Fast Exponential-Time Algorithms for the Forest Counting and the Tutte Polynomial Computation in Graph Classes

Heidi Gebauer[*] Yoshio Okamoto[†]

Abstract

We prove #P-completeness for counting the number of forests in regular graphs and chordal graphs. We also present algorithms for this problem, running in $O^*(1.8494^m)$ time for 3-regular graphs, and $O^*(1.9706^m)$ time for unit interval graphs, where m is the number of edges in the graph and O^*-notation ignores a polynomial factor. The algorithms can be generalized to the Tutte polynomial computation.
Keywords: chordal graph; exponential-time algorithm; forest; regular graph; Tutte polynomial; unit interval graph.
1991 Mathematics Subject Classification: 22E46, 53C35, 57S20

1 Introduction

Counting is a fundamental task in combinatorics, and algorithmic aspects of counting problems have also been studied. One of the most interesting phenomena around algorithmic counting is that we can count the number of spanning trees in a graph in polynomial time [8] while it is #P-complete to count the number of forests in a graph, even in a bipartite planar graph [13]. These two counting problems fit into a general concept of the Tutte polynomial of a graph (or of a matroid), and this connection yields a fruitful development in algorithmic counting.

The #P-complete counting problems have been tackled mainly via two different approaches. One is the approximate approach, and the other is the exact approach. In the approximate method, we try to quickly approximate the desired value within a certain guarantee by, for example, a Markov chain Monte Carlo method. See Jerrum's book [7]. In the exact approach, we stick to the exact correct value, and try to reduce the running time as much as possible. When a given problem is #P-complete, we cannot expect the algorithm to run in polynomial time. Hence, we try to make the exponent of the exponential running time closer to constant, or try to make the base closer to 1.

This paper takes the latter exact approach. First we prove that the forest counting problem is #P-complete for regular graphs and chordal graphs. Then, we design exact algorithms for the problem when the input graphs are restricted to the regular graphs or to the unit interval graphs. The running time of our algorithm is $O^*(1.8494^m)$ time for 3-regular graphs, and $O^*(1.9706^m)$ for unit interval graphs, where m is the number of edges in the graph and O^*-notation ignores a polynomial factor. It has to be noted here that the algorithms can be generalized to the Tutte polynomial computation.

Note that for general graphs the contraction-deletion formula for the number of forests yields an algorithm running in $O^*(\min\{2^m, 1.6181^{n+m}\})$ time, where n and m represent the numbers of vertices and edges in a given graph (refer to a book by Wilf [16] where he obtained this bound for the chromatic polynomial but the idea can be applied to any quantity that is governed by the contraction-deletion formula). For 3-regular graphs it holds that $m = 3n/2$, and hence the latter expression in this bound gives $1.6181^{n+m} = 1.6181^{2m/3+m} = 1.6181^{5m/3} > 2.2301^m$. This means that our algorithm with running time 1.8494^m is much faster than a direct application of the contraction-deletion formula.

Related Work There are several papers studying the forest counting problem (or the Tutte polynomial computation, more generally) via the exact approach. The basis is the hardness result due to Jaeger, Vertigan & Welsh [6] showing that counting the number of forests in a graph is #P-complete. Vertigan [12] proved that the problem is

[*]Institute of Theoretical Computer Science, ETH Zurich, Zurich, CH-8092, Switzerland
[†]Department of Information and Computer Sciences, Toyohashi University of Technology, Hibarigaoka 1–1, Tempaku, Toyohashi, Aichi, 441-8580, Japan

#P-complete for planar graphs, and Vertigan & Welsh [13] proved that it is #P-complete even for bipartite planar graphs.

On the exact algorithmic side, not much is known for the forest counting problem. Andrzejak [1] and Noble [9] independently obtained a polynomial-time algorithm for the forest counting problem in graphs of bounded tree-width. To the authors' knowledge, this is the only non-trivial case where a polynomial-time solution is known. As mentioned above, for general graphs the contraction-deletion formula for the number of forests yields an algorithm running in $O^*(\min\{2^m, 1.6181^{n+m}\})$ time, where n and m represent the numbers of vertices and edges in a given graph throughout this article. Giménez, Hliněný & Noy [4] gave an algorithm in graphs of bounded clique-width. Their algorithm runs in $\exp(O(n^{1-1/(k+2)}))$ time where k is the clique-width of a given graph. Furthermore, Sekine, Imai & Tani [10] gave an $\exp(O(\sqrt{n}))$-time algorithm in planar graphs.

As for the approximation, Annan [2] gave a fully polynomial-time approximation scheme for the forest counting in dense graphs.

For some counting problems in regular graphs, Vadhan [11] gave #P-completeness results by utilizing the so-called interpolation technique and Fibonacci technique. These techniques are also used in this paper.

Preliminaries In this article, all graphs are finite and undirected. Let $G = (V, E)$ be a graph. The *degree* of a vertex $v \in V$ in G is the number of edges incident to v, and denoted by $\deg_G(v)$. A graph is k-*regular* if every vertex of it has degree k. A graph is *planar* if it can be drawn on the plane without any edge crossing. A graph is *bipartite* if the vertex set can be partitioned into two parts such that every edge has the endpoints in both parts. Other terms on graphs will be defined when they are first used, or can be found in any textbook on graphs like West [15].

A *forest* of a graph $G = (V, E)$ is a subset $F \subseteq E$ which embraces no cycle. Our goal is to count the number of forests in a given graph. The following is our problem template, where a class of graphs is denoted by Γ.

Problem: Γ-#FORESTS
Input: a graph $G \in \Gamma$;
Output: the number of forests in G.

We write $f(n) = O^*(g(n))$ if $f(n) = O(g(n)p(n))$ for some constant-degree polynomial $p(n)$. Namely, in the O^*-notation we ignore the polynomial factor.

Basic terminology on complexity theory like #P-completeness can be found in the book by Garey & Johnson [3].

2 Intractability

In this section, we concentrate on the intractability results. We prove #P-completeness of Γ-#FORESTS for various Γ.

2.1 Bounded-degree graphs

Denote by 3Δ the class of all graphs of maximum degree at most three, by BP the class of all bipartite planar graphs, and by 3ΔBP the class of all bipartite planar graphs of maximum degree at most three. We prove the following.

Theorem 2.1. The problem 3ΔBP-#FORESTS is #P-complete.

This theorem immediately gives the following corollary.

Corollary 2.2. The problem 3Δ-#FORESTS is #P-complete.

To prove the theorem, we use BP-#FORESTS, which is shown to be #P-complete by Vertigan & Welsh [13]. We first prove that the following variant of 3ΔBP-#FORESTS is #P-complete.

Problem: Γ-#FORESTS with inclusive edges
Input: a graph $G = (V, E) \in \Gamma$, and an edge set $S \subseteq E$;
Output: the number of forests in G which contain S.

Lemma 2.3. The problem 3ΔBP-#FORESTS with inclusive edges is #P-complete.

