

Counting independent sets in strongly orderable graphs

Marc Heinrich¹ and Haiko Müller*¹

¹School of Computing, University of Leeds, Leeds LS2 9JT, UK

January 7, 2021

Abstract

We consider the problem of devising algorithms to count exactly the number of independent sets of a graph G . We show that there is a polynomial time algorithm for this problem when G is restricted to the class of strongly orderable graphs, a superclass of chordal bipartite graphs. We also show that such an algorithm exists for graphs of bounded clique-width. Our results extends to a more general setting of counting independent sets in a weighted graph and can be used to count the number of independent sets of any given size k .

1 Introduction

Sometimes, knowing that there is a solution to a particular problem is not enough, and we would like instead to compute how many solutions there are. Ever since the work of Valiant [Val79a] introducing the complexity class $\#P$ which captures this type of problems, the counting version of many problems have been studied such as for example constraint satisfaction problems [Bul08], spanning trees [BP83], independent sets [Vad01], colourings [BDGJ99],...

Counting problems have been motivated in part by reliability questions. Indeed, the resilience of a network to certain attacks is connected to the number of subgraphs which satisfy certain properties [BP83], such as being connected for example. These problems are also motivated by some questions from statistical physics where models of particle interactions lead to computing some quantities called partition functions, which are often weighted generalisations of some form of counting problem. Finally, counting is also closely tied to the problem of random sampling [JVV86]: drawing a solution to a problem according to a certain (often uniform) distribution.

In this paper, we are interested in the problem of counting the number of independent sets in a graph. Ever since the work of [PB83], we know that this problem is $\#P$ -hard in general graphs. Many other results, both positive and negative were obtained on this problem for more restricted classes of graphs [Gre00, OUU08, FG04, DM19], and this paper follows into this direction by studying the complexity of the problem on two different classes of graphs. Our main result is an algorithm to compute exactly the number of independent sets in strongly orderable graphs, a superclass of chordal bipartite. This class was introduced by Dagan [Dra00a] as a way to study how the existence of particular elimination ordering in a graph influenced the complexity of various problems. Our second result is a dynamic programming algorithms to compute the number of independent sets in graphs of bounded clique-width. This class includes graphs of bounded treewidth, as well as other non-sparse classes of graphs such as cographs and distance hereditary graphs.

Existing work. There are several significant results on the problem of counting independent sets in a graph. The problem was first shown to be $\#P$ -hard, even for bipartite graphs in the work of Provan and Ball [PB83]. This result was later improved in several papers showing that the hardness

Work supported by EPSRC grants EP/S016562/1, "Sampling in hereditary classes".

holds for other classes of graphs such as graphs of maximum degree 3 [Gre00], planar graphs [Vad01] or comparability graphs [DM19]. From the point of view of parametrized complexity, counting the number of independent sets of a given size was shown to be $W[1]$ -hard [FG04], and hence is unlikely to have an FPT algorithm.

On the positive side, the problem has a polynomial time algorithm for several more restricted classes of graphs such as chordal graphs [OUU08], cocomparability graphs [DM19], graphs with bounded tree-width [WTZL18] or tolerance graphs [LS15], just to give a few examples. A picture representing all the known results, and the relations between the different classes can be found in Figure 1.

The problem of counting the number of matchings in a graph, *i.e.*, independent sets in the line-graph, has also been widely studied. Counting the number of perfect matchings in bipartite graphs was one of the first problem shown to be $\#P$ -complete by Valiant in her paper [Val79a] which introduced $\#P$ as a complexity class. This result was later extended to counting all matchings (not just the perfect ones) [Val79b]. Similarly to independent sets, the problem was studied for several classes of graphs, and was shown to be $\#P$ -hard on chordal and chordal bipartite graphs [OUU09]. Surprisingly, while it is also hard on planar graphs [DM19], the problem of counting only perfect matchings for this class is related to Pfaffian orientations and admits a polynomial time algorithm [Kas63].

The problem of counting the number of independent sets in a graph has also been studied from the point of view of approximation. Most of the results on the problem come in fact from statistical physics and the study of a particular model of particle interactions called the hard core model. This model can be seen as a weighted generalisation of counting the independent sets in the graph. More precisely, given a parameter λ , each independent set X has an associated weight of $\lambda^{|X|}$, and the partition function for the hard core model with fugacity λ corresponds to the following quantity:

$$\sum_{S \in \mathcal{I}(G)} \lambda^{|S|},$$

where $\mathcal{I}(G)$ is the set of all the independent sets of the graph G . We can observe that when the parameter λ is equal to 1, this is exactly counting the number of independent sets of G . We can also remark that the quantity above can be seen as a polynomial in λ , and the coefficients of this polynomial are the numbers of independent sets of G of a given size k for all possible values of k . Hence, asking to compute exactly this quantity for any parameter λ , is essentially the same as computing the number of independent sets of size k of the graph, for any possible k .

From the point of view of approximation, the complexity of this problem on graphs of maximum degree Δ is well understood. Indeed, there is a parameter $\lambda_c(\Delta) := (\Delta-1)^{\Delta-1}/(\Delta-2)^\Delta$, which has some physical interpretation (see [Wei06, GŠV16] for example), and the complexity of the problem depends only on whether $\lambda < \lambda_c$ or not. When $\lambda < \lambda_c$, the problem admits a FPTAS (Fully Polynomial-Time Approximation Scheme) [Wei06], *i.e.*, an algorithm which computes a $(1 + \varepsilon)$ -approximation of the result in time polynomial in both the size of the instance and $1/\varepsilon$. On the other hand, when $\lambda > \lambda_c$, no such approximation scheme exists, even if randomization is allowed, and assuming $NP \neq RP$ (this is the equivalent of P versus NP for randomized algorithms) [GŠV16, Sly10, MWW09, GGŠ⁺11]. Remark that when $\lambda = \lambda_c$ is exactly at the threshold the complexity is apparently still open. Note moreover that λ_c is larger than 1 when $\Delta \leq 5$, and is smaller than 1 when $\Delta \geq 6$. This implies that the number of independent sets of a graph of maximum degree Δ can be approximated in polynomial time when Δ is at most 5, but does not admit an approximation scheme when Δ is 6 or more. Finally, a last interesting fact is that approximating the number of independent sets for the class of bipartite graphs appears to have an intermediate complexity. More precisely, it is believed [CGG⁺16] that it does not admit an approximation scheme, nor is it AP-reducible to $\#SAT$, the problem of counting the number of satisfiable assignment of a boolean formula. This claim is supported by the fact that many problems were shown to be equivalent to approximating the number of independent sets in a bipartite graph, and they form a class sometimes called $\#BIS$.

