# Counting weighted independent sets beyond the permanent

Martin Dyer*
School of Computing
University of Leeds
Leeds LS2 9JT, UK
m.e.dyer@leeds.ac.uk

Mark Jerrum*
School of Mathematical Sciences
Queen Mary University of London
Mile End Road, London E1 4NS, UK
m.jerrum@qmul.ac.uk

Haiko Müller*
School of Computing
University of Leeds
Leeds LS2 9JT, UK
h.muller@leeds.ac.uk

Kristina Vušković
School of Computing
University of Leeds
Leeds LS2 9JT, UK
k.vuskovic@leeds.ac.uk

15 May 2020

**Abstract**

Jerrum, Sinclair and Vigoda (2004) showed that the permanent of any square matrix can be estimated in polynomial time. This computation can be viewed as approximating the partition function of edge-weighted matchings in a bipartite graph. Equivalently, this may be viewed as approximating the partition function of vertex-weighted independent sets in the line graph of a bipartite graph.

Line graphs of bipartite graphs are perfect graphs, and are known to be precisely the class of (claw, diamond, odd hole)-free graphs. So how far does the result of Jerrum, Sinclair and Vigoda extend? We first show that it extends to (claw, odd hole)-free graphs, and then show that it extends to the even larger class of (fork, odd hole)-free graphs. Our techniques are based on graph decompositions, which have been the focus of much recent work in structural graph theory, and on structural results of Chvátal and Sbihi (1988), Maffray and Reed (1999) and Lozin and Milanič (2008).

## 1 Introduction

Independent sets are central objects of study in graph theory.[1] In general, finding a largest independent set is a very hard problem. Indeed, it is known to be hard to approximate the

*Work supported by EPSRC grants EP/S016562/1 and EP/S016694/1, "Sampling in hereditary classes".

[1]Definitions not given here appear in Section 1.1 below.

size of this set within a ratio $n^{1-\varepsilon}$ for graphs on $n$ vertices and any $\varepsilon > 0$, unless $\mathsf{P} = \mathsf{NP}$. This has led to an emphasis on studying this problem in particular classes of graphs.

In particular, there has been a focus on *hereditary* classes, that is, classes that are closed under vertex deletions. Equivalently, such a class can be defined by a (not necessarily finite) set of *forbidden (induced) subgraphs*.

*Matchings* are a particular case. A matching in a graph $G = (V, E)$ is an independent set in the *line graph* of the root graph $G$. ("Graph" will mean "simple undirected graph", unless otherwise stated.) Edmonds [14] showed that a maximum weighted matching in any graph can be found in polynomial time. Beineke [1] showed that line graphs can be characterised by nine forbidden subgraphs. Of these, the *claw* (see Fig. 1) seems the most important for algorithmic questions. Thus Minty [30] extended Edmonds' algorithm to the larger class of *claw-free* graphs.

Claw-free graphs have been studied extensively by several authors, including Chudnovsky and Seymour in a long sequence of papers culminating in [8]. These papers give a decomposition of claw-free graphs, which unfortunately is non-algorithmic. However, this has been simplified and extended in [15, 31] to give an efficient decomposition which supports finding a maximum weighted independent set.

In this paper we are concerned with counting problems, and for these it is important to distinguish weighted and unweighted (or unary weighted) variants, even more so than with optimisation problems. For example, there is an efficient approximation algorithm for counting unweighted matchings in a general graph, but the existence of an approximation algorithm for counting weighted matchings remains an open question. It is weighted counting problems that we focus on here. Hereditary classes are particularly suitable for counting problems, since they are self-reducible by vertex deletion.

Since claw-free graphs include line graphs, the #P-completeness result of Valiant [37] for matchings implies that *exact counting* of independent sets in polynomial time is unlikely. Even polynomial time *approximate* counting remains an open question for general line graphs in the weighted setting. However, building on an earlier pseudopolynomial algorithm of Jerrum and Sinclair [21], Jerrum, Sinclair and Vigoda [22] made an important breakthrough in approximate counting. They showed that approximately counting weighted *perfect matchings* in a *bipartite* graph is in polynomial time, the *permanent* approximation problem.

Our goal is to extend the result of [22] to larger classes of graphs. It might be expected that the right direction for this would be to matchings in general graphs but, as noted above, this remains an open problem, and a positive solution seems increasingly unlikely. An important requirement of the proof of [22] is that the graph should have no odd cycles, which places it precisely in the class of bipartite graphs. Indeed, Štefankovič, Vigoda and Wilmes [32] have given a family of nonbipartite graphs for which the algorithm of [22] does not run in polynomial time. Interestingly, from the viewpoint of this paper, they also show that weighted matchings in these graphs, and in a more general class of graphs that are "close to bipartite", can be counted in polynomial time, using the algorithm of [22] with a graph decomposition technique. In [32], this is the Gallai-Edmonds decomposition.

Here we take a different direction to generalise [22], regarding approximating the permanent as the problem of approximately counting weighted independent sets in line graphs of bipartite graphs. We show that these two problems are polynomial time equivalent. That approximating the permanent is reducible to approximately counting weighted independent sets is shown in Section 2.5, and that counting arbitrarily weighted independent sets in line graphs of bipartite graphs is reducible to approximating the permanent is shown in Section 2.4.4.

An important property of line graphs of bipartite graphs is that they are *perfect*. So it might be hoped that the appropriate generalisation of the result of [22] would be to counting independent sets in perfect graphs. That this class can be recognised in polynomial time was shown by Chudnovsky, Cornuéjols, Liu, Seymour and Vušković [9]. The maximum independent set in a perfect graph can also be found in polynomial time, using a convex optimisation algorithm of Grötschel, Lovász, and Schrijver [17], though no combinatorial algorithm is yet known for this problem.

However, approximately counting independent sets in perfect graphs appears intractable in general. Bipartite graphs are perfect, but approximately counting independent sets in bipartite graphs defines the complexity class #BIS. This class was introduced by Dyer, Goldberg, Greenhill and Jerrum [13], and hardness for the class has since been used as evidence for the intractability of various approximate counting problems.

Trotter [36] suggested the smaller class of *line-perfect* graphs. These are graphs whose line graph is perfect. Trotter showed that a graph is line-perfect if and only if it contains no odd cycle of size larger than three. Independent sets in the line graph (matchings in the root graph) appear a natural target for generalising [22] but, in fact, they are a proper subclass of those that we will consider here.

Line graphs of bipartite graphs have a simple set of forbidden subgraphs [19]. These are the claw, all odd holes and the *diamond* (see Fig. 1). See Maffray and Reed [28, Thm. 4], who also gave the corresponding result [28, Thm. 5] for line graphs of bipartite *multigraphs*. These have the claw, *gem*, *4-wheel* and odd holes (see Fig. 1) as forbidden subgraphs. Of these, excluding the claw and the odd holes appears important in extending the algorithm of [22]. This results from the "canonical paths" argument used in its proof. However, the diamond, gem or 4-wheel do not appear important in this respect.

Here we establish this claim. We extend the result of [22] to the class of graphs which excludes only claws and odd holes. This is essentially the class of *claw-free perfect* graphs. (See Section 2.3 below.) These form a main focus of this paper, and we show that the algorithm of [22] can be extended to approximate the total weight of independent sets for graphs in this class. The structure of graphs in this class was characterised by Chvátal and Sbihi [10] and Maffray and Reed [28]. They gave a polynomial time *decomposition* algorithm that splits the graph into simpler parts. We use their results to show that the algorithm of [22] can be applied directly to count weighted independent sets in (claw, odd hole)-free graphs, a slightly larger class than claw-free perfect graphs. Since line graphs of bipartite graphs are a proper subclass of (claw, odd hole)-graphs, this is a natural generalisation of the result of [22].

Chudnovsky and Plumettaz [5] have given a different decomposition of claw-free perfect

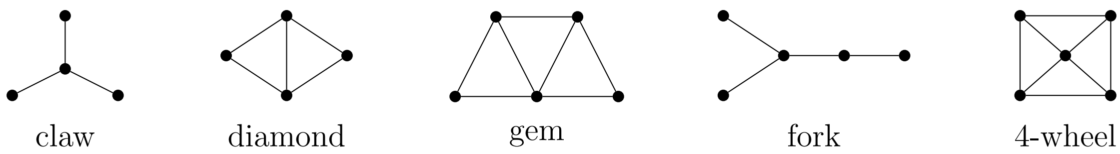



Fig. 1: Claw, diamond, gem, fork and 4-wheel

graphs, which has the additional property of *composability*. That is, the rules used to decompose the graph can be applied in reverse to create precisely the graphs in the class (and no more). Unfortunately, this results in a considerably more complex decomposition than that of [10, 28]. We make no use of this here, since we do not need composability. Moreover, [5] does not give a polynomial time algorithm for its decomposition. The ideas in [15, 31] would give a polynomial time decomposition, but it is unclear whether this supports counting.

In Section 2 we will develop a polynomial time algorithm for approximately counting all weighted independent sets in a (claw, odd hole)-free graph, generalising the algorithm of [22].

Observe that the algorithm of [12] runs in polynomial time, and counts all weighted independent sets in an arbitrary claw-free graph $G = (V, E)$ with unary weights. (See also Matthews [29].) This generalises Jerrum and Sinclair's matching algorithm [21].

So what do we achieve by restricting to (claw, odd hole)-free graphs? The gain is that our algorithm is genuinely polynomial time, whereas that of [12] is only pseudopolynomial. In particular, this allows us to approximate the total weight of independent sets of any given size $k$. We can estimate the total weight of *maximum* independent sets, which corresponds to counting maximum matchings in the root graph of a line graph. We then further relax the conditions on the class. We cannot relax first the odd hole condition, since this would take us into more general claw-free graphs, and might require counting matchings in general graphs. Therefore we consider relaxing the claw-free condition.

Lozin and Milanič [26] described a polynomial time algorithm for finding a maximum weighted independent set in a *fork-free* graph. That is, a graph with only the *fork* (see Fig. 1) as a forbidden subgraph. Clearly, this is a proper superclass of claw-free graphs, since the claw is a subgraph of the fork. In Section 3, we show how our methods can be combined with ideas of [26] to count arbitrarily weighted independent sets in (fork, odd hole)-free graphs. Again, these are a proper superclass of (claw, odd hole)-free graphs. This gives a further nontrivial generalisation of the result of [22].

## 1.1 Preliminaries

Let $\mathbb{N} = \{1, 2, \ldots\}$ denote the natural numbers, and $\mathbb{N}_0 = \mathbb{N} \cup \{0\}$. If $n \in \mathbb{N}$, let $[n] = \{1, 2, \ldots, n\}$. For a set $S$, $S^{(2)}$ will denote the set of subsets of $S$ of size exactly 2.

