \sum_{A \subseteq E} \mathbf{x}(V - S, A) \prod_{B \in \pi(S,A)} x_{w_\pi(B) + |B|}(y - 1)^{|A| - r(A)}. \tag{3.1}$$

For $G = (V, E)$ and $S \subseteq V$, let $\Pi(S)$ be the set of all weighted partitions of S such that the sum of the weights is at most n. Note that $|\Pi(S)| \leq n^{|S|} B(|S|)$ where $B(k)$ denotes the kth Bell Number. Let $\Pi_0(S)$ denote the set of all weighted partitions of S with each block having weight zero.

In the algorithm we compute the evaluation of several polynomials which resemble the states model expansion of U (2.1), except that we restrict the summation to those sets of edges inducing a particular weighted partition.

Let $G = (V, E)$ and $S \subseteq V$. Let π be a weighted partition of S. Then we define

$$U_G^S(\pi; \mathbf{x}, y) = \sum_{\substack{A \subseteq E: \\ \pi_G(S,A)=\pi}} \mathbf{x}(V - S, A)(y-1)^{|A|-r(A)}. \tag{3.2}$$

In order to be completely clear, $\mathbf{x}$ and y will be specified in the input so we will think of U_G^S as an evaluation of a polynomial rather than a polynomial.

In Section 5, we shall see that the algorithm works by building up the set of pairs

$$\mathcal{U}(H, S) = \{(\pi, U_H^S(\pi; \mathbf{x}, y)) : \pi \in \Pi(S),\ U_H^S(\pi; \mathbf{x}, y) \neq 0\},$$

for successively larger subgraphs H of G and certain sets $S \subseteq V(H)$. The key step occurs when G is the union of two edge-disjoint graphs G_1 and G_2 such that $V(G_1) \cap V(G_2) = S$. Then the following lemma shows that $\mathcal{U}(G, S)$ may be computed from $\mathcal{U}(G_1, S)$ and $\mathcal{U}(G_2, S)$.

Lemma 3.1. *Let $G_1 = (V_1, E_1)$, $G_2 = (V_2, E_2)$ be graphs having disjoint edge sets. Let $S = V_1 \cap V_2$ and let $G = G_1 \cup G_2$. Then for all $\pi \in \Pi(S)$,*

$$U_G^S(\pi; \mathbf{x}, y) = \sum U_{G_1}^S(\pi_1; \mathbf{x}, y) U_{G_2}^S(\pi_2; \mathbf{x}, y)(y-1)^{(\#\pi + |S| - \#\pi_1 - \#\pi_2)},$$

where the summation is over all π_1, $\pi_2 \in \Pi(S)$ such that $\pi_1 \vee \pi_2 = \pi$.

Proof. Let $V = V_1 \cup V_2$ and $E = E_1 \cup E_2$. Recall that the definition of U_G^S is as follows.

$$U_G^S(\pi; \mathbf{x}, y) = \sum_{\substack{A \subseteq E: \\ \pi(S,A)=\pi}} \mathbf{x}(V - S, A)(y-1)^{|A|-r(A)}.$$

Now let $A_1 \subseteq E_1$, $A_2 \subseteq E_2$ and let $A = A_1 \cup A_2$. We claim that $\pi_G(S, A) = \pi_{G_1}(S, A_1) \vee \pi_{G_2}(S, A_2)$. Suppose the weighted partitions induced on S by A_1, A_2 and A are π_1, π_2 and π respectively. Two vertices in the same block of either π_1 or π_2 must be in the same block of π. Hence the blocks of π are the blocks of $\pi_1 \vee \pi_2$. Let B be a block of π and for $i = 1, 2$ let $w_i(B)$ denote the number of vertices of $V_i - S$ that lie on a path beginning at a vertex in B and containing only edges of A_i. Then the label on B in π is $w_1(B) + w_2(B)$. Now B is the disjoint union of blocks of π_1. It is not difficult to see that the sum of the labels on these blocks is $w_1(B)$ and a similar result holds considering π_2. Hence $\pi = \pi_1 \vee \pi_2$ as required. Now $\pi_G(S, A_1) = \pi_{G_1}(S, A_1)$ and $\pi_G(S, A_2) = \pi_{G_2}(S, A_2)$, so we have

