Exact counting of Euler Tours for Graphs of Bounded Tree-width

Prasad Chebolu
Department of Computer Science
University of Liverpool
prasadc@liverpool.ac.uk

Mary Cryan†
School of Informatics
University of Edinburgh
mcryan@inf.ed.ac.uk

Russell Martin
Department of Computer Science
University of Liverpool
Russell.Martin@liverpool.ac.uk

Abstract

In this paper we give a simple polynomial-time algorithm to exactly count the number of Euler Tours (ETs) of any Eulerian graph of bounded treewidth. The problems of counting ETs are known to be $\#P$-complete for general graphs (Brightwell and Winkler, 2005 [4]). To date, no polynomial-time algorithm for counting Euler tours of any class of graphs is known except for the very special case of series-parallel graphs (which have treewidth 2).

1 Introduction

1.1 Background

Let $G = (V, E)$ be an undirected graph. An Euler tour (ET) of $G$ is any ordering $e_{\pi(1)}, \ldots, e_{\pi(|E|)}$ of the edges of $E$, and of the directions of those edges, such that for every $1 \leq i < |E|$, the target vertex of arc $e_{\pi(i)}$ is the source vertex of $e_{\pi(i+1)}$, and such that the target vertex of $e_{\pi(|E|)}$ is the source of $e_{\pi(1)}$. We use $ET(G)$ to denote the set of Euler tours of $G$, where two Euler tours are considered to be equivalent if one is a cyclic permutation of the other. It is a well-known fact that a given undirected graph $G$ has at least one Euler tour if and only if $G$ is connected and every vertex $v \in V$ has even degree.

The problem of counting the number of Euler tours is considerably more difficult than that of finding a single one, and few positive results exist. Brightwell and Winkler [4] have shown that the problem of counting ETs of undirected graphs is $\#P$-complete. This is in contrast to the problem of counting Euler tours of a directed Eulerian graph, where the number of Euler tours can be counted exactly in polynomial-time using the Matrix-Tree theorem[3] and the so-called “BEST” Theorem (after de Bruijn, van Aardenne, Smith and Tutte, [1,10], though apparently the first two deserve credit as the original discoverers). Creed[6] showed that the problem on undirected graphs remains $\#P$-complete even for planar graphs. Therefore it is unlikely that a polynomial-time algorithm exists for the general class of undirected graphs or planar undirected graphs.

It is well-known that many computational problems become tractable on graphs of bounded treewidth. For example, Noble [9] showed that any point of the Tutte polynomial can be evaluated on graphs of bounded treewidth in polynomial time. The number of ETs is not a point of the Tutte polynomial, nor do we know

---

*Supported by EPSRC grant EP/D043905/1
†Corresponding author
any reduction from Euler tours to a point of the Tutte (which would preserve treewidth). However, Cryan et al.\({\cite{5}}\) previously gave a direct polynomial-time algorithm for the special case of series-parallel graphs (which have treewidth 2). In this paper we extend this work to give a polynomial-time algorithm for graphs of treewidth \(d\), for bounded \(d \in \mathbb{N}_0\).

This is a preliminary paper; we believe that we can extend our results to compute any point of the Martin polynomial (whose coefficients equal the number of vertex-pairings of the graph inducing \(\kappa\) components, for \(\kappa = 1, 2, \ldots\)). Makowsky and Mariño\({\cite{8}}\) conjectured in 2001 that the Martin polynomial could be computed for graphs of bounded treewidth; however, no proof of this result exists yet.

### 1.2 Basic definitions

Throughout this paper that we are working with a connected Eulerian graph \(G = (V, E)\) - that is, a connected graph for which every vertex has even degree. We first introduce the well-known concepts of tree decomposition and treewidth of a graph:

**Definition 1.1** A tree decomposition of a graph \(G = (V, E)\) is a pair \((\{X_i \mid i \in I\}, T = (I, F))\), where \(T\) is a tree on \(I\) and where \(X_i \subseteq V\) for all \(i \in I\), such that the following properties are satisfied:

1. \(\cup_{i \in I} X_i = V\).
2. For every \((u, v) \in E\), there is some \(i \in I\) such that \(u, v \in X_i\).
3. For every \(v \in V\), the set of nodes \(\{i \in I \mid v \in X_i\}\) containing \(v\) induces a subtree of \(T\).

The width of a tree decomposition \((\{X_i \mid i \in I\}, T = (I, F))\) is \(\max_{i \in I} |X_i| - 1\). The treewidth of a graph \(G\) is the minimum width over all tree decompositions of \(G\).

The problem of computing the treewidth of a graph is well-known to be fixed parameter tractable in the treewidth of the graph itself (see for example, Bodlander\({\cite{2}}\)). Moreover, many graph problems which are NP-complete in general can be solved in polynomial-time on graphs of bounded treewidth - that is, graphs where the treewidth is at most \(k\) for some fixed \(k\). There are also encouraging results for counting/sampling for \#P-complete problems. In particular, Noble \({\cite{9}}\) has shown that any point of the Tutte polynomial can be evaluated in polynomial time, for graphs of bounded treewidth.

A refinement of tree decomposition is the concept of a nice tree decomposition:

**Definition 1.2** A nice tree decomposition of a graph \(G\) is defined to be any tree decomposition \((\{X_i \mid i \in I\}, T = (I, F))\), where a particular node \(\rho\) of \(T\) is considered to be the root of \(T\), and each node \(i \in I\) is one of the four following types:

- **leaf:** node \(i\) is a leaf of the tree \(T\), and \(|X(i)| = 1\).
- **join:** node \(i\) has exactly two children \(j\) and \(j'\), and \(X(i) = X(j) \cup X(j')\).
- **introduce:** node \(i\) has exactly one child \(i'\), and there is one vertex \(v \in V\) such that \(X(i) = X(i') \cup \{v\}\).
- **forget:** node \(i\) has exactly one child \(i'\), and there is one vertex \(v \in V\) such that \(X(i) = X(i') \setminus \{v\}\).

It is well-known that if a graph \(G\) has a tree decomposition of width \(k\), then it also has a nice tree decomposition of width \(k\) and with \(O(n)\) nodes. Also, for any constant \(k\), there are well-known algorithms to find a nice tree decomposition of width \(k\) in polynomial-time (see, for example, \({\cite{7}}\)), for any given graph which has treewidth at most \(k\).

We will give an exact algorithm for counting ETs of a graph of treewidth \(k\), for any constant \(k \in \mathbb{N}_0\).

First observe there is a many-to-one relationship between ETs and Eulerian Orientations of a graph. Let \(EO(G)\) denote the set of all Eulerian Orientations of the graph \(G\).
**Observation 1** Let \( G = (V, E) \) be any Eulerian multigraph. Then any Euler tour \( T \in ET(G) \) induces a unique Eulerian Orientation on \( G \).

The elements of \( ET(G) \) can be partitioned according to the particular Eulerian orientation they induce:

\[
ET(G) = \bigcup_{E \in EO(G)} ET(E),
\]

where \( ET(E) \) is the set of Euler tours of the directed graph given by \( E \). We can refine this relationship further using the well-known relationship between Euler tours and in-Arborescences on directed Eulerian graphs:

**Theorem 2** ([1],[10]) Let \( \vec{G} = (V, \vec{E}) \) be an Eulerian directed graph where each vertex \( v \in V \) has outdegree (and indegree) \( d_v \). Then for any \( r \in V \),

\[
|ET(\vec{G})| = \prod_{v \in V} (d_v - 1)! |ARB(\vec{G}, r)|.
\]

Equations 1 and 2 motivate the following definitions of an Orb of an undirected connected Eulerian graph:

**Definition 1.3** Let \( G = (V, E) \) be an connected Eulerian multigraph. An Orb \( O \) is a pair \( O = (E, A) \), where \( E \in EO(G) \) and \( A \) is an in-directed Arborescence (rooted at some arbitrary \( r \in V \)) on \( E \).

- Let \( ORB(G) \) denote the set of all orbs \( (E, A) \) on \( G \);
- For any \( r \in V \), let \( ORB(G, r) \) denote the set of all orbs \( (E, A) \) on \( G \), where \( A \) is rooted at \( r \).

For any \( E \in EO(G) \), we identify \( E \) with the directed graph it induces on \( V \).

