Available online at www.sciencedirect.com

ScienceDirect

Electronic Notes in Theoretical Computer Science

Electronic Notes in Theoretical Computer Science 346 (2019) 655–666

www.elsevier.com/locate/entcs

# FPT Algorithms to Enumerate and Count Acyclic and Totally Cyclic Orientations

Farley Soares Oliveira[1] Hidefumi Hiraishi[2] Hiroshi Imai[3]

*Department of Computer Science*
*The University of Tokyo*
*Tokyo, Japan*

**Abstract**

In this paper, we deal with counting and enumerating problems for two types of graph orientations: acyclic and totally cyclic orientations. Counting is known to be #P-hard for both of them. To circumvent this issue, we propose Fixed Parameter Tractable (FPT) algorithms. For the enumeration task, we construct a Binary Decision Diagram (BDD) to represent all orientations of the two kinds, instead of explicitly enumerating them. We prove that the running time of this construction is bounded by $O^*(2^{\mathrm{pw}^2/4+o(\mathrm{pw}^2)})$ with respect to the pathwidth pw. We then develop faster FPT algorithms to count acyclic and totally acyclic orientations, running in $O^*(2^{\mathrm{bw}^2/2+o(\mathrm{bw}^2)})$ time, where bw denotes the branch-width of the given graph. These counting algorithms are obtained by applying the observations in our enumerating algorithm to branch decomposition.

*Keywords:* Acyclic Orientations, Totally Cyclic Orientations, Parameterized Algorithms, FPT Algorithms, Path-width, Branch-width, Binary Decision Diagram

## 1 Introduction

A graph orientation of an undirected graph $G = (V, E)$ is a directed graph obtained by orienting all edges of $G$. In this paper, we design enumerating and counting algorithms parameterized by *path-width* and *branch-width* for acyclic and totally cyclic orientations. An orientation is said to be *acyclic* if it is a directed acyclic graph, and *totally cyclic* if each of its connected components is strongly connected. It is well-know that, for planar graphs, these two orientations are dual in the sense that there is a bijection between acyclic orientations of a planar graph and totally cyclic orientation of its dual graph.

[1] Email:diveira@is.s.u-tokyo.ac.jp
[2] Email:hiraishi1729@is.s.u-tokyo.ac.jp
[3] Email:imai@is.s.u-tokyo.ac.jp

https://doi.org/10.1016/j.entcs.2019.08.057

Acyclic and totally cyclic orientations are related to various objects in combinatorics and geometry. The most well-known relationship is the one between graph coloring and acyclic orientations. The number of acyclic orientations of $G$ equals $(-1)^n\chi_G(-1)$, where $\chi_G$ denotes its chromatic polynomial [26]. Due to the duality between coloring and nowhere-zero flows, the number of totally cyclic orientations can be obtained by evaluating the flow polynomial at $\chi^*_G(-1)$ [17]. This duality between the chromatic and flow polynomials (or still, between acyclic and totally cyclic orientations) was synthesized into the concept of the Tutte polynomial [7], and naturally extends to geometrical objects such as hyperplane arrangements, point configurations, polytopes, etc., in the lens of oriented matroids [16]. Generalized from the chromatic polynomial, the Tutte polynomial is a matroid-invariant bivariate polynomial $T(x, y)$ which encodes a wide range of information about matroids beyond graphs. In particular, the number of acyclic and totally cyclic orientations of a graph is obtained by evaluating $T(2, 0)$ [26] and $T(0, 2)$ [17], respectively.

It is conjectured that the spanning forests of a graph are closely related to these two types of orientations. Namely, the *Merino-Welsh conjecture* states that, for any given undirected graph without loops and bridges, the number of spanning forests is at most the maximum of the numbers of acyclic orientations and totally cyclic orientations; equivalently, $T(1, 1) \leq \max\{T(2, 0), T(0, 2)\}$. This inequality is also conjectured to hold for Tutte polynomials of matroids. There are partial results for several subclasses of graphs and matroids [27,18,20,8].

Acyclic and totally cyclic orientations are also intriguingly connected to hyperplane arrangements, an object well-studied in the fields of combinatorics and geometry. Given a graph $G = (V, E)$, the set of hyperplanes $x_i = x_j$ for each $\{i, j\} \in E$ is called *graphic arrangement*; its dual entity given from matroid duality, *cographic arrangement*. There is a bijection between the regions of the space divided by any given graphic arrangement and the acyclic orientations of the corresponding graph [13].

While one can find these two orientations of a given undirected graph in linear time, counting and enumerating them turn out to be #P-hard even for bipartite graphs [28]. One of the promising approaches for intractable problems is to design *Fixed Parameter Tractable*, or FPT, algorithms. An algorithm is FPT with respect to a structural parameter $k$ if it runs in $O(f(k)\,\mathrm{poly}(N))$ time, where $f$ is a computable function and $N$ is the input size. For the problems of counting acyclic and totally cyclic orientations, all of the known FPT algorithms can be obtained by evaluating the Tutte polynomial.

The canonical way of explicitly enumerating objects in graph theory and geometry is performing *reverse search*. In [1], reverse search techniques were developed for a wide variety of objects, including cells of hyperplane arrangements, and they can thus be extended for enumerating acyclic and totally cyclic orientations of graphs. There are also enumeration algorithms specialized in acyclic orientations [25,3,9].