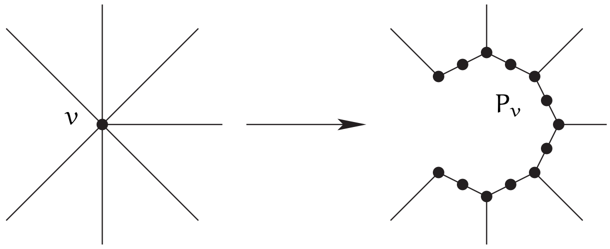

Figure 1: Replacing a vertex with a path (a local picture).

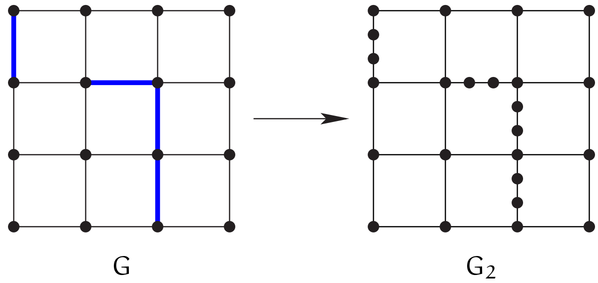

Figure 2: Replacing edges with paths. The thick edges belong to S, and each of them is replaced by a path of length three in G_2.

Proof. We reduce BP-#FORESTS to 3ΔBP-#FORESTS with inclusive edges. Let $G = (V, E)$ be a bipartite planar graph given as an input for BP-#FORESTS. Without loss of generality, we may assume that G has no vertex of degree zero. We fix a plane embedding of G (which can be obtained in linear time). From G, we construct another graph G' which is also bipartite planar and furthermore whose maximum degree is at most three. First we replace each vertex $v \in V$ with a path P_v of length $2\deg_G(v) - 2$, and the path is embedded as if it surrounded the vertex v. The neighbors of v are joined to every second vertex of P_v in the same circular order. See Figure 1. We perform this operation for all vertices of G, and G' is the resulting graph. Note that G' is bipartite planar since G is so, and that the maximum degree of G' is at most three.

Set S to be the set of edges in P_v for all $v \in V$. Then we can find a natural bijection from the family of forests in G to the family of forests in G' which include S. Thus the lemma is proved. □

Proof of Theorem 2.1. We reduce 3ΔBP-#FORESTS with inclusive edges to 3ΔBP-#FORESTS. Let $G = (V, E)$ be a bipartite planar graph with maximum degree at most three and $S \subseteq E$. Let $s = |S|$, and for each $\ell \in \{1, \dots, s+1\}$ we construct a graph $G_\ell = (V_\ell, E_\ell)$ from G by replacing each edge $e \in S$ with a path P_e of length $2\ell - 1$. Especially G_1 is isomorphic to G. Figure 2 shows an example for $\ell = 2$.

Fix $\ell \in \{1, \dots, s+1\}$. We define a map from the family of forests in G_ℓ to the family of forests in G as follows: We map a forest $F_\ell \subseteq E_\ell$ of G_ℓ to a forest $F \subseteq E$ of G if and only if

- when $e \in S \cap F$, all edges of P_e belong to F_ℓ,
- when $e \in S \setminus F$, at least one edge of P_e does not belong to F_ℓ, and
- when $e \notin S$, e belongs to F_ℓ if and only if e belongs to F.

We can observe that every forest F in G is the image of $(2^{2\ell-1} - 1)^{|S \setminus F|}$ forests in G_ℓ. Therefore the number of forests in G_ℓ is equal to

$$\sum_F (2^{2\ell-1} - 1)^{|S\setminus F|} = \sum_{i=0}^{s} \sum_{F:|S\setminus F|=i} (2^{2\ell-1} - 1)^i = \sum_{i=0}^{s} a_i x_\ell^i,$$

where $x_\ell = 2^{2\ell-1} - 1$ and a_i is the number of forests F in G such that $|S \setminus F| = i$. Since $x_\ell \neq x_{\ell'}$ for all $\ell, \ell' \in \{1, \dots, s+1\}$, $\ell \neq \ell'$, by knowing the number of forests in G_ℓ for all $\ell \in \{1, \dots, s+1\}$ we can compute $a_0, \dots, a_s$ in polynomial time. Since a_0 is the number of forests in G which contain S, this completes the reduction. □

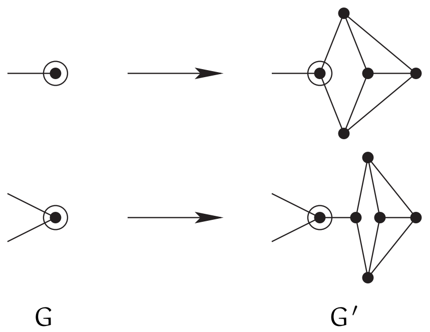

Figure 3: Attaching a graph to a degree-one vertex and a degree-two vertex.

2.2 Regular graphs

Denote by kREG the class of k-regular graphs, and by kREGP the class of k-regular planar graphs.

Theorem 2.4. The problem 3REGP-#FORESTS is #P-complete.

Proof. We reduce 3ΔBP-#FORESTS to 3REGP-#FORESTS. Let $G = (V, E)$ be a bipartite planar graph with maximum degree at most three. Without loss of generality, we may assume that G has no vertex of degree zero. We construct a 3-regular planar graph G' from G as follows. We attach the graph shown in Figure 3 (top) to each vertex of degree one, and attach the graph shown in Figure 3 (bottom) to each vertex of degree two. We can see that the resulting graph G' is 3-regular and still planar. Denote by n_1 and n_2 the number of degree-one vertices and degree-two vertices in G, respectively. Then the number of forests in G' is equal to the number of forests in G times $c_1^{n_1} c_2^{n_2}$ where c_1 and c_2 are the numbers of forests in the appended graphs (in Figure 3), thus constants. This completes our reduction. □

For general $k \geq 3$, we similarly have the following theorem.

Theorem 2.5. For every $k \geq 3$, the problem kREG-#FORESTS is #P-complete.

The proof is a bit more involved, and we have to distinguish the cases according to the parity of k.

Proof of Theorem 2.5 for odd k. We reduce 3REG-#FORESTS to kREG-#FORESTS. Let $G = (V, E)$ be a 3-regular graph. We construct a k-regular graph G' from G by attaching the graph shown in Figure 4 to each vertex of G. Namely, it is a graph having $(k-3)/2$ copies of K_{k+1}^- (a complete graph on $k+1$ vertices with one edge removed) and another vertex with edges to the $k-3$ vertices on the copies which were incident to the removed edges. Then, we can see that the resulting graph G' is k-regular, and the number of forests in G' is equal to the number of forests in G times c^n, where c is the number of forests in the appended graph which only depends on k. This completes our reduction. □

When k is even, we produce a sequence of reductions. First we consider the following problem.

Problem: Γ-#FORESTS with exclusive edges
Input: a graph $G = (V, E) \in \Gamma$, and an edge set $S \subseteq E$;
Output: the number of forests in G which do not contain any edges in S.

Lemma 2.6. For even $k \geq 4$, the problem kREG-#FORESTS with exclusive edges is #P-complete.