- Let \( ARB(E) \) denote the set of all in-Arborescences on \( E \);
- Let \( ARB(E, r) \) denote the set of all in-Arborescences on \( E \) which are rooted into \( r \).

Combining 1 and 2, we have the following interpretation of the Euler tours of a given undirected graph \( G \) with degrees \( 2d_v \):

\[
|ET(G)| = \prod_{v \in V} (d_v - 1)! |ORB(G, r)|.
\]

Therefore in order to count Euler tours of an undirected Eulerian multigraph, it suffices to count Orbs of that graph. This is the approach we will take in this paper.

### 2 Counting Orbs

Consider a given Eulerian (multi)graph \( G = (V, E) \) of treewidth \( k \), and let \( (\{X(i) \mid i \in I\}, T = (I, F)) \) be a nice tree decomposition of \( G \) with width \( k \). Let \( \rho \in I \) be the root of the tree decomposition, and designate any vertex in \( X_\rho \) to be the distinguished vertex \( r \). We will show how to evaluate the cardinality of \( ORB(G, r) \) in \( O(n^{O(k)}) \) time.

In the following two pages we introduce various concepts which we will use in our dynamic programming algorithm, and make some simple observations.
Definition 2.1 Let \( \{ X(i) \mid i \in I \}, T = (I, F) \) be any tree decomposition of the graph \( G = (V, E) \), and let \( i \in I \). We define the set \( V(i) \) to be \( \cup_{j \in T} X(j) \), the set of all vertices of \( G \) which appear in the subtree of the tree decomposition rooted at \( i \).

Definition 2.2 Let \( \{ X(i) \mid i \in I \}, T = (I, F) \) be any tree decomposition of the graph \( G = (V, E) \). For any \( i \in I \), we define the following two subsets of \( E \):

\[
E(i) = \{ e = (u, v) \in E : u, v \in X(i) \}.
\]
\[
E_\ell(i) = \{ e = (u, v) \in E : u, v \in V(i), |\{u, v\} \cap X(i)| \leq 1 \}.
\]

Let \( G(i) \) and \( G_\ell(i) \) be the subgraphs of \( G \) induced by the edge sets above in turn. Observe that for any \( i \in I \), the edge sets of \( G(i) \) and \( G_\ell(i) \) are edge-disjoint.

Observation 3 Let \( \{ X(i) \mid i \in I \}, T = (I, F) \) be any tree decomposition of the graph \( G = (V, E) \) with root node \( \rho \).

Observe the union of \( G(\rho) \) and \( G_\ell(\rho) \) is the original graph \( G \).

Definition 2.3 Let \( \{ X(i) \mid i \in I \}, T = (I, F) \) be any tree decomposition of the graph \( G = (V, E) \). For any \( i \in I \) we define the two following disjoint subsets of \( X(i) \):

\[
L(i) = \{ x \in X(i) : \exists v : (x, v) \in E_\ell(i) \};
\]
\[
U(i) = \{ x \in X(i) : \exists v : (x, v) \in E_\ell(i) \}.
\]

For a given Orb, we can define induced orientations and arc sets at any node of the tree decomposition:

Definition 2.4 Let \( G = (V, E) \) be a Eulerian multigraph with tree decomposition \( \{ X(i) \mid i \in I \}, T = (I, F) \). Let \( \rho \) be the root of the decomposition, and consider the designated root \( r \in X(\rho) \). Then for any orb \( O = (E, A) \in ORB(G, r) \), and any node \( i \) of the decomposition, we define the following:

(i) \( \mathcal{E}_\ell(i) \) is the restriction of \( \mathcal{E} \) to the subgraph \( G_\ell(i) \);

(ii) \( \mathcal{E}(i) \) is the restriction of \( \mathcal{E} \) to the subgraph \( G(i) \);

(iii) \( \mathcal{A}_\ell(i) \) is the restriction of \( \mathcal{A} \) to arcs in \( \mathcal{E}_\ell(i) \).

(iv) \( \mathcal{A}(i) \) is the restriction of \( \mathcal{A} \) to arcs belonging to \( \mathcal{E}(i) \);

We now make a simple observation concerning the structures induced at the root node \( \rho \):

Observation 4 Suppose we are given an Eulerian multigraph \( G = (V, E) \), with the nice tree decomposition \( \{ X(i) \mid i \in I \}, T = (I, F) \), having root \( \rho \). Let \( r \in X(\rho) \), and let \( (E, A) \in ORB(G, r) \). Then

(i) \( \mathcal{E}_\ell(\rho) \) is an orientation on \( G_\ell(\rho) \) that satisfies the Eulerian condition at all \( v \in V(\rho) \setminus X(\rho) \), but not necessarily at \( v \in X(\rho) \).

(ii) \( \mathcal{E}(\rho) \) is an orientation on \( G(\rho) \), such that for every \( v \in X(\rho) \), the difference \( (out_{\mathcal{E}(\rho)}(v) - in_{\mathcal{E}(\rho)}(v)) \) is equal to \( (in_{\mathcal{E}_\ell(\rho)}(v) - out_{\mathcal{E}_\ell(\rho)}(v)) \).

(iii) \( \mathcal{A}_\ell(\rho) \) is an in-directed forest on \( \mathcal{E}_\ell(\rho) \) with some root set \( R \subseteq L(\rho) \), such that every \( v \in (V(\rho) \setminus X(\rho)) \cup (L(\rho) \setminus R) \) has a out-arc in \( \mathcal{A}_\ell(\rho) \).

(iv) \( \mathcal{A}(\rho) \) is a set of arcs from \( \mathcal{E}(\rho) \), containing exactly one out-arc for every \( x \in ((X(\rho) \setminus L(\rho)) \cup R) \setminus \{r\} \) such that \( \mathcal{A}(\rho) \) is an in-directed tree rooted at \( r \).
We will exploit Observation 4 in the design of our dynamic programming algorithm to count Orbs of all Eulerian graphs with constant treewidth. Our algorithm will count pairs of the form \((\mathcal{E}_t(\rho), A_t(\rho))\). In fact, we will count pairs of this type for all nodes \(i\) of our treewidth decomposition, partitioning the set of these pairs according to two parameters which we will call the charge vector and the root vector.

We now define the sets of Orientations we will consider in building the dynamic programming table.

**Definition 2.5** Let \(G = (V, E)\) be an Eulerian multigraph with tree decomposition \(\{X(i) \mid i \in I\}, T = (I, F)\), and consider any \(i \in I\).

- Let \(D_t(i)\) denote the set of all orientations of the edges of \(G_t(i)\) which are Eulerian at every vertex \(v \in V(i) \setminus X(i)\), but not necessarily at \(v \in X(i)\).
- Let \(D(i)\) denote the set of all orientations (not necessarily Eulerian) of the edges of \(G(i)\).

Note that parts (i) and (ii) of Observation 4 could be re-stated by saying that \(\mathcal{E}_t(\rho) \in D_t(\rho)\) and \(\mathcal{E}(\rho) \in D(\rho)\) respectively.

We will partition the orientations of \(D(i)\) and \(D_t(i)\) in terms of the “charge” (outdegree - indegree) induced on the \(X(i)\) vertices by the orientation.

**Definition 2.6** Let \(G = (V, E)\) be a connected Eulerian multigraph with tree decomposition \(\{X_i \mid i \in I\}, T = (I, F)\), and let \(i \in I\). We define the following sets of “charge vectors”:

- \(C(i) \subseteq \mathbb{Z}^{|X(i)|}\) is the set of all vectors \(c\) which can be generated by some orientation in \(D(i)\).
- \(C_t(i) \subseteq \mathbb{Z}^{|X(i)|}\) is the set of all vectors \(c\) which can be generated by some orientation in \(D_t(i)\).

When discussing a specific orientation \(\mathcal{D} \in D(i)\) or \(\mathcal{D} \in D_t(i)\), we write \(c(\mathcal{D})\) for the charge vector induced by \(\mathcal{D}\). Observe that the vertex set induced by an orientation \(\mathcal{D} \in D_t(i)\) is \((V(i) \setminus X(i)) \cup L(i)\), because the vertices of \(U(i) (= X(i) \setminus L(i))\) are not endpoints of any edge in \(G_t(i)\). Therefore, in relation to Definition 2.5, the only vertices for which \(\mathcal{D}\) will violate the Eulerian property are the vertices in \(L(i)\), rather than all of \(X(i)\). To be consistent with the charge vectors of \(C(i)\), we describe the charge vectors of \(C_t(i)\) in terms of \(X(i)\), although every \(c \in C_t(i)\) is guaranteed to have \(c_v = 0\) for all vertices \(v \in U(i)\).

