Fast Exponential-Time Algorithms for the Forest Counting in Graph Classes

Heidi Gebauer[1] **Yoshio Okamoto**[2]

[1]Department of Computer Science, ETH Zurich, Zurich, CH-8092, Switzerland

[2]Department of Information and Computer Sciences, Toyohashi University of Technology, Hibarigaoka 1–1, Tempaku, Toyohashi, Aichi, 441-8580, Japan
E-mail: okamotoy@ics.tut.ac.jp

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, unit interval graph

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 (Kirchhoff 1847) while it is #P-complete to count the number of forests in a graph, even in a bipartite planar graph (Vertigan & Welsh 1992). 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 (Jerrum 2003). 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 for the algorithm to run in polynomial time. Hence, we try to obtain a subexponential-time algorithm, or try to make the base of the exponential running time 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.

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 (Jaeger, Vertigan & Welsh 1990) showing that counting the number of forests in a graph is #P-complete. Vertigan (Vertigan 2006) proved that the problem is #P-complete for planar graphs, and Vertigan & Welsh (Vertigan & Welsh 1992) proved that it is #P-complete even for bipartite planar graphs.

On the algorithmic side, not much is known for the forest counting problem. Andrzejak (Andrzejak 1998) and Noble (Noble 1998) 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. Giménez, Hliněný & Noy (Giménez, Hliněný & Noy 2005) gave a subexponential-time algorithm in graphs of bounded clique-width, and Sekine, Imai & Tani (Sekine, Imai & Tani 1995) gave a subexponential-time algorithm in planar graphs.

For some counting problems in regular graphs, Vadhan (Vadhan 2001) 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.

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 $G \in \Gamma$;
Question: 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.

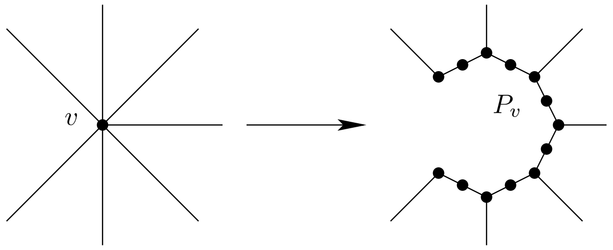

Figure 1: Replacing a vertex with a path (a local picture).

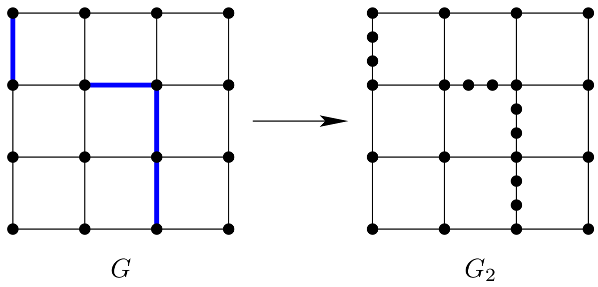

Figure 2: Replacing edges with paths. The blue thick edges belong to S, and each of them is replaced by a path of length three in G_2.

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. In particular, 3Δ-#FORESTS is #P-complete.

To prove the theorem, we use BP-#FORESTS, which is shown to be #P-complete by Vertigan & Welsh (Vertigan & Welsh 1992). 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 $S \subseteq E$; **Question**: the number of forests in G which contain S.

Lemma 2.2. The problem 3ΔBP-#FORESTS with inclusive edges is #P-complete.

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 two vertices 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 $2\ell - 1$. An example for $\ell = 2$ is displayed in Figure 2. Note that G_1 is isomorphic to G.

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 belongs 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 $\ell, \ell' \in \{1, \dots, s+1\}$, 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. □

2.2 Regular graphs

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

Theorem 2.3. 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. □

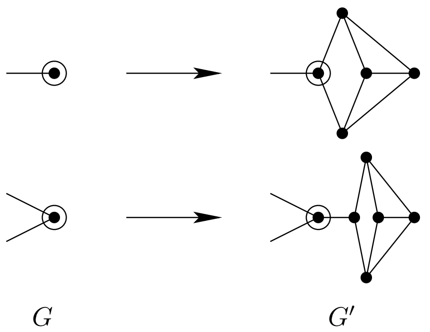

Figure 3: Attaching a graph to a degree-one vertex and a degree-two vertex.

