Evaluating the Tutte Polynomial of a Rooted Graph with Bounded Tree-Width

Christopher Knapp, Steve Noble

Department of Mathematical Sciences, Brunel University London, UK

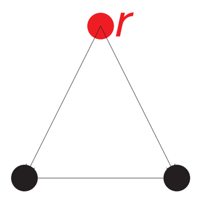
The Rooted Tutte Polynomial

A rooted graph is a graph with a fixed source vertex called the root. The Tutte polynomial of a rooted graph $G = (V, E)$ with root vertex r is given by

$$T(G, r; x, y) = \sum_{A \subseteq E} (x-1)^{\rho(E)-\rho(A)}(y-1)^{|A|-\rho(A)},$$

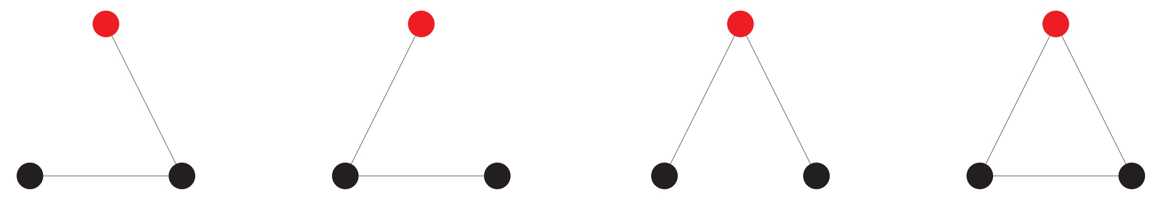
where $\rho(A) = \max_{F \subseteq A}\{|F| : F \text{ is a rooted tree}\}$ is the branching rank of A.

Example

$G =$ 

$T(G, r; x, y) = x^2y - 2xy + 2x + 2y.$

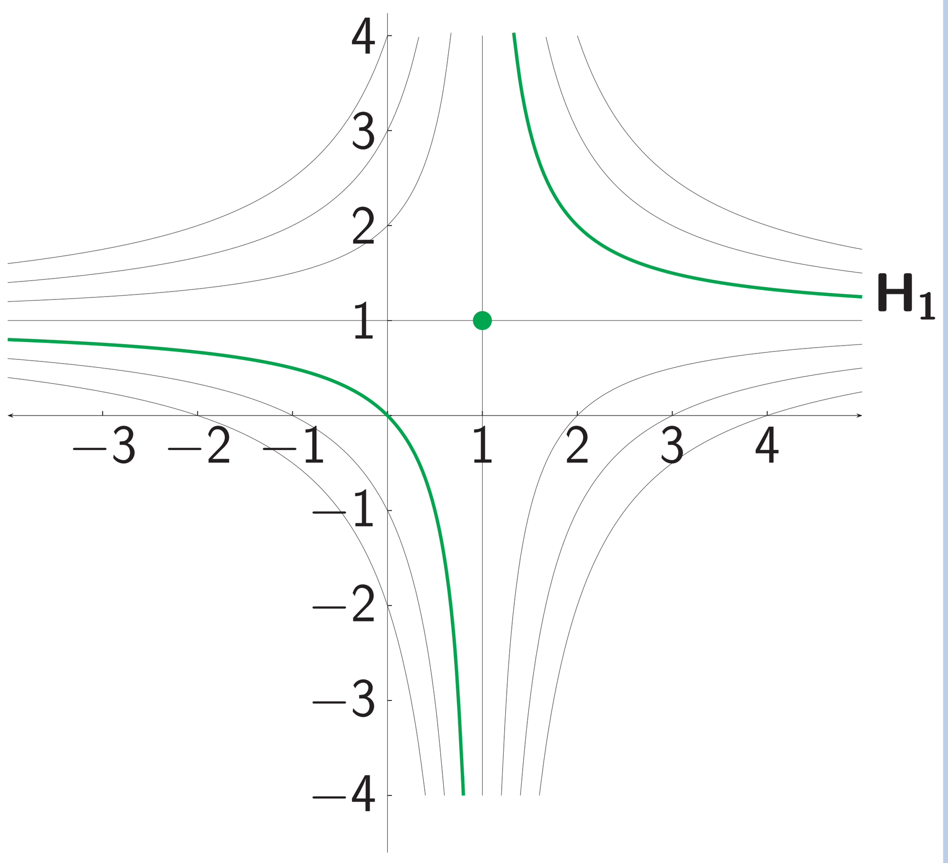
The matroidal Tutte polynomial and the rooted Tutte polynomial coincide when $x = 1$. A consequence of this is that they share evaluations along this line. For instance for G as above we have
$T(G, r; 1, 2) = \#(\text{ spanning connected subgraphs of } G) = 4$. These such graphs can be seen below.