$$\begin{aligned}
&\sum_{\substack{A \subseteq E: \\ \pi_G(S,A)=\pi}} \mathbf{x}(V - S, A)(y-1)^{|A|-r(A)} \\
&\qquad = \sum_{\substack{\pi_1, \pi_2: \\ \pi_1 \vee \pi_2 = \pi}} \sum_{\substack{A_1 \subseteq E_1: \\ \pi_{G_1}(S,A_1)=\pi_1}} \sum_{\substack{A_2 \subseteq E_2: \\ \pi_{G_2}(S,A_2)=\pi_2}} \mathbf{x}(V - S, A_1 \cup A_2)(y-1)^{|A_1|+|A_2|-r(A_1 \cup A_2)}.
\end{aligned}$$

Edges of G_1 do not have either endpoint in $V_2 - S$ and similarly edges of G_2 do not have either endpoint in $V_1 - S$. Consequently if $A_1 \subseteq E_1$ and $A_2 \subseteq E_2$ then for all i, $c(V - S, A_1 \cup A_2, i) = c(V_1 - S, A_1, i) + c(V_2 - S, A_2, i)$ which implies that

$$\mathbf{x}(V - S, A_1 \cup A_2) = \mathbf{x}(V_1 - S, A_1)\mathbf{x}(V_2 - S, A_2).$$

Furthermore for $i = 1, 2$ we have

$$r(A_i) = |V_i| - c(V_i - S, A_i) - \#\pi(S, A_i)$$

and similarly

$$r(A) = |V| - c(V - S, A) - \#\pi(S, A).$$

So

$$\begin{aligned} r(A) &= |V| - c(V - S, A) - \#\pi(S, A) \\ &= |V_1| - c(V_1 - S, A_1) - \#\pi(S, A_1) \\ &\quad + |V_2| - c(V_2 - S, A_2) - \#\pi(S, A_2) \\ &\quad + (\#\pi(S, A_1) + \#\pi(S, A_2) - \#\pi(S, A) - |S|) \\ &= r(A_1) + r(A_2) + \#\pi(S, A_1) + \#\pi(S, A_2) - \#\pi(S, A) - |S|. \end{aligned}$$

Finally we get

$$\begin{aligned} &\sum_{\substack{\pi_1,\pi_2: \\ \pi_1 \vee \pi_2 = \pi}} \sum_{\substack{A_1 \subseteq E_1: \\ \pi_{G_1}(S,A_1)=\pi_1}} \sum_{\substack{A_2 \subseteq E_2: \\ \pi_{G_2}(S,A_2)=\pi_2}} \mathbf{x}(V - S, A_1 \cup A_2)(y-1)^{|A_1|+|A_2|-r(A_1 \cup A_2)} \\ &\quad = \sum_{\substack{\pi_1,\pi_2: \\ \pi_1 \vee \pi_2 = \pi}} \sum_{\substack{A_1 \subseteq E_1: \\ \pi_{G_1}(S,A_1)=\pi_1}} \sum_{\substack{A_2 \subseteq E_2: \\ \pi_{G_2}(S,A_2)=\pi_2}} \mathbf{x}(V_1 - S, A_1)(y-1)^{|A_1|-r(A_1)} \\ &\qquad \cdot \mathbf{x}(V_2 - S, A_2)(y-1)^{|A_2|-r(A_2)}(y-1)^{(\#\pi+|S|-\#\pi_1-\#\pi_2)} \\ &\quad = \sum_{\substack{\pi_1,\pi_2: \\ \pi_1 \vee \pi_2 = \pi}} U^S_{G_1}(\pi_1; \mathbf{x}, y) U^S_{G_2}(\pi_2; \mathbf{x}, y)(y-1)^{(\#\pi+|S|-\#\pi_1-\#\pi_2)}, \end{aligned}$$