In this paper, we design parameterized algorithms for implicitly enumerating acyclic and totally cyclic orientations of graphs using *Binary Decision Diagrams* (abbr. BDD). BDDs were originally devised by Bryant [6] to efficiently compress

Boolean functions. They allow one to perform Boolean operations on the compressed data without decompressing it. This outstanding virtue of BDDs has yielded a stream of BDD-based approaches to combinatorial optimization and enumeration problems. For example, an algorithm for computing the Tutte polynomial was developed by constructing the BDD associated with spanning forests in a top-down manner [23]. BDD-based algorithms were also developed to solve graph optimization and enumeration problems such as max clique, graph coloring, graph covering [11] and path enumeration [15].

We propose FPT algorithms with respect to path width pw for constructing BDDs of all acyclic and totally cyclic orientations, resulting in a $O^*(2^{\mathrm{pw}^2/4+o(\mathrm{pw}^2)})$ running time, where the notation $O^*$ ignores polynomial factors in the number of edges. To the best of our knowledge, the currently fastest FPT algorithms for counting acyclic orientations run in $O^*(2^{\mathrm{tw}^2+o(\mathrm{tw}^2)})$ time with respect to the tree-width tw of the graph [19], and in $O^*(2^{\mathrm{pw}\log(\mathrm{pw})})$ time with respect to the path-width pw of the graph [23]. Note that the inequality $\mathrm{tw} \leq \mathrm{pw}$ always holds, so the time complexity of these algorithms is lower than the one of the algorithm provided here. However, the BDD constructed by our algorithm has several uses beyond counting. It can be used for, among other things, enumerating the acyclic and totally cyclic orientations of the graph and to sample these orientations uniformly at random. It can also be combined with BDDs of other Boolean functions defined over the same domain, in terms of Boolean operations. Once the BDD is constructed, all the acyclic (resp. totally cyclic) orientations can be enumerated in $O(mN)$ time, where $N$ denotes the total number of acyclic (resp. totally cyclic) orientations. The enumeration of acyclic orientations has been applied to resource management in distributed scheduling mechanisms [2,3]. On the other hand, the enumeration of totally cyclic orientations has been applied to finding feasible directions for the one-way street problem in road networks [10].

For the counting task, we provide FPT algorithms which count the number of acyclic and totally cyclic orientations using branch decomposition and branch-width. These concepts were originally introduced in [22], where the branch-width bw of a graph was shown to be linearly correlated with its tree-width tw, a more standard graph width in parameterized algorithm design for graph problems, as follows: $\lceil 2\,\mathrm{tw}/3 \rceil + 1 \leq \mathrm{bw} \leq \mathrm{tw} + 1$. Our counting algorithms for acyclic and totally cyclic orientations run in $O^*(2^{\mathrm{bw}^2/2+o(\mathrm{bw}^2)})$ time, faster than the $O^*(2^{\mathrm{tw}^2+o(\mathrm{tw}^2)})$ of the aforementioned algorithm in [19] which evaluates the Tutte polynomial. Finally we mention that branch decomposition can be exploited to design parameterized counting algorithms for other types of graph orientations. The FPT algorithm with respect to carving decomposition in [24] can be naturally converted into an XP algorithm with respect to branch decomposition, where an algorithm is said to be XP if it runs in $O^*(N^{f(k)})$ time for some computable function $f$ and input size $N$.

## 2 Enumerating Acyclic and Totally Cyclic Orientations

Unless otherwise stated, all the graphs in this section are finite and simple. For a graph $G$, we denote its set of vertices by $V(G)$ and its set of edges by $E(G)$. Furthermore, we define $n(G) := \mathrm{card}(V(G))$ and $m(G) := \mathrm{card}(E(G))$. In case the graph $G$ is clear from the context, we simply write $V$, $E$, $n$ and $m$. A *partial orientation* of a graph is a partially directed graph obtained by orienting some of its edges. More formally, a partial orientation $P$ of $G$ is the edge-disjoint union of a subgraph $G' = (V, E')$, where $E' \subseteq E$, and a digraph $D = (V, A)$, where $A$ is a set of directed edges obtained by substituting each edge $\{u, v\}$ of $E \setminus E'$ by either the directed edge $(u, v)$ or $(v, u)$. For an arbitrary partial orientation $P$ of $G$, we write $E(P) := E'$ and $A(P) := A$. Note that graphs and directed graphs are partial orientations in particular. For $S \subseteq E(P) \cup A(P)$, we denote by $G[S]$ the partial orientation induced by $S$, and we say that $v \in V$ is adjacent to $S$ if it is adjacent to at least one element of $S$.