In counting \((\mathcal{E}_t(i), A_t(i))\) pairs, we will partition the set of such pairs according to the charge vector of the orientation, but will also consider the root-status of the \(X(i)\) vertices in relation to \(A_t\). For every \(x \in X(i)\), there are three possible scenarios in regard to \(A_t(i)\):

1. \(x \in U(i)\), in which case \(x\) does not belong to the induced multigraph \(G_t(i)\).
2. \(x \in L(i)\), and \(x\) is a (possibly isolated) root of a subtree in \(A_t(i)\);
3. \(x \in L(i)\), and \(x\) is not the root of a subtree in \(A_t(i)\). There is some \(y \in L(i)\) such that \(y\) is a root in \(A_t(i)\), such that \(x\) belongs to \(y's\) subtree (via an out-arc \(x \rightarrow z\) for some \(z \in V(i) \setminus X(i)\)).

We use this distinction between vertices of \(X(i)\) to define the concept of a root vector for an arbitrary node \(i\) of the tree decomposition:

**Definition 2.7** Let \(G = (V, E)\) be a connected Eulerian multigraph, and suppose we have a tree decomposition \(\{X(i) \mid i \in I\}, T = (I, F)\) of \(G\) of width \(k\). Let \(i \in I\). We define the set \(S(i)\) of root vectors induced by \((\mathcal{E}_t(i), A_t(i))\) pairs to be the set \(S(i)\) of all \(s \in |X(i)|^{X(i)}\) satisfying the following properties:

- For every \(x \in U(i)\), \(s_x = x\);
- There is at least one \(x \in L(i)\) such that \(s_x = x\);
• Let \( R(s) \subseteq L(i) \) be the set of vertices \( x \in R(s) \) such that \( s_x = x \). Then for every \( y \in L(i) \setminus R(s) \), \( s_y \in R(s) \).

By Observation \([4]\) and by our discussion above Definition \([7]\) every induced forest \( A_\ell(i) \) of an orb \((E, A) \in ORB(G, r)\) is consistent with a unique \( s \in S(i) \). For a specific forest \( F \) on \( G_\ell(i) \), we will write \( s(F) \) to denote the vector of \(|X(i)|^{X(i)}\) which indicates, for each of the vertices in \( X(i) \), the root of the tree in \( F \) which contains it. Note that the information carried by the vectors of \( S(i) \) could also be encoded as vectors in \(|L(i)|^{L(i)}\), however to have consistency with the charge vectors, we assume the root vectors have length \(|X(i)|\), while enforcing the constraint that all vertices \( x \) of \( U(i) \) have \( s_x \) set to \( x \).

Now we define the concepts of forests and forest Orbs for nodes of the tree decomposition.

**Definition 2.8** Let \( G = (V, E) \) be an Eulerian multigraph, and suppose we have a tree decomposition \((\{X(i) \mid i \in I\}, T = (I, F)) \) of \( G \) of width \( k \). Let \( i \in I \). Let \( D \in D_\ell(i) \). We define a forest with respect to \( D \) to be any in-directed forest \( F \) on \( D \) such that

- There is some set \( R(F) \subseteq L(i) \) such that \( R(F) \) is the set of roots of \( F \);
- Every \( v \in (V(i) \setminus X(i)) \cup (L(i) \setminus R(F)) \) has an out-arc in \( F \).

We write \( FOR(D) \) to denote the set of all forests on \( D \) (wrt \( i \)). We define a forest Orb to be any pair \((D, F)\) such that \( D \in D_\ell(i) \) and \( F \in FOR(D) \).

Our algorithm will construct a table \( \Psi(i) \), indexed by pairs \((c, s)\) for \( c \in C(i) \) and \( s \in S(i) \). The table will store the value

\[
\psi(i, c, s) = \sum_{D \in D_\ell(i)} \sum_{F \in FOR(D)} I_{c(D) = c, s(F) = s}.
\]

In the following section, we will show how to build the table \( \Psi(i) \) for every node \( i \) of the tree decomposition in polynomial-time, in a bottom-up fashion of the tree \((I, F)\). However we first show how, once we have this table constructed for the root \( \rho \) of the tree decomposition, we can then compute the number of Orbs of the original graph \( G \).

**NOT DONE YET.** It will be similar to the proof of the ‘forget’ case.

We now have the following observation about “charge vectors”.

**Observation 5** Suppose \( G = (V, E) \) is a graph (or multi-graph) with tree decomposition \((\{X_i \mid i \in I\}, T = (I, F))\), and let \( i \in I \). Then every \( c \in C(i) \) satisfies \( c(x) \in \{-d_{G_\ell(i)}(x), -d_{G_\ell(i)}(x) + 2, \ldots, d_{G_\ell(i)}(x) - 2, d_{G_\ell(i)}(x)\} \).

Combining Observation \([5]\) together with bounded treewidth, we can derive specific bounds on the size of \(|C(i)|\).

**Observation 6** Suppose \( G = (V, E) \) is a simple graph with tree decomposition \((\{X_i \mid i \in I\}, T = (I, F))\) of treewidth \( k \), and let \( i \in I \). Then by simplicity, we know \( d_{G_\ell(i)}(x) \leq (n - 1) \) for all \( x \in X_i \). Therefore by Observation \([5]\) \(|C(i)| \leq n^k\).

For the case of multi-graphs, we have a lesser observation:

**Observation 7** Suppose \( G = (V, E) \) is a multi-graph with tree decomposition \((\{X_i \mid i \in I\}, T = (I, F))\) of treewidth \( k \), and let \( i \in I \). Let \( m = |E| \). Then we have \( d_{G_\ell(i)}(x) \leq (m - 1) \) for every \( x \in X_i \). Hence \(|C(i)| \leq m^k\).
Finally, we present the following bound on the number of root vectors:

**Observation 8** Suppose $G = (V,E)$ is a multi-graph with tree decomposition $(\{X_i \mid i \in I\}, T = (I,F))$ of treewidth $k$, and let $i \in I$. Then the number of root vectors $|S(i)|$ satisfies the bound $S(i) \leq |L(i)||L(i)| \leq k^k$.

### 2.1 Our algorithm

We now discuss the bottom-up computation of the table $\Psi(i)$, storing the values $\psi(i, c, s)$ for all $c \in C(i)$, $s \in S(i)$.

Note that if $G = (V,E)$ is simple, then by Observations 6 and 8 the table $\Psi(i)$ contains at most $n^k k^k$ entries, where $n = |V|$. Alternatively, if $G$ is not necessarily simple, then by Observations 7 and 8 $\Psi(i)$ contains at most $m^k k^k$ entries.

We now show how to build $\Psi(i)$ for all nodes of the tree decomposition $(\{X_i \mid i \in I\}, T = (I,F))$. This is done in a bottom-up dynamic programming fashion, with the tables for node $i$ only being built after the corresponding tables for the child node (or nodes) of $i$ have already been constructed. Recall that every node of a nice treewidth decomposition has at most two child nodes.

#### 2.1.1 Leaf

In the case of a leaf node $l$, we have $X(l) = \{w\}$ for some vertex $w \in V$. $G_l(l)$ is an empty graph with no vertices or edges. There is exactly one charge vector in $C(l)$ - this is the vector $c^*$ of length $|X(l)| = 1$ which assigns charge-0 to $w$.

To consider possible sets of root-vectors, note that $X(l) = \{w\}$, and $L(l) = \emptyset$. Therefore the only root vector in $S(l)$ is the vector $s^*$ of length 1 which assigns $s^*_w = w$.