Proof. We reduce $(k-1)$REG-#FORESTS to kREG-#FORESTS with exclusive edges. Note that since k is even and at least four, $k-1$ is odd and at least three. Hence, $(k-1)$REG-#FORESTS is #P-complete by Theorem 2.5.

Let $G = (V, E)$ be a $(k-1)$-regular graph. Since $k-1$ is odd, G has even number of vertices. Take an arbitrary partition of V into $|V|/2$ parts of size two, and for each part $\{u_i, v_i\}$, $i \in \{1, \ldots, |V|/2\}$, we attach an edge $e_i = \{u_i, v_i\}$ to G. The resulting graph $G' = (V, E \cup \{e_i \colon 1 \leq i \leq |V|/2\})$ is k-regular. We set $S = \{e_i \colon i \in \{1, \ldots, |V|/2\}\}$, the set of attached edges. Then we may observe that the set of forests of G is the set of forests of G' which contain no edge of S. This completes the reduction. □

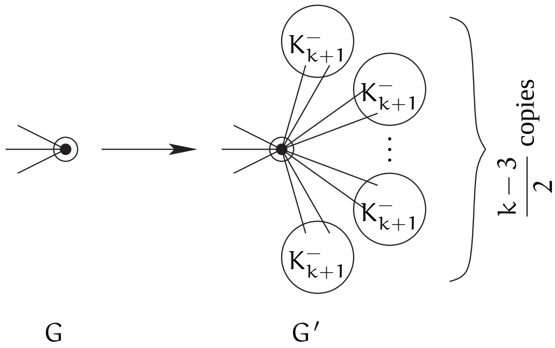

Figure 4: Attaching a graph to a degree-three vertex. Here K^-_{k+1} represents a complete graph on $k+1$ vertices with one edge removed, and two edges leave each K^-_{k+1} from the vertices of degree $k-1$, i.e., the vertices incident to the removed edge.

Next we consider the following auxiliary problem. Denote by $(2,k)$REG the class of graphs in which every vertex has degree 2 or k.

Lemma 2.7. For even $k \geq 4$, the problem $(2,k)$REG-#FORESTS is #P-complete.

Proof. We reduce kREG-#FORESTS with exclusive edges to $(2,k)$REG-#FORESTS. Let $G = (V, E)$ be a k-regular graph, where $k \geq 4$ is even, and $S \subseteq E$. Let $s = |S|$, and for each $\ell \in \{1, \dots, s+1\}$ we construct a graph $G_\ell = (V_\ell, E_\ell)$ from G by replacing each edge $e \in S$ with a path P_e of length ℓ. We can see that every vertex of G_ℓ has degree 2 or k.

Fix $\ell \in \{1, \dots, s+1\}$ and we define a map from the family of forests in G_ℓ to the family of forests in G as follows: We map a forest $F_\ell \subseteq E_\ell$ of G_ℓ to a forest $F \subseteq E$ of G if and only if

- when $e \in S \cap F$, all edges of P_e belong to F_ℓ,
- when $e \in S \setminus F$, at least one edge of P_e does not belong to F_ℓ,
- when $e \notin S$, e belongs to F_ℓ if and only if e belongs to F.

As in the proof of Lemma 2.3, we can observe that every forest F in G is the image of $(2^\ell - 1)^{|S \setminus F|}$ forests in G_ℓ. Therefore, the number of forests in G_ℓ is equal to

$$\sum_F (2^\ell - 1)^{|S \setminus F|} = \sum_{i=0}^{s} \sum_{F:\ |S \setminus F| = i} (2^\ell - 1)^i = \sum_{i=0}^{s} a_i x_\ell^i,$$

where $x_\ell = 2^\ell - 1$ and a_i is the number of forests F in G such that $|S \setminus F| = i$. Since $x_\ell \neq x_{\ell'}$ for all $\ell, \ell' \in \{1, \dots, s+1\}$, by knowing the numbers of forests in G_ℓ for all $\ell \in \{1, \dots, s+1\}$ we can compute $a_0, \dots, a_s$ in polynomial time. Since a_s is the number of forests in G which exclude S, this completes the reduction. □

We are now ready to prove Theorem 2.5 for even $k \geq 4$.

Proof of Theorem 2.5 for even $k \geq 4$. We reduce $(2,k)$REG-#FORESTS to kREG-#FORESTS when $k \geq 4$ is even. Let $G = (V, E)$ be a graph whose vertices are of degree two or k. We construct a k-regular graph G' from G by attaching the graph shown in Figure 5 to each degree-two vertex of G. Namely, it is a graph having $(k-2)/2$ copies of K^-_{k+1} (a complete graph on $k+1$ vertices with one edge removed) and another vertex with edges to the $k-2$ vertices on the copies which were incident to the removed edges. Then we can see that the resulting graph G' is k-regular and the number of forests in G' is equal to the number of forests in G times c^{n_2}, where c is the number of forests in the appended graph and n_2 is the number of degree-two vertices. Note that c depends on k only. □

Note that the resulting graph G' in the proof of Theorem 2.5 is not planar unless $k = 3$.

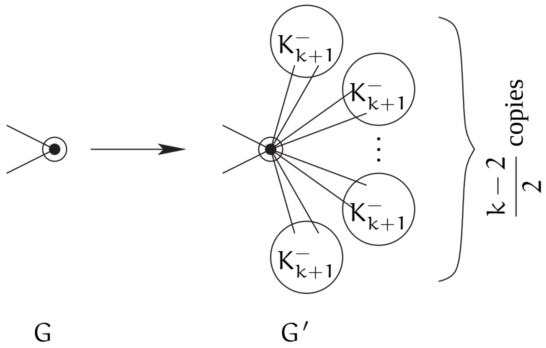

Figure 5: Attaching a graph to a degree-two vertex. Here K^-_{k+1} represents a complete graph on $k+1$ vertices with one edge removed, and two edges leave each K^-_{k+1} from the vertices of degree $k-1$, i.e., the vertices incident to the removed edge.

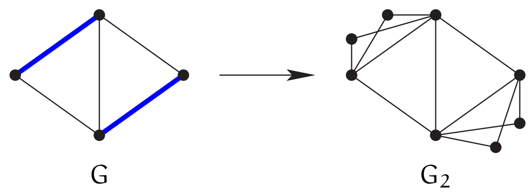

Figure 6: Joining paths of length two.

2.3 Chordal graphs

A graph G is *chordal* if every induced cycle is of length three. Denote by CHORDAL the class of chordal graphs.

Theorem 2.8. The problem CHORDAL-#FORESTS is #P-complete.

To prove Theorem 2.8, we use the following lemma about exclusive edges.

Lemma 2.9. The problem CHORDAL-#FORESTS with exclusive edges is #P-complete.

Proof. We use any graph class Γ such that Γ-#FORESTS is #P-complete. For example, set $\Gamma =$ BP. From a given graph $G = (V, E) \in \Gamma$, we construct a chordal graph $G' = (V', E')$ by $V' = V$ and $E' = \binom{V}{2}$. Namely, G' is a complete graph on V. Set $S = \binom{V}{2} \setminus E$. Then, we can see that the forests of G have a one-to-one correspondence to the forests of G' which exclude S. □

Now comes the main part of the proof.