Our result. The main result of this paper is a proof that the number of independent sets of a graph can be computed in polynomial time for strongly orderable graphs. This class (that we define in the next section) is a subclass of weakly chordal graphs, and contains the chordal bipartite graphs. In fact,

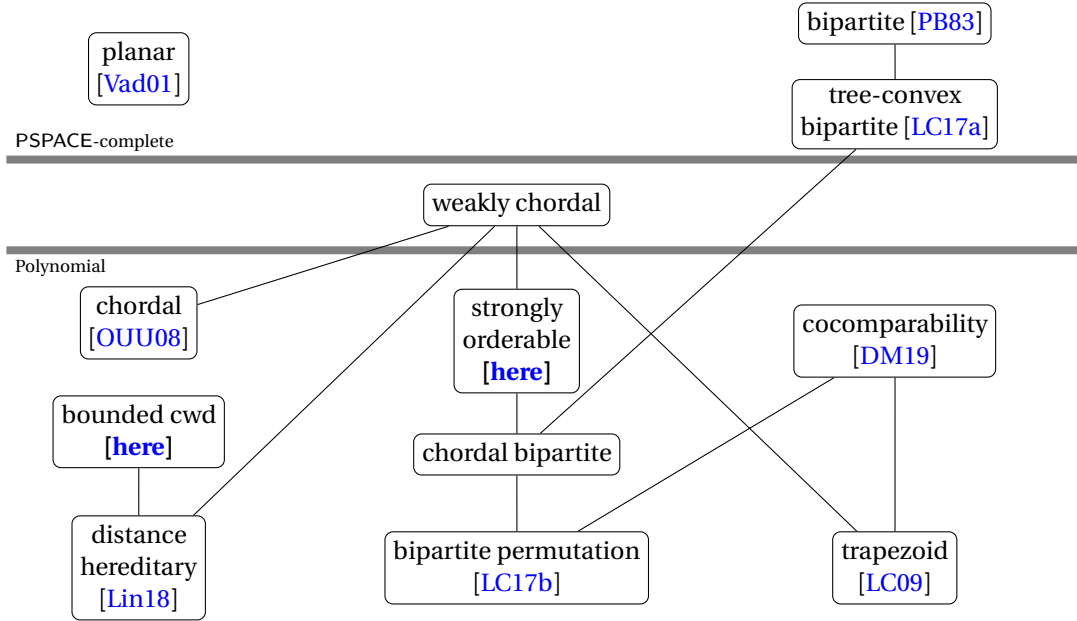


Figure 1: Complexity landscape for the problem of counting exactly the number of independent sets in graphs. The complexity for weakly chordal graphs is still open.

our algorithm applies to a slightly more general problem. Given a weight function w on the vertices of the graph G , our goal is to compute the following quantity:

$$\#\text{WIS}(G, w) = \sum_{S \in \mathcal{I}(G)} \prod_{v \in S} w(v).$$

Note that the number of independent sets of a graph corresponds to the case where all the weights are equal to 1. The partition function for the hard core model with fugacity λ corresponds to the case where all the weights are equal to λ . This quantity can be interpreted as a more general version of the partition function of the hard core model where each vertex v is given an individual fugacity of $w(v)$. Our main result is the following theorem:

Theorem 1. *$\#\text{WIS}(G, w)$ can be computed in polynomial time if G is a strongly orderable graph, and w is an arbitrary positive weight function.*

By the remark above, the case of counting the number of independent sets, or computing the partition function for this class of graph is also polynomial since these are special cases of the result above. If we compare this result to the problem of counting matchings, it appears to be somewhat surprising. Indeed, it is known that counting independent sets in bipartite graphs is $\#\text{P}$ -hard, and even the approximation of the problem seem unlikely to have a polynomial time approximation scheme. On the other hand, there are polynomial time approximation schemes for matchings in bipartite graphs, and counting the number of matching exactly is $\#\text{P}$ -hard. Hence, even though the counting problem appears to be easier for matchings than for independent sets on bipartite graphs, it is in fact harder when we restrict further to the class of chordal bipartite graphs. Our result proves that the situation for strongly orderable graphs is similar to that of chordal graphs in the sense that counting independent sets on this class is polynomial, while counting matchings is $\#\text{P}$ -hard.

In addition, our result is a significant improvement on [LC17b] which proved that a polynomial-time algorithm existed for computing the number of independent sets for bipartite permutation graphs, a subclass of chordal bipartite graphs. We also show that problem is polynomial for graphs of bounded clique-width:

Theorem 2. $\#WIS(G, w)$ can be computed in polynomial time if G has bounded bounded clique-width, and w is an arbitrary positive weight function.

This generalizes at the same time a result from [WTZL18] which proved it for graphs of bounded tree-width, and a result from [Lin18], which concerned distance-hereditary graphs which have clique-width at most 3. Again our algorithm works for the more general problem of the hard core model with fugacity λ , even when the fugacity is different for each vertex.

Overview The rest of the paper is organized as follows. In Section 2 we give some basic definitions from graph theory and formally define our problem. In Section 3 we first give an overview on how the proof for strongly orderable graphs works, and we then prove two technical lemmas which are used later in the proof. Section 4 contains the description of the algorithm and the proof for strongly orderable graphs. Finally, Section 5 proves our result for graphs of bounded clique-width.

2 Problem definition and notations

2.1 Graphs and graph classes

We start with some basic definitions and notations from graph theory, as well as the definitions of some classes of graphs that will be used in the rest of the paper.

For subsets $U, W \subseteq V$, let $U \times W := \{\{u, w\} \mid u \in U \text{ and } w \in W \setminus \{u\}\}$ denote the set of unordered pairs with one element in U and one in W . A graph $G = (V, E)$ is described by a finite set of vertices V , and a set of edges $E \subseteq V \times V$. Given two vertices $u, w \in V$, we will denote by $uw := \{u, w\}$ the edge between u and w . Unless stated otherwise all the graphs in this paper are simple, there are no loops and no multiple edges, and undirected; if $uw \in E$ then $wu \in E$.

Given a graph $G = (V, E)$, and a set of vertices $U \subseteq V$, we denote by $G[U]$ the subgraph induced by the vertices in U , i.e., $G[U] := (U, E \cap (U \times U))$. The notation $G \setminus U$ is shorthand for $G[V \setminus U]$ and for $v \in V$ we abbreviate $G \setminus \{v\}$ by $G \setminus v$. If U and W are disjoint subset of vertices, then $G[U, W]$ denotes the bipartite graph with colour classes U and W , and edges $E \cap (U \times W)$. Given a vertex v , we denote by $N_G(v)$ the neighbours of v , and $N_G^2(v)$ the vertices at distance 2, and we will drop the subscript G when the graph is clear from the context.

A **weight function** w on G is an assignment of non-negative rational weights to the vertices of G . We can see it as a function $w : V \rightarrow \mathbb{Q}$, but also as a vector of size $|V|$. We denote by $\bar{1}$ the weight functions equal to 1 for all the vertices of the graph. If H is a subgraph of G , then we will denote by $w|_H$ the restriction of w to the vertices of H .