Throughout this paper, graphs are always simple and undirected. Let $G = (V, E)$ be a graph. We denote its vertex set by $V(G)$, and its edge set by $E(G) \subseteq V^{(2)}$. We write an edge $e \in E$ between $v$ and $w$ in $G$ as $e = vw$, or $e = \{v, w\}$ if the $vw$ notation is ambiguous. If $U, W \subseteq V$

with $U \cap W = \varnothing$, we will denote the $U, W$ *cut* by $(U, W) = \{uw \in E : u \in U, w \in W\}$. We also consider *multigraphs*, in which $E$ may have *parallel edges*. That is, $E$ is a *multiset* with elements in $V^{(2)}$.

For a graph $G = (V, E)$, we will write $n = |V|$ and $m = |E|$, unless stated otherwise. The *empty graph* $G = (\varnothing, \varnothing)$ is the unique graph with $n = 0$. Also, $G = (V, V^{(2)})$, is the complete graph on $n$ vertices. The *complement* of any graph $G = (V, E)$ is $\overline{G} = (V, V^{(2)} \setminus E)$.

The *neighbourhood* of $v \in V$ will be denoted $\mathrm{N}(v)$, and $\mathrm{N}[v] = \mathrm{N}(v) \cup \{v\}$. Then the *degree* $\deg(v)$ of $v$ is $|\mathrm{N}(v)|$. More generally, the neighbourhood of a set $U \subseteq V$, is $\mathrm{N}(U) = \{v \in V \setminus U : uv \in E$ for some $u \in U\}$, and $\mathrm{N}[U] = U \cup \mathrm{N}(U)$. We will say that a vertex $v \in V \setminus U$ is *complete* to $U$ if $U \subseteq \mathrm{N}(v)$, and *anticomplete* if $U \cap \mathrm{N}(v) = \varnothing$. More generally, a set $U \subseteq V$ is complete to a set $W \subseteq V \setminus U$ if every $u \in U$ is adjacent to every $w \in W$, and anticomplete if $(U, W) = \varnothing$. Observe that this is a symmetric relation between $U$ and $W$. The graph $G = (V, E)$ is *connected* if $V$ cannot be partitioned into sets $U, W$ that are anticomplete.

The term "induced" subgraph will always mean a vertex-induced subgraph. If $U \subseteq V$, we will write $G[U]$ for the subgraph of $G$ induced by $U$. Where "subgraph" is used without qualification, it will always mean induced subgraph. Then a class $\mathcal{C}$ of graphs is called *hereditary* if $G[U] \in \mathcal{C}$ for all $G \in \mathcal{C}$ and $U \subseteq V$. If $U \subseteq V$, we will often write $G \setminus U$ as shorthand for $G[V \setminus U]$.

We say a graph $G$ *contains* a graph $H$ if it has an induced subgraph isomorphic to $H$, and $H$ is a *forbidden subgraph* for the graph class $\mathcal{C}$ if no graph in $\mathcal{C}$ contains $H$. It is easy to see that any hereditary class can be characterised by a (possibly infinite) set $\mathcal{H}$ of minimal forbidden subgraphs. In this case we refer to $\mathcal{C}$ as the class of $\mathcal{H}$-free graphs.

An *odd hole* in a graph $G$ is a subset $H \subseteq V$, with $|H| \geq 5$ and odd, such that $G[H]$ is a simple cycle. A *perfect* graph $G$ is such that neither $G$ nor its complement $\overline{G}$ contains an odd hole. A hole in $\overline{G}$ is called an *antihole* in $G$. Perfect graphs were originally defined differently, but the equivalence to this definition was proved in [6].

The *line graph* $L(G) = (E, \mathcal{E})$ of a multigraph $G = (V, E)$ has $\mathcal{E} = \big\{\{xy, yz\} : xy, yz \in E\big\}$. We will write $G = L^{-1}(G')$ for the inverse operation, when it is defined, and call $G$ the *root* multigraph of the line graph $G'$. Note that $G = L^{-1}(G')$ is not unique when $G$ is a multigraph, whereas it is unique for $|V| > 4$ when $G$ is a graph. When $L^{-1}(G')$ is not uniquely defined, we may choose it to be any multigraph $G$ such that $G' = L(G)$.

A set $S \subseteq V$ is *independent* (or *stable*) in $G$ if $G[S]$ is edgeless. The empty set $\varnothing$ is an independent set in every graph. By $\mathcal{I}(G)$ we denote the set of all independent sets of $G$, and $\mathcal{I}_k(G) = \mathcal{I}(G) \cap V^{(k)}$ for $k \in \mathbb{N}_0$. The largest $k$ for which $\mathcal{I}_k(G) \neq \varnothing$ is the *independence number* $\alpha(G)$ of $G$.

For further information on graph theory, see [3, 11], for example.

We will suppose that the vertices $v \in V(G)$ are equipped with non-negative *weights* $w(v) \in \mathbb{R}$. We will denote such a vertex-weighted graph by $(G, w)$, or simply $G$ when the vertex weights $w$ are understood. Two weighted graphs $(G_1, w_1)$, $(G_2, w_2)$ will be called *isomorphic* if $G_1, G_2$ are isomorphic as graphs, though we may have $w_1 \neq w_2$. Since we are considering only approximation, we may assume here that $w(v) \in \mathbb{Q}$. The weight of a subset $S$ of $V$

is then defined to be $w(S) = \prod_{v\in S} w(v)$.[2] Then let $W_k(G) = \sum_{S\in\mathcal{I}_k(G)} w(S)$, and $W(G) = \sum_{S\in\mathcal{I}(G)} w(S) = \sum_{k=0}^{\alpha(G)} W_k(G)$. In particular, we have $W_0(G) = 1$, $W_1(G) = \sum_{v\in V} w(v)$ and $W_2(G) = \sum_{uv\notin E} w(u)w(v)$.

We will use only the following simple properties of $W(G)$. If $G$ has connected components $C_1, C_2, \ldots, C_r$, then $W(G) = \prod_{i=1}^{r} W(C_i)$, and if $\mathcal{S}_1, \mathcal{S}_2, \ldots, \mathcal{S}_s$ partitions $\mathcal{I}(G)$, then $W(G) = \sum_{i=1}^{s} \sum_{I\in\mathcal{S}_i} w(I)$.

We will say that a vertex-weighted graph $(G', w')$ is *equivalent* to a vertex-weighted graph $(G, w)$ if $W_k(G) = W_k(G')$, for all $k \in \mathbb{N}_0$, and hence $W(G) = W(G')$. In particular, this implies $\alpha(G) = \alpha(G')$. Observe that equivalent weighted graphs are not necessarily isomorphic, and isomorphic weighted graphs are not necessarily equivalent.

Note that, if $w(v) = 0$ for any $v \in V$, then $G$ is equivalent to $G[V \setminus \{v\}]$, thus we can consider such vertices as present in or deleted from $G$, whichever is more convenient. We will assume that such vertices are deleted before carrying out computations, so we may assume that $w(v) > 0$ for all $v \in V$.

If $w(v) = 1$ for all $v \in V$, then $W_k(G) = |\mathcal{I}_k(G)|$, the number of independent sets of size $k$ in $G$, and $W(G) = |\mathcal{I}(G)|$ counts all independent sets in $G$. However, we also refer to the case with non-unit weights as "counting".

A central theme of structural graph theory has been *decomposition*, that is, breaking a graph into smaller pieces that have stronger properties than the original, such that the pieces are all connected to each other in some canonical fashion. Our counting algorithms for (claw, odd hole)-free and (fork, odd hole)-free graphs are based on two graph decompositions, clique cutset decomposition and modular decomposition, respectively. We will describe these in Section 2 and Section 3 respectively.

We consider approximating $W(G)$ and $W_k(G)$ in the following sense. An FPRAS (fully polynomial randomized approximation scheme) is an algorithm which produces an estimate $\widehat{W}$ of a quantity $W$ such that

$$\Pr\big((1-\varepsilon)W \le \widehat{W} \le (1+\varepsilon)W\big) \ \ge\ 3/4\,.$$

The key FPRAS we employ here is that of Jerrum, Sinclair and Vigoda [22] for the *permanent*. This uses the Markov chain approach to approximate counting, but we will not need the interior details of the algorithm. Essentially, we use [22] as a "black box" here. We note the equivalence of approximate counting with approximate random generation [23], but we make no direct use of this here.

For further information on approximate counting, see [20], for example.

## 2 Approximating $W(G)$ in (claw, odd hole)-free graphs

We develop an algorithm for approximating $W(G)$ in (claw, odd hole)-free graph using clique cutset decomposition. But first we describe this method, and its application to counting, in a general setting.

[2] Note the difference from the corresponding definition $\sum_{v\in S} w(v)$ used in optimisation.

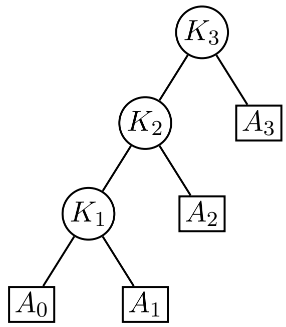



Fig. 2: Clique decomposition tree

## 2.1 Clique cutset decomposition

A *clique* $K$ in a graph $G = (V, E)$ is a subset of $V$ such that $G[K]$ is a complete graph. $K \subseteq V$ is a *clique cutset* of $G$ if $K$ is a clique and $G \setminus K$ is disconnected. In particular, $\varnothing$ is a clique cutset of every disconnected graph. For a clique cutset $K$ of $G$, let $(A, B)$ be a partition of $V \setminus K$ such that $A$ is anticomplete to $B$. The subgraphs $G[A \cup K]$ and $G[B \cup K]$ are *blocks* of the decomposition of $G$ by $K$. These blocks may themselves have clique cutsets, so may contain further blocks. A block with no clique cutset is called an *atom*. The decomposition can be presented in the form of a tree, in which the interior vertices are cliques, and the leaves are atoms. Tarjan [33] gave $O(mn)$ algorithm for a particular tree representation. This gives a binary decomposition tree in which all the interior nodes (cliques) form a path. If the tree has height $h$, we will number the atoms $A_0, A_1, \ldots, A_h$ and cliques $K_1, K_2, \ldots, K_h$ from the bottom up in this tree. While the atoms are all different, a clique can occur several times in a decomposition tree. See Fig. 2, and see [33] for further information. We describe how this decomposition may be used for computing $W(G)$ in Section 2.2.

## 2.2 Approximating $W(G)$ using clique cutset decomposition

Let $\mathcal{C}$ be a hereditary class of graphs such that all graphs in $\mathcal{C}$ have a clique cutset decomposition with all atoms in some hereditary class $\mathcal{A} \subset \mathcal{C}$, and we can approximate $W(G)$ for any weighted $G = (V, E)$ in $\mathcal{A}$ in time $T_{\mathcal{A}}(n) = \Omega(n)$, where $T$ is assumed convex. We show how to determine $W(G)$ for the entire graph $G$ in time $T_{\mathcal{C}}(n) \leq 2nT_{\mathcal{A}}(n)$.