Complexity of Evaluating the Rooted Tutte Polynomial

Evaluating the two-variable rooted Tutte polynomial anywhere in the xy-plane is #P-hard apart from at $(1, 1)$ and along the hyperbola H_1: $(x-1)(y-1) = 1$. In both of these cases there is a polynomial time algorithm to evaluate $T(G, r; x, y)$. The complexity plane showing some of the curves $H_\beta : (x-1)(y-1) = \beta$ for $\beta \in \mathbb{R}$ is given to the right, where the 'easy' points are coloured green.

Complexity Plane



Tree-Decomposition

Let $G = (V, E)$ be a rooted graph. A *nice tree-decomposition* of G is a pair $\tau = (\{S_i | i \in I\}, T = (I, B))$ with a root node S_r such that T is a tree with branches B, and for every node i of T, we have a subset S_i of V, such that:

1. $\bigcup_{i \in I} S_i = V$.
2. for every edge $(v, w) \in E$, there exists a leaf l of T such that $\{v, w\} \subseteq S_l$.
3. for all $i, j, k \in I$, if j is on the path from i to k in T, then $S_i \cap S_k \subseteq S_j$.
4. for all $i \in I$, S_i must contain the root vertex.
5. every node $i \in I$ must be one of the following types:
   - Leaf: node i is a leaf of T.
   - Join: node i has exactly two child nodes j and k in T and $S_i = S_j = S_k$.
   - Introduce: node i has exactly one child j in T, and there is a vertex $a \in V \setminus S_j$ with $S_i = S_j \cup \{a\}$.
   - Forget: node i has exactly one child j in T, and there is a vertex $a \in V \setminus S_i$ with $S_j = S_i \cup \{a\}$.
6. for every node $i \in I$ which isn't a forget node, there exists a leaf l of T such that $S_i = S_l$.

Tree-Width

The *tree-width* of τ is given by

$$\max_{i \in I} |S_i| - 1.$$

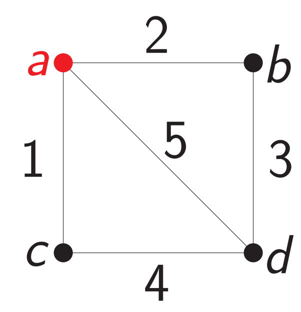
The tree-width of G is then said to be the minimum tree-width taken over all possible nice tree-decompositions of G.

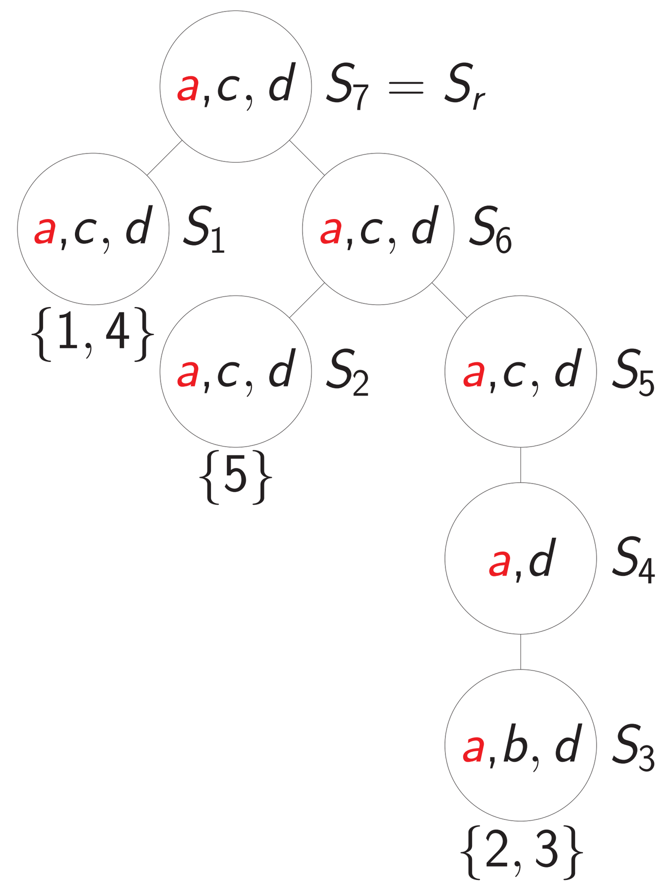
The notion of tree-width has proven to be very important in algorithmic graph theory. This is because many algorithmic problems that are intractable for arbitrary graphs, can be solved efficiently in polynomial and often linear time when restricted to the class of graphs with bounded tree-width.