The first class of graph that we consider are the chordal bipartite graphs, which are the bipartite equivalent of chordal graphs defined as follows:

Definition 3. A graph G is **chordal bipartite** if it is bipartite and does not contain any induced cycle of length 5 or more.

Similarly to chordal graphs, there are other characterizations of chordal bipartite graphs based on some particular ordering of its vertices, however these properties will not be needed in the rest of the paper. Instead we focus on a special subclass of chordal bipartite called chain graphs with the following definition:

Definition 4 ([Yan82]). A bipartite graph G is a **chain graph** if for every vertices u and v on the same side of the bipartition, we have either $N(u) \subseteq N(v)$ or the contrary $N(v) \subseteq N(u)$.

Chain graphs will play an important role in the proof of our main result which concerns graphs called strongly orderable. The latter class is a generalisation of both strongly chordal graphs and chordal bipartite graphs.

Definition 5 ([VBW92]). A graph G is **strongly orderable** if there exists an ordering of the vertices v_1, \dots, v_n such that for all $i < j$ and $k < l$, if $v_i v_k, v_i v_l$ and $v_k v_j$ are edges of G , then $v_j v_l$ is also an edge of G . An ordering satisfying this property is called a **strong ordering**.

Moreover, graphs in this class can be recognized in polynomial time, and a strong ordering can be computed in time $O(|E| \cdot |V|)$ [Dra00b]. Finally, the class of cographs will be used in the proofs.

Definition 6. *Cographs are graphs constructed starting from single vertices by the operations of:*

- *disjoint union: if (V, E) and (W, F) are cographs, then $(V \cup W, E \cup F)$ is a cograph;*
- *complete join: if (V, E) and (W, F) are cographs, then $(V \cup W, E \cup F \cup (V \times W))$ is a cograph.*

There are several other ways to characterize cographs. We will only need the following property in the rest of our proof:

Proposition 7 ([CLS81]). *A graph is a cograph if and only if it does not contain P_4 , a path on four vertices, as an induced subgraph.*

For further information on graphs in these restricted classes, their properties and references to original work see [Gol80, BLS99]. We will also need the concept of clique-width. The formal definition of this parameter can be found in Section 5.

2.2 Arithmetic circuit

Throughout this paper, an **arithmetic circuit** over the set of variables x_1, \dots, x_n is a directed acyclic graph such that each vertex is either:

- A vertex of in-degree zero, and labelled with either x_i for some $i \geq 0$, or some fixed constant $z \in \mathbb{Q}$,
- A vertex of in-degree two, and labelled with one of the four following operations: $+$, $-$, \times , $/$.

Note that in the case of subtraction and division gates, the order of the two inputs is important. Remark that this is not the standard definition of arithmetic circuit, as they usually do not include division gates. We include them here because they will play an important role in our proofs. A circuit C describes a function $\mathbb{Q}^n \rightarrow \mathbb{Q}^k$, where k is the number of outputs (*i.e.*, vertices with out-degree zero) which is a rational fraction: it is the quotient of two polynomial functions. In the following, a circuit will be called **positive** if it does not contain any subtraction gates, and all the constants in the circuit are positive. Positive circuits will play an important role as they behave nicely from the point of view of approximate computations.

2.3 Problem definition

Given a graph G , we denote by $\#IS(G)$ the number of independent sets in G . We will consider a more general problem of computing a weighted number of independent sets. Given a graph G , and a weight functions w , we will denote by $\#WIS(G, w)$ the number of weighted independent sets defined by the following quantity:

$$\#WIS(G, w) = \sum_{S \in \mathcal{I}(G)} \prod_{v \in S} w(v).$$

For a fixed graph G , the function $w \mapsto \#WIS(G, w)$ is a polynomial on n variables whose maximum degree is the size of the largest independent set in G . We will denote by P_G this polynomial, which is sometimes called the independence polynomial of the graph G [GH83]. Our goal is to investigate for which classes of graphs this quantity can be computed exactly. It is easy to see that computing the number of independent sets corresponds to the case where the weight function w is $\mathbf{1}$. More generally, computing the partition function of the hard core model with fugacity λ is also a special case of this problem where the weight function w is $\lambda \cdot \mathbf{1}$.

3 Outline and technical results

The proof of Theorem 1 proceeds in essentially two steps. The first step consists in building an arithmetic circuit computing the independence polynomial P_G by using the structural properties of strongly orderable graphs described in Lemma 8 below. The input to this circuit are the different coefficients of the weight function w , and it outputs $\#WIS(G, w)$ for the input graph G . Once this arithmetic circuit is built, one can think that evaluating it for a specific weight function w would be straightforward, unfortunately some technicalities force us to take a slight detour. Indeed, a direct evaluation of the circuit might not be possible since intermediate results might required a very large (super-polynomial) number of bits to be represented exactly. On the other hand, the presence of division gates implies that the standard trick of making all the computation modulo some (large enough) prime p does not work since this might lead to divisions by zero if we were not careful in the choice of the prime p .

Instead, our proof proceeds by first observing that the circuit built in the previous step is positive, and then use this property to evaluate the circuit approximately, up to a multiplicative $(1 + \varepsilon)$ factor. Finally, since the polynomial P_G computed by this circuit has a degree at most n , the exact value can be recovered from the approximation by a simple rounding, provided ε is chosen sufficiently small (but still requiring only a polynomial number of bits to represent).

We now proceed by proving two technical lemmas used in the proof of our theorem. The first one, Lemma 8 below gives the structural properties of strongly orderable graphs that we will need in the first step of the proof. The second one concerns the evaluation of positive arithmetic circuits using approximations which is used in the second step of the proof.

Lemma 8. *Let G be a strongly orderable graphs, and u be the first vertex in a strong ordering of G . Then:*

- $G[N(u)]$ is a cograph,
- $G[N(u), N^2(u)]$ is a chain graph.

Proof. Let us first consider the first point. Let us assume by contradiction that $N(u)$ is not a cograph. By Proposition 7 this means $N(u)$ contains an induced path P on four vertices. Let v_1, \dots, v_4 be the vertices in this path. Up to symmetry, we can assume that either v_1 or v_2 is the first vertex of P according to the strong ordering of G . We consider these two cases:

- v_1 is the first vertex of P according to the strong ordering of G . In this case, we know that uv_1 , uv_4 and v_1v_2 are edges of G , and u appears before v_2 , and v_1 appears before v_4 in the strong ordering of G . Hence it follows that v_2v_4 is an edge of G by the property of the strong ordering, a contradiction of the assumption that P is an induced path.
- v_2 is the first vertex of P according to the strong ordering of G . In this case, we know that uv_2 , uv_4 and v_2v_1 are edges of G , and u appears before v_1 , and v_2 appears before v_4 in the strong ordering of G . Hence it follows that v_1v_4 is an edge of G , contradicting again the assumption that P is an induced path.