The decomposition tree in Section 2.1 has cliques $K_1, K_2, \ldots, K_h$ and atoms $A_0, A_1, \ldots, A_h$, where $h \leq n$. The root of the tree is $K_h$. Let $A'_i = A_i \setminus K_i$, $s_i = |K_i|$, $a'_i = |V(A'_i)|$. Let $G_i$ be the graph formed by deleting the vertices of $A'_h, A'_{h-1}, \ldots, A'_{i+1}$ from $G$. Thus $G_h = G$, and $G_{i-1} = G_i \setminus V(A'_i)$. We will account for the independent sets intersecting $A'_i$ by revising the weights on the vertices in $K_i$, in a similar way to Tarjan's [33] approach to the maximum weight independent set problem.

Since $K_i$ is a clique cutset in $G_i$, we may partition $\mathcal{I}(G_i)$ by the value of $I \cap K_i$, which is either $\{v\}$ ($v \in K_i$) or $\varnothing$. Then we partition $G_i$ into three vertex-disjoint subgraphs $G_{i-1} \setminus K_i$,

$G[K_i]$, and $A'_i$. So we may write

$$\begin{aligned}W(G_i) &= W(G_{i-1} \setminus K_i)\, W(A'_i) + \sum_{v \in K_i} W(G_{i-1} \setminus \mathrm{N}[v])\, w(v)\, W(A'_i \setminus \mathrm{N}(v)) \\ &= W(A'_i) \Big( W(G_{i-1} \setminus K_i) + \sum_{v \in K_i} W(G_{i-1} \setminus \mathrm{N}[v])\, w(v)\, W(A'_i \setminus \mathrm{N}(v))/W(A'_i) \Big) \\ &= W(A'_i)\, W(G_{i-1}) \ ,\end{aligned}$$

where the vertex weights in $G_{i-1}$ relate to those in $G_i$ by

$$w(v) \leftarrow w(v)\, W(A'_i \setminus \mathrm{N}(v))/W(A'_i) \quad (v \in K_i), \qquad w(v) \leftarrow w(v) \quad (v \notin K_i) \ .$$

Thus, since $G_h = G$, we may compute $W(G)$ by induction as

$$W(G) = W(A_0) \prod_{i=1}^{h} W(A'_i) \ ,$$

where we update the vertex weights as above at each stage. Note that the weights of some vertices may change several times in this process, since the cliques of Tarjan's decomposition are not necessarily vertex-disjoint.

At stage $i$, we have to perform $s_i + 1$ computations on subgraphs of $A'_i$, which are all in $\mathcal{A}$. Thus the total time is $T_{\mathcal{C}}(n) = \sum_{i=1}^{h} (s_i + 1) T_{\mathcal{A}}(a'_i) \le 2n T_{\mathcal{A}}(n)$, since $h \le n$, $s_i \le n$, $\sum_{i=1}^{h} a'_i \le n$ and $T$ is convex.

Note that this analysis deals only with applications of the algorithm for $\mathcal{A}$. It ignores the effect of the bit-size of the vertex weights on $T_{\mathcal{A}}(n)$. This distinction is not so important for optimisation, but is much more important for counting, since contracting modules (see 3.1) can cause exponential growth in the weights. The same comment applies to the algorithm of Section 3. However, we do not pursue this issue further in this paper.

#### 2.2.1 Error Analysis

We are only approximating the weight of graphs in $\mathcal{A}$, so we must show that the resulting error in $W(G)$ can be controlled for $G \in \mathcal{C}$.

Suppose we approximate to a factor $(1 \pm \varepsilon/n^2)$ throughout. Then, by induction, the weights in $A_{h-i}$ will have relative error at most $(1 \pm \varepsilon/n^2)^i$. Thus the estimate of the total weight of $A_0$ will have relative error at most $(1 \pm \varepsilon/n^2)^h$. The error in $W(A'_{h-i})$ will be at most $(1 \pm \varepsilon/n^2)^i$, so the error in $\prod_{i=1}^{h} W(A'_i)$ is at most $(1 \pm \varepsilon/n^2)^{h(h-1)/2}$. Hence the error in $W(G)$ is at most $(1 \pm \varepsilon/n^2)^{h(h+1)/2}$. Since $h < n-1$, the error is a most $(1 \pm \varepsilon/n^2)^{n^2/2}$, which is at most $(1 \pm \varepsilon)$ for $\varepsilon < 1$. So the overall error can be kept within any desired relative error $\varepsilon$ by performing the weight estimations for all graphs in $\mathcal{A}$ to within error $\varepsilon/n^2$.

### 2.3 Structure of claw-free perfect graphs

In our application of the method of Section 2.2, $\mathcal{C}$ will be the class of (claw, odd hole)-free graphs, and $\mathcal{A}$ will be a class of graphs that we will define below. We must examine approximate counting in this class, but first we review the structural results which allow us to apply clique cutset decomposition.

Chvátal and Sbihi [10] investigated the structure of claw-free *perfect* graphs as a special class of perfect graphs. These are closely related to (claw, odd hole)-free graphs. The difference is that odd antiholes are also forbidden. The following lemma of Ben Rebea explains that relation.

**Lemma 1** (Ben Rebea)**.** *Let $G$ be a connected claw-free graph with $\alpha(G) \geq 3$. If $G$ contains an odd antihole then it contains a hole of length five.* □

**Corollary 1.** *A claw-free graph with $\alpha(G) \geq 3$ is perfect if and only if it has no odd hole.*

*Proof.* From the Strong Perfect Graph Theorem [6], Lemma 1 implies that a claw-free graph with $\alpha(G) \geq 3$ is perfect if and only if it contains no odd hole. □

Chvátal and Sbihi [10] gave a decomposition theorem via clique cutsets for claw-free perfect graphs. As described in Section 2.1, a clique cutset decomposition can be described by a binary tree whose interior vertices are cliques, and whose leaves are atoms.

**Theorem 1** (Chvátal and Sbihi)**.** *If a claw-free perfect graph has no clique cutset then it is either elementary or peculiar.* □

We will describe elementary and peculiar graphs below. These will be the atoms of the decomposition.

### 2.3.1 Peculiar graphs

A *peculiar* graph is constructed as follows. A set $K$ of vertices, initially a clique, is partitioned into six non-empty subsets $A_1, A_2, A_3, B_1, B_2, B_3$. At least one edge is removed from each of the edge sets $(A_1, B_2)$, $(A_2, B_3)$ and $(A_3, B_1)$. Finally, three disjoint nonempty cliques $K_1, K_2, K_3$ are added, and each vertex in $K_i$ is made adjacent to every vertex in $K \setminus (A_i \cup B_i)$ for $i = 1, 2, 3$.

The smallest peculiar graph, with $|A_i|, |B_i|, |K_i| = 1$ $(i = 1, 2, 3)$ is shown in Fig. 3. The black vertices are $A$'s, the white $B$'s and the grey $K$'s. This graph is a template for all peculiar graphs, as shown by Chvátal and Sbihi [10].

We will need the following simple observation about peculiar graphs.

**Lemma 2.** *A peculiar graph $G = (V, E)$ has independence number $\alpha(G) = 3$. Any independent set of size three has one vertex in each of $K_1, K_2, K_3$.*

*Proof.* Note that $K_1 \cup A_3 \cup B_2$, $K_2 \cup A_1 \cup B_3$, $K_3 \cup A_2 \cup B_1$ are three cliques which cover $V$, so $\alpha(G) \leq 3$. However, we can form an independent set of size three by taking one vertex from each of $K_1, K_2, K_3$, so $\alpha(G) \geq 3$.

Let $I = \{v_1, v_2, v_3\}$ be any maximum independent set in $G$. Suppose first $I \cap K_i = \varnothing$ for all $i = 1, 2, 3$. Then $I$ is contained in $K$. But $K$ is a perfect graph with vertices contained in

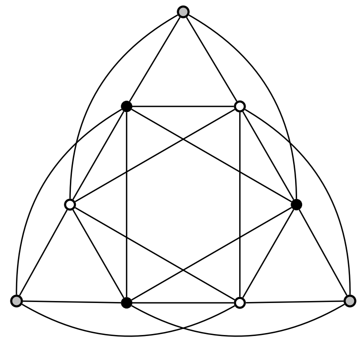


Fig. 3: Minimal peculiar graph from [10]

two disjoint cliques, $A_1 \cup A_2 \cup A_3$ and $B_1 \cup B_2 \cup B_3$. Thus $K$ is a cobipartite graph, with $\alpha(K) \leq 2$. Hence $\alpha(G) \leq 2$, a contradiction.

Thus, without loss of generality, assume $v_1 \in K_1 \cap I$. Now $\mathrm{N}[v] = (K \cup K_1) \setminus (A_1 \cup B_1)$. So $v_2, v_3 \in A_1 \cup B_1 \cup K_2 \cup K_3$. But $A_1 \cup B_1 \cup K_i$ is a clique for $i = 2, 3$. Thus, if $v_2 \in A_1 \cup B_1$, $v_3$ cannot exist. Thus $v_2 \in K_2$, without loss of generality, and then we must have $v_3 \in K_3$. $\square$

In Fig. 3, the three corner triangles cover all the vertices, and the three corner vertices form an independent set.

Peculiar graphs do not form an hereditary class. If in a subgraph $G$ of a peculiar graph, $K_i = \varnothing$ holds for any $i = 1, 2, 3$, it follows that $\alpha(G) \leq 2$. However, if $\alpha(G) \leq 2$, then $G$ is a clique or a cobipartite graph. But both of these are elementary graphs, as defined in Section 2.3.2 below. Thus we may insist that a peculiar graph has $K_i \neq \varnothing$ $(i = 1, 2, 3)$ and $\alpha(G) = 3$.

### 2.3.2 Elementary graphs

Chvátal and Sbihi called a graph $G = (V, E)$ *elementary* if $E$ can be two-(edge)-coloured so that edges $xy, yz \in E$ have distinct colours whenever $xz \notin E$. Such a colouring is called elementary. It is clear from this that elementary graphs form a hereditary class. Whether $G$ has an elementary colouring can be checked by forming the *Gallai graph* $\mathrm{Gal}(G) = (E, \mathcal{E})$, where $\{xy, yz\} \in \mathcal{E}$ if and only if $(x, y, z)$ is a $P_3$. Clearly $\mathrm{Gal}(G)$ can be constructed in time $O(mn)$, by taking all pairs of $xy \in E$ and $z \in V$, and checking that $yz \in E$, $xz \notin E$. Then an elementary colouring exists if and only if $\mathrm{Gal}(G)$ is bipartite. If so, the two colour classes of $\mathrm{Gal}(G)$ give an elementary colouring of $G$. This can be recognised in $O(|\mathcal{E}|)$ time by breadth-first search, so this gives an $O(mn)$ algorithm for recognising elementary graphs and determining an elementary colouring. Note that some edges of $G$ may be left uncoloured by this process. These can be coloured arbitrarily if a full colouring is required.