In this section, we propose algorithms to to construct BDDs which can be used for, among other things, enumerate all the acyclic and totally cyclic orientations of a given graph $G$. A *binary decision diagram* (BDD) is an acyclic graph used to compactly represent Boolean functions, originally proposed in [6]. Sekine, Imai, Tani [23] proposed proceeding with the BDD construction in a top-down fashion (as opposed to the bottom-up fashion in [6]), an approach which we also follow here. In this approach, one assigns values to the Boolean variables in succession. In the diagram, children (nodes) are obtained from their parents by the assignment of a value to a single variable. One then considers the equivalence of substructures (in our case, partial orientations) in relation to the Boolean function. Two substructures are said to be equivalent if, for all possible assignments of the remaining variables (the remaining variables are assumed to be the same for both substructures), the output of the first equals that of the second in relation to the Boolean function. When two substructures are equivalent, it is redundant to compute both of them, so they can be merged before completely constructing the data-structure, reducing memory usage as a result. An example of this process is shown in Fig. 1.

Here, we consider a quasi-reduced ordered binary decision diagram (QOBDD), as in [23], by only applying a merging rule we shall specify later in this section. Fixing a variable ordering in advance, a QOBDD is constructed as a layered diagram such that all nodes in $i$-th level corresponds to $i$-th variable in the order. Let $G$ be a graph. Fix any ordering of the set of vertices $V$, and take a characteristic vector $v \in \{0,1\}^E$ to indicate the digraph obtained by substituting $\{u, v\}$ by $(u, v)$ if $v(\{u, v\}) = 0$ and by $(v, u)$ otherwise, for each $\{u, v\} \in E$ where $u < v$. The Boolean functions we consider are $f_{\mathrm{acyc}} : \{0,1\}^E \to \{0,1\}$ which takes the value 1 for acyclic orientations and 0 otherwise, and $f_{\mathrm{tcyc}} : \{0,1\}^E \to \{0,1\}$ which takes the value 1 for totally cyclic orientations and 0 otherwise.

In the remaining part of this section, we establish two tests which can be used to check if two partial orientations of a graph are equivalent in relation to the Boolean functions above, describe an ordering of the edges based on path decompositions of the graph, and finally give a bound on the size of BDDs parameterized by the

path-width of the graph using these tests and ordering.

**Definition 2.1** Let $P$ be a partial orientation. We say that the *elimination front* of $P$, denoted by $\alpha(P)$, is the set of vertices which are adjacent to at least one edge in $E(P)$ and one directed edge in $A(P)$.

**Definition 2.2** Let $P$ be a partial orientation of $G$. The *reachability relation* of $P$ is the binary relation $\mathcal{R}_P$ defined on its elimination front $\alpha(P)$ given by

$$\forall (u,v) \in \alpha(P)^2, (u \mathcal{R}_P v \iff u \neq v \text{ and } \exists \text{ a directed path from } u \text{ to } v \text{ in } G[A(P)]).$$

From a graph $G$ and an ordering $e_1, e_2, \cdots, e_m$ of its edges, we can construct the BDD for totally cyclic orientations as follows. The test to check whether two partial orientations are equivalent is shown in Steps b and c.

(i) Add the node $G$ to the 0-th level.

(ii) For each $k \in \{1, 2, \cdots, m\}$,
   (a) Add the pairs $(P, \mathrm{pp}(P))$ corresponding to all the partial orientations $P$ obtained by orienting the edge $e_k$ of each of the nodes $P_0$ of the $(k-1)$-th level, as well as a pointer to $P_0$ (their parent in the BDD), to a temporary set $T_k$.
   (b) For each pair $(P, \mathrm{pp}(P))$ in $T_k$, if there exists no vertex of $P$ non-adjacent to $E(P)$ which is a source or a sink, add $P$ to the $k$-th level, map the parent of $P$ to $P$, map $P$ to $\mathbf{0}$, and set $T_k := T_k \setminus \{(P, \mathrm{pp}(P))\}$.
   (c) Divide the remnant set $T_k$ into its equivalence classes induced by $P$ having the same reachability relation, choose *one* element of each of these equivalence classes, add its $P$ to the $k$-th level of the BDD, and map the parent of $P$ to $P$.

(iii) In the $m$-th level, map the remaining nodes to either $\mathbf{0}$ or $\mathbf{1}$.

We can use a similar algorithm to construct a BDD for all acyclic orientations by, instead of what is done in Step b above, mapping the partial orientations directly to $\mathbf{0}$ when they contain at least one directed cycle. In this case, whether the descendents of a node $P$ contain a directed cycle or not is completely characterized by the elimination front $\alpha(P)$, the reachability relation $\mathcal{R}_P$ and the orientation of the remaining undirected edges $E(P)$, not depending on $A(P)$ (up to $\mathcal{R}_P$). Thus, it is not necessary to store information about the directed edges in the case of acyclic orientations, but only information about the reachability relation. In contrast, in the algorithm for enumerating totally cyclic orientations, it is possible that a directed edge in $A(P)$ contained in a directed path connecting vertices of the elimination front loses this property in some descendant of $P$, and so we must keep information about each partial orientation.

We show examples of the constructions in Fig. 1.

The correctness of our methods is based on the two results below.