In both cases we obtain a contradiction, hence the first property holds. Let us now prove the second point. Let u_1, u_2 be two vertices in $N(u)$ such that u_1 appears before u_2 in the strong ordering of G . Let v_1 be a neighbour of u_1 in $H := G[N(u), N^2(u)]$. Then by construction, we know that uu_1 , uu_2 and u_1v_1 are edges of G , and x appears before v_1 , and u_1 appears before u_2 in the strong ordering of G . By the property of the strong ordering, it follows that u_2v_1 is an edge of G . Since this holds for every neighbour of u_1 in H , it follows that $N_H(u_1) \subseteq N_H(u_2)$. Since this holds for any pair of vertices u_1, u_2 in $N(u)$, it implies that H is a chain graph. \square

The second technical result below allows us to evaluate positive arithmetic circuits.

Lemma 9. *Let C be a positive arithmetic circuit with n inputs and one output computing a polynomial P of degree at most d . Let x a positive rational vector of dimension n such that both x and $C(x)$ can be represented using n_b bits. Then $C(x)$ can be computed exactly in time polynomial in $|C|, n, d$ and n_b .*

Note that in the lemma above, the fact that the circuit is positive, and that $x > 0$ is important. Indeed, without these conditions, when doing the computations using floating point arithmetic, an important loss of relative precision can occur when subtracting two quantities of close values. Remark also that a direct evaluation of the circuit is not necessarily possible. Indeed, because there are some division gates, it is possible for the intermediate results to require more than a polynomial number of bits to be represented exactly, even if the end-result does not.

Proof. We will use floating point arithmetic to evaluate the circuit C on the input x . In the rest of the proof, b_m will represent the number of bits we use for the mantissa of the floating point numbers, and b_e is the number of bits used for the exponent. Since we know that each coefficient of x can be written using only n_b bits, it follows that the coefficient of x satisfy that for all $i \leq n$: $2^{-n_b} \leq x_i \leq 2^{n_b}$. Moreover, we make the following claim:

Claim 9.1. *All the intermediate values resulting from computing $C(x)$ are between $\frac{1}{A}$ and A where $A = 2^{n_b \cdot 2^{|C|}}$.*

Proof. We prove the result by induction on $|C|$. If $|C| = 0$, then the result holds immediately. Otherwise, consider the circuit C' obtained from C by removing one of the output gates (*i.e.*, with out-degree zero). Using the induction hypothesis, all the intermediate values of C' have size at most $2^{n_b \cdot 2^{|C|-1}}$. If the gate we just removed is an addition gate, then its output is at most $2^{n_b \cdot 2^{|C|-1} + 1}$. If it is a multiplication or a division gate, then it is at most $(2^{n_b \cdot 2^{|C|-1}})^2$. If it is a gate with in-degree zero, then the result holds immediately. The lower bounds holds using a similar argument. \square

A consequence of this fact is that if we use at least $b_e \geq \log(n_b) + |C| + 1 > \log(A)$ bits for the exponent, all the computation can be done using floating point arithmetic without any risk of overflow. Hence, the only thing that remains to be done is to investigate how large the mantissa needs to be to get a sufficient precision and retrieve the exact result in the end.

Let $\varepsilon = 2^{-b_m}$, and let $K = \frac{1+\varepsilon}{1-\varepsilon}$. Let us denote by $\tilde{+}, \tilde{\times}, \tilde{/}$ the operations $+, \times, /$ on floating point numbers. We know that for any operator $\circ \in \{+, \times, /\}$, and any values a and b represented as floating point numbers, we have $(1-\varepsilon)(a \circ b) \leq a \tilde{\circ} b \leq (1+\varepsilon)(a \circ b)$. In other words, this means that every time we perform a floating point operation, we loose at most a $1 \pm \varepsilon$ factor in the relative precision.

We will show by induction on $|C|$ that for all intermediate value y in the computation of $C(x)$, if \tilde{y} is the corresponding value computed using floating point arithmetic, then $yK^{-3^{|C|}} \leq \tilde{y} \leq yK^{3^{|C|}}$. If $|C| = 0$, then there are no gates in the circuit and the result trivially holds. Otherwise, consider the circuit C' obtained after deleting one of the output gates. We know that the induction hypothesis holds for C' . Moreover, let y be the exact output of the gate we just deleted, and \tilde{y} be its approximate output computed using floating point arithmetic. Then we have:

- If the gate we deleted is an addition gate, then let a and b be the exact values of its input, and \tilde{a} and \tilde{b} the approximate values computed using floating point arithmetic. Then we know that:

$$\tilde{y} = \tilde{a} \tilde{+} \tilde{b} \leq (1+\varepsilon)(\tilde{a} + \tilde{b}) \leq (1+\varepsilon)K^{3^{|C|-1}}(a+b) \leq K^{3^{|C|}} y,$$

where we used the induction hypothesis on \tilde{a} and \tilde{b} which are positive numbers and intermediate results in the circuit C' . In a similar fashion, we have:

$$\tilde{y} = \tilde{a} \tilde{+} \tilde{b} \geq (1-\varepsilon)(\tilde{a} + \tilde{b}) \geq (1-\varepsilon)K^{-3^{|C|-1}}(a+b) \geq K^{-3^{|C|}} y,$$

- If the gate we deleted was a multiplication gate, then again let a and b be its exact inputs, and \tilde{a} and \tilde{b} the approximate inputs computed using floating point arithmetic. Then it holds that:

$$\tilde{y} = \tilde{a} \tilde{\times} \tilde{b} \leq (1+\varepsilon)(\tilde{a} \times \tilde{b}) \leq (1+\varepsilon)K^{2 \cdot 3^{|C|-1}}(a+b) \leq K^{3^{|C|}} y,$$

and,

$$\tilde{y} = \tilde{a} \tilde{\times} \tilde{b} \geq (1-\varepsilon)(\tilde{a} \times \tilde{b}) \geq (1-\varepsilon)K^{-2 \cdot 3^{|C|-1}}(a+b) \geq K^{-3^{|C|}} y.$$

- Finally, if the gate was a division gate, we have in a similar way:

$$\tilde{y} = \tilde{a} / \tilde{b} \leq (1 + \varepsilon)(\tilde{a} / \tilde{b}) \leq (1 + \varepsilon)K^{2 \cdot 3^{|C|-1}}(a/b) \leq K^{3^{|C|}} y ,$$

and,

$$\tilde{y} = \tilde{a} / \tilde{b} \geq (1 - \varepsilon)(\tilde{a} / \tilde{b}) \geq (1 - \varepsilon)K^{-2 \cdot 3^{|C|-1}}(a/b) \geq K^{-3^{|C|}} y .$$