Maffray and Reed [28] characterised elementary graphs in a very precise way. They showed

**Theorem 2** (Maffray and Reed)**.** *G is elementary if and only if it is an augmentation of the line graph of bipartite multigraph.* $\square$

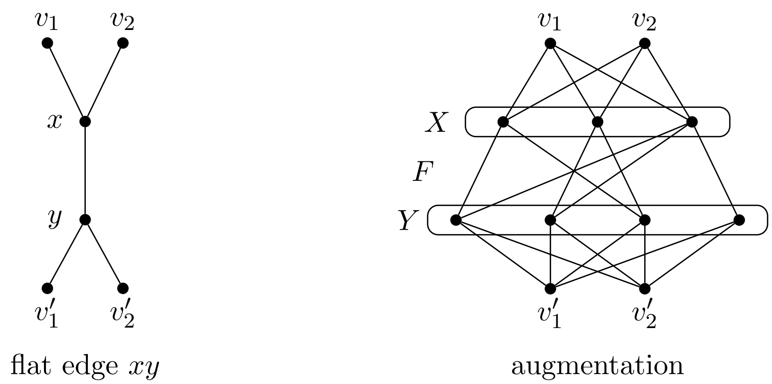



Fig. 4: Augmenting a flat edge

We must describe the "augmentation" in this theorem. An edge $xy$ in $G$ is called *flat* if $x, y$ have no common neighbour. Then we augment the flat edge by replacing $x$ by a clique $X$, $y$ by a clique $Y$, and $xy$ by any non-empty edge set $F \subseteq (X, Y)$. That is, we replace $xy$ by a cobipartite graph called an *augment*. Finally we add all edges between $X$ and $\mathrm{N}(x) \setminus \{y\}$ and all edges between $Y$ and $\mathrm{N}(y) \setminus \{x\}$, see Fig. 4. Then a graph $G'$ is an augmentation of a graph $G$ if $G'$ can be obtained from $G$ by applying one or more such steps to independent flat edges in $G$. See Fig. 5, where the first graph is a line graph of a bipartite multigraph, as we will show below. The second and third show augmentations using the *independent* flat edges $x_1y_1$ and then $x_2y_2$ and two $2 \times 2$ cobipartite graphs as replacements.

We observe that an augmentation of a line graph of a bipartite multigraph need not be a line graph of a bipartite multigraph, as can be seen in Fig. 5 where the two augmentations contain a gem which is an excluded structure for the class of line graphs of bipartite multigraphs by the following characterisation of this class.

**Theorem 3** (Maffray and Reed [28]). *A graph is the line graph of bipartite multigraph if and only if it is (claw, gem, 4-wheel, odd hole)-free. (See Fig. 1.)*

Maffray and Reed [28] show how to recover the structure of an elementary graph as the line graph of bipartite multigraph with augmented flat edges, using an elementary colouring of the graph. This can be done in $O(mn)$ time, so there is an $O(mn)$ time algorithm for determining the graph structure. (Maffray and Reed claim only the looser bound $O(m^2)$.)

## 2.4 Counting in (claw, odd hole)-free graphs

From Corollary 1, Lemma 2 and Theorem 1 we have

**Lemma 3.** *Every (claw, odd hole)-free graph $G$ without a clique cutset and with $\alpha(G) > 3$ is elementary.*

This gives us a clique cutset decomposition in which the atoms are in the hereditary class $\mathcal{A}$ of graphs that are either elementary or have $\alpha(G) \leq 3$. To apply the method of Section 2.2, we must consider how to approximate $W(G)$ in these graphs.

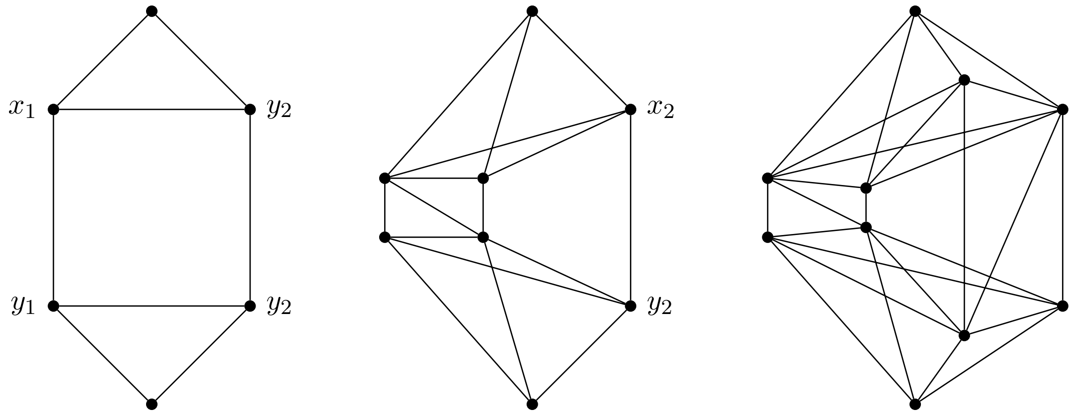



Fig. 5: Augmentation of flat edges

#### 2.4.1 Computing $W(G)$ in graphs with $\alpha(G) \leq 3$

Let $G$ be a (claw, odd hole)-free atom. For any $k$, we can determine $\mathcal{I}_k(G)$ in $O(n^k)$ time by listing all $k$-tuples of vertices and checking which are independent in $G$. Thus we can determine $W_k(G)$ for $k = 0, 1, 2, 3, 4$ in $O(n^4)$ time. If $W_4(G) > 0$, we conclude that $G$ must be elementary, by Lemma 3. Otherwise, we set $W(G) = \sum_{k=0}^{3} W_k(G)$.

#### 2.4.2 Approximating $W(G)$ in elementary graphs

If a (claw, odd hole)-free atom $G$ has $\alpha(G) > 3$, then it is elementary by Lemma 3. We use the $O(mn)$ time algorithm of Maffray and Reed [28] to identify $G$ *as the line graph of a bipartite multigraph with augments*. If this algorithm fails, we conclude that the original graph was not (claw, odd hole)-free, and halt. Otherwise, we have an elementary atom, and we continue.

An elementary graph $G$ is not necessarily the line graph of bipartite multigraph because of the augments, as discussed in Section 2.3.2. However, we will replace $G$ by an equivalent $G'$, such that $G'$ is the line graph of bipartite multigraph. We do this by replacing the augments in $G$ by "gadgets" which are line graphs of bipartite multigraphs.

#### 2.4.3 Augmentation gadgets

Suppose the augment $Z = X \cup Y$ in $G = (V, E)$, with vertex weights $w(v)$ ($v \in V$), comprises a cobipartite graph with cliques on $X, Y$ and connecting bipartite graph $(X \cup Y, F)$. Clearly $W_k(Z) \neq 0$, only for $k = 0, 1, 2$.

If $U \subseteq V$, we will write $W_k(U)$ for $W_k(G[U])$. Then $W_1(X) = \sum_{v \in X} w(v)$, $W_1(Y) = \sum_{v \in Y} w(v)$ and $W_2(Z) = \sum_{uv \notin F} w(u)w(v)$. These are respectively the total weights of independent sets in $Z$ which involve $X$ alone, $Y$ alone, or both. We will also write $\overline{W}_2(Z) = W_1(X)W_1(Y) - W_2(Z) = \sum_{uv \in F} w(u)w(v)$.

Consider the gadget $Z'$ shown in Fig. 6, where $\rho, \sigma, \overline{\rho}, \overline{\sigma} \geq 0$ are to be determined, Let $X' = \{x_1, x_2\}$, $Y' = \{y_1, y_2\}$. Note that both vertices in the clique $X'$ have the same neighbours external to $Z'$ as all vertices in $X$ have to vertices external to $Z$, and similarly for $Y'$ and $Y$.

If we set $\overline{\rho} = W_1(X) - \rho$, then $W_1(X') = \rho + \overline{\rho} = W_1(X)$, and if we set $\overline{\sigma} = W_1(Y) - \sigma$, then $W_1(Y') = \sigma + \overline{\sigma} = W_1(Y)$. Thus the total weight of independent sets using $X'$ but not $Y'$ is $W_1(X)$, and the total weight of independent sets using $Y'$, but not $X'$ is $W_1(Y)$, as required.

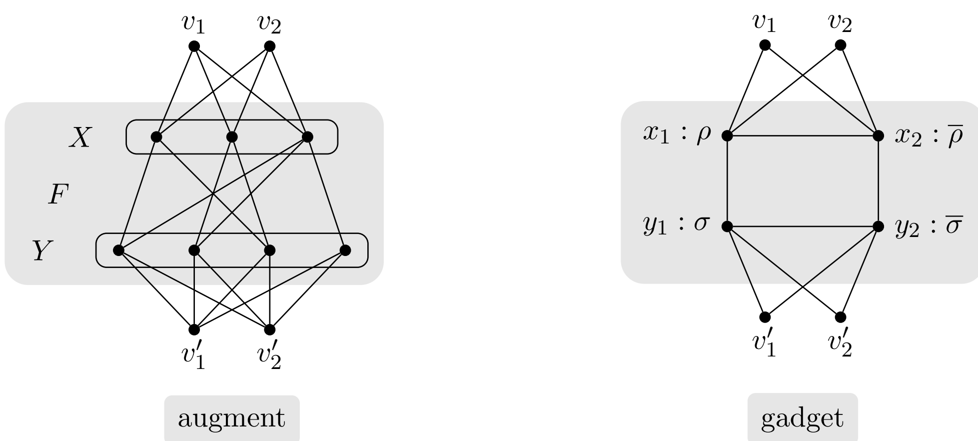



Fig. 6: Augmentation and equivalent gadget with vertex weights

The weight of independent sets using both $X'$ and $Y'$ is

$$W_2(Z') = \rho\overline{\sigma} + \overline{\rho}\sigma = \rho(W_1(Y) - \sigma) + \sigma(W_1(X) - \rho) = W_2(Z) ,$$

again as required, provided

$$\rho = \frac{W_2(Z) - \sigma W_1(X)}{W_1(Y) - 2\sigma} .$$

It is convenient to break symmetry by requiring $\sigma < \overline{\sigma}$, making the denominator of the above fraction positive. We must also have $\rho, \overline{\rho} \geq 0$, so $0 \leq \rho \leq W_1(X)$. Thus we require

$$0 \leq \sigma \leq \min\{W_2(Z), \overline{W}_2(Z)\}/W_1(X).$$

Otherwise we can choose $\sigma$ arbitrarily. Then the gadget $Z'$ is equivalent to $Z$. The particular choice $\sigma = 0$ deletes $y_1$ and its incident edges, and gives an even smaller gadget. Note that $Z'$ is itself an augment, equivalent to $Z$ for computing $W_k(G)$ for any $0 \leq k \leq \alpha(G)$.

The gadget $Z'$ is the line graph of a simple bipartite graph, see Fig. 7. Every augment $Z$ that has less vertices than $Z'$ is also line graph of a suitable bipartite graph. So using this gadget to replace all augments on at least four vertices in $G$ will result in it becoming the line graph $G'$ of bipartite multigraph, as required. Moreover, the size of $G'$ does not exceed the size of $G$, which becomes relevant in the following analysis.