**Proposition 2.3** *At any given level of the BDD for totally cyclic orientations, we are allowed to merge the partial orientations $P_i$ and $P_j$ if $\mathcal{R}_{P_i} = \mathcal{R}_{P_j}$ and, for both*

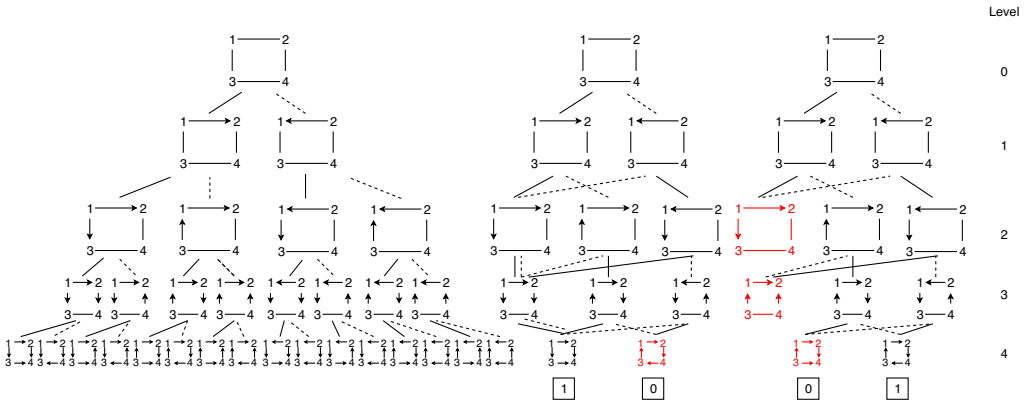


Fig. 1. (Left) Binary decision tree generated by the cycle graph with four vertices $C_4$. The edges are ordered as follows: $\{1,2\} < \{1,3\} < \{2,4\} < \{3,4\}$. Solid and dotted lines indicate the edge is oriented from smallest to biggest and from biggest so smallest vertex, respectively. (Center) Reduced binary decision diagram used to enumerate acyclic orientations constructed using our algorithm. Red partial orientations denote a direct cycle has been found and there is no need to keep developing the branch emanating from them. (Right) Reduced binary decision diagram used to enumerate totally cyclic orientations constructed from our algorithm. Red partial orientations denote that a sink or source non-adjacent to $E(P)$ has been found and thus there is no need to keep developing the branch emanating from them.

*of them, there is no vertex non-adjacent to any undirected edge which is a source or a sink.*

**Proof.** Let $P$ denote a partial orientation, in the BDD, in which there is no vertex of $P$ non-adjacent to $E(P)$ which is a source or a sink, and let $D(P)$ denote the digraph resulting from a fixed, arbitrary orientation of $E(P)$. To prove the theorem, it is sufficient to show a map $s$ which takes each $P$ to a simplified version $s(P)$ in such a way that (i) $D(s(P))$ is totally cyclic iff $D(P)$ is totally cyclic, and (ii) $D(s(P))$ is totally characterized by $\mathcal{R}_P$. We construct the map as follows: we initially perform vertex identification of each of the strongly connected components of $G[A(P)]$, obtaining $s_1(P)$. Since the elements of any pair of vertices of a strongly connected component are mutually reachable, we have that $D(P)$ is totally cyclic iff $D(s_1(P))$ is totally cyclic. We then define $s(P)$ from $s_1(P)$ by setting $A(s(P)) := \{(u,v) \in \alpha(s_1(P))^2 : u\mathcal{R}_{s_1(P)}v\}$ and deleting the (newly) isolated vertices not contained in $\alpha(s_1(P))$. It follows from the definition that $D(s(P))$ is completely characterized by $\mathcal{R}_P$.

Let $\alpha := \alpha(s_1(P)) = \alpha(s(P))$. To prove that a necessary condition for $D(s_1(P))$ to be totally cyclic is that $D(s(P))$ is totally cyclic (sufficiency is immediate), we proceed by contradiction. Assume $D(s_1(P))$ is totally cyclic, but there exists $v_0, v_l \in \alpha$ $(v_0 \neq v_l)$ such that $v_0\mathcal{R}_{s(P)}v_l$ and there exists no walk from $v_l$ to $v_0$ on $D(s(P))$. Note that $v_0\mathcal{R}_{s(P)}v_l$ implies that $v_0\mathcal{R}_{s_1(P)}v_l$, and thus there exists a directed walk $(v_0,v_1),(v_1,v_2),\ldots,(v_{l-1},v_l)$ contained in $A(s_1(P))$. Since all directed edges of $A(s_1(P))$ are contained in some cycle after the orientation, there must exist walks $b_i, p_i, a_i$, where $a_i, b_i$ are vertices and $p_i$ is an arbitrary length walk, contained in $E(s_1(P))$ and taking each $v_i$ to $v_{i-1}$ $(i = 1,2,\cdots,l)$ after the orientation. It follows that the walk $(v_l,b_l),p_l,(a_l,v_{l-1}),(v_{l-1},b_{l-1}),\ldots,(a_2,v_1),(v_1,b_1),p_1,(a_1,v_0)$ in

$D(s(P))$ (up to vertex identification of cycles) connects $v_l$ to $v_0$, which is a contradiction, completing the proof. □

**Proposition 2.4** *At any given level of the BDD for acyclic orientations, we are allowed to merge the partial orientations $P_i$ and $P_j$ if $\mathcal{R}_{P_i} = \mathcal{R}_{P_j}$ and both contain no directed cycles.*