as required. $\square$

4 Tree-width

We begin with definitions of tree-decompositions and of tree-width. A *tree-decomposition* of a graph $G = (V, E)$ is a pair $\big(S = \{S_i | i \in I\}, T = (I, F)\big)$ where S is a family of subsets of V, one for each vertex of T, and T is a tree such that

- $\bigcup_{i \in I} S_i = V$.
- for all edges $\{v, w\} \in E$, there exists $i \in I$ such that $\{v, w\} \subseteq S_i$.
- for all $i, j, k \in I$, if j is on the path from i to k in T, then $S_i \cap S_k \subseteq S_j$.

The *width* of a tree-decomposition is $\max_{i \in I} |S_i| - 1$. The *tree-width* of a graph G is the minimum width of a tree-decomposition of G.

Given a simple graph with tree-width at most k, the algorithm given in [6] will, in time $O(g(k)n)$, produce a tree-decomposition of width at most k. Note however that

$$g(k) = k^5(2k+1)^{2k-1}((4k+5)^{4k+5}(2^{2k+5}/3)^{4k+5})^{4k+1}.$$

Let $\mathcal{T}' = (S', T')$ be the output of the algorithm. Suppose we arbitrarily give T' a root r. Then it is easy to modify $\mathcal{T}'$ to produce a tree-decomposition $\mathcal{T} = \big(\{S_i | i \in I\}, T = (I, F)\big)$ satisfying the following properties.

1. T is rooted.
2. For all $i \in I$, $|S_i| = k + 1$.
3. If S_i and S_j are joined by an edge of T then $|S_i \cap S_j| \geq k$.
4. For all $i \in I$, there is a leaf l of T such that $S_l = S_i$.
5. For all $i \in I$, either i is a leaf of T or i has two children.
6. $|I| \leq 2n$.

This follows using an easy induction and the procedure may be carried out in time $O(g(k)n)$. We call such a tree-decomposition, a *reduced rooted* tree-decomposition.

5 The algorithm

We now describe how the algorithm works and discuss its complexity. Let k be a fixed strictly positive integer. We assume that we are given a graph G with tree-width at most k, and rationals $x_1, \ldots, x_n$ and y. Remove all but one edge from each parallel class to give G'. Define $m : E(G') \to \mathbb{Z}^+$ so that $m(e)$ is the size of the parallel class (that is the maximal set of mutually parallel edges) containing e in G. Compute a reduced rooted tree-decomposition $(\mathcal{S}, T)$ of G' with width at most k.

Using property (4) of a reduced rooted tree-decomposition, we see that we may arbitrarily associate each edge $e = \{u, v\}$ of G' with a leaf l of T such that $\{u, v\} \subseteq S_l$. Let E_l denote the set of edges associated with leaf l. Then the collection $\{E_l : l \text{ is a leaf of } T\}$ forms a partition of $E(G')$.

Removing multiple edges and loops from G and defining $m(e)$ requires time $O(m)$. Computing a tree-decomposition using the algorithm in [6] requires time $O(g(k)n)$ and producing a reduced tree-decomposition from this requires time $O(g(k)n)$. Finally computing the partition $\{E_l : l \text{ is a leaf of } T\}$ needs time $O(k^2 n)$.

For $i, j \in I$, we write $i \preceq j$ if $i = j$ or i is a descendant of j in T. Now for each $i \in I$, let G_i denote the subgraph of G for which the vertex set is $\bigcup_{j \preceq i} S_j$ and the edge set consists of all edges of G for which the corresponding edge of G' is in E_l for some l

that is a descendant i in T. (It is not necessary for the algorithm to explicitly compute or construct any of these subgraphs.)