In particular, we can compute the output of the circuit up to a multiplicative factor of $K^{3^{|C|}} \leq 1 + 3^{|C|+1} \varepsilon$. Let $D \leq 2^{n_b \cdot n}$ be the least common multiple of all the denominators of the input vector x . Since the polynomial P computed by C has degree d , it follows that $D^d P(x)$ is an integer. Hence, we only need to choose ε such that the error term satisfies $y D^d 3^{|C|+1} \varepsilon < \frac{1}{2}$. Indeed, if this holds, then if we denote by \tilde{y} the output of the circuit computed using floating point arithmetic, and y is the exact output, then we have:

$$D^d y - \frac{1}{2} < D^d (1 - 3^{|C|+1} \varepsilon) y \leq D^d \tilde{y} \leq D^d (1 + 3^{|C|+1} \varepsilon) y < D^d y + \frac{1}{2} .$$

In particular, since $D^d y$ is an integer, and the fact that the interval above contains only a single integer, this value can be retrieved by rounding $D^d \tilde{y}$ to the closest integer.

Moreover, the inequality $y D^d 3^{|C|+1} \varepsilon < \frac{1}{2}$ holds as soon as $\frac{1}{\varepsilon} > 2y D^d 3^{|C|+1}$. Since we have $\varepsilon = 2^{-b_m}$, it is sufficient to take $b_m \geq \log(3) \cdot (|C| + 1) + \log(y D^d) + 1$ for this inequality to hold. Since $D^d y$ is a positive integer and is at most $2^{n_b \cdot n \cdot d + n_b}$, it follows that taking b_m greater than $\log(3) \cdot (|C| + 1) + n_b \cdot n \cdot d + 1$ is sufficient. This completes the proof and shows that we only need a polynomial number of bits to achieve the required precision for the exact result to be recovered at the end. \square

4 Proof of the Theorem

We now have all the tools we need to prove the Theorem 1. The proof first proceeds by showing how to construct a polynomial-size positive circuit computing the independence polynomial P_G . This circuit will be built using the strong ordering of the graph. Before we can construct the circuit for the whole graph, we start by considering the case where we restrict ourselves to a smaller class, which will then be used as a subroutine for the main construction.

Lemma 10. *If G is a cograph, then we can build in polynomial time a positive circuit of size $O(n)$ computing P_G .*

Proof. We will show by induction on the size of the graph G that there is a positive circuit computing $P_G - 1$, which is the polynomial corresponding to the weighted number of non-empty independent sets. The induction is done using the recursive definition of cographs in Definition 6. If G contains a unique vertex v_1 , then $P_G - 1 = x_1$, and trivially has a constant-size positive circuit.

If G is a disjoint union of two cographs G_1 and G_2 , then any independent set of G is the union of an independent set of G_1 and an independent set of G_2 . From this it follows that $P_G = P_{G_1} P_{G_2}$, and consequently we have: $P_G - 1 = (P_{G_1} - 1)(P_{G_2} - 1) + (P_{G_1} - 1) + (P_{G_2} - 1)$ and we can easily construct a circuit for $P_G - 1$ given circuits for both $P_{G_1} - 1$ and $P_{G_2} - 1$.

Finally, if G is a complete join of two cographs G_1 and G_2 , then any non-empty independent set of G is either a non-empty independent set of G_1 or a non-empty independent set of G_2 , but not both. This implies that $P_G - 1 = (P_{G_1} - 1) + (P_{G_2} - 1)$. This implies again that the circuit for $P_G - 1$ can be computed from the circuits for both $P_{G_1} - 1$ and $P_{G_2} - 1$.

This proves that the polynomial $P_G - 1$, and hence P_G has a positive circuit. This circuit has size $O(n)$ since at each step of the induction only a constant number of gates are added, and is positive since we use only addition and multiplication gates. \square

The main element of the proof will be the following lemma which consists in showing that, up to a modification of the weight function, we can safely remove the first vertex in a strong ordering of G . More precisely, we want to show that if v is the first vertex in the strong ordering of G , then there is a weight function w' of $G \setminus v$ such that $\#\text{WIS}(G, w) = K \cdot \#\text{WIS}(G \setminus v, w')$, where K and the coefficients of w' can be computed by positive circuits. If this holds, then constructing the positive circuit for the whole graph is simply a matter of iterating this procedure and removing the vertices one by one.

Let us first describe in more details how this weight function w' can be constructed. Let G be a strongly orderable graph, with v the first vertex in a strong ordering of G , and w a weight function on G . By Lemma 8 we know that $H = G[N(v), N^2(v)]$ induces a chain graph. We denote by v_1, \dots, v_k the vertices in $N(v)$ ordered according to the inclusion of their neighbourhoods in H . In other words, $N_H(v_i) \subseteq N_H(v_{i+1})$. Let us denote by G_i the graph induced by $\{v_1, \dots, v_{i-1}\} \setminus N(v_i)$. We will take the convention that G_0 is the empty graph, and $G_k = G[N(v)]$. We consider the weight function w' on $G \setminus v$ such that w' and w agree on $G \setminus N[v]$, and for all the vertices in $N(v)$:

$$w'(v_i) = w(v_i) \cdot (1 + w(v)) \cdot \frac{\#\text{WIS}(G_i, w|_{G_i})}{\#\text{WIS}(G_i, w'|_{G_i})}. \quad (1)$$

Note that in this definition, $w'(v_i)$ depends only on $w'(v_j)$ for $j < i$, and consequently this defines w' uniquely. Moreover, since G_i is a cograph, then by Lemma 10 it follows immediately that the coefficients of w' can be computed using only positive circuits. Moreover, the following result holds:

Lemma 11. *We have $\#\text{WIS}(G, w) = (1 + w(v)) \cdot \#\text{WIS}(G \setminus v, w')$.*

Proof. The proof proceeds by partitioning the set of independent sets of G , $\mathcal{I}(G)$, depending on how the independent sets intersect the vertex set of $G \setminus N[v]$. More precisely, let us denote by \mathcal{I}_j the set of independent sets $I \in \mathcal{I}(G \setminus N[v])$ such that j is the largest index such that v_j is not a neighbour of any vertex in I , with the convention that \mathcal{I}_0 are the independent sets adjacent to all the vertices of $N(v)$. Note that the $(\mathcal{I}_j)_{0 \leq j \leq k}$ is a partition of $\mathcal{I}(G \setminus N[v])$.