**Proof.** Let $\alpha := \alpha(P_i) = \alpha(P_j)$. For an arbitrary orientation of the remaining undirected edges, let $D_i$ and $D_j$ be the digraphs obtained from $P_i$ and $P_j$, respectively. By symmetry, it is sufficient to prove that if $D_i$ is cyclic, then $D_j$ is also cyclic. Assume $D_i$ contains a directed cycle $C$. This cycle must necessarily pass through $\alpha$, otherwise it would follow that $D_j$ is acyclic. We can replace the directed path $C \cap A(P_i)$ with the directed path $C \cap A(P_j)$ in the cycle $C$ to obtain a new cycle $C'$ of $D_j$. □

The number of nodes in the BDD constructed by the algorithm above depends on the ordering of edges, and finding the optimum ordering is a co-NP-complete problem in general [6]. In particular, it is important to consider the *width* of the BDD, defined as the maximum number of nodes in a level over all levels. We describe an ordering of the edges, denoted *path-width ordering*, based on path decompositions of a graph which allows us to bound the width of the BDD, originally introduced in [21].

For a detailed account of path decompositions and the corresponding path-width of a graph, we refer the reader to [4]. Here, we note that the pathwidth can be computed and a corresponding path decomposition can be found in time $2^{O(\mathrm{pw}^2)}n$ for graphs of pathwidth pw [12], and recall one of the characterizations of path-width based on interval graphs. Given $n$ intervals on a line, an *interval graph* is their intersection graphs, i.e. the graph on these intervals where two intervals are adjacent iff their intersection is non-null. Furthermore, the *interval thickness* of a (not necessarily interval) graph $G$ is one less than the maximum clique size in an interval supergraph of $G$. It is known that the path-width of a graph is at most $k-1$ iff its interval thickness is at most $k$ [4].

Given a graph $G$ with path-width pw, we conclude from the above that there exists an interval graph, which is also a supergraph of $G$, on $n$ intervals whose maximum clique size is at most $\mathrm{pw}+1$. Without loss of generality, we can assume that the endpoints of the intervals have distinct coordinates in $\{1, 2, \ldots, 2n\}$. If we order the intervals in increasing order of their right endpoints, we can define the path-width ordering of the edges as their lexicographical ordering based on this ordering of the intervals.

**Lemma 2.5** *[21] For any graph $G$, the size of the elimination front for a path-width ordering of its edges is bounded by* $\mathrm{pw}+1$*, where* pw *denotes the path-width of $G$.*

A binary relation is said to be a *strict partial order* if it is irreflexive, transitive and asymmetric. It is a well known and easy to verify fact that, for any finite set $S$, the set of all partial orders of $S$ is in bijection with the set of all strict partial orders of $S$.

**Theorem 2.6** *The width of the BDDs described in this section is bounded by*

$$2^{\frac{\mathrm{pw}^2}{4}+o(\mathrm{pw}^2)}$$

*for a path-width ordering of the edges, where* pw *denotes the path-width of the graph.*

**Proof.** Let us first consider the BDD for acyclic orientations. For a given partial orientation $P$, where $A(P)$ is acyclic, it is simple to check that the reachability relation $\mathcal{R}_P$ is a strict partial order on the elimination front $\alpha(P)$. As explained above, there exists a one-to-one correspondence between the set of strict partial orders on $\alpha(P)$ and the set of partial orders on $\alpha(P)$. The following bound to the number of partial orders $P_n$ of a finite set of $n$ elements is given in [14]:

$$\log_2 P_n = \frac{n^2}{4} + \frac{3n}{2} + O(\log n),$$

from which the desired result follows.

In the case of totally cyclic orientations, there may exist subsets of vertices in the elimination front whose elements are mutually reachable. If we perform vertex identification on those subsets, we can use the result in [14] by considering the reachability relations on sets of cardinality $1, 2, \ldots, \mathrm{pw}$, respectively. The total number of possible reachability relations becomes bounded by

$$\begin{aligned} 2^{\frac{1^2}{4}+o(\mathrm{pw}^2)} + 2^{\frac{2^2}{4}+o(\mathrm{pw}^2)} + \cdots + 2^{\frac{\mathrm{pw}^2}{4}+o(\mathrm{pw}^2)} &= 2^{\frac{\mathrm{pw}^2}{4}+o(\mathrm{pw}^2)} + (\mathrm{pw}-1)2^{\frac{(\mathrm{pw}-1)^2}{4}+o(\mathrm{pw}^2)} \\ &= 2^{\frac{\mathrm{pw}^2}{4}+o(\mathrm{pw}^2)} + 2^{\frac{2\,\mathrm{pw}-1}{4}} 2^{\frac{(\mathrm{pw}-1)^2}{4}+o(\mathrm{pw}^2)} \\ &= 2^{\frac{\mathrm{pw}^2}{4}+o(\mathrm{pw}^2)}. \end{aligned}$$