Proof of Theorem 2.8. We reduce CHORDAL-#FORESTS with exclusive edges to CHORDAL-#FORESTS. Let $G = (V, E)$ be a chordal graph and $S \subseteq E$. Let $s = |S|$, and for each $\ell \in \{0, \dots, s\}$ we construct a graph $G_\ell = (V_\ell, E_\ell)$ from G by joining ℓ paths of length two, in parallel, to the endpoints of every edge $e \in S$. Especially, G_0 is isomorphic to G. Figure 6 shows an example for $\ell = 2$.

Fix $\ell \in \{0, \dots, s\}$, and denote by $P_e^1, P_e^2, \dots, P_e^\ell$ the newly added paths in G_ℓ between the endpoints of e. We define a map from the family of forests in G_ℓ to the family of forests in G as follows: We map a forest $F_\ell \subseteq E_\ell$ of G_ℓ to a forest $F \subseteq E$ of G if and only if

- when $e \in S \cap F$, F_ℓ contains one of the paths among $P_e^1, \dots, P_e^\ell$ completely or contains e,
- when $e \in S \setminus F$, F_ℓ contains none of the paths among $P_e^1, \dots, P_e^\ell$ completely or does not contain e, and
- when $e \notin S$, e belongs to F_ℓ if and only if e belongs to F.

We can observe that every forest F in G is the image of $(3^\ell + \ell 3^{\ell-1})^{|S\cap F|}3^{\ell|S\setminus F|}$ forests in G_ℓ. Therefore the number of forests in G_ℓ is equal to

$$\begin{aligned}\sum_F (3^\ell + \ell 3^{\ell-1})^{|S\cap F|}3^{\ell|S\setminus F|} &= \sum_{i=0}^{s}\sum_{F\colon |S\cap F|=i}(3^\ell + \ell 3^{\ell-1})^{i}3^{\ell(s-i)}\\ &= 3^{\ell s}\sum_{i=0}^{s}\sum_{F\colon |S\cap F|=i}(1+\ell/3)^i = 3^{\ell s}\sum_{i=0}^{s} a_i x_\ell^i,\end{aligned}$$

where $x_\ell = 1 + \ell/3$ and a_i is the number of forests F in G such that $|S \cap F| = i$. Since $x_\ell \neq x_{\ell'}$ for all $\ell, \ell' \in \{0, \ldots, s\}$, $\ell \neq \ell'$, by knowing the number of forests in G_ℓ for all $\ell \in \{0, \ldots, s\}$ we can compute $a_0, \ldots, a_s$ in polynomial time. Since a_0 is the number of forests in G which exclude S, this completes the reduction. □

Note that the proof actually shows that counting the number of forests in a split graph is #P-complete, where a graph is *split* if the vertex set can be partitioned into a clique and an independent set. Denote by SPLIT the class of split graphs.

Theorem 2.10. The problem SPLIT-#FORESTS is #P-complete.

Proof. The proof of Lemma 2.9 shows that it is #P-complete to count the number of forests in a complete graph which do not contain any edges in a given edge subset S. Therefore, the given graph G in the proof of Theorem 2.8 can be restricted to a complete graph, and then we can see that the constructed graphs $G_0, \ldots, G_s$ are all split graphs. □

3 Algorithms

In this section, we concentrate on faster (exponential-time) algorithms for the forest counting problem. The trivial algorithm runs in $O^*(2^m)$ time, and the goal is to beat this bound. Throughout the section, n and m denote the numbers of vertices and edges in a given graph respectively.

Denote by $\mathcal{F}(G)$ the family of forests in G. To state a fundamental property of $|\mathcal{F}(G)|$, we need to introduce the deletion and the contraction of an edge in a graph. For a graph $G = (V, E)$ and an edge $e \in E$, the *deletion* of e from G is an operation to obtain another graph, denoted by $G\backslash e$, where the vertex set of $G\backslash e$ is the same as that of G and the edge set of $G\backslash e$ is $E \setminus \{e\}$. The *contraction* of e in G is an operation to obtain another graph, denoted by G/e in the following way: we first remove the edge e and then identify the endpoints of e. Note that contraction may introduce a loop or multiple edges in the graph. Here, an edge is called a *loop* if its endpoints are identical. As a basic property of $|\mathcal{F}(G)|$, the following so-called contraction-deletion formula is well-known (see also Section 4):

$$|\mathcal{F}(G)| = \begin{cases} 1 & \text{if } G \text{ has no edge,}\\ |\mathcal{F}(G\backslash e)| & \text{if an edge } e \text{ is a loop of } G,\\ |\mathcal{F}(G/e)| + |\mathcal{F}(G\backslash e)| & \text{if an edge } e \text{ is not a loop of } G.\end{cases}$$

As mentioned in the introduction, the direct application of this formula will yield the running time bound $O^*(\min\{2^m, 1.6181^{n+m}\})$. In the sequel, we give improved algorithms for regular graphs, bounded-degree graphs, and unit interval graphs.

3.1 Regular graphs and bounded-degree graphs

To illustrate the general strategy, we start with an algorithm for 3REG-#FORESTS (i.e., counting the number of forests in 3-regular graphs).

Theorem 3.1. We can count the number of forests in a 3-regular graph with m edges in $O^*(1.8494^m)$ time.

Proof. The idea for our algorithm is as follows. Let $G = (V, E)$ be a given 3-regular graph. Each vertex v of G is incident to exactly three edges, say, e_1, e_2, e_3. Then by the contraction-deletion formula, we have

$$\begin{aligned}|\mathcal{F}(G)| =& |\mathcal{F}(G/e_1/e_2/e_3)| + |\mathcal{F}(G/e_1/e_2\backslash e_3)| + |\mathcal{F}(G/e_1\backslash e_2/e_3)| + |\mathcal{F}(G/e_1\backslash e_2\backslash e_3)|\\ &+ |\mathcal{F}(G\backslash e_1/e_2/e_3)| + |\mathcal{F}(G\backslash e_1/e_2\backslash e_3)| + |\mathcal{F}(G\backslash e_1\backslash e_2/e_3)| + |\mathcal{F}(G\backslash e_1\backslash e_2\backslash e_3)|.\end{aligned}$$

The central observation is that the four graphs $G/e_1\backslash e_2\backslash e_3$, $G\backslash e_1/e_2\backslash e_3$, $G\backslash e_1\backslash e_2/e_3$ and $G\backslash e_1\backslash e_2\backslash e_3$ are all isomorphic (up to the existence of isolated vertices). Therefore, the formula above may be written in the following way:

$$\begin{aligned}|\mathcal{F}(G)| =&|\mathcal{F}(G/e_1/e_2/e_3)| + |\mathcal{F}(G/e_1/e_2\backslash e_3)| + |\mathcal{F}(G/e_1\backslash e_2/e_3)| + |\mathcal{F}(G\backslash e_1/e_2/e_3)| \\ &+ 4|\mathcal{F}(G\backslash e_1\backslash e_2\backslash e_3)|.\end{aligned}$$

Note that in each of the graphs $G/e_1/e_2/e_3$, $G/e_1/e_2\backslash e_3$, $G/e_1\backslash e_2/e_3$, $G\backslash e_1/e_2/e_3$ and $G\backslash e_1\backslash e_2\backslash e_3$ on the right-hand side the number of edges is exactly $m-3$. Thus, from the given instance with n vertices and m edges, we obtained five subinstances with $n-1$ vertices and $m-3$ edges.