We can remark that if an independent set I of $G \setminus N[v]$ is in \mathcal{I}_j , then the vertices of $N[v]$ it is not adjacent to are exactly v_1, \dots, v_j . Indeed, since $G[N(v), N^2(v)]$ induces a chain graph, then any vertex $w \in G \setminus N[v]$ that is adjacent to v_j is also adjacent to v_i for all $i \geq j$. Hence, it follows from this observation that we can write

$$\mathcal{I}(G) = \bigcup_{j=0}^k \{S \cup I \mid S \in \mathcal{I}(G[\{v, v_1, \dots, v_j\}]) \text{ and } I \in \mathcal{I}_j\}.$$

In other words, this means that the independent sets of G are obtained by extending for all $j \geq 0$ the elements of \mathcal{I}_j with an independent set of $G[\{v, v_1, \dots, v_j\}]$. And conversely, any set obtained with this method is an independent set of G . Hence we have

$$\#\text{WIS}(G, w) = \sum_{j=0}^k \alpha_j \sum_{I \in \mathcal{I}_j} \prod_{u \in I} w(u),$$

where $\alpha_j = \#\text{WIS}(G[\{v, v_1, \dots, v_j\}])$. Again, an independent set $I \in \mathcal{I}(G[\{v, v_1, \dots, v_j\}])$ can be one of the following:

- I is the empty set,
- I is a singleton containing only v ,
- or I is an independent set containing v_i for some $i \leq j$ but no $v_{i'}$ for any $i' > i$.

This case distinction implies that the coefficient α_j can be rewritten as:

$$\alpha_j = 1 + w(v) + \sum_{i=1}^j w(v_i) \cdot \#\text{WIS}(G_i, w).$$

In this sum, the 1 corresponds to the empty set, and the term $w(v)$ corresponds to the independent set containing only v .

Using exactly the same argument as above for the graph $G - v$ using the weight function w' , we can write in a similar fashion:

$$\#\text{WIS}(G \setminus v, w') = \sum_{j=0}^k \alpha'_j \sum_{I \in I_j} \prod_{u \in I} w(u),$$

where $\alpha'_j = \#\text{WIS}(G[\{v_1, \dots, v_j\}])$. Again, these coefficients satisfy the equality:

$$\alpha'_j = 1 + \sum_{i=1}^j w'(v_i) \#\text{WIS}(G_i, w').$$

Hence, to prove the result, it is enough to show that for all j , $\alpha'_j = (1 + w(v))\alpha_j$, which is equivalent to proving that for all i , we have $w'(v_i) \cdot \#\text{WIS}(G_i, w') = (1 + w(v)) \cdot w(v_i) \cdot \#\text{WIS}(G_i, w)$, which corresponds exactly to the definition of w' from Equation (1). \square

Corollary 12. *There is an algorithm that, given a strongly orderable graph G , outputs a (polynomial size) positive circuit for P_G in polynomial time.*

Proof. The proof follows from applying the result of Lemma 9 repeatedly. More precisely, we show the result by induction on the number of vertices in G . If G is empty, $P_G = 1$ and the result trivially holds. Otherwise, let v be the first vertex in a strong ordering of G . Using the induction hypothesis, we can build a positive circuit C' which computes $P_{G \setminus v}$. Then, we know that for any weight function w , we can define a weight function w' as in Equation (1), and it follows from Lemma 11 that $\#\text{WIS}(G, w) = (1 + w(v))\#\text{WIS}(G \setminus v, w')$. We know by Lemma 8 that $G[N(v)]$ is a cograph. Hence, by definition of w' , and using Lemma 10, it follows that the coefficients of w' can be computed with a positive circuit from the coefficients of w . Plugging this into the inputs of the circuit C' , and dividing the result by $1 + w(v)$ creates a positive circuit which computes P_G . \square

The proof of the main theorem then follows immediately by combining the different lemmas together.

Proof of Theorem 1. By Corollary 12, we can build in polynomial time a positive circuit C which computes P_G . Since P_G has degree at most n , it follows from Lemma 9 that we can evaluate exactly and in polynomial time the output of the circuit C on any positive weight function w , and the result follows. \square

5 Graphs of bounded clique-width

We now prove that counting the weighted number of independent sets can be done in polynomial time for graphs of bounded clique-width. Let us start by giving a formal definition of the clique-width of a graph before proving this result.

Let Λ be a finite set of labels and let U be a (countable) set of vertices. Recursively we define Λ -expressions and the labelled graphs they describe. Here a **labelled graph** is a triple (V, E, λ) where (V, E) is a graph and $\lambda : V \rightarrow \Lambda$ assigns labels to the vertices. Λ -expressions are built using the four following operations:

Create a vertex. For every label $i \in \Lambda$ and every vertex $v \in U$ the string $\bullet_i(v)$ is a Λ -expression. It describes the labelled graph $(\{v\}, \emptyset, \lambda)$ with $\lambda(v) = i$.

Disjoint union. Let σ and τ be Λ -expressions describing (V, E, λ) and (W, F, μ) . If $V \cap W = \emptyset$ then $\sigma \oplus \tau$ is a Λ -expression. It describes the disjoint union of the two graphs described by σ and τ , in other words the graph $(V \cup W, E \cup F, \gamma)$ where:

$$\gamma(v) = \begin{cases} \lambda(v) & \text{if } v \in V, \\ \mu(v) & \text{if } v \in W. \end{cases}$$

Add edges. For different labels $i, j \in \Lambda$ and a Λ -expression σ the string $\eta_{i \times j}(\sigma)$ is a Λ -expression. It describes the graph where all the edges between colours i and j were added. Formally, if σ corresponds to the graph (V, E, λ) then $\eta_{i \times j}(\sigma)$ describes $(V, E \cup F, \lambda)$ where $F = \{vw \mid \lambda(v) = i \text{ and } \lambda(w) = j\}$.

Relabel. For different labels $i, j \in \Lambda$ and a Λ -expression σ the string $\rho_{i \rightarrow j}(\sigma)$ is a Λ -expression describing the graph where vertices labelled i were relabelled to j . Formally, if σ describes (V, E, λ) then $\rho_{i \rightarrow j}(\sigma)$ describes (V, E, μ) where:

$$\mu(v) = \begin{cases} j & \text{if } \lambda(v) = i, \\ \lambda(v) & \text{if } \lambda(v) \neq i. \end{cases}$$

The **clique-width** of a graph $G = (V, E)$, denoted by $\text{cwd}(G)$, is the minimum size of a set Λ such that there is a labelling $\lambda : V \rightarrow \Lambda$ and a Λ -expression describing (V, E, λ) .

If we additionally allow the empty string to be a Λ -expression then (\emptyset, \emptyset) is the only graph of clique-width zero. A graph $G = (V, E)$ has clique-width at most one if and only if $E = \emptyset$. The graph G has clique-width at most two if and only if G is a cograph. There is a polytime algorithm recognising graphs of clique-width at most three [CHL⁺12]. In general it is NP-complete to decide whether for given G and k , $\text{cwd}(G) \leq k$ holds [FRRS06]. In contrast to tree-width, the parameterised problem for clique-width is not known to be fixed parameter tractable.