Then for each $i \in V(T)$ the algorithm iteratively computes the set of pairs

$$\mathcal{U}(G_i, S_i) = \{(\pi, U_{G_i}^{S_i}(\pi; \mathbf{x}, y)) : \pi \in \Pi(S_i),\ U_{G_i}^{S_i}(\pi; \mathbf{x}, y) \neq 0\},$$

by working upwards through the tree computing $\mathcal{U}(G_i, S_i)$ only when the sets corresponding to each of its descendants have been computed. Let $\beta(n, m, k, \mathbf{x}, y)$ denote the maximum time needed for one multiplication or addition during the computation of $U_G(\mathbf{x}, y)$.

We first deal with the computation at leaves of T.

Lemma 5.1. *If l is a leaf, then $\mathcal{U}(G_l, S_l)$ can be computed in time $O(2^{(k+1)^2} k^2 \log(p)\beta)$.*

Proof. Since $V(G_l) = S_l$, $U_{G_l}^{S_l}(\pi; \mathbf{x}) = 0$ unless the weight of each block of π is zero. So we may restrict our attention to weighted partitions where each block has weight zero. If $\pi \in \Pi_0(S_i)$ and $y \neq 1$ then

$$U_{G_l}^{S_l}(\pi; \mathbf{x}) = \sum_{A \subseteq E_l : \pi(S_l, A) = \pi} (y-1)^{-r(A)} \prod_{e \in A} (y^{m(e)} - 1).$$

If $\pi \in \Pi_0(S_l)$ and $y = 1$ then

$$U_{G_l}^{S_l}(\pi; \mathbf{x}) = \sum_{\substack{A \subseteq E_l : r(A) = |A| \\ \pi(S_l, A) = \pi}} \prod_{e \in A} m(e).$$

We compute all these sums in parallel by making one pass through all $A \subseteq E_l$, determining $\pi(S_l, A)$ in time $O(k^2)$, computing $(y-1)^{-r(A)} \prod_{e \in A}(y^{m(e)} - 1)$ or $\prod_{e \in A} m(e)$ in time $k^2 \log(p)\beta$ and adding the result on to the appropriate sum. $\square$

The next two lemmas deal with the computation at vertices of T that are not leaves. Suppose that j is the child of i in T. Recall that $S_i \setminus S_j$ contains at most one vertex. Define G_j^+ as follows: if $S_i = S_j$ then let $G_j^+ = G_j$ and otherwise form G_j^+ from G_j by adding the unique vertex of $S_i \setminus S_j$ as an isolated vertex.

First we show how to compute $\mathcal{U}(G_j^+, S_i)$ from $\mathcal{U}(G_j, S_j)$. This is really a bookkeeping exercise but its description is slightly complicated. If $S_i = S_j$ then there is nothing to be done. Otherwise let $S_j \setminus S_i = \{s\}$ and $S_i \setminus S_j = \{t\}$. Let $A \subseteq E(G_j)$ and let $\pi = \pi_{G_j}(S_j, A)$. Suppose the block of π containing s is B. Now there are two cases to consider. If $B \neq \{s\}$ then $\pi_{G_j^+}(S_i, A)$ is formed from $\pi_{G_j}(S_j, A)$ by adding $\{t\}$ as a block with weight zero, deleting s from B and incrementing the weight of B by one. Furthermore $\mathbf{x}(V(G_j^+) \setminus S_i, A) = \mathbf{x}(V(G_j) \setminus S_j, A)$. On the other hand if $B = \{s\}$ then $\pi_{G_j^+}(S_i, A)$ is formed from $\pi_{G_j}(S_j, A)$ by adding $\{t\}$ as a block with weight zero and deleting B. Now $\mathbf{x}(V(G_j^+) \setminus S_i, A) = \mathbf{x}(V(G_j) \setminus S_j, A) x_{w(B)+1}$. Notice that in either case $\pi_{G_j^+}(S_i, A)$ and $\mathbf{x}(V(G_j^+) \setminus S_i, A)$ depend on A only via $\pi_{G_j}(S_j, A)$ and $\mathbf{x}(V(G_j) \setminus S_j, A)$. Consequently for any $\pi \in \Pi(S)$ we define π_s^t by adding $\{t\}$ as a block with weight zero and then proceeding as follows. If $\{s\}$ is a block of π then delete it, otherwise increment the weight of the block containing s by one and then delete s. If B is the block of π containing s then we define