The discussion above leads to the following algorithm.

1. Choose an arbitrary maximal independent set I of G.
2. Output the value returned by the call to $A(G, I)$.

Below is a description of $A(G, I)$, which outputs the number of forests in G with the information that I is an independent set of G.

1. If I is non-empty,
   (a) choose an arbitrary vertex $v \in I$. Let e_1, e_2, e_3 be the edges incident to v.
   (b) Output the sum of the values returned by $A(G/e_1/e_2/e_3, I \setminus \{v\})$, $A(G/e_1/e_2\backslash e_3, I \setminus \{v\})$, $A(G/e_1\backslash e_2/e_3, I\setminus\{v\})$, $A(G\backslash e_1/e_2/e_3, I\setminus\{v\})$ and 4 times the value returned by $A(G\backslash e_1\backslash e_2\backslash e_3, I\setminus \{v\})$.
2. Otherwise, compute $|\mathcal{F}(G)|$ by the contraction-deletion formula and output it.

Note that in the call to $A(G, I)$ (at any point) the vertex v is incident to three edges since I is an independent set of G (at any point). Therefore, by the discussion above, the algorithm correctly outputs the number of forests in a given 3-regular graph.

We now bound the running time of our algorithm. The number of subinstances we get in the end (namely, subinstances (G, I) with $I = \emptyset$) is $5^{|I|}$, and each of such subinstance has $n - |I|$ vertices and $m - 3|I|$ edges. By the contraction-deletion formula, the number of forests in each subinstance can be computed in

$$O^*(\min\{2^{m-3|I|}, 1.6181^{(n-|I|)+(m-3|I|)}\})$$

time. Note that $n = 2m/3$ for 3-regular graphs, and so

$$1.6181^{(n-|I|)+(m-3|I|)} = 1.6181^{(5m/3-4|I|)} > 2.2301^m/6.8553^{|I|}.$$

Therefore, $\min\{2^{m-3|I|}, 1.6181^{(n-|I|)+(m-3|I|)}\} = 2^{m-3|I|}$, and hence, the total running time of the algorithm is bounded from above by $O^*(5^{|I|} \times 2^{m-3|I|}) = O^*(2^m \times (5/8)^{|I|})$.

Thus, we need a lower bound for the size of a maximal independent set.

Lemma 3.2. Every maximal independent set of a graph of maximum degree k with n vertices contains at least $n/(k+1)$ vertices.

Proof. Let $G = (V, E)$ be a graph of maximum degree k and $I \subseteq V$ be an arbitrary maximal independent set of G. We count the number of edges between I and $V \setminus I$ in two ways. On one hand, each vertex of I is incident to at most k edges. Therefore, the number of edges between I and $V \setminus I$ is at most $k|I|$. On the other hand, every vertex of $V \setminus I$ has at least one of its neighbors in I since I is maximal. Therefore, the number of edges between I and $V \setminus I$ is at least $|V \setminus I|$. Thus, we obtain $k|I| \geq |V \setminus I| = n - |I|$. This results in $|I| \geq n/(k+1)$. □

Consequently, the running time of our algorithm is bounded by $O^*(2^m \times (5/8)^{|I|}) \leq O^*(2^m \times (5/8)^{n/4}) = O^*(2^m \times (5/8)^{m/6}) = O^*(1.8494^m)$. This completes the proof. □

For k-regular graphs G we may obtain a similar algorithm. To this end, we again take an arbitrary maximal independent set I of a given k-regular graph G. Each vertex v of I is incident to exactly k edges, and they give rise to 2^k subinstances from the contraction-deletion formula, but we can see that $k+1$ of them are isomorphic. Therefore, the number of subinstances we get in the end is $(2^k - k)^{|I|}$, and each of these instances has $n - |I|$ vertices and $m - k|I|$ edges. Thus, by the same argument as Theorem 3.1, we obtain the running time bound $O^*(2^m \times ((2^k - k)/2^k)^{|I|})$. By Lemma 3.2 we get $|I| \geq \frac{n}{k+1} = \frac{2m}{k(k+1)}$, and hence obtain the following theorem.

Theorem 3.3. For any $k \geq 2$, we can count the number of forests in a k-regular graph in $O^*((2(1-\frac{k}{2^k})^{\frac{2}{k(k+1)}})^m)$ time.

Note that 2REG-#FORESTS can be solved in polynomial time (not by the algorithm above) since every connected component of a 2-regular graph is a cycle.

For graphs of maximum degree at most k, the same algorithm works and the worst-case running time is also the same.

Theorem 3.4. For any $k \geq 2$, we can count the number of forests in a graph of maximum degree k in $O^*((2(1 - \frac{k}{2^k})^{\frac{2}{k(k+1)}})^m)$ time.

Proof. The algorithm is exactly the same as ours for k-regular graphs: we choose an arbitrary maximal independent set I and from each vertex of I we obtain a number of subinstances. Then, compute the number of forests in every subinstance we get in the end.

For each $i \in \{0, \ldots, k\}$, let n_i denote the number of vertices in I of degree i. Then, the number of subinstances we get in the end is $\prod_{i=0}^{k}(2^i - i)^{n_i}$, and each of these instances has $m - \sum_{i=0}^{k} in_i$ edges. Therefore, up to a polynomial factor, the running time is bounded by

$$\begin{aligned}\prod_{i=0}^{k}(2^i - i)^{n_i} \times 2^{m-\sum_{i=0}^{k} in_i} &= 2^m \frac{\prod_{i=0}^{k}(2^i - i)^{n_i}}{2^{\sum_{i=0}^{k} in_i}} = 2^m \prod_{i=0}^{k}\left(\frac{2^i - i}{2^i}\right)^{n_i} = 2^m \prod_{i=0}^{k}\left(1 - \frac{i}{2^i}\right)^{n_i} \\ &\leq 2^m \prod_{i=0}^{k}\left(1 - \frac{k}{2^k}\right)^{n_i} = 2^m\left(1 - \frac{k}{2^k}\right)^{\sum_{i=0}^{k} n_i} = 2^m\left(1 - \frac{k}{2^k}\right)^{|I|} \\ &\leq 2^m\left(1 - \frac{k}{2^k}\right)^{\frac{n}{k+1}} \leq 2^m\left(1 - \frac{k}{2^k}\right)^{\frac{2m}{k(k+1)}}.\end{aligned}$$

Here in the second last inequality we applied Lemma 3.2 and in the last inequality we used the fact that $2m \leq kn$ (a consequence of double-counting). □

3.2 Unit interval graphs

Theorem 2.8 states that counting the number of forests in a chordal graph is #P-complete. The main goal of this section should have been to give a faster (exponential-time) algorithm for chordal graphs, but so far attempts were not that successful. Therefore, we focus on a subclass of the chordal graphs, namely, the class of unit interval graphs.