The result we want to show is the following

Theorem 13. *There is an algorithm which, given a graph G , a Λ -expression of G with $|\Lambda| = \ell$ and a weight function w , computes $\#\text{WIS}(G, w)$ in time $O(2^{\ell} \ell^2 \text{poly}(n))$.*

The exponent in the polynomial in the result above is independent of the bound ℓ on the clique-width of the graph. This result implies Theorem 2. Indeed, if the Λ -expression is not given as input to the algorithm, then using the result from [OS06], a Λ -expression with $|\Lambda| \leq (2^{3 \text{cwd}(G)+1} + 1)$ can be computed in polynomial time (if $\text{cwd}(G)$ is constant). This implies that the algorithm above is still polynomial on graphs of bounded clique-width, even when the corresponding expression is not given to the algorithm, however the dependency in the clique-width then becomes a double exponential.

Proof. Let us start with a few notations. In all the rest of this proof, Λ will be a finite set of labels, and given a Λ -expression σ , we will denote by G_σ the labelled graph described by this expression. Moreover, if H a labelled graph with a labelling λ , and $\Gamma \subseteq \Lambda$ is a subset of labels, then $H(\Gamma)$ denotes the subgraph induced by the vertices with a label in Γ , in other words: $H(\Gamma) = H[\{v \mid \lambda(v) \in \Gamma\}]$.

Let G be the input graph with the weight function w , and let μ be the Λ -expression describing G . The algorithm will proceed using dynamic programming by computing, for some Λ -expression σ and for every different subsets of labels $\Gamma \subseteq \Lambda$ the quantity $\#\text{WIS}(G_\sigma(\Gamma), w)$ that will be denoted by $c(\sigma, \Gamma)$ to simplify notations. These quantities will be computed by induction using the recursive definition of the Λ -expressions.

The subexpressions of μ form a decomposition tree. Its leaves are the \bullet -nodes, exactly one for each vertex of G . Moreover the tree contains $n-1$ inner nodes with two children, these are the \oplus -nodes. All other nodes have exactly one child and are η - or ρ -nodes. There is always a decomposition tree such that each path between two \oplus -nodes (or an \oplus -node and the root) has $O(|\Lambda|^2)$ nodes of η - or ρ -type. In short, we can assume that the size of the decomposition tree is $O(|\Lambda|^2 n)$.

Let σ be a subexpression of μ , and $\Gamma \subseteq \Lambda$ be a subset of labels. Let us show how $c(\sigma, \Gamma)$ can be computed using the recursive structure of Λ -expressions:

Create a vertex. Assume first that $\sigma = \bullet_i(v)$ for some label $i \in \Lambda$ and a vertex $v \in U$. Then we have:

$$c(\sigma, \Gamma) = \begin{cases} 1 + w(v) & \text{if } i \in \Gamma, \\ 1 & \text{if } i \notin \Gamma. \end{cases}$$

Indeed, if $i \in \Gamma$ then G contains a single vertex v , and there are two possible independent sets: the empty set and the set containing v , and if $i \notin \Gamma$, then G is the empty graph and the empty set is the only independent set of G .

Disjoint union. Assume that $\sigma = \tau_1 \oplus \tau_2$, then we have:

$$c(\sigma, \Gamma) = c(\tau_1, \Gamma) \cdot c(\tau_2, \Gamma).$$

Indeed, since G_σ is the disjoint union of G_{τ_1} and G_{τ_2} , any independent set of G_σ is the union of an independent set of G_{τ_1} and an independent set of G_{τ_2} . And reciprocally, every set of this form is an independent set of G_σ . This holds even when the graph is restricted to the vertices with a label in Γ .

Add edges. Assume now that $\sigma = \eta_{i \leftrightarrow j}(\tau)$. Each independent set of $G_\sigma \langle \Gamma \rangle$ is independent in $G_\tau \langle \Gamma \setminus \{i\} \rangle$ or $G_\tau \langle \Gamma \setminus \{j\} \rangle$ and the independent sets of $G_\tau \langle \Gamma \setminus \{i, j\} \rangle$ are independent both in $G_\tau \langle \Gamma \setminus \{i\} \rangle$ and in $G_\tau \langle \Gamma \setminus \{j\} \rangle$. Therefore we have

$$c(\sigma, \Gamma) = c(\tau, \Gamma \setminus \{i\}) + c(\tau, \Gamma \setminus \{j\}) - c(\tau, \Gamma \setminus \{i, j\}).$$

Relabel. Finally, assume that $\sigma = \rho_{i \rightarrow j}(\tau)$. Then we have:

$$c(\sigma, \Gamma) = \begin{cases} c(\tau, \Gamma \setminus \{i\}) & \text{if } j \notin \Gamma, \\ c(\tau, \Gamma \cup \{i\}) & \text{if } j \in \Gamma. \end{cases}$$

This follows from the fact that relabelling the vertices only change the subset of labels Γ under consideration.

A dynamic programming algorithm evaluating this recurrence will perform at most $O(2^\ell \ell^2 n)$ steps where $\ell = |\Lambda|$. Each step consists only in performing some arithmetic operations on numbers represented by a polynomial number of bits, hence the whole algorithm runs in polynomial time, which completes the proof of the theorem. \square

6 Conclusion

We have shown that the number of independent sets of a graph can be computed in polynomial time for strongly orderable graphs and graphs of bounded clique-width. One natural question is whether these results can be generalized further. A natural class of graph to consider is the class of weakly chordal graphs for which the complexity is still open. One other remark is that our algorithm for bounded clique-width graphs is doubly exponential in the clique-width of the graph when a lambda expression is not given as input to the algorithm. This is due to the fact that there is currently no FPT algorithm to compute the clique-width of a graph. One way to get around this problem could be to try to adapt the algorithm for the rank-width, a parameter closely related to clique-width which can be computed in FPT time.

Acknowledgement

The authors are grateful to Martin Dyer for stimulating discussions.

References

- [BDGJ99] Russ Bublely, Martin Dyer, Catherine Greenhill, and Mark Jerrum. On approximately counting colorings of small degree graphs. *SIAM Journal on Computing*, 29(2):387–400, 1999.
- [BLS99] Andreas Brandstädt, Van Bang Le, and Jeremy P. Spinrad. *Graph classes: a survey*. SIAM Monographs on Discrete Mathematics and Applications. Society for Industrial and Applied Mathematics, Philadelphia, 1999.