Finally, the only orientation on $G_l(l)$ to satisfy $c^*$ (or indeed any charge vector) is the empty one $D^*$; also the only forest on $D^*$ to satisfy $s^*$ is again the empty forest $F^*$ consisting of no arcs, and the single isolated vertex $w$. Hence the table $\Psi(l)$ consists of the following single entry:

$$\psi(l, c^*, s^*) = 1.$$  

#### 2.1.2 Introduce

For the case of introduce, our current node $i \in I$ has a single child $i'$, and $X(i) = X(i') \cup \{w\}$ for some $w \not\in X(i')$. By the properties of a nice treewidth decomposition, we know that for every $v \in V(i') \setminus X(i')$, there is no edge of the form $(w, v)$ in $G$. Therefore the adjacent vertices to $w$ are all either in $X(i)$ or in $V \setminus V(i)$, and $L(i) = L(i')$. The graph $G_l(i)$ is identical to $G_l(i')$. If we adopt the convention that the entry for $w$ is at the end of the charge vectors in $C(i)$, then

$$C(i) = \{c.0 \mid c \in C(i')\}.$$  

Now we consider the set of root vectors $S(i)$ in relation to $S(i')$. Assuming that the entry for $w$ will be stored at the end of the root vectors for $i$, then by Definition 2.7 and by $L(i) = L(i')$, we have

$$S(i) = \{s.w \mid s \in S(i')\}.$$  

Next we consider the value of $\psi(c.0, s.w)$, for any $c \in C(i')$, $s \in S(i')$ in relation to the table $\Psi(i')$ which has previously been computed. Given that $G_l(i) = G_l(i')$, and by Definition 2.5 we know that an orientation $D \in D_l(i)$ satisfies $c(D) = c.0$ if and only if we have $c(D) = c$ at node $i'$. Also, for any
\[ \mathcal{F} \in \text{FOR}(D), \text{we have} \ s(\mathcal{F}) = s.w \text{ at } i \text{ if and only if we have} \ s(D) = s \text{ at node } i'. \text{ Hence the values for} \]
\[ \text{the table } \Psi(i) \text{ are, for every } c \in C(i'), \text{ every } s \in S(i'), \]
\[ \psi(i, c, 0, s.w) = \psi(i', c, s). \]

2.1.3 Forget

For the case of forget, our current node \( i \in I \) has a single child \( i' \), and \( X(i) = X(i') \setminus \{w\} \) for some \( w \notin X(i') \). Note that the charge vectors in \( C(i) \) will be of length 1 less than those in \( C(i') \), because any \( c \in C(i) \) will not include an entry for \( w \). Similarly the root vectors of \( S(i) \) will be of length 1-less than those in \( S(i') \) for the same reason.

In using the values of table \( \Psi(i') \) to create the table \( \Psi(i) \), we will need a few more definitions. First of all, we define some subclasses of edges:

- For any \( u, v \in V \), we define \( E_{u,v} = \{ e \in E, e = (u, v) \} \), and \( m_{u,v} = |E_{u,v}| \);
- \( E(i', w) = \bigcup_{v \in V(i') \setminus X(i')} E_{w,v} \);
- \( E(i', w) = \bigcup_{v \in X(i') \setminus \{w\}} E_{w,v} = \bigcup_{v \in X(i)} E_{w,v} \).

Observe that \( E(i', w) \) and \( E(i', w) \) are mutually disjoint and their union is the set of all edges adjacent to \( w \).

We will use the definitions above to relate forest-Orbs for \( i \) with forest-Orbs for \( i' \). We represent a forest-Orb of \( \text{FOR}(i) \) as \((D, \mathcal{F})\) for \( D \in D_i \) and \( \mathcal{F} \in \text{FOR}(D) \). For every such \((D, \mathcal{F})\), we define

- \( \mathcal{D}' \) to be the restriction of \( D \) to the graph \( G_i(i') \) (ie, to the edges of \( E_i(i') \));
- \( Q_{w,v} \subseteq E_{u,v} \) to be the edges of \( E_{w,v} \) directed away from \( w \) in \( D \), for any \( v \) adjacent to \( w \) in \( G \). We also define \( q_{w,v} = |Q_{w,v}| \);
- \( \mathcal{F}' \) to be the restriction of \( \mathcal{F} \) to the arcs \( \mathcal{D}' \);

The following theorem specifies the relationship between elements of \( \text{FOR}(i) \) and \( \text{FOR}(i') \) in the forget case:

**Theorem 9** Let \( G = (V, E) \) be an Eulerian multigraph with tree decomposition \( \{X(i) \mid i \in I\}, T = (I, F) \), and let \( i \in I \) be a forget node such that \( X(i) = X(i') \setminus \{w\} \) for some \( w \in X(i') \), where \( w' \) is the single child of \( i \).

Suppose \( D \) is an orientation of the edges of \( G_i(i) \) and \( \mathcal{F} \) is some set of arcs of \( \mathcal{D} \). Then \((D, \mathcal{F}) \in \text{FOR}(i)\) with charge vector \( c \in C(i) \) and root vector \( s \in S(i) \) if and only if \( \mathcal{D} \) is the disjoint union of \( \mathcal{D}' \in D(i') \) and some orientation \( Q \) of \( E(i, w) \) (with induced values \( q_{w,x} \) for \( x \in A_w \cap X(i) \)), and \( \mathcal{F} \) is the disjoint union of some \( \mathcal{F}' \in \text{FOR}(D') \) and some arc set \( Q \subseteq Q \) such that all the following conditions hold:

(a) \((\mathcal{D}', \mathcal{F}') \in \text{FOR}(i')\);

(b) \( \sum_{x \in X(i)} (m_{w,x} - 2q_{w,x}) = c(\mathcal{D}')w \);

(c) \( c_x = c(\mathcal{D}')x \) for \( x \in X(i) \setminus A_w, c_x = c(\mathcal{D}')x - 2q_{w,x} + m_{w,x} \) for \( x \in X(i) \cap A_w \);

(d) The forests \( \mathcal{F}, \mathcal{F}' \) and their root sets \( R = \{x \in L(i) : s(\mathcal{F})_x = x\} \) and \( R' = \{x \in L(i') : s(\mathcal{F'})_x = x\} \) are related in one of the following ways:

\( w \notin R' \): In this case \( R = (R' \setminus L) \cup U \) for some pair of sets \( U \) and \( L \) such that:
(ii) \( R' \subseteq (R' \cap \{ x \in A_w : q_{w,x} < m_{w,x} \}) \setminus \{ s(F')_w \}; \)

The set \( Q \) is the union of exactly one arc of the form \( (x \rightarrow w) \) (from the \( m_{w,x} - q_{w,x} \) possibilities), for every \( x \in L \cup ((A_w \cap U(i')) \setminus U). \)

\( w \in R' \): In this case \( R = (R' \setminus (L \cup \{ w \})) \setminus (\{ w^* \} \cap U(i')), \) for a vertex \( w^* \) and sets \( L, U \) such that:

(I) \( w^* \in (X(i) \cap \{ x \in A_w : q_{w,x} > 0 \}) \setminus \{ x \in L(i') : s(F')_x = w \}; \)

(II) \( L \subseteq (R' \cap \{ x \in A_w : q_{w,x} < m_{w,x} \}) \setminus \{ w, w^*, s(F')_{w^*} \}; \)

(III) \( U \subseteq (U(i') \cap \{ x \in A_w : q_{w,x} = m_{w,x} \}). \)

The set \( Q \) is the union of exactly one arc of the form \( (x \rightarrow w) \) (from the \( m_{w,x} - q_{w,x} \) possibilities), for every \( x \in L \cup ((A_w \cap U(i')) \setminus (U \cup \{ w^* \})). \) together with one arc of the form \( (w \rightarrow w^*). \)

Proof: We prove this Lemma in two parts.

I: We first show the “if” of our claim. Assume that we have \( D' \subseteq D_t(i'), F' \subseteq FOR(D'), Q \subseteq D(i) \) and \( Q \) such that conditions (a)-(d) are satisfied. We will show that then \( (D' \cup Q, F' \cup Q) \) is an element of \( FOR(i) \) with charge vector \( c \) and root vector \( s \), and with root set \( R \) as described.

We proceed in two stages. We first prove that \( D' \cup Q \) is an orientation of \( D_t(i) \) with the claimed charge vector \( c \). We know that \( D' \) is an orientation on \( G_t(i') \) which is Eulerian at every vertex \( v \in V(i') \setminus X(i') \), and which has some charge \( c(D')_x \) for every \( x \in L(i') \) (and 0 charge at every \( x \in U(i') \)). By definition, the graph \( G_t(i) \) is equal to \( G_t(i') \) together with the set of edges \( E(i, w) = \{ e = (w, x) : x \in X(i) \} \). \( Q \) is an orientation on the set \( E(i, w) \). Therefore \( D' \cup Q \) is an orientation on the graph \( G_t(i) \).