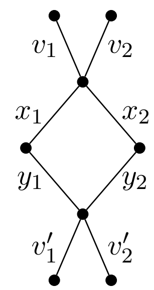



Fig. 7: Root graph of the gadget depicted in Fig. 6

### 2.4.4 Reduction to permanent approximation

We now need to determine $W_k(G')$, where $G'$ is the line graph of a bipartite multigraph. We determine its root multigraph $G'' = L^{-1}(G')$ and verify that it is bipartite in $O(m)$ time, using (for example) the algorithm of Lehot [25]. The vertex weights in the line graph $G'$ become edge weights in $G''$, and independent sets become matchings. The graph $G''$ may have parallel edges, but for our purposes we can reduce this to a simple edge-weighted bipartite graph $G^*$ by adding the weights on parallel edges. Now $M_k(G^*) = W_k(G')$ will be the total weight of all *matchings* of size $k$ in $G^*$.

Suppose $G^* = (V_1 \cup V_2, E^*)$, where $n_1 = |V_1|$, $n_2 = |V_2|$. We wish to use the permanent algorithm of [22] to determine $M_k(G^*)$. However, this algorithm only computes $M_n(G^*)$ for the perfect matching case $n_1 = n_2 = n$. It is possible the algorithm of [22] can be modified to the general case, but the general case can be reduced to the permanent, as follows.

To determine $M_k(G^*)$, let $n_1' = n_2 - k$, $n_2' = n_1 - k$. We add a set $V_1'$ of $n_1'$ vertices to $G^*$, and the edges of a complete bipartite graph $K_{n_1',n_2} = (V_1' \cup V_2, V_1' \times V_2)$, and add a set $V_2'$ of $n_2'$ vertices and the edges of a complete bipartite graph $K_{n_1,n_2'} = (V_1 \cup V_2', V_1 \times V_2')$. See Fig. 8, where $n_1 = 5$, $n_2 = 4$, $k = 2$. The weights assigned to the added edges are all 1. Let this weighted graph be $G^+ = (V_1^+ \cup V_2^+, E^+)$, where $V_1^+ = V_1 \cup V_1'$, $V_2^+ = V_2 \cup V_2'$, and $E^+$ is $E^*$ plus the edges of the complete bipartite graphs. Let $n^+ = |V_1^+| = |V_2^+| = n_1 + n_2 - k = n - k$.

Now observe that there is a correspondence between matchings of size $k$ in $G^*$ and perfect matchings in $G^+$. For each $k$-matching in $G^*$, we match the $n_1 - k$ unmatched vertices in $V_1$ with vertices in $V_2'$, and we match the $n_2 - k$ unmatched vertices in $V_2$ with vertices in $V_1'$. Given a perfect matching $M^+$ in $G^+$, we can uniquely recover a $k$-matching $M^*$ in $G^*$ of the same weight. However, there are $n_1'!n_2'! = (n_1 - k)!(n_2 - k)!$ matchings $M^+$ corresponding to any $M^*$. Thus $M_{n^+}(G^+) = (n_1 - k)!(n_2 - k)!M_k(G^*)$.

Thus our algorithm will use the permanent method of [22] to compute $M_{n^+}(G^+)$, and then divide this by $(n_1 - k)!(n_2 - k)!$ to obtain $M_k(G^*)$. Thus we can determine $W_k(G')$ for any elementary graph $G'$ and any $0 \le k \le \alpha(G')$. We then compute $W(G') = \sum_{k=1}^{\alpha(G')} W_k(G')$. This has time complexity is $O(nT_{\mathrm{P}}(n, \varepsilon))$, where $T_{\mathrm{P}}(n, \varepsilon)$ is the time to approximate the total weight of perfect matchings in an $n \times n$ bipartite graph with relative error $1 \pm \varepsilon$. The clique decomposition in Section 2.2 gives another $n$ factor, so the overall time complexity of approximating $W(G)$ in a (claw, odd hole)-free graph $G$ with $n$ vertices is $T_{\mathcal{C}}(n, \varepsilon) =$

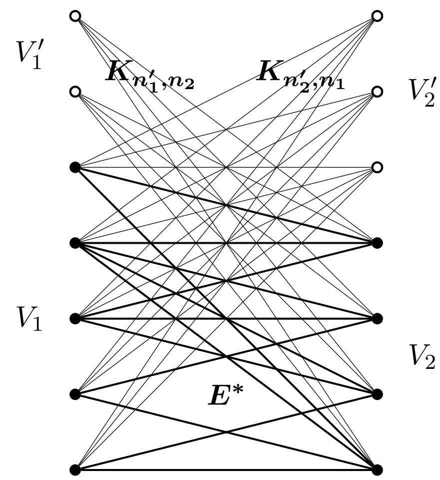



Fig. 8: Equivalent permanent problem

$O(n^2 T_{\mathrm{P}}(n, \varepsilon n^{-2}))$, where the accuracy parameter comes from the error analysis in Section 2.2.1. The best bound for $T_{\mathrm{P}}(n,\varepsilon)$ known is $O(n^7 \log^4 n + n^6 \log^5(n)\varepsilon^{-2})$ [2]. Thus the overall time complexity for (claw, odd hole)-free graphs is $T_{\mathcal{C}}(n,\varepsilon) = O(n^{12}\log^5(n)\varepsilon^{-2})$. This analysis is clearly loose and could be tightened. However, without a radical improvement in the bound for $T_{\mathrm{P}}(n,\varepsilon)$, the overall time complexity cannot be improved to anything practically relevant.

## 2.5 Approximating $W_k(G)$

The freedom to use very large vertex weights allows us to approximate $W_k(G)$ for any $0 \le k \le \alpha(G)$. For $k < \alpha(G)$, we would need to use the algorithm described in [12], which is an improvement of the approach of that in [21]. This method is based on the fact that the *independence polynomial*

$$P_G(\lambda) = \sum_{i=0}^{\alpha(G)} \lambda^k \, W_k(G) \tag{2.1}$$

has only real negative roots when $G$ is claw-free. This was proved in [7] for unit weights, and extended to general weights in [12]. However, the algorithm requires the reduction from approximate counting to approximate random generation [23]. Therefore, we will not give further details here, but it is a straightforward application of the above algorithm for $W(G)$.

However, for $\alpha = \alpha(G)$ the total weight $W_\alpha(G)$ of maximum independent sets, which correspond exactly to perfect matchings in a graph when $G$ is a line graph, there is a simpler approach, which we will describe. Note that this gives a complete generalisation of the result of [22].

We have a multiplier $\lambda$ for every vertex weight, as in (2.1) above, so $w(v) \leftarrow \lambda w(v)$ for all $v \in V$. Then $W(G)$ becomes

$$W(\lambda G) = \sum_{I \in \mathcal{I}(G)} w(I)\, \lambda^{|I|} = \sum_{k=0}^{\alpha(G)} W_k(G)\, \lambda^k = P_G(\lambda)\ ,$$

Thus, if $\lambda \geq 1$ and $\alpha = \alpha(G)$,

$$W_\alpha(G)\, \lambda^\alpha \leq W(\lambda G) < W(G)\lambda^{\alpha-1} + W_\alpha(G)\, \lambda^\alpha$$

so

$$W_\alpha(G) \leq W(\lambda G)/\lambda^\alpha < W_\alpha(G) + W(G)/\lambda\ .$$

So we need $\lambda \geq W(G)/\varepsilon W_\alpha(G)$ to achieve relative error $\varepsilon$. Let $w_{\min} \leq w(v) \leq w_{\max}$ for all $v \in V$. Then $W(G) \leq 2^n w_{\max}^\alpha$ and $W_\alpha(G) \geq w_{\min}^\alpha$, since $G$ has at most $2^n$ independent sets and at least one of size $\alpha$. Thus it suffices to take $\lambda \geq 2^n (w_{\max}/w_{\min})^\alpha/\varepsilon$ in order that $W(\lambda G)/\lambda^\alpha$ approximates $W_\alpha(G)$ with relative error $\varepsilon$. Since the time complexity of the algorithm is polynomial in $\log\lambda$, it is clearly polynomial in $n$ and the bit size of the $w(v)$'s, as required.

# 3 Approximating $W(G)$ in (fork, odd hole)-free graphs

We will extend the result for claw-free graphs to fork-free graphs using modular decomposition, as described in Section 3.1. Our algorithm is inspired by Lozin and Milanič's [26] approach to computing the maximum weight independent set problem using modular decomposition. Again, we will first describe the modular decomposition approach in a general context.

## 3.1 Modular decomposition

*Modules* were first introduced by Gallai [16, 27, Thm. 3.1.2], using different terminology. If $S \subseteq V$ in $G = (V, E)$, we say that any vertex $x \in V \setminus S$ *distinguishes* $S$ if there exist $u, v \in S$ with $ux \in E$, $vx \notin E$. Then a set $M \subseteq V$ is a *module* of $G$ if no vertex of $V \setminus M$ distinguishes it. Alternatively, $M$ is a module if $\mathrm{N}(u) \setminus M = \mathrm{N}(v) \setminus M$ holds for all $u, v \in M$. Thus $\varnothing$, $V$ and all the singletons $\{v\}$ $(v \in V)$ are modules of $G$. These are the *trivial* modules; all other modules are *nontrivial*.

Another way of defining a module $M$ is that every vertex $v \in V \setminus M$ must be either complete or anticomplete to $M$. It follows that $M$ is a module in $G$ if and only if it is a module in $\overline{G}$, since $v$ is complete to $M$ in $G$ if and only if it is anticomplete to $M$ in $\overline{G}$. It is also easy to show that the modules of $G$ are closed under intersection. However, if $M_1, M_2$ are modules, then $M_1 \cup M_2$ is only guaranteed to be a module if $M_1 \cap M_2 \neq \varnothing$. Otherwise some $v$ could be complete to $M_1$ and anticomplete to $M_2$, so neither complete nor anticomplete to $M_1 \cup M_2$. Modules are also not generally closed under complementation, since if $u \notin M$ is complete to $M$ and $v \notin M$ is anticomplete to $M$, then any vertex of $M$ distinguishes $V \setminus M$.

**Observation 1.** *If $M$ is a module of $G = (V, E)$ and $U \subseteq V$ then $M \cap U$ is a module of $G[U]$.*

*Proof.* Otherwise, two vertices $u, v \in M \cap U$ distinguishable by $x \in U \setminus M$ belong to $M$, that is $u, v \in M$, and are distinguished by $x \in V \setminus M$. □

If $M \neq \varnothing$ is a module of $G$ then $G/M$ denotes the graph obtained from $G$ by *contracting* $M$ to a single vertex, with the same adjacencies in $G/M$ as all vertices in $M$. We will label this vertex as $v_M$ in $G/M$. For $|M| \leq 1$ let $G/M = G$. Note that

**Observation 2.** *If $M$ is a module in $G \in \mathcal{C}$, for some hereditary class $\mathcal{C}$, then $G/M \in \mathcal{C}$.*

*Proof.* Contracting $M$ deletes all but one of its vertices, and relabels the remaining vertex $v_M$. Since $\mathcal{C}$ is a class of unlabelled graphs, $G/M \in \mathcal{C}$ by heredity. □

If $M$ is a module of $G$, and $M'$ is a module of $G[M]$, then $M'$ is also a module of $G$.