A graph $G = (V, E)$ is a *unit interval graph* if there exist a family $\mathcal{I} = \{I_1, \ldots, I_n\}$ of unit closed intervals on a line and a bijection $\psi\colon V \to \mathcal{I}$ such that $\{u, v\} \in E$ if and only if $\psi(u) \cap \psi(v) \neq \emptyset$. For a unit interval graph G, the set $\mathcal{I}$ of unit intervals as in the definition is called the *unit interval representation* of G. We can determine whether a given graph is a unit interval graph or not, and if so generate a unit interval representation of the graph in linear time [5]. Therefore, for our purpose, we may assume that a unit interval graph is given through a unit interval representation $\mathcal{I}$ of it.

The main result of this section is as follows.

Theorem 3.5. The number of forests in a unit interval graph can be counted in $O^*(1.9706^m)$ time.

Proof. Let $G = (V, E)$ be a unit interval graph and fix a unit interval representation $\mathcal{I}$ of it with the corresponding bijection ψ. First of all, we may assume that G is 2-connected (namely it is connected and the removal of any vertex does not make it disconnected) since the number of forests in a graph is the product of the numbers of forests of all 2-connected components (i.e., maximal 2-connected subgraphs). Then, we make the following preprocessing. We look at the leftmost interval I_1 in $\mathcal{I}$, and collect the intervals in $\mathcal{I}$ which intersect I_1. Denote by C_1 the vertices in G corresponding to the collected intervals. Now, we dispose the collected intervals from $\mathcal{I}$ and look for the leftmost interval I_2 in the remaining $\mathcal{I}$, collecting the intervals in $\mathcal{I}$ which intersect I_2. Denote by C_2 the vertices in G corresponding to the collected intervals. We dispose the collected intervals from $\mathcal{I}$, and proceed along the same way. Thus, we obtain a partition $\{C_1, \ldots, C_k\}$ of the vertex set V, which we call the *clique partition* of G (with respect to $\mathcal{I}$), satisfying the following properties.

1. For each $i \in \{1, \ldots, k\}$, the set C_i is a clique of G.

2. For each $i, j \in \{1, \ldots, k\}$, $i < j$, there exists an edge between C_i and C_j if and only if $j = i + 1$.

Note that the clique partition of G can be obtained in linear time [5].

An edge $e \in E$ is called *non-bridging* if it connects two vertices of some C_i. Otherwise, the edge is *bridging*. From the construction and the assumption that G is 2-connected, we may observe that $|C_i| \geq 3$ for each $i \in \{1, \ldots, k-1\}$, and $|C_k| \geq 1$. The following is an important lemma for our algorithm.

Lemma 3.6. Under the assumption above, the number of bridging edges in G is at most $2m/3$, where m is the number of edges in G.

Proof. Let n_i be the size of C_i. When $k = 1$, we have no bridging edge; Thus the lemma holds.

To illustrate the general case, let us first consider when $k = 2$. Then, we have to show that the number of bridging edges is at most two thirds times $\binom{n_1}{2} + \binom{n_2}{2}$ plus the number of bridging edges. Since the number of bridging edges is at most $(n_1 - 1)n_2$ by construction, it suffices to show that $(n_1 - 1)n_2 \leq n_1(n_1 - 1) + n_2(n_2 - 1)$. This inequality always holds, and we are done for this case.

For general k, the number of bridging edges is at most $\sum_{i=1}^{k-1}(n_i - 1)n_{i+1}$ and the number of non-bridging edges is exactly $\sum_{i=1}^{k}\binom{n_i}{2}$. By the same argument as the case $k = 2$, it suffices to show that $\sum_{i=1}^{k-1}(n_i - 1)n_{i+1} \leq \sum_{i=1}^{k} n_i(n_i - 1)$. This can be shown as follows with noting that $x^2 + y^2 \geq 2xy$ for all $x, y \in \mathbb{R}$ and $x^2/2 - x \geq 0$ for all $x \geq 2$:

$$\begin{aligned}
\sum_{i=1}^{k} n_i(n_i - 1) &= \sum_{i=1}^{k} n_i^2 - \sum_{i=1}^{k} n_i = \sum_{i=1}^{k-1}(n_i^2/2 + n_{i+1}^2/2) + n_1^2/2 + n_k^2/2 - \sum_{i=1}^{k} n_i \\
&\geq \sum_{i=1}^{k-1} n_i n_{i+1} + n_1^2/2 + n_k^2/2 - n_1 - \sum_{i=2}^{k} n_i \geq \sum_{i=1}^{k-1} n_i n_{i+1} - \sum_{i=2}^{k} n_i \\
&\geq \sum_{i=1}^{k-1} n_i n_{i+1} - \sum_{i=1}^{k-1} n_{i+1} = \sum_{i=1}^{k-1} n_{i+1}(n_i - 1).
\end{aligned}$$

Thus the lemma is verified. □

We now describe our algorithm. The correctness again follows from the contraction-deletion formula.

1. Compute a clique partition $\{C_1, \ldots, C_k\}$ of G.
2. Enumerate all forests of the subgraph $G[C_i] = (C_i, E_i)$ of G induced by C_i for all $i \in \{1, \ldots, k\}$.
3. For each choice of the forests $F_1, \ldots, F_k$ from $G[C_1], \ldots, G[C_k]$
   (a) construct the graph G' from G by deleting the edges in $E_1 \setminus F_1, \ldots, E_k \setminus F_k$ and contracting the edges in $F_1, \ldots, F_k$.
   (b) Compute $|\mathcal{F}(G')|$ by the contraction-deletion formula.
4. Output the sum of the $|\mathcal{F}(G')|$'s computed in the previous step.

To bound the running time, we need to estimate the number of forests in $G[C_i]$ (for Step 2), and the number of edges in G' (for Step 3). From Lemma 3.6 we already know that G' has at most $2m/3$ edges since all edges in G' were bridging edges of G. Thus, it suffices to resolve the former one.

The number of forests in $G[C_i]$ is at most $\sum_{j=0}^{n_i-1}\binom{\binom{n_i}{2}}{j}$. So the number of exhaustive search executions can be bounded by $\prod_{i=1}^{k}\sum_{j=0}^{n_i-1}\binom{\binom{n_i}{2}}{j}$. The following lemma gives an estimate.

Lemma 3.7. For $n \geq 3$, it holds that

$$\left(\sum_{j=0}^{n-1}\binom{\binom{n}{2}}{j}\right)^{1/\binom{n}{2}} \leq 7^{1/3}.$$

Proof. Set $f(n) = (\sum_{j=0}^{n-1} \binom{\binom{n}{2}}{j})^{1/\binom{n}{2}}$. A direct calculation shows $f(3) = 7^{1/3} \geq 1.9129, f(4) = 42^{1/6} \leq 1.8644, f(5) = 386^{1/10} \leq 1.8141, f(6) = 13212^{1/15} \leq 1.8825, f(7) = 82160^{1/21} \leq 1.7141$. So, it suffices to show $f(n) \leq 1.9$ for $n \geq 8$.