Consider the charge induced by \( D' \cup Q \) on \( G_t(i) \), by considering 4 cases: \( V(i') \setminus X(i'), w, L(i) \) and \( U(i) \).

- The charge induced on any \( v \in V(i') \setminus X(i') \) is 0, because \( D' \) induces charge 0 on these vertices, and these vertices do not appear in \( E(i, w) \) (and hence \( Q \) induces no charge).

- The charge induced on \( w \) is \( c(D')_w + \sum_{x \in U(i') \cap A_w} (2q_{w,x} - m_{w,x}) \), where the \( D' \) contributes \( c(D')_w \), and the second expression is the contribution from \( Q \).

Under assumption (b), this evaluates to 0.

- Let \( x \in L(i) = L(i') \setminus \{ w \} \). These vertices belong to \( \Gamma_t(i) \), to \( D' \) and to \( Q \). The charge induced by \( D' \cup Q \) is \( c(D')_x + (m_{w,x} - 2q_{w,x}) \) if \( x \in A_w \), and \( c(D')_x \) otherwise. In both these cases assumption (c) implies an overall charge of \( c_x \), as required.

- The charge induced on any vertex of \( U(i) = U(i') \setminus A_w \) is 0. These vertices do not belong to \( D', Q \) or to \( G_t(i) \).

Note that \( V(i) \setminus X(i) = (V(i') \setminus X(i')) \cup \{ w \} \). Hence the overall orientation \( D' \cup Q \) is Eulerian at all vertices \( V(i) \setminus X(i) \). The charge vector \( c(D' \cup Q) \) has the value \( c_x \) at all \( x \in L(i) \), and 0 at all \( x \in U(i) \), as required. Hence \( D' \cup Q \subseteq D_t(i) \) with charge vector \( c \).

Now consider \( F' \cup Q \), where \( F' \subseteq FOR(D') \), \( Q \) is a subset of the arcs in \( Q \), and \( F', Q \) satisfy (d). We will show that under these circumstances \( F' \cup Q \) is a forest with the claimed root set \( R \) on \( D' \cup Q \). To show that \( F' \cup Q \) is a forest with root set \( R \), we must show:

(a) That no vertex \( x \in R \) has an outgoing arc in \( F' \cup Q; \)

(b) That every vertex \( v \in V(i) \setminus (U(i) \cup R) \) has exactly one outgoing arc in \( F' \cup Q; \)

(c) That \( F' \cup Q \) contains no directed cycle.
We will prove \( (\alpha) \)-\((\gamma) \) individually, first considering the \( w \not\in R' \) case, then the \( w \in R' \) case.

\((\alpha)\): Our goal is to show (in both the \( w \not\in R' \) case and the \( w \in R' \) case) that no vertex of \( R \) has an outgoing arc in \( F' \) or in \( Q \). By definition, \( F' \) contains outgoing arcs for every \( v \in V(i') \setminus (R' \cup U(i')) \). There is no outgoing arc for any \( x \in R' \cup U(i') \) in \( F' \). Note that regardless of whether \( R = (R' \setminus L) \cup U \) (\( w \not\in R' \) case) or \( R = (R' \setminus (L \cup \{w\})) \cup U \cup \{w* \} \cap U(i') \) (\( w \in R' \) case), we have \( R \subseteq R' \cup U(i') \). So there are no outgoing arcs for vertices of \( R \) in \( F' \). We now consider the arcs of \( Q \).

In the \( w \not\in R' \) case, we have \( R' = (R \setminus L) \cup U \) for \( L, U \) as specified in (d). The arcs of \( Q \) are of the form \((x \to w)\) for \( x \in L \cup ((U(i') \cap A_w) \setminus U) \). Note that \((R' \setminus L) \cup U \) and \( U \) each have an empty intersection with \( L \cup (U(i') \cap A_w) \setminus U \). Hence for \( w \not\in R' \), no vertex of \( R \) has an outgoing arc in \( F' \cup Q \).

In the \( w \in R' \) case, \( R \) is the disjoint union of \((R' \setminus (L \cup \{w\})) \cup U \) and \( \{w* \} \cap U(i') \). The set \( Q \) contains the arc \((w \to w*)\), together with an arc \((x \to w)\) for every \( x \in L \setminus ((U(i') \cap A_w) \setminus (U \cup \{w* \})) \). Note that \( w \not\in R \), hence we need not consider the arc \((w \to w*)\) further. Note next that \( L \) has an empty intersection with each of \((R' \setminus (L \cup \{w\})) \cup U \) and \( \{w* \} \cap U(i') \), and therefore \( L \cap R = \emptyset \). Finally, note that \((U(i') \cap A_w) \setminus (U \cup \{w* \}) \) also has an empty intersection with each of \((R' \setminus (L \cup \{w\})) \cup U \) and \( \{w* \} \cap U(i') \). Therefore no arc of \( Q \) is outgoing from a vertex of \( R \). Therefore in the case of \( w \in R' \), no vertex of \( R \) has an outgoing arc in \( F' \cup Q \), and \( (\alpha) \) holds.

\((\beta)\): We must show that every vertex \( v \in V(i) \setminus (U(i) \cup R) \) has exactly one outgoing arc in \( F' \cup Q \). We first note that \( V(i') = V(i) \). For every \( v \in V(i') \setminus X(i') \), we know that \( v \) has exactly one outgoing arc in \( F' \). Also, if \( v \in V(i') \cap X(i') \), there is no outgoing arc from \( v \in Q \) (since \( v \neq w \), and \( v \not\in A_w \cap X(i') \)). Hence every \( v \in V(i') \setminus X(i') \) has exactly one outgoing arc in \( F' \cup Q \), as required.

We now show that every \( x \in X(i') \setminus (U(i) \cup R) \) has exactly one outgoing arc in \( F' \cup Q \). First observe that \( U(i) = U(i') \cap \overline{A_w} \).

First consider the case \( w \not\in R' \). In this case \( R = (R' \setminus L) \cup U \). We now partition \( X(i') \) into six sets as follows:

\[
R' \cap L, \ R' \setminus L, \ L(i') \setminus R', \ U(i') \cap \overline{A_w}, \ U, \ (U(i') \cap A_w) \setminus U.
\]

Then \( X(i') \setminus (U(i) \cup R) \) is the union of the three disjoint sets \((R' \cap L), L(i') \setminus R' \) (which includes \( w \)) and \((U(i') \cap A_w) \setminus U \). We will show that every vertex in these sets has exactly one outgoing arc in \( F' \cup Q \). For \( x \in R' \cap L \), we know that \( x \) has no outgoing arc in \( F' \). However by construction, \( Q \) contains exactly one arc \((x \to w)\). For \( x \in L(i') \setminus R' \), \( F' \) contains an outgoing arc for \( x \); however, there is no arc leaving \( x \in Q \), so again \( x \) has exactly one outgoing arc in \( F' \cup Q \). Finally, for \( x \in (U(i') \cap A_w) \setminus U \), \( x \) is not in \( G_x(i') \) and therefore has no outgoing arc in \( F' \); however, by construction, \( Q \) contains exactly one arc of the form \((x \to w)\) leaving \( x \). So in all three cases, there is one outgoing arc for \( x \) in \( F' \cup Q \), as required. Hence \( (\beta) \) holds in the \( w \not\in R' \) case.