A module that does not overlap with any other module is strong [24], more formally, $M \subseteq V$ is a *strong module* of $G = (V, E)$ if $M$ is a module of $G$ and for all modules $M'$ of $G$ we have $M \subseteq M'$ or $M \supseteq M'$ or $M \cap M' = \varnothing$. Every trivial module of $G$ is strong. $G$ is a *prime graph* if every strong module of $G$ is trivial. Like connectedness, primeness is an intrinsic property of the graph, but not a hereditary property. For example, none of the graphs in Fig. 1 is prime.

The strong modules of $G = (V, E)$ are partially ordered by set inclusion. The unique top element of this poset is $V$, and $\varnothing$ is the unique bottom element. The layer above $\varnothing$ consists of the singletons $\{v\}$ for all $v \in V$. The strong modules in the next layer up are called *prime modules* of $G$. A prime module $M$ of $G$ induces a prime subgraph $G[M]$.

Let $M_1$ and $M_2$ be strong modules such that $M_1 \subset M_2$. We say $M_2$ *covers* $M_1$ if, for all strong modules $M$, $M_1 \subseteq M$ and $M \subseteq M_2$ imply $M_1 = M$ or $M = M_2$. If $|V| > 1$ then the strong modules of $G = (V, E)$ except those of size at most one are the nodes of a tree rooted at $V$, where the arcs of the tree are given by the cover relation. Note that the singleton modules are often included as leaves in this tree, but we will not do so here. See Fig. 9, where $M_7 = V$. We will call this the *standard decomposition tree*. Equivalent definitions exist, see [18]. Several algorithms are known to compute the standard decomposition tree in linear time, see for example [34].

We will use this tree in an equivalent form. Let us number the modules $M_1, M_2, \ldots, M_h = V$, according to a *postorder* on the standard tree. This order places all the descendants of a vertex before the vertex itself, as in Fig. 9. (See, for example, [35, Ch. 3].) We will call this the *extended* decomposition tree. Here $G_0 = G$, and $G_i = G_{i-1}/\tilde{M}_i$ ($i \in [h]$), where $\tilde{M}_i$ is $M_i$ after $M_1, \ldots, M_{i-1}$ have been contracted in order to single vertices. We will denote this by $G_{i-1} = G/(M_1, M_2, \ldots, M_{i-1})$, and similarly and hence $\tilde{M}_i = M_i/(M_1, M_2, \ldots, M_{i-1})$ is a module in $G_{i-1}$. We can represent the extended decomposition tree as shown in Fig. 9, where $G_{i-1}$ ($i \in [h]$) are the internal vertices, and $\tilde{M}_1, \tilde{M}_2, \ldots, \tilde{M}_h$ and $G_h$ are the leaves.

Note that $\tilde{M}_i$ is a prime module in $G_{i-1}$, and in particular $G_{i-1}[\tilde{M}_i]$ is a prime graph. Also $G_h$ is a single vertex, since $M_h = V$ has been contracted to a single vertex. Note also that $\tilde{M}_i$ is isomorphic to the graph obtained by contracting only its children in the standard tree, but its vertex weights require contracting the whole subtree of which it is the root.

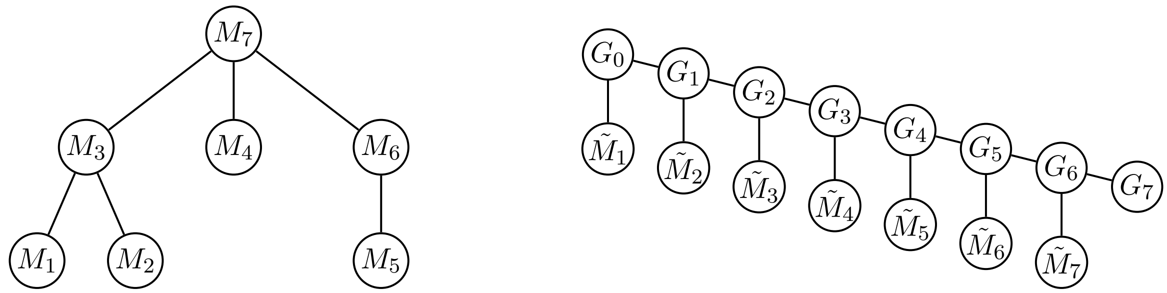



Fig. 9: A standard modular decomposition tree and its extended tree

Let $t_i = |\tilde{M}_i|$, so $2 \le t_i < n$. Then $|V(G_0)| = n$, and $|V(G_i)| = |V(G_{i-1})| - t_i + 1$, so $1 = |V(G_h)| = n - \sum_{i=1}^h t_i + h \le n - 2h + h = n - h$. Thus $h \le n - 1$, so the decomposition tree contains at most $(n-1)$ modules. Also $n - \sum_{i=1}^h t_i + h = 1$ implies $\sum_{i=1}^h t_i = n - h - 1 \le n - 2$. Thus the extended decomposition tree can be represented explicitly in $O(n)$ space.

We can compute the extended tree from the standard tree in a further $O(n)$ time, using postorder tree traversal [35]. We can contract modules and form the modules $\tilde{M}_i$, during the traversal. Since $\sum_{i=1} |\tilde{M}_i| < n$, the additional time complexity for contracting modules is also $O(n)$, so the total remains $O(m)$. Of course, this excludes the time to compute the vertex weight of $v_{M_i}$ in $G_i$, as will be detailed in Section 3.2 below. However, these computations can also be integrated into the tree traversal. Performing the algorithm this way, the extended tree is purely a useful notional device, and never computed explicitly.

Of course, we could compute the extended tree explicitly by successively finding a prime module and contracting it, until the contracted graph is prime. While this may be conceptually simpler, it is computationally inefficient. Finding any prime module appears to be an $\Omega(m)$ computation, so the time complexity of producing the whole extended tree becomes $\Omega(mn)$. The inefficiency clearly results from discarding information gained in earlier searches when carrying out the later searches.

We will make use of the following.

**Observation 3.** *Each of the modules $\tilde{M}_i$ $(i \in [h])$ in the extended tree is isomorphic to a prime subgraph of $G$.*

*Proof.* Since $\tilde{M}_i$ is $M_i$ with all its submodules contracted, it follows from Observation 2, that $\tilde{M}_i$ is isomorphic to some subgraph $\tilde{M}'_i$ of $G$. Note that $\tilde{M}'_i$ is not unique. As observed above, each of the $\tilde{M}_i$ is prime, so $\tilde{M}'_i$ is also prime. Since they are vertex-weighted graphs, the isomorphism between $\tilde{M}_i$ and $\tilde{M}'_i$ is in the sense defined in Section 1.1. $\square$

## 3.2 Approximating $W(G)$ using modular decomposition

Let $\mathcal{C}$ be a hereditary class, and $\mathcal{P} \subseteq \mathcal{C}$, the (non-hereditary) class of prime graphs in $\mathcal{C}$. Let $T_{\mathcal{C}}(n)$ be the time to compute $W(G)$ for any connected $n$-vertex graph $G \in \mathcal{C}$, and let $T_{\mathcal{P}}(n)$

bound the time to compute $W(G)$ for any $n$-vertex prime graph $G \in \mathcal{C}$. We may assume that $T_{\mathcal{P}}(n)$ is a monotonically increasing function that is linear or convex. We will show $T_{\mathcal{C}}(n) \leq T_{\mathcal{P}}(n) + O(m)$. This strengthens the result of [26, Thm. 1], with an easier proof.

We use the notation of Section 3.1. We construct the extended decomposition tree as described in Section 3.1, and begin with $G_0 = G$. At step $i$, we contract the module $\tilde{M}_i$ in $G_{i-1}$ to give $G_i$, giving $v_{\tilde{M}_i}$, the vertex that represents $\tilde{M}_i$ in $G_i$, weight $W(\tilde{M}_i)$. Then $W(G_i) = W(G_{i-1})$, since the set $\tilde{M}_i$ has the same neighbourhood in $G_{i-1}$ as the vertex $\tilde{M}_i$ in $G_i$. Thus, by induction, $W(G) = W(G_0) = W(G_h) = w(v)$ for the unique vertex $v \in V(G_h)$.

If $h = 1$, $T_{\mathcal{C}}(n) = T_{\mathcal{P}}(n)$. Otherwise, $2 \leq h \leq n-1$ so, omitting the time to compute the modular decomposition, we have

$$\begin{aligned} T_{\mathcal{C}}(n) &\leq \max\left\{ \textstyle\sum_{i=1}^{h} T_{\mathcal{P}}(t_i) : \sum_{i=1}^{h} t_i = n+h, 2 \leq t_i \leq n, i \in [h] \right\} \\ &\leq T_{\mathcal{P}}(n-h+2) + (h-1)T_{\mathcal{P}}(2), \\ &\leq T_{\mathcal{P}}(n) + (n-2)T_{\mathcal{P}}(2) \;=\; T_{\mathcal{P}}(n) + O(n), \end{aligned}$$

where the second line follows from the first since $T_{\mathcal{P}}$ is convex, so $\sum_{i=1}^{h} T_{\mathcal{P}}(t_i)$ is maximised by setting $t_1 = n-h+2$, $t_i = 2$, $i = 2, \ldots, h$. The third line follows from the second because $T_{\mathcal{P}}$ is increasing, so $T_{\mathcal{P}}(n-h+2) \leq T_{\mathcal{P}}(n)$ for $h \geq 2$. Adding the $O(m)$ time to compute the modular decomposition [34], we have $T_{\mathcal{C}}(n) \leq T_{\mathcal{P}}(n) + O(m)$. Thus we can approximate $W(G)$ in any graph in $\mathcal{C}$, with only an $O(m)$ overhead, if we can approximate it in all the prime graphs in $\mathcal{C}$.

Note that this analysis deals only with applications of the algorithm for $\mathcal{P}$, as does that in [26, Thm. 1]. It ignores the effect of the bit-size of the vertex weights on $T_{\mathcal{P}}(n)$. This distinction is not so important for optimisation, but is much more important for counting, since contracting modules can cause exponential growth in the weights. However, we will not pursue this issue further.