- [BP83] Michael O. Ball and J. Scott Provan. Calculating bounds on reachability and connectedness in stochastic networks. *Networks*, 13(2):253–278, 1983.
- [Bul08] Andrei A. Bulatov. The complexity of the counting constraint satisfaction problem. In *International Colloquium on Automata, Languages, and Programming*, pages 646–661. Springer, 2008.
- [CGG⁺16] Jin-Yi Cai, Andreas Galanis, Leslie Ann Goldberg, Heng Guo, Mark Jerrum, Daniel Štefankovič, and Eric Vigoda. #BIS-hardness for 2-spin systems on bipartite bounded degree graphs in the tree non-uniqueness region. *Journal of Computer and System Sciences*, 82(5):690–711, 2016.
- [CHL⁺12] Derek G. Corneil, Michel Habib, Jean-Marc Lanlignel, Bruce A. Reed, and Udi Rotics. Polynomial-time recognition of clique-width ≤ 3 graphs. *Discrete Applied Mathematics*, 160(6):834–865, 2012.
- [CLS81] Derek G. Corneil, Helmut Lerchs, and Lorna Stewart Burlingham. Complement reducible graphs. *Discrete Applied Mathematics*, 3(3):163–174, 1981.
- [DM19] Martin Dyer and Haiko Müller. Counting independent sets in cocomparability graphs. *Information Processing Letters*, 144:31–36, 2019.
- [Dra00a] Feodor F Dragan. Strongly orderable graphs a common generalization of strongly chordal and chordal bipartite graphs. *Discrete applied mathematics*, 99(1-3):427–442, 2000.
- [Dra00b] Feodor F Dragan. Strongly orderable graphs: A common generalization of strongly chordal and chordal bipartite graphs. *Discrete Applied Mathematics*, 99(1–3):427–442, 2000.
- [FG04] Jörg Flum and Martin Grohe. The parameterized complexity of counting problems. *SIAM Journal on Computing*, 33(4):892–922, 2004.
- [FRRS06] Michael R. Fellows, Frances A. Rosamond, Udi Rotics, and Stefan Szeider. Clique-width minimization is NP-hard. In Jon M. Kleinberg, editor, *Proceedings of the 38th Annual ACM Symposium on Theory of Computing, Seattle, WA, USA, May 21–23, 2006*, pages 354–362. ACM, 2006.
- [GGŠ⁺11] Andreas Galanis, Qi Ge, Daniel Štefankovič, Eric Vigoda, and Linji Yang. Improved inapproximability results for counting independent sets in the hard-core model. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*, pages 567–578. Springer, 2011.
- [GH83] Ivan Gutman and Frank Harary. Generalizations of the matching polynomial. *Utilitas Mathematica*, 24(1):97–106, 1983.
- [Gol80] Martin Charles Golumbic. *Algorithmic Graph Theory and Perfect Graphs*. Academic Press, New York, 1980.
- [Gre00] Catherine Greenhill. The complexity of counting colourings and independent sets in sparse graphs and hypergraphs. *Computational Complexity*, 9(1):52–72, 2000.
- [GŠV16] Andreas Galanis, Daniel Štefankovič, and Eric Vigoda. Inapproximability of the partition function for the antiferromagnetic ising and hard-core models. *Combinatorics, Probability and Computing*, 25(4):500–559, 2016.
- [JVV86] Mark R Jerrum, Leslie G Valiant, and Vijay V Vazirani. Random generation of combinatorial structures from a uniform distribution. *Theoretical Computer Science*, 43:169–188, 1986.
- [Kas63] Pieter W. Kasteleyn. Dimer statistics and phase transitions. *Journal of Mathematical Physics*, 4(2):287–293, 1963.

- [LC09] Min-Sheng Lin and Yung-Jui Chen. Counting the number of vertex covers in a trapezoid graph. *Information Processing Letters*, 109(21-22):1187–1192, 2009.
- [LC17a] Min-Sheng Lin and Chien-Min Chen. Counting independent sets in tree convex bipartite graphs. *Discrete Applied Mathematics*, 218:113–122, 2017.
- [LC17b] Min-Sheng Lin and Chien-Min Chen. Linear-time algorithms for counting independent sets in bipartite permutation graphs. *Information Processing Letters*, 122:1–7, 2017.
- [Lin18] Min-Sheng Lin. Simple linear-time algorithms for counting independent sets in distance-hereditary graphs. *Discrete Applied Mathematics*, 239:144–153, 2018.
- [LS15] Min-Sheng Lin and Sheng-Huang Su. Counting independent sets in a tolerance graph. *Discrete Applied Mathematics*, 181:174–184, 2015.
- [MWW09] Elchanan Mossel, Dror Weitz, and Nicholas Wormald. On the hardness of sampling independent sets beyond the tree threshold. *Probability Theory and Related Fields*, 143(3-4):401–439, 2009.
- [OS06] Sang-il Oum and Paul Seymour. Approximating clique-width and branch-width. *Journal of Combinatorial Theory, Series B*, 96(4):514–528, 2006.
- [OUU08] Yoshio Okamoto, Takeaki Uno, and Ryuhei Uehara. Counting the number of independent sets in chordal graphs. *Journal of Discrete Algorithms*, 6(2):229–242, 2008.
- [OUU09] Yoshio Okamoto, Ryuhei Uehara, and Takeaki Uno. Counting the number of matchings in chordal and chordal bipartite graph classes. In *International Workshop on Graph-Theoretic Concepts in Computer Science*, pages 296–307. Springer, 2009.
- [PB83] J. Scott Provan and Michael O. Ball. The complexity of counting cuts and of computing the probability that a graph is connected. *SIAM Journal on Computing*, 12(4):777–788, 1983.
- [Sly10] Allan Sly. Computational transition at the uniqueness threshold. In *2010 IEEE 51st Annual Symposium on Foundations of Computer Science*, pages 287–296. IEEE, 2010.
- [Vad01] Salil P. Vadhan. The complexity of counting in sparse, regular, and planar graphs. *SIAM Journal on Computing*, 31(2):398–427, 2001.
- [Val79a] Leslie G. Valiant. The complexity of computing the permanent. *Theoretical Computer Science*, 8(2):189–201, 1979.
- [Val79b] Leslie G. Valiant. The complexity of enumeration and reliability problems. *SIAM Journal on Computing*, 8(3):410–421, 1979.
- [VBW92] Dirk J. Verschuur, A. J. Berkhout, and C. P. A. Wapenaar. Adaptive surface-related multiple elimination. *Geophysics*, 57(9):1166–1177, 1992.
- [Wei06] Dror Weitz. Counting independent sets up to the tree threshold. In *Proceedings of the thirty-eighth annual ACM Symposium on Theory of Computing*, pages 140–149, 2006.
- [WTZL18] Pengfei Wan, Jianhua Tu, Shenggui Zhang, and Binlong Li. Computing the numbers of independent sets and matchings of all sizes for graphs with bounded treewidth. *Applied Mathematics and Computation*, 332:42–47, 2018.
- [Yan82] Mihalis Yannakakis. The complexity of the partial order dimension problem. *SIAM Journal on Algebraic Discrete Methods*, 3(3):351–358, 1982.