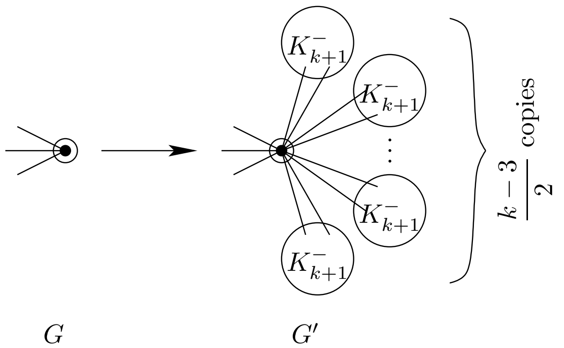

Figure 4: Attaching a graph to a degree-three vertex. Here K^-_{k+1} represents a complete graph on $k{+}1$ vertices with one edge removed, and two edges leaves each K^-_{k+1} from the vertices of degree $k-1$, i.e., the vertices incident to the removed edge.

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

Theorem 2.4. 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 2.4 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 $S \subseteq E$;
Question: the number of forests in G which do not contain any edges in S.

Lemma 2.5. 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 2.4.

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 $G' = (V, E \cup \{e_i \mid i \in \{1, \ldots, |V|/2\}\})$ is k-regular. We set $S = \{e_i \mid i \in \{1, \ldots, |V|/2\}$, the set of attached edges. Then we may observe that the forests of G is the forests of G' which contain no edge of S. This completes the reduction. □

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 2.6. 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 $\ell \in \{1, \ldots, s+1\}$ 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 $\ell \in \{1, \ldots, s+1\}$ 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 belongs 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 2.2, 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 $\ell, \ell' \in \{1, \ldots, s+1\}$, by knowing the numbers of forests in G_ℓ for all $\ell \in \{1, \ldots, s+1\}$ we can compute $a_0, \ldots, a_s$ 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 2.4 for even $k \geq 4$.

Proof of Theorem 2.4 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

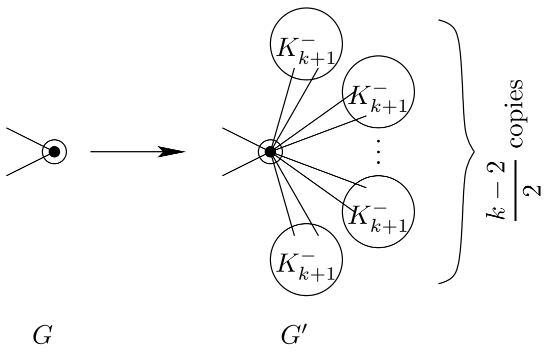

Figure 5: Attaching a graph to a degree-two vertex. Here K_{k+1}^- represents a complete graph on $k+1$ vertices with one edge removed, and two edges leaves each K_{k+1}^- from the vertices of degree $k-1$, i.e., the vertices incident to the removed edge.

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 2.4 is not planar unless $k = 3$.

2.3 Chordal graphs

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

Theorem 2.7. The problem $\mathsf{CHORDAL}$-#$\mathsf{FORESTS}$ is #P-complete.

To prove Theorem 2.7, we use the following lemma about exclusive edges.

Lemma 2.8. The problem $\mathsf{CHORDAL}$-#$\mathsf{FORESTS}$ with exclusive edges is #P-complete.

Proof. We use any graph class Γ such that Γ-#$\mathsf{FORESTS}$ is #P-complete. For example, set $\Gamma = \mathsf{BP}$. 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 2.7. We reduce $\mathsf{CHORDAL}$-#$\mathsf{FORESTS}$ with exclusive edges to $\mathsf{CHORDAL}$-#$\mathsf{FORESTS}$. 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 $e \in S$. An example for $\ell = 2$ is displayed in Figure 6. Note that G_0 is isomorphic to G.

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 $P_e^1, \dots, P_e^\ell$,

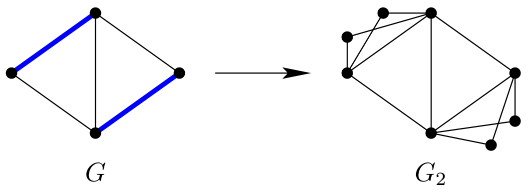

Figure 6: Joining paths of length two.

- when $e \in S \setminus F$, F_ℓ contains none of the paths among $P_e^1, \dots, P_e^\ell$, 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

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

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 $\ell, \ell' \in \{0, \dots, s\}$, by knowing the number of forests in G_ℓ for all $\ell \in \{0, \dots, s\}$ we can compute $a_0, \dots, a_s$ 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.