### 3.3 Structure of fork-free graphs

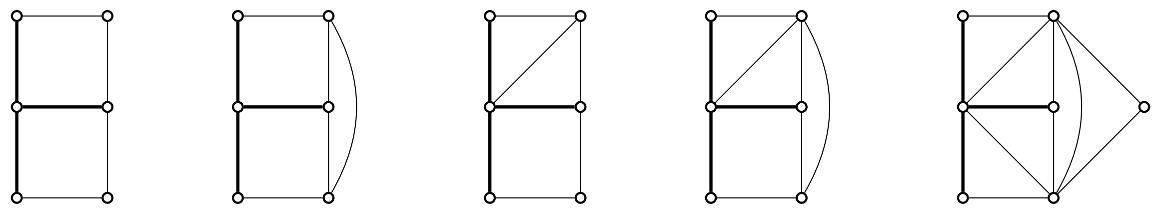
Lozin and Milanič [26] used a modular decomposition approach to determine the maximum weight independent set in a fork-free graph. However, there seems to be a flaw in their algorithm and its analysis. Consequently, we will re-work most of their development, in addition to extending it from optimisation to counting.

The approach of [26] is based on a structural result given in [26, Thm. 3]. We begin with a more useful version of this theorem, the original being too weak for its application. We first repeat two structural lemmas from [26]. We will also make use of the following simple observation, which was used as the basis of the algorithm in [34].

**Observation 4.** *If $v$ is any vertex of $G$, and $M$ is a module not containing $v$, then either $M \subseteq \mathrm{N}(v)$ or $M \subseteq V \setminus \mathrm{N}[v]$.*

*Proof.* Otherwise, $v$ distinguishes $M$, contradicting it being a module. $\square$

**Lemma 4** (see [26], Thm. 3 and also [4])**.** *If a prime fork-free graph contains a claw, then it contains one of the graphs $H_1, \ldots, H_5$ (see Fig. 10).* $\square$



$H_1$ $H_2$ $H_3$ $H_4$ $H_5$

Fig. 10: The five minimal fork-free prime graphs extending a claw.

**Lemma 5** ([26], Lemma 1)**.** *Let $G$ be a fork-free graph and let $v$ be any vertex of $G$. Assume that $H \in \{H_1, \ldots, H_5\}$ is an induced subgraph of $G \setminus \mathrm{N}[v]$. Then no neighbour of $v$ distinguishes $V(H)$.* □

Note that we could omit $H_2$ for our application, because it contains a 5-hole, but we give the result for general claw-free graphs, as used in [26], since what follows could be used for computing a maximum weight independent set in a claw-free graph.

The statement and proof of the following theorem modifies the weaker result of [26, Thm. 3].

**Theorem 4.** *Let $G$ be a fork-free graph and $v$ a vertex of $G$. If $G$ is prime and $M$ is a prime subgraph of $G \setminus \mathrm{N}[v]$, then $M$ is claw-free.*

*Proof.* Assume by contradiction that $M$ contains a claw. Then by Lemma 4, $M$ contains $H$, one of the graphs $H_1, \ldots, H_5$. Hence, by Lemma 5, $\mathrm{N}(v)$ can be partitioned into sets $Y$ and $Z$, such that $Y$ is anticomplete to $H$ and $Z$ is complete to $H$.

Let $W$ be an (inclusionwise) maximal subset of vertices of $G \setminus \mathrm{N}[v]$ satisfying the following properties:

(i) $V(H) \subseteq W$,

(ii) $G[W]$ is connected,

(iii) $\overline{G[W]}$ is connected,

(iv) $Z$ is complete to $W$,

(v) $Y$ is anticomplete to $W$.

Note that such a set $W$ exists since $V(H)$ satisfies all these properties. Clearly, $5 < |V(H)| \leq |W| < |V(G)|$. Since $G$ is prime, $W$ cannot be a nontrivial module of $G$, and hence some $u \in V(G) \setminus W$ distinguishes $W$. Note that $u \neq v$. We will obtain a contradiction (to the existence of $W$) by showing that the set $W' = W \cup \{u\}$ also satisfies properties (i)–(v).

Since $W$ satisfies (iv) and (v), $u \in V \setminus \mathrm{N}(v)$ and hence $W' \subseteq V \setminus \mathrm{N}(v)$. Clearly $W'$ satisfies (i). Since $W$ satisfies (ii) and (iii), and since $u$ has both a neighbour and a non-neighbour in $W$, it follows that $W'$ satisfies (ii) and (iii).

Suppose that $u$ has a non-neighbour $z \in Z$. Since $u$ distinguishes $W$ and $\overline{G[W]}$ is connected, $u$ distinguishes a pair of nonadjacent vertices $w_1, w_2 \in W$. But then $\{u, w_1, z, w_2, v\}$ induces a fork, a contradiction. Therefore, $Z$ is complete to $W'$ and hence $W'$ satisfies (iv).

Finally, suppose that $u$ has a neighbour $y \in Y$. Since $G[W]$ is connected, and $u$ distinguishes $W$, there is a shortest path $P = (v_0, \dots, v_k)$ connecting $V(H)$ and $u$ in $G[W']$ with $v_0 \in V(H)$ and $v_k = u$. Let $v_{k+1} = y$ and $v_{k+2} = v$. Note that $v$ is anticomplete to $V(P)$, and $y$ is anticomplete to $V(P) \setminus \{v_k\}$, and hence $(v_0, \dots, v_{k+2})$ is a chordless path. Since $v_2$ has no neighbour in $H$, by Lemma 5, $v_1$ is complete to $V(H)$. But then any two nonadjacent vertices of $H$, together with $v_1, v_2, v_3$ induce a fork in $G$, a contradiction. Therefore, $Y$ is complete to $W'$ and hence $W'$ satisfies (v). □

Note that Theorem 4 cannot obviously be strengthened. If $G$ is a fork-free prime graph and $v$ a vertex of $G$, then $G \setminus \mathrm{N}[v]$ is not necessarily claw-free nor prime. Consider, for example, a $n \times n$ complete bipartite graph with a perfect matching removed. This is easily shown to be fork-free and prime. The case $n = 4$ is shown in Fig. 11. The graph $G \setminus \mathrm{N}[v]$ for the vertex labelled $v$ is also shown. This graph is clearly neither claw-free nor prime. Consequently, Theorem 3 of [26] is inapplicable, even in this simple case.

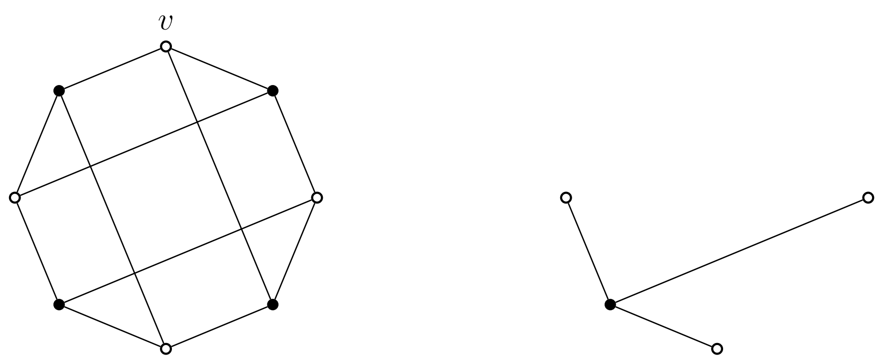



Fig. 11: Graph $G$ and a derived $G \setminus \mathrm{N}[v]$

### 3.4 Approximating $W(G \setminus \mathrm{N}[v])$ for prime $G$ and $v \in V$

To apply Theorem 4, we need the following strengthening.

**Corollary 2.** *Let $G$ be prime and $v$ be a vertex of $G$. The modules $\tilde{M}_i$ ($i \in [h]$) in the extended decomposition tree for $G \setminus \mathrm{N}[v]$ are claw-free.*

*Proof.* This follows directly from Observation 3 and Theorem 4. □

To determine $W(G)$ for a prime graph, we first show how to determine $W(G \setminus \mathrm{N}[v])$ for any $v \in V$. Let $G_v$ denote $G \setminus \mathrm{N}[v]$. As we have seen, $G_v$ is not prime in general, so we must approximate $W(G_v)$ using the modular decomposition approach of Section 3.1.

The algorithm is then as follows. We construct the extended decomposition tree for $G_v$, with modules $\tilde{M}_i$ ($i \in [h]$). For each $i = 1, 2, \dots, h$, we determine $W(G_{i-1}[\tilde{M}_i])$, using this as the weight for $v_{\tilde{M}_i}$ in $G_i$. From Corollary 2, $G_{i-1}[\tilde{M}_i]$ is claw-free, so we may use the algorithm of Section 2 in this computation. Finally $W(G_v) = w(u)$, where $u$ is the unique vertex in $G_h$.

More generally, suppose $G \in \mathcal{C}$ for some hereditary class $\mathcal{C}$ and, given $v \in V$, all prime subgraphs of $G_v$ are in some smaller hereditary class $\mathcal{A}$. Then we can use this method to approximate $W(G_v)$ for graphs in $\mathcal{C}$, using modular decomposition and an algorithm for approximating $W$ for graphs in $\mathcal{A}$. In our application $\mathcal{C} = (\text{fork}, \text{odd hole})\text{-free}$ and $\mathcal{A} = (\text{claw}, \text{odd hole})\text{-free}$.

### 3.5 Approximating $W(G)$ for prime $G$

The algorithm described in Section 3.4 approximates $W(G \setminus \mathrm{N}[v])$ for prime $G$ and $v \in V$. Let $S(v) = \{I \in \mathcal{I}(G) : v \in I\}$. Then $w(v)\, W(G \setminus \mathrm{N}[v]) = \sum_{I \in S(v)} w(I)$, the total weight of all independent sets containing $v$. The classes $\mathcal{C}$, $\mathcal{A}$ are as in Section 3.4.

We can write $V = \{v_1, v_2, \ldots, v_n\}$ and determine $w(v_i)\, W(G \setminus \mathrm{N}[v_i])$ for $i \in [n]$, similarly to [26], but the sum of these greatly overestimates $W(G)$, since $\{S(v_i) : i \in [n]\}$ is a cover of $\mathcal{I}(G)$, not a partition.

Let $V_i = \{v_i, \ldots, v_n\}$ and $S'(v_i) = \{I \in \mathcal{I}(G) : v_i \in I \text{ and } I \subseteq V_i\}$. The sets $\{S'(v_i) : i \in [n]\}$ form a partition of $\mathcal{I}(G) \setminus \{\varnothing\}$, and so

$$W(G) = 1 + \sum_{i=1}^{n} \sum_{I \in S'(v_i))} w(I) = 1 + \sum_{i=1}^{n} w(v_i)\, W(G[V_i \setminus \mathrm{N}[v_i]]) \,. \tag{3.1}$$

So we must approximate $W(G[V_i \setminus \mathrm{N}[v_i]])$ for $i \in [n]$. We do this by constructing the extended decomposition tree for $G \setminus \mathrm{N}[v_i]$ with leaf modules $\tilde{M}_1, \tilde{M}_2, \ldots, \tilde{M}_h$, as in Section 3.4. From Corollary 2 we know that the modules $\tilde{M}_1, \tilde{M}_2, \ldots, \tilde{M}_h$ in this decomposition are in $\mathcal{A}$. We transform this extended decomposition tree of $G \setminus \mathrm{N}[v_i]$ into an extended decomposition tree for $G[V_i \setminus \mathrm{N}[v_i]]$. For a fixed $i \in [n]$ we take $G'_0 = G[V_i \setminus \mathrm{N}[v_i]]$ and for $j \in [h]$ we set $G'_j = G_j \setminus \{v_1, v_2, \ldots, v_{i-1}\}$ and $\tilde{M}'_j = \tilde{M}_j \setminus \{v_1, v_2, \ldots, v_{i-1}\}$. For each $j \in [h]$ the set $\tilde{M}'_j$ is a module in $G'_{j-1}$ by Observation 1.