$$x(s,\pi) = \begin{cases} x_{w(B)+1} & \text{if } B = \{s\}, \\ 1 & \text{otherwise.} \end{cases}$$

Lemma 5.2. *Let j be a child of i in T with $S_i \neq S_j$ and let $\pi_0 \in \Pi(S_i)$. Then using the notation above*

$$U_{G_j^+}^{S_i}(\pi_0; \mathbf{x}, y) = \sum_{\pi \in \Pi(S_j): \pi_s^t = \pi_0} U_{G_j}^{S_j}(\pi; \mathbf{x}, y) x(s,\pi).$$

Furthermore $\mathcal{U}(G_j^+, S_i)$ can be computed from $\mathcal{U}(G_j, S_j)$ in time $O(B(k)n^{k+1}\beta)$.

Proof. First note that if $\{t\}$ is not a block of π_0 then $U_{G_j^+}^{S_i}(\pi_0; \mathbf{x}, y) = 0$. Otherwise by (3.2) we have

$$U_{G_j^+}^{S_i}(\pi_0; \mathbf{x}, y) = \sum_{\substack{A \subseteq E(G_j^+): \\ \pi(S_i, A) = \pi_0}} \mathbf{x}(V(G_j^+) \setminus S_i, A)(y-1)^{|A| - r(A)}. \tag{5.1}$$

Furthermore we have

$$\sum_{\pi \in \Pi(S_j): \pi_s^t = \pi_0} U_{G_j}^{S_j}(\pi; \mathbf{x}, y) x(s,\pi)$$
$$= \sum_{\substack{\pi \in \Pi(S_j): \\ \pi_s^t = \pi_0}} \sum_{\substack{A \subseteq E(G_j): \\ \pi(S_j, A) = \pi}} \mathbf{x}(V(G_j) \setminus S_j, A)(y-1)^{|A| - r(A)} x(s,\pi). \tag{5.2}$$

From the discussion preceding the lemma and the fact that $E(G_j) = E(G_j^+)$, a set A contributes to the sum in (5.1) if and only if it contributes to the right-hand side of (5.2). Furthermore for such a set A, the discussion preceding the lemma implies that $\mathbf{x}(V(G_j^+) \setminus S_i, A) = \mathbf{x}(V(G_j) \setminus S_j, A)x(s,\pi)$. Hence the first part of the lemma follows.

To see that the complexity calculation is correct first recall that $|\Pi(S_i)| = n^{k+1}B(k+1)$. However if t does not occur as a singleton block of weight zero in π_0 then $U_{G_j^+}^{S_i}(\pi_0; \mathbf{x}, y) = 0$, so we only have to compute $U_{G_j^+}^{S_i}(\pi_0; \mathbf{x}, y)$ for $n^k B(k)$ different weighted partitions π_0.

For such a weighted partition π_0, we must determine which weighted partitions of S_j appear in the sum in (5.1). There are two types, those in which s appears as a singleton block and those in which s does not. In the former case we may add s to π_0 as a singleton block with any of the possible $O(n)$ weights and in the latter case we may add s to any of the at most k blocks of π_0. In both cases we remove the singleton block containing t. It remains to calculate $\mathbf{x}(s,\pi)$ for each of these $O(n)$ partitions and finally compute the sum. The total time required is $O(B(k)n^{k+1}\beta)$ as required. ∎

The second lemma follows from Lemma 3.1.