Next consider the case \( w \in R' \). In this case we have \( R = (R' \setminus (L \cup \{w\})) \cup U \cup \{w* \} \cap U(i') \). We partition \( X(i') \) into eight sets in this case:

\[
\{w\}, \ R' \cap L, \ R' \setminus (L \cup \{w\}), \ L(i') \setminus R', \ U(i') \cap \overline{A_w}, \ U, \ \{w* \} \cap U(i'), \ (U(i') \cap A_w) \setminus (U \cup \{w* \}).
\]

Then by definition of \( R \), \( x \in X(i') \setminus (R \cup U(i)) \) if and only if \( x \) belongs to one of \( \{w\}, R' \cap L, L(i') \setminus R' \) and \((U(i') \cap A_w) \setminus (U \cup \{w* \}) \). We show that every vertex in each of these four sets has exactly one outgoing arc. For the vertex \( w \), \( F' \cup Q \) contains no outgoing arc from \( w \) (as \( w \in R' \)), but \( Q \) contains one arc of the form \((w \to w*)\), hence \( F' \cup Q \) has exactly one outgoing arc from \( w \). Let \( x \in R' \cap L \). In this case, by \( x \in R' \), we know that \( x \) has no outgoing arc in \( R' \); also, by \( x \in L \), we know that \( Q \) contains exactly one arc leaving \( x \) (an arc of the form \((x \to w)\)). Hence for \( x \in R' \cap L \), \( F' \cup Q \) contains exactly one arc leaving \( x \). Now suppose \( x \in L(i') \setminus R' \). In this case \( F' \) already contained one outgoing arc from \( x \). However,
by $x \in L(i') \setminus R'$ we know $x \not\in L$ and $x \not\in U(i')$, hence none of the arcs of $Q$ is outgoing from $x$. So $F' \cup Q$ contains exactly one arc leaving $x$ for $x \in L(i') \setminus R'$. Finally assume $x \in (U(i') \cap A_w) \setminus (U \cup \{w^*\})$. By $x \in U(i')$, we know such an $x$ will have no outgoing arc in $F'$. By definition of $eQ$, there is exactly one arc of the form $(x \to w)$ in $Q$ for such an $x$. So again, there is one outgoing arc in $F' \cup Q$ for every $x \in (U(i') \cap A_w) \setminus (U \cup \{w^*\})$. Hence $(\beta)$ holds in the case of $w \in R'$.

$(\gamma)$: Next we show that there is no simple directed cycle in $F' \cup Q$ (together with $\alpha$ and $\beta$, this will imply the non-existence of any cycle in the undirected image of $F' \cup Q$). By our assumption that $F'$ is a forest on $D'$, there can be no directed cycle in $F'$. Therefore any simple directed cycle that could exist in $F' \cup Q$ would need to contain at least one arc from $Q$.

We will treat the $w \not\in R'$ and $w \in R'$ cases separately. One observation which we will use repeatedly is the following - if $x \in L(i') \setminus R'$, there is exactly one $z \in R'$ such that there is a path from $x$ to $z$ in $F'$. This is because there existed $z, z', z \neq z'$ satisfying this condition, this would imply a directed path in $F'$ between $z$ and $z'$, where both $z$ and $z'$ are in $R'$ (and hence neither has a outgoing arc in $F'$). Note the unique $z$ is $z = s(F')_x$.

We consider the $w \not\in R'$ case first. In this case, every arc of $Q$ is of the form $(x \to w)$ for some $x \in X(i)$. Therefore if a simple directed cycle exists in $F' \cup Q$, then it must contain exactly one $Q$ arc. Also, since no vertices of $U(i') \cup Q$ belong to $F'$, the arc must be $(x^* \to w)$ for some $x^* \in L$. Consider such a hypothetical cycle consisting of $(x^* \to w)$, together with a directed path $p$ from $w$ to $x^*$ lying entirely in $F'$. Moreover, since $w \in L(i') \setminus R'$ and $x^* \in L \subseteq R'$, by our observation above we must have $x^* = s(F')_w$. Now observe that (d)(i) excludes $s(F')_w$ from being a member of $L$, therefore there can be no arc from $x^* = s(F')_w$ to $w$ in $Q$. This proves that for the $w \not\in R'$ case, there can be no simple directed cycle in $F' \cup Q$.

We now consider the case of $w \in R'$. In this case, $Q$ consists of one arc $(w \to w^*)$, together with one arc of the form $(x \to w)$ for every $x \in L \cup ((U(i') \cap A_w) \setminus (U \cup \{w^*\}))$. A simple directed cycle may visit $w$ at most once, hence a simple directed cycle may either contain exactly one $Q$ arc (either $(w \to w^*)$ or one of the $(x \to w)$ arcs) or exactly two $Q$ arcs, where in the latter case this must be one of the $(x \to w)$ arcs followed immediately in the cycle by $(w \to w^*)$. We consider each of these cases in turn. First consider a hypothetical cycle consisting of the arc $(w \to w^*)$ and a path $p$ in $F'$ from $w^*$ to $w$. By existence of an outgoing path from $w^*$ in $F'$, we can deduce that $w^* \in L(i') \setminus R'$. We know $w \in R'$. Then by our observation, we must have $w = s(F')_{w^*}$. Now recall that (d)(I) specifies that $w^*$ cannot be any vertex which lies in the subtree of $w$ in $F'$. So we have a contradiction for the case of a cycle containing $(w \to w^*)$ and no other $Q$ arcs. Next consider a hypothetical cycle consisting of one arc of the form $(x \to w)$ from $Q$ and a path in $F'$ from $w$ to $x$. Observe that the existence of a path leaving $w$ in $F'$ would imply that $w$ must be an element of $L(i') \setminus R'$, in direct contradiction to the fact that $w \in R'$. Hence there is no simple directed cycle in $F' \cup Q$ containing exactly one $Q$ arc. Consider the final possibility for a simple cycle in $F' \cup Q$, where we have $(x \to w)$ (from $Q$) for some $x$ followed directly by the arc $(w \to w^*)$, and then by a path $p$ in $F'$ from $w^*$ to $x$. Note that for such a path to exist in $F'$, given that $x, w^* \in X(i')$, we must have $w^* \in L(i') \setminus R'$ and $x \in R', x = s(F')_{w^*}$. Now recall that by $x \in R'$, we know $x \in L(i')$, and therefore the arc $(x \to w)$ of $Q$ is from $x \in L$. However, (d)(II) specifically states that $s(F')_{w^*}$ is not an element of $L$. Hence we have a contradiction. So in all three possible subcases of $w \in R'$ we have shown that a cycle is impossible in $F' \cup Q$.

only if: It is also true that given an orientation $D \in D_4(i)$ and a forest $F \in FOR(D)$ with root set $R$, charge vector $c \in C(i)$ and root vector $s \in S(i)$, that conditions (a)-(d) are satisfied. Note this is the easier direction of the proof.

We now apply Theorem 9 to the calculation of $\psi(i, c, s)$ for $c \in C(i), s \in S(i)$ in the forget case. We know that if $(D, F)$ is in $FOR(i)$ with charge vector $c$ and root vector $s$ if and only if all of conditions (a)-(d) hold for $(D', F') = (D(i'), F(i'))$ and $(Q, Q) = (D(i'), F(i'))$. We now make some observations concerning
conditions (a)-(d):

**Observation 10** Let \( G = (V, E) \) be an Eulerian multigraph with tree decomposition \( \{X_i \mid i \in I\}, T = (I, F) \), and let \( i \in I \) be a forget node such that \( X(i) = X(i') \setminus \{w\} \) for some \( w \in X(i') \), where \( i' \) is the single child of \( i \).

Consider the task of counting pairs \( (D, F) \in \text{FOR}(i) \) with charge vector \( c \in C(i) \) and root vector \( s \in S(i) \). For any \( (D', F') \in \text{FOR}(i') \), every orientation \( Q \) on \( E(i', w) \), and every subset \( Q \subseteq Q' \) with the induced values \( q_{w,x}, x \in A_w \cap X(i) \), conditions (b)-(d) can be expressed solely in terms of \( c, s, c' = c(D') \), \( s' = s(F') \), the edge counts \( m_{w,x} = |E_{w,x}| \) for \( x \in X(i) \), the out-of-\( w \) edge counts \( q_{w,x} = |Q_{w,x}| \) for \( x \in X(i) \), and finally, the collection of arcs \( Q \).

**Proof:** That this is true is immediately clear for conditions (b) and (c), which are describe in terms of \( c \) and \( c' \).

Condition (d) takes more consideration. First observe that we can test whether \( w \in R' \) or \( w \notin R' \) (and identify which set of tests need to be carried out) by checking whether \( s(F')_w \) is equal to \( w \) or not. Also note that we already know the sets of vertices \( L(i'), U(i'), A_w \), \( \text{L}(i) = (L(i') \setminus \{w\}) \cup U(i') \cap A_w \), \( U(i) = U(i') \setminus A_w \), and \( X(i) = L(i) \cup U(i) \), as these can be determined from the tree decomposition of \( G \).