□

# 3 FPT Algorithm for Counting Acyclic Orientations Using Branch Decomposition

We review *branch decomposition* and *branch-width*. Let $G = (V, E)$ be an undirected graph, where $V$ and $E$ are its set of vertices and of edges respectively. A *branch decomposition* $D$ of $G$ is a pair $(T, \phi)$ where $T$ is a rooted binary tree with $|E|$ leaves and $\phi$ is a bijection from $E$ to the set of leaves of $T$. To avoid confusion, we call a vertex of $T$ a *node*, and an edge of $T$ a *link*. Given a branch decomposition $D = (T, \phi)$ and its link $e$, the values $w(D)$ and $w_e$ are assigned in the following way. Deleting $e$ in $T$, we obtain two subtrees $T_1$ (containing the root) and $T_2$ (not containing the root). Let $L_i\,(i = 1, 2)$ be the set of leaves in $T_i$. Then we can obtain the partition $(E_1, E_2)$ of $E$, where $E_i = \phi^{-1}(L_i)\,(i = 1, 2)$ with $\phi^{-1}(L_i) = \{\phi^{-1}(l) : l \in L_i\}$. Set $\mathrm{mid}(e)$ as a set of common vertices among $G[E_1]$ and $G[E_2]$ where $G[E_i]\,(i = 1, 2)$ are edge-induced subgraphs of $G$ by $E_i$, respectively. The value $w_e$

is defined to be $|\mathrm{mid}(e)|$, i.e. the number of vertices common between $G[E_1]$ and $G[E_2]$. The value $w(D)$ is the maximum value $w_e$ among all links $e$ of $D$. The branch-width $\mathrm{bw}(G)$ (or abbr. bw) of $G$ is the minimum value $w(D)$ among all possible branch decompositions $D$ of $G$. For any fixed $k$, there exists a linear time algorithm which checks whether a graph has branch-width bounded by $k$ and, in case it does, outputs a branch decomposition of minimum branch-width [5].

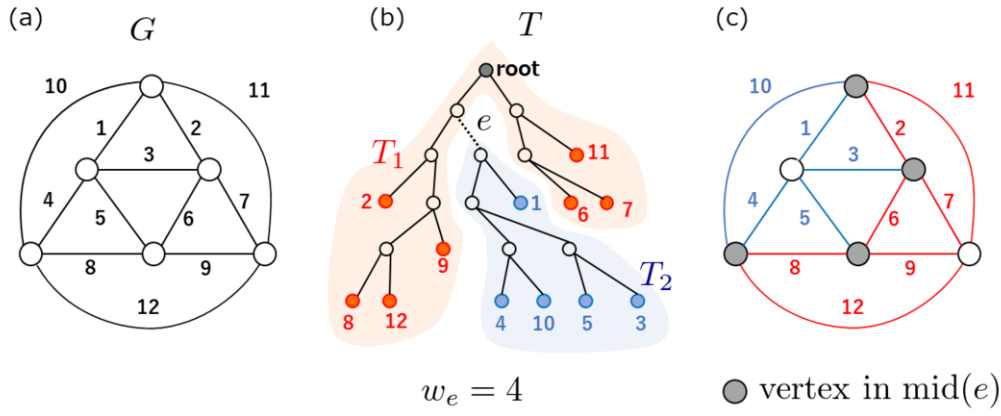


Fig. 2. Branch decomposition: (a) a given graph $G$, (b) a decomposition tree $T$ of $G$, and (c) two edge-induced subgraphs by a link $e$ of $T$

Given an optimal branch decomposition $D = (T, \phi)$ of $G$, we construct FPT algorithms to count the number of acyclic orientations and totally cyclic orientations. We first introduce some notations to construct our algorithm. For our convenience, we fix a linear order $<$ among the vertices in advance. For a given node $u$, we denote its parent by $u_0$, and its two children by $u_1$ and $u_2$, if there is any. Note that a non-leaf node $u$ has two children $u_1$ and $u_2$, and a non-root node $u$ has its parent $u_0$. For a non-root node $u$, let $e_0$ be $\{u, u_0\}$. By deleting $e_0$ in $T$, we obtain two subtrees. Denote one which does not contain the root by $T_0$, the set of leaves of $T_0$ by $L_0$, and $\phi^{-1}(L_0)$ by $E_0$. For a root node $u$, we set $E_0 := E$.

Our algorithm computes the number of acyclic and totally cyclic orientations by performing dynamic programming on the branch decomposition in a bottom-up manner from the leaves to the root. Before starting with the dynamic programming, we prepare an array $\mathrm{DP}_u$ for each node $u$. If a node $u$ is not the root, the array $\mathrm{DP}_u$ is indexed by the set $\{0,1,2,3\}^{W_u}$, where $W_u = \{(a,b) \in \mathrm{mid}(e_0)^2 : a < b\}$. If a node $u$ is the root, we prepare the array $\mathrm{DP}_u$ with just one index. Each element of $\mathrm{DP}_u$ is a pair $(i, O)$ where $i$ is an integer and $O$ is a graph orientation of an edge-induced subgraph $G[E_0]$ of $G$ by $E_0$. For each $\mathrm{DP}_u$, initialize all elements by a pair of 0 and an arbitrary graph orientation of $G[E_0]$.