Lemma 5.3. *Let j and k be the children of i in T. Then $\mathcal{U}(G_i, S_i)$ can be computed from $\mathcal{U}(G_j^+, S_i)$ and $\mathcal{U}(G_k^+, S_i)$ in time $O((B(k+1))^2 kn^{2k+2}(k+\beta))$.*

Proof. Applying Lemma 3.1 we see that For all $\pi \in \Pi(S_i)$,

$$U_{G_i}^{S_i}(\pi; \mathbf{x}, y) = \sum U_{G_j^+}^{S_i}(\pi_1; \mathbf{x}, y) U_{G_k^+}^{S_i}(\pi_2; \mathbf{x}, y)(y-1)^{(\#\pi+k+1-\#\pi_1-\#\pi_2)},$$

where the summation is over all π_1, $\pi_2 \in \Pi(S)$ such that $\pi_1 \vee \pi_2 = \pi$.

We can compute $\mathcal{U}(G_i, S_i)$ by making one pass through all pairs $(\pi_1, U_{G_j^+}^{S_i}(\pi_1; \mathbf{x}, y))$ and $(\pi_2, U_{G_k^+}^{S_i}(\pi_2; \mathbf{x}, y))$ of elements from $\mathcal{U}(G_j^+, S_i)$ and $\mathcal{U}(G_k^+, S_i)$ respectively, computing $\pi_1 \vee \pi_2$ and adding the contribution from this pair to the sum giving $(\pi_1 \vee \pi_2, U_{G_i}^{S_i}(\pi_1 \vee \pi_2; \mathbf{x}, y))$. To find $\pi_1 \vee \pi_2$ requires time $O(k^2 + k\beta)$; to find $(y-1)^{(\#\pi+|S|-\#\pi_1-\#\pi_2)}$ requires time $O(k\beta)$. Consequently the complexity estimate follows. ☐

Together, the preceding two lemmas show that for any $i \in I$ with children j and k, we can calculate $\mathcal{U}(G_i, S_i)$ from $\mathcal{U}(G_j, S_j)$ and $\mathcal{U}(G_k, S_k)$. Finally we can recover $U_G(\mathbf{x}, y)$ given $\mathcal{U}(G_r, S_r)$, where r is the root of T.

Lemma 5.4. *Let r be the root of T. Then*

$$U_G(\mathbf{x}, y) = \sum_{\pi \in \Pi(S_r)} U_{G_r}^{S_r}(\pi; \mathbf{x}, y) \prod_{B \in \pi} x_{(w_\pi(B)+|B|)}.$$

Furthermore $U_G(\mathbf{x}, y)$ can be computed from $\mathcal{U}(G_r, S_r)$ in time $O(B(k+1)n^{k+1}k\beta)$.

Proof. By applying the definition of U_G^S and the fact that $G_r = G$, we have

$$\sum_{\pi \in \Pi(S_r)} U_{G_r}^{S_r}(\pi; \mathbf{x}, y) \prod_{B \in \pi} x_{(w_\pi(B)+|B|)} = \sum_{A \subseteq E(G)} \mathbf{x}(V - S_r, A)(y-1)^{|A|-r(A)} \prod_{B \in \pi(S_r, A)} x_{(w_{\pi(S_r,A)}(B)+|B|)}.$$

Comparing this expression with (3.1), we see that this is exactly U_G. The sum contains $O(B(k+1)n^{k+1})$ terms and to compute each term requires time $O(k\beta)$, so the complexity calculation is correct. ☐