For simplicity, let $z = \binom{n}{2}$. Since $n \geq 8$, we have $z \geq 28$. Let $g(z) = (\sum_{j=0}^{\sqrt{2z}} \binom{z}{j})^{1/z}$, then we have $f(n) \leq g(z)$ where $z = \binom{n}{2}$. By using the bound $\sum_{i=0}^{b} \binom{a}{i} \leq (ea/b)^b$, we obtain

$$g(z) = \left(\sum_{j=0}^{\sqrt{2z}} \binom{z}{j} \right)^{1/z} \leq \left(\left(\frac{ez}{\sqrt{2z}} \right)^{\sqrt{2z}} \right)^{1/z} = \left(\frac{e}{\sqrt{2}} \sqrt{z} \right)^{\sqrt{2/z}}.$$

Let $h(z) = (\frac{e}{\sqrt{2}}\sqrt{z})^{\sqrt{2/z}}$. We have the monotonicity: $h(z') \geq h(z)$ for $z \geq z' \geq 28$. Therefore, $g(z) \leq h(z) \leq h(28) < 1.9$. This completes the proof. □

Armed with Lemma 3.7, we may bound the running time from above as follows. Let m' be the number of edges in G'. Since $m' \leq 2m/3$, the running time is at most

$$\begin{aligned}
\prod_{i=1}^{k} \sum_{j=0}^{n_i-1} \binom{\binom{n_i}{2}}{j} \times O^*(2^{m'}) &= \prod_{i=1}^{k} \left(\left(\sum_{j=0}^{n_i-1} \binom{\binom{n_i}{2}}{j} \right)^{1/\binom{n_i}{2}} \right)^{\binom{n_i}{2}} \times O^*(2^{m'}) \\
&\leq (7^{1/3})^{\sum_{i=1}^{k} \binom{n_i}{2}} O^*(2^{m'}) \\
&= (7^{1/3})^{m-m'} O^*(2^{m'}) \\
&\leq O^*(7^{m/9} 2^{2m/3}) = O^*(1.9706^m).
\end{aligned}$$

This completes the proof of Theorem 3.5. □

4 Extension to the Tutte polynomials

The *Tutte polynomial* of an undirected graph $G = (V, E)$ is a two-variate polynomial $T(G; x, y)$. A standard reference for Tutte polynomials is a book by Welsh [14]. It is well-known that the Tutte polynomial can be defined via the following contraction-deletion formula:

$$T(G; x, y) = \begin{cases} 1 & \text{if } G \text{ has no edge,} \\ xT(G/e; x, y) & \text{if an edge } e \text{ is an isthmus of } G, \\ yT(G \backslash e; x, y) & \text{if an edge } e \text{ is a loop of } G, \\ T(G/e; x, y) + T(G \backslash e; x, y) & \text{if an edge } e \text{ is neither an isthmus nor a loop of } G, \end{cases}$$

where an *isthmus* of a graph is an edge whose removal increases the number of connected components. Note that $T(G; 2, 1)$ is equal to the number of forests in G.

In this section, we discuss how the method of this paper can easily be generalized to the Tutte polynomial computation.

4.1 Regular graphs and bounded-degree graphs

Let $G = (V, E)$ be a 3-regular graph. The basic idea is the same as the algorithm from Section 3.1. However, we need a little change. To this end we introduce a notation. For an edge $e \in E$, we may rewrite the contraction-deletion formula above as follows:

$$T(G; x, y) = \begin{cases} 1 & \text{if } G \text{ has no edge,} \\ \alpha_e(x, y)T(G/e; x, y) + \beta_e(x, y)T(G \backslash e; x, y) & \text{otherwise,} \end{cases}$$

where $\alpha_e(x, y)$ and $\beta_e(x, y)$ depend on the edge e. If e is an isthmus of G, we set $\alpha_e(x, y) = x$ and $\beta_e(x, y) = 0$; if e is a loop of G, we set $\alpha_e(x, y) = 0$ and $\beta_e(x, y) = y$; otherwise we set $\alpha_e(x, y) = \beta_e(x, y) = 1$.

Consider an arbitrary vertex $v \in V$ and the edges $e_1, e_2, e_3 \in E$ incident to v. By applying the rule above, we may write $T(G; x, y)$ as

$$\begin{aligned}T(G;x,y) =& f_{123}(x,y)T(G/e_1/e_2/e_3;x,y) + f_{12}(x,y)T(G/e_1/e_2\backslash e_3;x,y)\\ &+ f_{13}(x,y)T(G/e_1\backslash e_2/e_3;x,y) + f_1(x,y)T(G/e_1\backslash e_2\backslash e_3;x,y)\\ &+ f_{23}(x,y)T(G\backslash e_1/e_2/e_3;x,y) + f_2(x,y)T(G\backslash e_1/e_2\backslash e_3;x,y)\\ &+ f_3(x,y)T(G\backslash e_1\backslash e_2/e_3;x,y) + f_\emptyset(x,y)T(G\backslash e_1\backslash e_2\backslash e_3;x,y),\end{aligned}$$

with some coefficient $f_S(x, y)$ for each $S \subseteq \{e_1, e_2, e_3\}$ (here we use the abbreviation $f_{12}(x, y)$ instead of writing $f_{\{e_1,e_2\}}(x, y)$ for example). Note that the value of $f_S(x, y)$ only depends on x, y and a local structure of G around v. Hence, for each S we can determine $f_S(x, y)$ in polynomial time. Therefore the values of $f_S(x, y)$ for all can be obtained in polynomial time.

The following is our algorithm to evaluate the Tutte polynomial of G at an arbitrarily given point (x, y).

1. Choose an arbitrary maximal independent set I of G.
2. Output the value returned by the call to $A(G, I)$.

Below is a description of $A(G, I)$.

1. If I is non-empty,
   (a) choose an arbitrary vertex $v \in I$. Let e_1, e_2, e_3 be the edges incident to v.
   (b) calculate $A(G/e_1/e_2/e_3, I \setminus \{v\})$, $A(G/e_1/e_2\backslash e_3, I \setminus \{v\})$, $A(G/e_1\backslash e_2/e_3, I \setminus \{v\})$, $A(G\backslash e_1/e_2/e_3, I \setminus \{v\})$ and $A(G\backslash e_1\backslash e_2\backslash e_3, I \setminus \{v\})$.
   (c) Let $E_v = \{e_1, e_2, e_3\}$ (for notational convenience). For each subset $S \subseteq E_v$ determine the coefficient $f_S(x, y)$ in the formula $T(G; x, y) = \sum_{S\subseteq E_v} f_S(x, y)T(G/S\backslash(E_v \setminus S)); x, y)$.
   (d) Output $\sum_{S\subseteq E_v} f_S(x, y)A(G/S\backslash(E_v \setminus S); x, y)$ using the identity $A(G\backslash e_1\backslash e_2\backslash e_3, I \setminus \{v\}) = A(G/e_1\backslash e_2\backslash e_3, I \setminus \{v\}) = A(G\backslash e_1/e_2\backslash e_3, I \setminus \{v\}) = A(G\backslash e_1\backslash e_2/e_3, I \setminus \{v\})$.
2. Otherwise, compute $T(G; x, y)$ by the contraction-deletion formula and output it.