For each non-root node $u$, we say that an graph orientation $O$ of $G[E_0]$ is *compatible* with an index $\boldsymbol{v} \in \{0,1,2,3\}^{W_u}$ if the following condition is satisfied: for all $w = (a,b) \in W_u$,

- $\boldsymbol{v}_w = 0$ if neither $b$ is reachable from $a$, nor $a$ from $b$ in $O$,

- $\boldsymbol{v}_w = 1$ if $b$ is reachable from $a$, but $a$ is not reachable from $b$ in $O$,
- $\boldsymbol{v}_w = 2$ if $b$ is not reachable from $a$, but $a$ is reachable from $b$ in $O$,
- $\boldsymbol{v}_w = 3$ otherwise,

where $\boldsymbol{v}_w$ is the element of $\boldsymbol{v}$ indexed by $w$. For notational convenience, if $u$ is the root node, any graph orientation $O$ of $G$ is regarded as compatible with the unique index of $\mathrm{DP}_u$.

We start dynamic programming from the leaves. For each leaf node $u$, we substitute $(1, O)$ into all elements of $\mathrm{DP}_u$ where $O$ is an orientation of an edge $\phi(u)$ compatible with each index. Then we finish the computation on $u$. For each leaf, this computation terminates in constant time.

For a non-leaf node $u$, we assume that the computation on its children $u_1$ and $u_2$ has been finished, and we update $\mathrm{DP}_u$ as follows. Take an index $\boldsymbol{v}_1$ of $\mathrm{DP}_{u_1}$ and an index $\boldsymbol{v}_2$ of $\mathrm{DP}_{u_2}$. Denote by $(i_1, O_1)$ the element of $\mathrm{DP}_{u_1}[\boldsymbol{v}_1]$, and by $(i_2, O_2)$ the element of $\mathrm{DP}_{u_2}[\boldsymbol{v}_2]$. We construct a graph orientation $O'$ on $G[E_0]$ by merging $O_1$ and $O_2$, i.e. by identifying each pair of vertices common with $O_1$ and $O_2$ into one vertex. Find the (unique) index $\boldsymbol{v}$ of $\mathrm{DP}_u$ with which $O'$ is compatible. Then update the element $(i, O)$ of $\mathrm{DP}_u[\boldsymbol{v}]$ to $(i + i_1 \cdot i_2, O')$, if the following condition is satisfied: (i) for counting acyclic orientations, $O$ is acyclic, (ii-1) for counting totally cyclic orientations and a non-root node $u$, all directed edges in $O$ participate in a directed cycle or a path connecting vertices of $\mathrm{mid}(e_0)$, and (ii-2) for counting totally cyclic orientations and the root node $u$, all directed edges in $O$ participate in a directed cycle. We finish the computation on a node $u$, after repeating the above procedure for all indices of $\mathrm{DP}_{u_1}$ and $\mathrm{DP}_{u_2}$. If we denote the integer in the array element $\mathrm{DP}_u[\boldsymbol{v}]$ after the computation is performed by $n_{u,v}$, the recurrence formula for the dynamic programming is given by

$$n_{u,v} = \sum_{v_1, v_2} n_{u_1,v_1} n_{u_2,v_2},$$

where the sum extends over all values of $v_i \in \{0, 1, 2, 3\}^{W_{u_i}}$ $(i = 1, 2)$ satisfying the conditions above. After the computation on the root node is finished, we output the integer in the array of the root node.

The correctness of these algorithms on branch decomposition can be proved in the same way as Propositions 2.3 and 2.4. For each node $u$, an array $\mathrm{DP}_u$ is constructed by taking each index of $\mathrm{DP}_{u_i}$ $(i = 1, 2)$. While the number of indices for $\mathrm{DP}_{u_i}$ $(i = 1, 2)$ is at most $2^{(\mathrm{bw}^2 - \mathrm{bw})/2}$, it can be verified in the same way as Theorem 2.6 that the number of indices to be taken into consideration is at most $2^{\mathrm{bw}^2/4 + o(\mathrm{bw}^2)}$. Therefore the time complexity of our algorithms is $O^*(2^{\mathrm{bw}^2/2 + o(\mathrm{bw}^2)})$. In summary, we obtain the following theorem.

**Theorem 3.1** *There is an algorithm to count acyclic orientations and totally cyclic orientations of a given undirected graph $G$ in $O^*(2^{\mathrm{bw}^2/2 + o(\mathrm{bw}^2)})$ time.*

# 4 Concluding Remarks

In this paper, we propose FPT algorithms for enumerating and counting acyclic orientations and totally cyclic orientations.

For the enumerating problem, we implicitly list all acyclic and totally cyclic orientation by constructing their respective BDDs instead of explicitly generating all targeted orientations in $O^*(2^{\mathrm{pw}^2/4+o(\mathrm{pw}^2)})$ time. The main advantage of enumeration by BDDs is that we can simultaneously manipulate the enumeration by taking logical operations on BDDs without tweaking each enumerated object one by one. For the counting problem, we construct FPT algorithms based on branch decomposition whose running time is $O^*(2^{\mathrm{bw}^2/2+o(\mathrm{bw}^2)})$.

As future work, we list the three following problems. Firstly, one can consider further improving the running time of the algorithms parameterized by path-width and branch-width presented in this paper. Secondly, one may consider investigating the (parameterized) computational complexity of manipulating BDDs associated with acyclic and totally cyclic orientations. Finally, we suggest developing algorithms parameterized by other structural graph invariants. Path-width and branch-width tend to be close to the input size $N$ when the input graph is dense. It is thus an important task to design efficient parameterized algorithms with respect to other parameters robust against dense graphs.