By combining Lemmas 5.1–5.4 and the remarks at the beginning of this section, we see that the overall running time of the algorithm is $O(g(k)n^{2k+3}\log(p)\beta)$. Finally we compute bounds on the numbers involved in the computations. When $y \neq 1$ the numbers involved other than the weights of the blocks of partitions may be written in the following form:

$$\sum_{A \in \mathcal{A}} x_1^{a_1} \cdots x_n^{a_n} (y-1)^{|A|-r(A)}$$

where $\mathcal{A}$ is a collection of subsets of $E(G)$ and $\sum_{i=1}^n a_i \leq n$. Recall that for all i, $x_i = p_i/q_i$ and $y = p_0/q_0$. Let

$$M = \max\{|p_0|, |p_1|, \ldots, |p_n|, |q_0|, |q_1|, \ldots, |q_n|\}.$$

Then

$$\begin{aligned}
&\sum_{A \in \mathcal{A}} x_1^{a_1} \cdots x_n^{a_n} (y-1)^{|A|-r(A)} \\
&\quad = \sum_{A \in \mathcal{A}} \frac{p_1^{a_1} \ldots p_n^{a_n} (p_0 - q_0)^{|A|-r(A)}}{q_1^{a_1} \ldots q_n^{a_n} q_0^{|A|-r(A)}} \\
&\quad = \sum_{A \in \mathcal{A}} \frac{p_1^{a_1} q_1^{n-a_1} \ldots p_n^{a_n} q_n^{n-a_n} (p_0 - q_0)^{|A|-r(A)} q_0^{m+r(A)-|A|}}{q_1^n \ldots q_n^n q_0^m}.
\end{aligned}$$

Considering the denominator, we have $|q_1^n \ldots q_n^n q_0^m| \leq M^{n^2+m}$. For the numerator we have

$$\sum_{A \subseteq E(G)} p_1^{a_1} q_1^{n-a_1} \ldots p_n^{a_n} q_n^{n-a_n} (p_0 - q_0)^{|A|-r(A)} q_0^{m+r(A)-|A|} \leq 2^m M^{n^2} (2M)^m.$$

Similar bounds hold when $y = 1$. Hence a bound on the size of any of the numbers occurring in the algorithm is $M^{n^2+m} 4^m$. To add, subtract, multiply or divide two b-bit integers takes at most $O(b \log(b) \log(\log(b)))$ time [1, 12]. So the overall running time of the algorithm is

$$O(g(k) n^{2k+3} (n^2 + m) \log(p) r \log(r(n+m)) \log(\log(r(n+m)))),$$

where $r = \log(\max\{|p_0|, \ldots, |p_n|, |q_0|, \ldots, |q_n|\})$.

When the graph is simple, $m \leq kn$ and so the running time is at most

$$O(g(k) n^{2k+5} r \log(rn) \log(\log(rn))).$$

Acknowledgement

We thank Janos Makowsky and the anonymous referee for their helpful comments.

References

[1] A. V. Aho, J. E. Hopcroft, and J. D. Ullman. *The Design and Analysis of Computer Algorithms*. Addison-Wesley, Reading Massachusetts, 1974.

[2] A. Andrzejak. A polynomial-time algorithm for computation of the Tutte polynomials of graphs of bounded treewidth. *Discrete Mathematics*, 190:39–54, 1998.

[3] S. Arnborg, D. G. Corneil, and A. Proskurowski. Complexity of finding embeddings in a k-tree. *SIAM Journal of Algebraic and Discrete Methods*, 8:277–284, 1987.

[4] S. Arnborg and A. Proskurowski. Linear time algorithms for NP-hard problems restricted to partial k-trees. *Discrete Applied Mathematics*, 23:11–24, 1989.

[5] H. L. Bodlaender. A tourist guide through treewidth. *Acta Cybernetica*, 11:1–21, 1993.

[6] H. L. Bodlaender. A linear time algorithm for finding tree-decompositions of small treewidth. *SIAM Journal on Computing*, 25:1305–1317, 1996.

[7] B. Bollobás and O. Riordan. A Tutte polynomial for coloured graphs. *Combinatorics, Probability and Computing*, 8:45–94, 1999.