To compute $W(G[V_i \setminus \mathrm{N}[v_i]])$, we note that restricting to $V_i \setminus \mathrm{N}[v_i]$ is equivalent to putting $w(v) = 0$ for all $v \notin V_i \setminus \mathrm{N}[v_i]$. Thus we can use the algorithm of section 3.4, with exactly the same justification, after setting $w(v) = 0$ for $v \notin V_i \setminus \mathrm{N}[v_i]$. Of course, in carrying out the algorithm we actually delete the vertices in $V \setminus (V_i \setminus \mathrm{N}[v_i])$. Thus the algorithm approximates $W(G[V_i \setminus \mathrm{N}[v_i]])$ for $i \in [n]$, using the algorithm of Section 2, and then combines the estimates using (3.1).

### 3.6 Approximating $W(G)$ for all graphs in $G \in \mathcal{C}$

Since we can now approximate $W(G)$ for any prime $G$, we can use the algorithm of Section 3.2 to lift this to arbitrary $G$. This completes the description of our algorithm.

The algorithm will fail if the $(\text{claw}, \text{odd hole})$-free algorithm fails on any of the modules $\tilde{M}'_j$ for any prime $G$ and any $G[V_i \setminus \mathrm{N}[v_i]]$ $(i \in [h])$. In that case we conclude that $G$ is not $(\text{fork}, \text{odd hole})$-free and terminate.

The basis of the algorithm is modular decomposition. From Section 3.2 this gives only a negligible overhead to the algorithm for prime graphs in $\mathcal{C}$. From Section 3.4, the algorithm for $G \setminus \mathrm{N}[v]$ is modular decomposition, so this adds a negligible overhead. However, the algorithm for prime $G$ in Section 3.5 requires $n$ applications of the algorithm for $G \setminus \mathrm{N}[v]$. Thus if $T_{\mathcal{A}}(n, \varepsilon)$ is the time complexity of the subroutine for $\mathcal{A}$, the overall time complexity is $O(nT_{\mathcal{A}}(n, \varepsilon n^{-2}))$. (The error analysis is similar to the one seen earlier in the reduction to permanent approximation, and is given below.) For $\mathcal{A}$ = (claw, odd hole)-free graphs, $T_{\mathcal{A}}(n, \varepsilon) = O(n^{12} \log^5(n)\varepsilon^{-2})$, so for $\mathcal{C}$ = (fork, odd hole)-free graphs, $T_{\mathcal{C}}(n, \varepsilon) = O(n^{17} \log^5(n)\varepsilon^{-2})$.

### 3.6.1 Error Analysis

We can only approximate the total weight for graphs in $\mathcal{A}$, in our application the class of (claw, odd hole)-free graphs. So we must show that the resulting error in $W(G)$ can be controlled for all $G \in \mathcal{C}$, in our application the class of (fork, odd hole)-free graphs.

Suppose we approximate to a factor $(1 \pm \varepsilon/2n^2)$ for $G \in \mathcal{A}$. In step $i$ of the algorithm of Section 3.2, we approximate $W(\tilde{M}_i)$, and contract $\tilde{M}_i$ with this weight. Thus one weight in $G_{i+1}$ has error $(1 \pm \varepsilon/2n^2)$. We do this at most $n$ times, so the error in $W(G)$ becomes at most $(1 \pm \varepsilon/2n)$. We do this $n$ times in the method of Section 3.5, and add the estimates. However, this does not increase the relative error. Finally, we apply the algorithm of Section 3.2 again, so the error is at most $(1 \pm \varepsilon/2n)^n$, which is at most $(1 \pm \varepsilon)$ for $\varepsilon < 1$.

## 3.7 Approximating $W_\alpha(G)$

We cannot use the method of Section 2.5 to approximate $W_k(G)$ for arbitrary $0 \le k \le \alpha(G)$, because there is no known analogue for fork-free graphs of the real-rootedness result of [8] for claw-free graphs. However, the method given in Section 2.5 for estimating $W_\alpha(G)$ is valid for any graph class, not necessarily even hereditary, where we can use arbitrary vertex weights. Therefore, the result of [22] for approximating the permanent can be completely generalised to approximating the total weight of maximum independent sets in (fork, odd hole)-free graphs.

# References


[1] L. Beineke (1970), Characterizations of derived graphs, *Journal of Combinatorial Theory* **9**, 129–135.

[2] I. Bezáková, D. Štefankovič, V. Vazirani and E. Vigoda (2008), Accelerating simulated annealing for the permanent and combinatorial counting problems, *SIAM Journal on Computing* **37**, 1429–1454.

[3] A. Brandstädt, V. B. Le and J. Spinrad, *Graph classes: a survey*, SIAM Monographs on Discrete Mathematics and Application, Philadelphia, 1999.

[4] A. Brandstädt, C. T. Hoàng, J.-M. Vanherpe (2004), On minimal prime extensions of a four-vertex graph in a prime graph. *Discrete Mathematics* **288** 9–17.

[5] M. Chudnovsky and M. Plumettaz (2013), The structure of claw-free perfect graphs, *Journal of Graph Theory* **75**, 203–220.

[6] M. Chudnovsky, N. Robertson, P. Seymour and R. Thomas (2006), The strong perfect graph theorem, *Annals of Mathematics* **164**, 51–229.

[7] M. Chudnovsky and P. Seymour (2007), The roots of the independence polynomial of a clawfree graph, *Journal of Combinatorial Theory (Series B)* **97** , 350–357.

[8] M. Chudnovsky and P. Seymour (2012), Claw-free graphs VII: Quasi-line graphs, *Journal of Combinatorial Theory (Series B)* **102**, 1267–1294.

[9] M. Chudnovsky, G. Cornuéjols, X. Liu, P. Seymour and K. Vušković (2005), Recognizing Berge graphs, *Combinatorica* **25**, 143–186.

[10] V. Chvátal and N. Sbihi (1988), Recognizing claw-free perfect graphs, *Journal of Combinatorial Theory (Series B)* **44**, 154–176.

[11] R. Diestel, *Graph Theory*, 4th edition, Graduate Texts in Mathematics **173**, Springer, 2012.

[12] M. Dyer, C. Greenhill and H. Müller (2018), Counting independent sets in graphs with bounded bipartite pathwidth, arXiv: abs/1812.03195.

[13] M. Dyer, L. A. Goldberg, C. Greenhill and M. Jerrum (2004), The relative complexity of approximate counting problems, *Algorithmica* **38**, 471–500.

[14] J. Edmonds (1965), Paths, trees, and flowers, *Canadian Journal of Mathematics* **17**, 449–467.

[15] Y. Faenza, G. Oriolo and G. Stauffer (2014), Solving the weighted stable set problem in claw-free graphs via decomposition, *Journal of the ACM* **61**, art. 20.

[16] T. Gallai (1967), Transitiv orientierbare Graphen, *Acta Mathematica Academiae Scientiarum Hungarica* **18**, 25–66.

[17] M. Grötschel, L. Lovász and A. Schrijver, *Geometric algorithms and combinatorial optimization*, Springer-Verlag, 1988.

[18] M. Habib and C. Paul (2010), A survey of the algorithmic aspects of modular decomposition, *Computer Science Review* **4**, 41–59.

[19] F. Harary and C. Holzmann (1974), Line graphs of bipartite graphs, *Rev. Soc. Mat. Chile* **1** 19–22.

[20] M. Jerrum, *Counting, Sampling and Integrating: Algorithms and Complexity*, Lectures in Mathematics – ETH Zürich, Birkhäuser, Basel, 2003.

[21] M. Jerrum and A. Sinclair (1989), Approximating the permanent, *SIAM Journal on Computing* **18**, 1149–1178.

[22] M. Jerrum, A. Sinclair and E. Vigoda (2004), A polynomial-time approximation algorithm for the permanent of a matrix with non-negative entries, *Journal of the ACM* **51**, 671–697.

[23] M. Jerrum, L. Valiant and V. Vazirani (1986), Random generation of combinatorial structures from a uniform distribution, *Theoretical Computer Science* **43**, 169–188.

[24] D. Kelly (1985), Comparability graphs, in I. Rival (ed) *Graphs and Orders*, D. Reidel, Dordrecht, NATO ASI Series **147**, 3–40.

[25] P. Lehot (1974), An optimal algorithm to detect a line graph and output its root graph, *Journal of the ACM* **21**, 569–575.

[26] V. Lozin and M. Milanič (2008), A polynomial algorithm to find an independent set of maximum weight in a fork-free graph, *Journal of Discrete Algorithms* **6**, 595–604.

[27] F. Maffray and M. Preissmann, A translation of Tibor Gallais article "Transitiv orientierbare Graphen" [16], in *Perfect Graphs*, Wiley, 2001.

[28] F. Maffray and B. Reed (1999), A description of claw-free perfect graphs, *Journal of Combinatorial Theory (Series B)* **75**, 134–156.

[29] J. Matthews (2008), *Markov chains for sampling matchings*, Ph.D. Dissertation, Edinburgh University.

[30] G. Minty (1980), On maximal independent sets of vertices in claw-free graphs, *Journal of Combinatorial Theory (Series B)* **28**, 284–304.

[31] P. Nobili and A. Sassano (2015–2019), An $\mathcal{O}(n^2 \log(n))$ algorithm for the weighted stable set problem in claw-free graphs, arXiv: abs/1501.05775.

[32] D. Štefankovič, E. Vigoda and J. Wilmes (2018), On counting perfect matchings in general graphs, in *Proc. Theoretical Informatics – 13th Latin American Symposium, (LATIN 2018)*, 873–885.

[33] R. Tarjan (1985), Decomposition by clique separators, *Discrete Mathematics* **55**, 221–232.

[34] M. Tedder, D. Corneil, M. Habib and C. Paul (2008), Simpler linear-time modular decomposition via recursive factorizing permutations, *Proc. 35th Internat. Colloquium on Automata, Languages and Programming (ICALP 2008)*, Springer LNCS **5125**, 634–645.

[35] G. Valiente, *Algorithms on Trees and Graphs*, Springer, 2002.

[36] L. Trotter (1977), Line perfect graphs, *Math. Programming* **12**, 255–259.

[37] L. Valiant (1979), The complexity of computing the permanent, *Theoretical Computer Science* **8**, 189–201.