# Acknowledgment

This work was supported by JSPS KAKENHI Grant Numbers 15H01677, 16K12392, 17K12639. We thank the three anonymous reviewers for their helpful comments and suggestions.

# References

[1] Avis, D. and K. Fukuda, *Reverse search for enumeration*, Discrete Applied Mathematics **65** (1996), pp. 21–46.

[2] Barbosa, V. C. and E. Gafni, *Concurrency in heavily loaded neighborhood-constrained systems*, ACM Transactions on Programming Languages and Systems (TOPLAS) **11** (1989), pp. 562–584.

[3] Barbosa, V. C. and J. L. Szwarcfiter, *Generating all the acyclic orientations of an undirected graph*, Information Processing Letters **72** (1999), pp. 71–74.

[4] Bodlaender, H. L., *A partial k-arboretum of graphs with bounded treewidth*, Theoretical Computer Science **209** (1998), pp. 1–45.

[5] Bodlaender, H. L. and D. M. Thilikos, *Constructive linear time algorithms for branchwidth*, in: *International Colloquium on Automata, Languages, and Programming*, Springer, 1997, pp. 627–637.

[6] Bryant, R. E., *Graph-based algorithms for Boolean function manipulation*, IEEE Transactions on Computers **35** (1986), pp. 677–691.

[7] Brylawski, T. and J. Oxley, *The Tutte polynomial and its applications*, Matroid Applications **40** (1992), pp. 123–225.

[8] Chávez-Lomelí, L. E., C. Merino, S. D. Noble and M. Ramírez-Ibáñez, *Some inequalities for the Tutte polynomial*, European Journal of Combinatorics **32** (2011), pp. 422–433.

[9] Conte, A., R. Grossi, A. Marino and R. Rizzi, *Listing acyclic orientations of graphs with single and multiple sources*, in: *LATIN 2016: Theoretical Informatics - 12th Latin American Symposium*, 2016, pp. 319–333.

[10] Conte, A., R. Grossi, A. Marino, R. Rizzi and L. Versari, *Directing road networks by listing strong orientations*, in: *International Workshop on Combinatorial Algorithms*, Springer, 2016, pp. 83–95.

[11] Coudert, O., *Solving graph optimization problems with ZBDDs*, in: *Proceedings of the 1997 European conference on Design and Test*, IEEE Computer Society, 1997, pp. 224–228.

[12] Fürer, M., *Faster computation of path-width*, in: *International Workshop on Combinatorial Algorithms*, Springer, 2016, pp. 385–396.

[13] Greene, C. and T. Zaslavsky, *On the interpretation of Whitney numbers through arrangements of hyperplanes, zonotopes, non-radon partitions, and orientations of graphs*, Transactions of the American Mathematical Society **280** (1983), pp. 97–126.

[14] Kleitman, D. J. and B. L. Rothschild, *Asymptotic enumeration of partial orders on a finite set*, Transactions of the American Mathematical Society **205** (1975), pp. 205–220.

[15] Knuth, D. E., "The Art of Computer Programming, Volume 4, Fascicle 1: Bitwise Tricks & Techniques; Binary Decision Diagrams," Addison-Wesley Professional, 2009, 12th edition.

[16] Las Vergnas, M., *Acyclic and totally cyclic orientations of combinatorial geometries*, Discrete Mathematics **20** (1977), pp. 51–61.

[17] Las Vergnas, M., *The Tutte polynomial of a morphism of matroids II. activities of orientations*, Progress in Graph Theory (1984).

[18] Lin, F., *A note on spanning trees and totally cyclic orientations of 3-connected graphs*, Journal of Combinatorics **4** (2013), pp. 95–104.

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

[20] Noble, S. D. and G. F. Royle, *The Merino–Welsh conjecture holds for series–parallel graphs*, European Journal of Combinatorics **38** (2014), pp. 24–35.

[21] Oliveira, F. S., H. Hiraishi and H. Imai, *Revisiting the top-down computation of BDD of spanning trees of a graph and its Tutte polynomial* Accepted for publication.

[22] Robertson, N. and P. D. Seymour, *Graph minors. X. Obstructions to tree-decomposition*, Journal of Combinatorial Theory, Series B **52** (1991), pp. 153–190.

[23] Sekine, K., H. Imai and S. Tani, *Computing the Tutte polynomial of a graph of moderate size*, in: *Algorithms and Computations*, Lecture Notes in Computer Science (1995), pp. 224–233.

[24] Shiroshita, S., T. Ogasawara, H. Hiraishi and H. Imai, *Fpt algorithms exploiting carving decomposition for eulerian orientations and ice-type models*, in: *International Workshop on Algorithms and Computation*, Springer, 2018, pp. 216–227.

[25] Squire, M. B., *Generating the acyclic orientations of a graph*, Journal of Algorithms **26** (1998), pp. 275–290.

[26] Stanley, R. P., *Acyclic orientations of graphs*, Discrete Mathematics **5** (1973), pp. 171–178.

[27] Thomassen, C., *Spanning trees and orientations of graphs*, Journal of Combinatorics **1** (2010), pp. 101–111.

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