The correctness argument goes along the same line as Section 3.1. As for the running time analysis, we only need to observe that the number of subinstances we get in the end is at most $5^{|I|}$. Thus, the analysis is verbatim.

Since the generalization to graphs of maximum degree k is also verbatim, we obtain the following theorem.

Theorem 4.1. For any fixed $k \geq 2$, we can compute the Tutte polynomial of a graph of maximum degree k in $O^*((2(1 - \frac{k}{2^k})^{\frac{2}{k(k+1)}})^m)$ time. In particular, the Tutte polynomial of a 3-regular graph can be computed in $O^*(1.8494^m)$ time.

4.2 Unit interval graphs

It is easy to see that the Tutte polynomial of a graph G is the product of the Tutte polynomials of the 2-connected components of G. Hence, we may assume that our unit interval graph $G = (V, E)$ is 2-connected. Let $\mathcal{I}$ be a unit interval representation of G with the corresponding bijection ψ. Similarly to the algorithm given in Section 3.2, we compute the Tutte polynomial, evaluated at an arbitrarily given point (x, y) in the following way. However, here we have to deal with isthmuses carefully. Let $\{C_1, \ldots, C_k\}$ be a clique partition of G, and $F_1, \ldots, F_k$ be forests of $G[C_1], \ldots, G[C_k]$ respectively. The algorithm given in Section 3.2 constructed a graph G' by contracting the edges in F_i and deleting the edges in $E(G[C_i]) \setminus F_i$ for all i and then computed the number of forests in G' in a naive way. Since the Tutte polynomial is independent from the order of contraction/deletion operations performed on the edges, we may first delete the edges in $E(G[C_i]) \setminus F_i$ and then contract the edges in F_i according to some order. Some of the edges in F_i can be isthmuses in the course of successive contractions, and we need to multiply x to the Tutte polynomial per encountered isthmus. Namely, we compute the Tutte polynomial of the obtained graph G' and output the polynomial multiplied by x^h where h is the total number of isthmuses we encountered.

Below is a more formal description of our algorithm.

1. Compute a clique partition $\{C_1, \ldots, C_k\}$ of G.

2. Enumerate all forests of the subgraph $G[C_i] = (C_i, E_i)$ of G induced by C_i for all $i \in \{1, \ldots, k\}$.

3. For each choice of the forests $F_1, \ldots, F_k$ from $G[C_1], \ldots, G[C_k]$

   (a) construct the graph G' from G by first deleting the edges in $E_1 \setminus F_1, \ldots, E_k \setminus F_k$ and then contracting the edges in $F_1, \ldots, F_k$.

   (b) Let h be the number of contracted isthmuses in the step above.

   (c) Compute $T(G'; x, y)$ by the contraction-deletion formula and store $x^h T(G'; x, y)$.

4. Output the sum of the $x^h T(G'; x, y)$'s computed in the previous step.

The correctness and the running time analysis go along the same line as Section 3.2. As a consequence, we obtain the following theorem.

Theorem 4.2. We can compute the Tutte polynomial of a unit interval graph in $O^*(1.9706^m)$ time.

5 Conclusion and open problems

We have seen #P-completeness results and fast (exponential-time) algorithms for the forest counting problem in some classes of graphs. We have further observed that the method can be generalized to the Tutte polynomial computation.

One of the major open questions is the complexity status of the forest counting (or the Tutte polynomial computation) for unit interval graphs. We do not even know that the problem is #P-complete or not for (not necessarily unit) interval graphs. For chordal graphs, we do not know any algorithm faster than the trivial $O^*(2^m)$-time algorithm. Finding such an algorithm seems a challenge.

Acknowledgments

The authors thank two anonymous referees for their valuable comments that improve the presentation of this paper significantly. The second author is partially supported by Grant-in-Aid for Scientific Research from Ministry of Education, Science and Culture, Japan, and Japan Society for the Promotion of Science.

References

[1] ANDRZEJAK, A. An algorithm for the Tutte polynomials of graphs of bounded treewidth. *Discrete Mathematics 190* (1998), 39–54.

[2] ANNAN, J. A randomised approximation algorithm for counting the number of forests in dense graphs. *Combinatorics, Probability and Computing 3* (1994), 273–283.

[3] GAREY, M., AND JOHNSON, D. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W.H. Freeman, 1979.

[4] GIMÉNEZ, O., HLINĚNÝ, P., AND NOY, M. Computing the Tutte polynomial on graphs of bounded clique-width. *SIAM Journal on Discrete Mathematics 20* (2006), 932–946.

[5] HERRERA DE FIGUEIREDO, C., MEIDANIS, J., AND PICININ DE MELLO, C. A linear-time algorithm for proper interval graph recognition. *Information Processing Letters 56* (1995), 179–184.

[6] JAEGER, F., VERTIGAN, D., AND WELSH, D. On the computational complexity of the Jones and Tutte polynomials. *Math. Proc. Camb. Phil. Soc. 108* (1990), 35–53.

[7] JERRUM, M. *Counting, Sampling and Integrating: Algorithms and Complexity*. Birkhäuser, Basel, 2003.

[8] KIRCHHOFF, G. Über die Auflösung der Gleichungen, auf welche man bei der Untersuchung der linearen Verteilung Galvanische Ströme geführt wird. *Annalen Physik Chemie 72* (1847), 497–508.

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

[10] SEKINE, K., IMAI, H., AND TANI, S. Computing the Tutte polynomial of a graph and the Jones polynomial of a knot of moderate size. In *Proc. 6th ISAAC* (1995), vol. 1004 of *Lecture Notes in Computer Science*, pp. 224–233.

[11] VADHAN, S. The complexity of counting in sparse, regular, and planar graphs. *SIAM Journal on Computing 31* (2001), 398–427.

[12] VERTIGAN, D. The computational complexity of Tutte invariants for planar graphs. *SIAM Journal on Computing 35* (2006), 690–712. Originally, a part of his doctoral thesis: 'The Computational Complexity of Tutte, Jones, Homfly, and Kauffman Invariants', DPhil thesis, Oxford University, Oxford, England, 1991.

[13] VERTIGAN, D., AND WELSH, D. The computational complexity of the Tutte plane: the bipartite case. *Combinatorics, Probability and Computing 1* (1992), 181–187.

[14] WELSH, D. *Complexity: Knots, Colourings and Counting*. Cambridge University Press, 1993.

[15] WEST, D. *Introduction to Graph Theory*, second ed. Prentice Hall, 2001.

[16] WILF, H. *Algorithms and Complexity*. Prentice Hall, 1986.