3 Algorithms

In this section, we concentrate on faster (exponential-time) algorithms for the forest counting problem. The trivial algorithm runs in $\mathrm{O}^*(2^m)$ time, and the goal is to beat this bound. Throughout the section, m denotes the number of edges in a given graph.

3.1 Regular graphs

We start with an algorithm for $\mathsf{3REG}$-#$\mathsf{FORESTS}$ (i.e., counting the number of forests in 3-regular graphs). The running time is $\mathrm{O}^*(1.8494^m)$.

For the analysis of our algorithm, we use the following simple lemma.

Lemma 3.1. Every maximal independent set of a k-regular graph with n vertices contains at least $n/(k+1)$ vertices.

Proof. Let $G = (V, E)$ be a k-regular graph 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, since each vertex of I is incident to k edges and since I is independent, these edges lie between I and $V \setminus I$. Therefore, the number of edges between I and $V \setminus I$ is $k|I|$. 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 $|I| \geq n/(k+1)$. □

Let G be a given 3-regular graph with n vertices. We first take an arbitrary maximal independent set I of G. Each vertex v of I is incident to exactly three edges, say, e_1, e_2, e_3. When we fix a forest F in G, exactly one of the following four is true.

1. No edge incident to v is contained in F.
2. Exactly one out of e_1, e_2, e_3 is contained in F.
3. Exactly two out of e_1, e_2, e_3 are contained in F.
4. All edges incident to v are contained in F.

This subdivision scheme gives an algorithm based on a search tree. When looking through all vertices in I, we will touch at least $3n/4 = m/2$ edges of G since $|I| \geq n/4$ by Lemma 3.1. So, at most $m/2$ edges are left untouched in G. By the exhaustive search, each of the left instances can be solved in $\mathrm{O}^*(2^{m/2})$ time.

Let us describe the subdivision scheme more precisely. For the edges e_1, e_2, e_3 incident to v in I, let $S = \{e_1, e_2, e_3\}$. For each subset $S' \subseteq S$, the algorithm tries to count the number of forests in G which contain S' and exclude $S \setminus S'$. The number of such forests is equal to the number of forests in G with S' contracted and $S \setminus S'$ deleted. An important observation is that the resulting graphs when $|S'| \leq 1$ are all identical (up to the existence of isolated vertices). Therefore, the subinstances we obtain from each vertex in I is at most five, and for each the number of edges decreases by three.

This gives the following recursion. Let $T(m)$ be the maximum number of nodes in the search tree created by the algorithm above when the input graph G has m edges. Then, we have $T(m) \leq \mathrm{O}^*(5^{n/4} \times 2^{m/2}) = \mathrm{O}^*(5^{m/6} \times 2^{m/2}) = \mathrm{O}^*(1.8494^m)$. Since each creation of subproblems can be done in polynomial time, we have proved the following theorem

Theorem 3.2. The problem 3REG-#FORESTS can be solved in $\mathrm{O}^*(1.8494^m)$ time.

For k-regular graphs G we 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, say, $e_1, e_2, \ldots, e_k$. When we fix a forest F in G, exactly one of the following $k+1$ conditions is true.

1. No edge incident to v is contained in F.
2. Exactly one out of $e_1, \ldots, e_k$ is contained in F.
3. Exactly two out of $e_1, \ldots, e_k$ are contained in F.

$\vdots$

k. Exactly $k-1$ of $e_1, \ldots, e_k$ are contained in F.

$k+1$. All edges incident to v are contained in F.

Note that the subinstances arising from Cases 1 and 2 are all identical. So the number of subinstances from each vertex in I is at most $2^k - k$, and the number of edges decreases by k. By Lemma 3.1, we can see that when we look through all vertices in I we will touch at least $kn/(k+1) = 2m/(k+1)$ edges of G. So at most $m - 2m/(k+1)$ edges are left untouched. By the exhaustive search, each of the left instances can be solved in $\mathrm{O}(2^{m-2m/(k+1)})$ time. Thus, by the same argument as Theorem 3.2, we 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 $\mathrm{O}^*((2^k - k)^{\frac{2m}{k(k+1)}} 2^{m - \frac{2m}{k+1}})$ 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.

3.2 Unit interval graphs

Theorem 2.7 states that counting the number of forests in a chordal graph is #P-complete. The main goal of this section should be to give a faster (exponential-time) algorithm for chordal graphs, but so far attempts are 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 set $\mathcal{I} = \{I_1, \ldots, I_n\}$ 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 (Herrera de Figueiredo, Meidanis & Picinin de Mello 1995). Therefore, for our purpose, we may assume that a unit interval graph is given through a unit interval representation $\mathcal{I}$ of it.