Suppose first that we are considering the case of \( w \notin R' \), hence we need to check (i), (ii) and also the details for \( Q \). We have the two root vectors \( s \in S(i) \) and \( s \in S(i') \); therefore from these root vectors we can identify \( R = \{ x \in L(i) : s_x = x \} \) and \( R' = \{ x \in L(i') : s_x = x \} \). Given the relationship that exists between \( R \) and \( R' \), we must have \( U = R \setminus R' \) and \( L = R' \setminus R \). Now we can check that (i) holds for \( L \) in polynomial-time, by taking the intersection of \( R' \) and \( \{ x \in A_w : q_{w,x} > 0 \} \) and excluding \( s'_w \) from this set. We can check that (ii) holds by calculating the set \( U(i') \cap A_w \) and checking that \( U \) is contained in its set; then calculating the set \( U(i') \cap \{ x \in A_w : q_{w,x} = m_{w,x} \} \) and checking that this set is contained in \( U \). Finally, if (i) and (ii) have been passed, we check that \( Q \) is the union of a single arc \((x \rightarrow w)\) for every \( x \in L \cup \{(A_w \cap U(i')) \setminus U\} \) by examining \( Q \) directly.

Next suppose we are considering the case of \( w \notin R' \), so need to check conditions (I)-(III) and also check which arcs lie in \( Q \). First note again that we can calculate \( R' \) from \( s' \) and \( R \) from \( s \). To check (I), we first identify the vertex \( w^* \) (this will be the target of the only arc outgoing from \( w \) in \( Q \)). We then compute the sets \( X(i) \cap \{ x \in A_w : q_{w,x} > 0 \} \) and \( \{ x \in L(i') : s'_x = w \} \). Then we check that \( w^* \) is in the first set, but not the second. Next we determine \( L \) and \( U \). If we take \( R \setminus R' \), this evaluates to \( U \cup \{ w^* \} \cap U(i') \). We already know the vertex \( w^* \), and whether it belongs to \( U(i') \) or not, therefore, we can recover the set \( U \) by deleting \( w^* \) is necessary. If we take \( R \setminus R' \) this evaluates to \( U \cup \{ w \} \). Excluding \( w \) gives us \( L \). To check condition (II), we calculate \( R' \cap \{ x \in A_w : q_{w,x} < m_{w,x} \} \) and exclude any of the vertices \( w, w^*, s^*_w \) which appear in this set. Then we check that every vertex of \( L \) appears in the computed set. The test (III) is exactly the same as test (ii) of the case \( w \notin R' \), and we evaluate it in exactly the same way. Finally we check that \( Q \) contains the necessary arcs by checking that it contains exactly the set of arcs described.

We now discuss how to compute the table \( \Psi(i) \).

We start by initialising the value \( \psi(i, c, s) \) to 0, for every \( c \in C(i) \) and \( s \in S(i) \).

Next we iterate through the table \( \Psi(i') \) one entry at a time, using the value \( \psi(i', c', s') \) (in conjunction with all possible orientations \( Q \) of \( E(i, w) \), and all relevant sets \( Q \), to increase the value of \( \psi(i, c, s) \) for any values of \( c, s \) which satisfy Theorem 9 (in conjunction with \( Q, Q' \) and in relation to \( c', s' \)). For each \( c', s' \), we perform the following steps:

(i) We check whether \( s'_w \) is \( w \) (ie, whether \( w \in R' \)) or otherwise \( w \notin R' \).
(ii) We consider each vector \( q \in \prod_{x \in A_w \cap X(i)} \{0, \ldots, m_{w,x} \} \) such that
\[
c'_w = \sum_{x \in A_w \cap X(i)} (m_{w,x} - 2q_{w,x})
\]
in turn, and compute the weight
\[
\chi(q) = \text{def} \prod_{x \in A_w \cap X(i)} \left( \frac{m_{w,x}}{q_{w,x}} \right),
\]
which is the number of different orientations of the edges of \( E(i, w) \) which have exactly \( q_{w,x} \) of the \( E_{w,x} \) edges oriented away from \( w \). For each vector \( q \), we define \( c^* \), the charge vector of \( F' \cup Q \) for any orientation \( Q \) consistent with \( q \), to be
\[
c^*_x = \begin{cases} 
  c'_x & \text{if } x \in X(i) \cap \overline{A_w} \\
  c'_x + (m_{w,x} - 2q_{w,x}) & \text{if } x \in X(i) \cap A_w.
\end{cases}
\]

(iii) We now have two cases, depending on whether \( w \in R' \) or not.

\( w \not\in R' \):

(a) We first consider each set \( L \subset (R' \cap \{ x \in A_w : q_{w,x} < m_{w,x} \}) \setminus \{ s'_w \} \) in turn; and also consider each subset \( U \) such that \( U \subseteq U(i') \cap A_w \) and \( U \supseteq U(i') \cap \{ x \in A_w : q_{w,x} = m_{w,x} \} \) in turn. Note that by \( |R'| \leq k \), \( s'_w \in R' \) and \( |U(i')| \leq (k - 1) \), we know there are at most \( 2^{k-1} \) possible pairs of sets \( L, U \) to be considered, which is constant (since the treewidth \( k \) is constant).

By Theorem\[9\] recall that in order for \( F' \cup Q \) to be a forest on the orientation \( D \cup U \) of \( G_\ell(i) \), that \( Q \) must be the union of exactly one arc \( (x \rightarrow w) \) for every \( x \in L \cup (U(i') \cap A_w) \setminus U \).

The number of ways we can choose these arcs is
\[
\kappa(L, U, q) = \prod_{x \in L} (m_{w,x} - q_{w,x}) \prod_{x \in (U(i') \cap A_w) \setminus U} (m_{w,x} - q_{w,x}).
\]

Observe that by our conditions on \( L \) and \( U \), we know that for every \( x \in L \) and every \( x \in (U(i') \cap A_w) \setminus U \), that \( m_{w,x} - q_{w,x} \), the number of arcs from \( x \) to \( w \), is non-zero.

(b) We now define the root vector of \( F' \cup Q \), for any \( Q \) which induces the vector \( q \). This will be \( s^* = s(q, L, U) \), defined as
\[
s^*_x = \begin{cases} 
  s'_w & \text{if } x \in L(i') \setminus \{ w \}, s'_x \in L \\
  s'_x & \text{if } x \in L(i') \setminus \{ w \}, s'_x \not\in L \\
  s'_w & \text{if } x \in (U(i') \cap A_w) \setminus U \\
  x & \text{if } x \in (U(i') \cap A_w) \cup U
\end{cases}
\]

(c) Finally, we add the value \( \chi(q) \times \kappa(L, U, q) \times \psi(i', c^*, s^*) \) to the table entry for \( \psi(i, c^*, s^*) \).

\( w \in R' \):

(a) We consider every possible \( w^* \) in the set \( (X(i) \cap \{ x \in A_w : q_{w,x} > 0 \}) \setminus \{ x \in L(i') : s(F')_x = w \} \) in turn. Note that for certain orientations \( Q \), the set of potential \( w^* \) vertices may be empty.

In these cases, we skip part (iii) and try another \( q \) vector (as described in (ii)).
Conditional on this $w^*$, we consider every possible $L \subset (R' \cap \{x \in A_w : q_{w,x} < m_{w,x}\}) \setminus \{w, w^*, s^*_w\}$ in turn; and also consider each subset $U$ such that $U \subseteq U(i') \cap A_w$ and $U \supseteq U(i') \cap \{x \in A_w : q_{w,x} = m_{w,x}\}$ in turn. By $|X(i)| \leq k$, there are at most $k-1$ possible values for $w^*$. For each particular $w^*$, there are at most $2^{k-1}$ possible pairs of sets $L, U$ to be considered, which is constant. So we will consider at most $(k-1)^2 2^{k-1} = (k-1)2^{k-1}$ triples $(w^*, L, U)$.

By Theorem 9 in the $w \in R'$ case, recall that for $F' \cup Q$ to be a forest on the orientation $D \cup Q$ of $G_E(i)$, that $Q$ must be the union of one arc of the form $(w \rightarrow w^*)$, together with exactly one arc $(x \rightarrow w)$ for every $x \in L \cup ((U(i') \cap A_w) \setminus (U \cup \{w^*\}))$. The number of ways we can choose these arcs is

$$\kappa(w^*, L, U, q) = q_{w,w^*} \prod_{x \in L} (m_{w,x} - q_{w,x}) \cdot \prod_{x \in (U(i') \cap A_w) \setminus (U \cup \{w^*\})} (m_{w,x} - q_{w,x}).$$

Observe that by our conditions on $L$ and $U$, we know that for every $x \in L$ and every $x \in (U(i') \cap A_w) \setminus (U \cup \{w^*\})$, that $m_{w,x} - q_{w,x}$, the number of arcs from $x$ to $w$, is non-zero.