[8] T. H. Brylawski and J. G. Oxley. The Tutte polynomial and its applications. In N. White, editor, *Matroid applications*, pages 123–225. Cambridge University Press, Cambridge, 1992.

[9] S. V. Chmutov, S. V. Duzhin, and S. K. Lando. Vassiliev knot invariants: I. Introduction. *Advances in Soviet Mathematics*, 21:117–126, 1994.

[10] S. V. Chmutov, S. V. Duzhin, and S. K. Lando. Vassiliev knot invariants: II. Intersection graph for trees. *Advances in Soviet Mathematics*, 21:127–134, 1994.

[11] S. V. Chmutov, S. V. Duzhin, and S. K. Lando. Vassiliev knot invariants: III. Forest algebra and weighted graphs. *Advances in Soviet Mathematics*, 21:135–145, 1994.

[12] J. Edmonds. Systems of distinct representatives and linear algebra. *Journal of Research of the National Bureau of Standards Section B*, 71B:241–245, 1967.

[13] G. E. Farr. A correlation inequality involving stable sets and chromatic polynomials. *Journal of Combinatorial Theory Series B*, 58:14–21, 1993.

[14] E. J. Farrell. On a general class of graph polynomials. *Journal of Combinatorial Theory Series B*, 26:111–122, 1979.

[15] P. Hliněný. The Tutte polynomial for matroids of bounded branch-width. *Combinatorics, Probability and Computing*, 15:397–409, 2006.

[16] F. Jaeger, D. L. Vertigan, and D. J. A. Welsh. On the computational complexity of the Jones and Tutte polynomials. *Mathematical Proceedings of the Cambridge Philosophical Society*, 108:35–53, 1990.

[17] J. A. Makowsky. Coloured Tutte polynomials and Kaufmann brackets for graphs of bounded tree width. *Discrete Applied Mathematics*, 145:276–290, 2005.

[18] J. A. Makowsky. Personal Communication, 2008.

[19] J. A. Makowsky and J. P. Mariño. Farrell polynomials on graphs of bounded tree width. *Advances in Applied Mathematics*, 30:160–176, 2003. Special issue of FPSAC'01 papers.

[20] S. D. Noble. Evaluating the Tutte polynomial for graphs of bounded tree-width. *Combinatorics, Probability and Computing*, 7:307–321, 1998.

[21] S. D. Noble. Complexity of graph polynomials. In G. Grimmett and C. J. H. McDiarmid, editors, *Combinatorics, Complexity, and Chance: A Tribute to Dominic Welsh*. Oxford University Press, Oxford, 2007.

[22] S. D. Noble and D. J. A. Welsh. A weighted graph polynomial from chromatic invariants of knots. *Annales de l'Institut Fourier*, 49:1057–1087, 1999.

[23] J. G. Oxley and G. P. Whittle. A characterization of Tutte invariants of 2-polymatroids. *Journal of Combinatorial Theory Series B*, 59:210–244, 1993.

[24] N. Robertson and P. D. Seymour. Graph minors II. Algorithmic aspects of tree-width. *Journal of Algorithms*, 7:309–322, 1986.

[25] N. Robertson and P. D. Seymour. Graph minors IV. Tree-width and well-quasi-ordering. *Journal of Combinatorial Theory Series B*, 48:227–254, 1990.

[26] I. Sarmiento. The polychromate and a chord diagram polynomial. *Annals of Combinatorics*, 4:227–236, 2000.

[27] R. P. Stanley. A symmetric function generalisation of the chromatic polynomial of a graph. *Advances in Mathematics*, 111:166–194, 1995.

[28] D. L. Vertigan and D. J. A. Welsh. The computational complexity of the Tutte plane: the bipartite case. *Combinatorics Probability and Computing*, 1:181–187, 1992.

[29] D. J. A. Welsh. *Complexity : Knots, Colourings, and Counting*. Number 186 in London Mathematical Society Lecture Note Series. Cambridge University Press, Cambridge, 1993.