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 since the number of forests in a graph is the product of the numbers of forests for all 2-connected components. 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 $i = j + 1$.

Note that the clique partition of G can be obtained in linear time (Herrera de Figueiredo et al. 1995).

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 $|C_k| \geq 2$. The following is an important lemma for our algorithm.

Lemma 3.4. 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 $(n_1-1)n_2 \le n_1(n_1-1)+n_2(n_2-1)$. 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 $\sum_{i=1}^{k-1}(n_i-1)n_{i+1} \le \sum_{i=1}^{k} n_i(n_i-1)$. This can be shown as follows with noting that $x^2+y^2 \ge 2xy$ for all $x, y \in \mathbb{R}$ and $x^2/2 - x \ge 0$ for all $x \ge 2$:

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

Thus the lemma is verified. □

Having the clique partition $\{C_1, \ldots, C_k\}$ of G, we enumerate all forests of the subgraph $G[C_i]$ of G induced by C_i for all $i \in \{1, \ldots, k\}$. For each forest F_i of each C_i, we obtain the graph G' obtained from G by contracting each connected component of $F_1, \ldots, F_k$. On G' we make the exhaustive search. This is our algorithm. The correctness follows from the well-known contraction-deletion formula for the number of forests (or, the Tutte polynomial).

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 is at most $\prod_{i=1}^{k}\sum_{j=0}^{n_i-1}\binom{\binom{n_i}{2}}{j}$. The following lemma gives an estimate.

Lemma 3.5. For $n \ge 3$, it holds that

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

Proof. Set $f(n) = (\sum_{j=0}^{n-1}\binom{\binom{n}{2}}{j})^{1/\binom{n}{2}}$. A direct calculation shows $f(3) = 7^{1/3} \ge 1.9130, f(4) = 42^{1/6} \le 1.8644, f(5) = 386^{1/10} \le 1.8141, f(6) = 13212^{1/15} \le 1.8825, f(7) = 82160^{1/21} \le 1.7141$. So, it suffices to show $f(n) \le 1.9$ for $n \ge 8$.

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

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

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

Armed with Lemma 3.5, we may bound the running time from above as follows. Let m' be the number of edges in G', which is by construction the same as the number of bridging edges. Since $m' \le 2m/3$, the running time is at most

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

Thus, we conclude the following theorem.

Theorem 3.6. We can count the number of forests in a unit interval graph in $O^*(1.9706^m)$ time.

4 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. 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 faster algorithm than the trivial $O^*(2^m)$-time algorithm. Finding such an algorithm seems a challenge.

Acknowledgment

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

Andrzejak, A. (1998), 'An algorithm for the Tutte polynomials of graphs of bounded treewidth', *Discrete Mathematics* **190**, 39–54.

Giménez, O., Hliněný, P. & Noy, M. (2005), Computing the Tutte polynomial on graphs of bounded clique-width, *in* 'Proceedings of 31st International Workshop on Graph-Theoretic Concepts in Computer Science (WG 2005)', Vol. 3787 of *Lecture Notes in Computer Science*, pp. 59–68. Full version will appear in *SIAM Journal on Discrete Mathematics*.

Herrera de Figueiredo, C., Meidanis, J. & Picinin de Mello, C. (1995), 'A linear-time algorithm for proper interval graph recognition', *Information Processing Letters* **56**, 179–184.

Jaeger, F., Vertigan, D. & Welsh, D. (1990), 'On the computational complexity of the Jones and Tutte polynomials', *Math. Proc. Camb. Phil. Soc.* **108**, 35–53.

Jerrum, M. (2003), *Counting, Sampling and Integrating: Algorithms and Complexity*, Birkhäuser, Basel.

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

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

Sekine, K., Imai, H. & Tani, S. (1995), Computing the Tutte polynomial of a graph and the Jones polynomial of a knot of moderate size, *in* 'Proc. 6th ISAAC', Vol. 1004 of *Lecture Notes in Computer Science*, pp. 224–233.

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

Vertigan, D. (2006), 'The computational complexity of Tutte invariants for planar graphs', *SIAM Journal on Computing* **35**, 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.

Vertigan, D. & Welsh, D. (1992), 'The computational complexity of the Tutte plane: the bipartite case', *Combinatorics, Probability and Computing* **1**, 181–187.