(b) Next we compute the root vector $s^*_x$ of $F' \cup Q$, for the current $Q$:

$$s^*_x = \begin{cases} 
    s^*_w \quad &\text{if } x \in L(i') \setminus \{w\}, s'_x \in L \cup \{w\} \\
    s'_x \quad &\text{if } x \in L(i') \setminus \{w\}, s'_x \notin L \cup \{w\} \\
    s^*_w \quad &\text{if } x \in (U(i') \cap A_w) \setminus U \\
    x \quad &\text{if } x \in (U(i') \cap \overline{A_w}) \cup U
\end{cases}.$$

(c) Finally, we add the value $\chi(q) \times \kappa(w^*, L, U, q) \times \psi(i', c', s')$ to the table entry for $\psi(i, c^*, s^*)$.

### 2.1.4 Join

In the case of a join node $j$, we know that $j$ has two child nodes $i$ and $i'$, and that $X(j) = X(i) = X(i')$. Observe that $V(j) = V(i) \cup V(i')$. Also note that by the rules of a join for a nice tree decomposition, that $V(j) \setminus X(j)$ is the disjoint union of $(V(i) \setminus X(i))$ and $(V(i') \setminus X(i'))$. Also, $G$ does not contain any edges connecting vertices of $V(i) \setminus X(i)$ with vertices of $V(i') \setminus X(i')$. Therefore the graph $G_E(j)$ is the disjoint union of the graphs $G_E(i)$ and $G_E(i')$.

Observe now that the charge vectors in $C(j)$ have the same length and are indexed by the same set of vertices $X(j)$, as the charge vectors of $C(i)$ and of $C(i')$. Also, the root vectors in $S(j)$ have the same length and are indexed by the same set of vertices $X(j)$, as the root vectors of $S(i)$ and of $S(i')$. We now have the following observation about the decomposition of any forest $\Orb(D, F) \in \Orb(j)$:

**Observation 11** Let $j$ be a join-node of the nice tree decomposition $(\{X(i) \mid i \in I\}, T = (I, F))$ of the Eulerian multi-graph $G$, and let $i$ and $i'$ be the child nodes of $j$. Then $(D, F)$ is a forest $\Orb(j)$ with charge vector $c$ and root vector $s$ if and only if $D$ is the disjoint union of $\overline{D} \in D(i)$ and $D' \in D(i')$, and $F$ is the disjoint union of $\overline{F} \in \Orb(D)$ and $F' \in \Orb(D')$ such that

(a) $c_x = c(\overline{D})_x + c(D')_x$ for all $x \in C(j)$;

(b) For every $L(i) \cap L(i')$, at least one of $s(\overline{F})_x = x$ and $s(F')_x = x$ holds;

(c) For every $x \in L(i) \setminus \overline{R}, y \in L(i') \setminus R'$ either $s(\overline{F})_x \neq y$ or $s(F')_y \neq x$.

14
(d) If we let \( \hat{s} \) denote the root vector of \( \hat{F} \) and \( F' \) denote the root vector of \( F' \), then \( s \) satisfies the following:

\[
s_x = \begin{cases} 
\hat{s}_x & \text{if } x \in L(i), \hat{s}_x \in L(i) \setminus L(i') \\
\hat{s}'_x & \text{if } x \in L(i), \hat{s}_x \in L(i) \cap L(i') \\
\hat{s}'_x & \text{if } x \in L(i'), \hat{s}'_x \in L(i') \setminus L(i) \\
\hat{s}_x & \text{if } x \in L(i'), \hat{s}_x \in L(i) \cap L(i') \\
x & \text{if } x \in X(j) \setminus (L(i) \cup L(i'))
\end{cases}
\]

Proof: Condition (a) is trivial.

For the forest conditions, it is not difficult to check that if \( F \in \text{FOR}(j) \), then all of (b), (c), (d) hold.

To prove that (b), (c), (d) imply that \( \hat{F} \cup F' \) is a forest (with the root vector \( s \)) on \( \hat{D} \cup D' \), note that properties (\( \alpha \)) and (\( \beta \)) for a forest follow easily from the fact that \( \hat{F} \) and \( F' \) are forests, and from (b). Checking (\( \gamma \)) takes a little bit more work, but is implied by (c).

We now describe how to fill table \( \Psi(j) \) when \( j \) is a join node. First, for every \( c \in C(j), s \in S(j) \), we initialise \( \psi(j,c,s) \) to 0. Next we iterate through the table \( \Psi(i) \) one entry at a time, using the value \( \psi(i,\hat{c},\hat{s}) \) in conjunction with the table \( \Psi(i') \) to increase the value of \( \psi(j,c,s) \) entries. For each \( \hat{c}, \hat{s} \), we perform the following steps.

(i) We compute the following sets using \( G \) and \( \hat{s} \):

\[
L(i) = \{ x \in X(i) : A_x \cap (V(i) \setminus X(i)) \neq \emptyset \}, \quad R = \{ x \in L(i) : \hat{s}_x = x \}.
\]

(ii) We consider each index \( (i', c', s') \) of \( \Psi(i') \) in turn.

- We define the charge vector \( c^* = \hat{c} + c' \).
- We compute the sets \( L(i') = \{ x \in X(i') : A_x \cap (V(i') \setminus X(i')) \neq \emptyset \}, \quad R' = \{ x \in L(i') : \hat{s}'_x = x \} \).

(iii) If properties (b) and (e) hold for \( \hat{s}, s' \) then we compute \( s^* \) as

\[
s^*_x = \begin{cases} 
\hat{s}_x & \text{if } x \in L(i), \hat{s}_x \in L(i) \setminus L(i') \\
\hat{s}'_x & \text{if } x \in L(i), \hat{s}_x \in L(i) \cap L(i') \\
\hat{s}'_x & \text{if } x \in L(i'), \hat{s}'_x \in L(i') \setminus L(i) \\
\hat{s}_x & \text{if } x \in L(i'), \hat{s}_x \in L(i) \cap L(i') \\
x & \text{if } x \in X(j) \setminus (L(i) \cup L(i'))
\end{cases}
\]

and add a value of 1 to the current value for \( \psi(j, c^*, s^*) \), otherwise we do nothing.

Then we return to (ii) and consider a new index of \( \Psi(i') \).

Observe that for the join computation, the issue of whether \( G \) was a simple graph or a multi-graph is not relevant (except in bounding the size of the table).

References

[1] T. van Aardenne-Ehrenfest and N.G. de Bruijn, Circuits and trees in oriented linear graphs, Simon Stevin (= Bull. Belgian Math. Soc.), 28, pp. 203–217, 1951.
[2] H.L. Bodlaender and A.M.C.A. Koster, Combinatorial Optimization on Graphs of Bounded Treewidth, *The Computer Journal, 51* (3), pp. 255–269, 2008.

[3] C.W. Borchardt, Über eine der Interpolation entsprechende Darstellung der Eliminations-Resultante, *J. reine angew. Math.*, 57, pp. 111–121, 1860.

[4] G. Brightwell and P. Winkler, Counting Eulerian circuits is $\#P$-complete, *Proceedings of the Second Workshop on Analytic Algorithmics and Combinatorics (ANALCO 2005)*, pp. 259–262, 2005.

[5] P. Chebolu, M. Cryan and R. Martin, Exact counting of Euler Tours for generalized series-parallel graphs. *Journal of Discrete Algorithms, 10*, pp. 110–122, January 2012.

[6] P. Creed, *Counting and Sampling Problems on Eulerian Graphs*, submitted PhD thesis, University of Edinburgh, 2009.

[7] T. Kloks, Treewidth, Computations and Approximations. *Lecture Notes in Computer Science, 842*, Springer-Verlag, 1994.

[8] J.A. Makowsky and J.P. Mariño, Farrell Polynomials on Graphs of Bounded Treewidth. *Advances in Applied Mathematics, 30*, pp. 160-176, 2003 (Special Issue of FPSAC ’01 papers).

[9] S. Noble, Evaluating the Tutte polynomial for graphs of bounded tree-width, *Combinatorics, Probability and Computing, 7*, pp. 303-323, 1998.

[10] C.A.B. Smith and W.T. Tutte, On unicursal paths in networks of degree 4. *American Mathematical Monthly, 48*:233-237, 1941.