Example Showing a Tree-Decomposition of a Graph

Example

Here we give a rooted graph G with root vertex a, together with a possible rooted tree-decomposition of it with tree-width 2.

$G =$ 



Main Idea

If we restrict the class of rooted graphs we are considering to be those with bounded tree-width, then we can find a linear-time algorithm to evaluate the rooted Tutte polynomial of such graphs anywhere in the xy-plane.

Concept of the Algorithm

Let $G = (V, E)$ be a simple rooted graph with tree-width at most k.

Let $\tau = (\{S_i | i \in I\}, T = (I, B))$ be a nice tree-decomposition of G positioned such that the root node S_r is at the top and all other nodes in T are descendants of it (like in the previous Example).

We begin by partitioning the edges of G among the leaf nodes of T. To each node $i \in I$, we associate a set of vertices $V_i \subseteq V$ and edges $E_i \subseteq E$ and let

$$V_i' = \bigcup_{j \leq i} V_j \quad \text{and} \quad E_i' = \bigcup_{j \leq i} E_j,$$

where the union is taken over every leaf j such that j is a descendant of i in T. Therefore to every node $i \in I$ in τ we associate a subgraph $G_i = (V_i', E_i')$.

We define a state α on a node set S_i in T to be a partition π_α of some subset S_α of S_i into non-empty blocks $B_1, \ldots, B_k$ such that the root vertex is in block B_1. We let $B_0 = S_i \setminus S_\alpha$ and $|\pi_\alpha|$ denote the number of blocks in S_α.

We essentially say that a subset of edges $A_i \subseteq E_i'$ induces a state α if $G_i | A_i$ partitions the connected components of the vertices in S_i into the same blocks as α.

Let

$$f(A, \alpha) = |\text{ Vertices in } G_i \setminus S_i \text{ that are not connected to } S_\alpha\ |,$$

and

$$g(A, \alpha) = |A| - |V(G)| + f(A, \alpha) + |S| - |S_\alpha| + |\pi_\alpha|.$$

For every node i in τ we compute

$$T_{G_i}(\alpha; x, y) = \sum (x-1)^{f(A,\alpha)}(y-1)^{g(A,\alpha)} \qquad (1)$$

where we are summing over all subsets of edges $A_i \subseteq E_i'$ which induce state α, for all possible states on S_i.

If we begin by computing this for each state on the leaf nodes in τ then we can easily adapt these polynomials as we work our way up the nice tree-decomposition towards S_r, eventually calculating the required rooted Tutte polynomial of G. This can all be done in linear time.

Example (Continued)

Computing (**??**) for each state on S_r with $|\pi_\alpha| = 1$ in the nice tree-decomposition of G we get

- **acd** : $y^2 + 2y + xy + 2x + 2$
- **ac|d** : $x^2y - 2xy + x + y - 1$
- **ad|c** : $x^2 + xy - y - 1$
- **a|cd** : $x^3y^2 - 3x^2y^2 + x^2y + 3xy^2 - 2xy - y^2 + y$.

Summing these polynomials gives
$T(G, a; x, y) = x^3y^2 - 3x^2y^2 + 3xy^2 + 2x^2y - 2xy + x^2 + 3x + 3y.$