

On approximately counting perfect matchings

Olivier Lemer

School of Computer and Communication Sciences
Computer Science Theory
Master Thesis

Supervisor

Professor Ola Svensson
EPFL IC THL2

June 25, 2021

Abstract

In this thesis, we study the problem of approximately counting the number of perfect matchings on a given graph, with the aim of contributing to the understanding of the problem, and making a step towards a solution on arbitrary graphs. We explore ways to generalize the best known solution on bipartite graphs, introduce new approaches and tools to solve the problem, and apply them to provide various reductions of the problem. This thesis describes a relation between integral and half-integral perfect matchings as a tool to extend bipartite solutions to the general graph. We introduce a new set of tools that modify the structure of a given graph with known impacts on matching counts, and apply them to reduce the problem to tripartite graphs of maximum degree 3. We also describe algorithms for exactly counting perfect matchings on select types of factor-critical graphs. This thesis also contains various other results, including a reduction to the problem of counting matchings with up to $n^{\frac{k}{k+1}}$ unmatched vertices.

Contents

1 Introduction **3**
1.1 Notations and definitions . . . 3
1.2 Counting perfect matchings . . . 4

2 Relating half-integral matchings on general graphs to the bipartite graph **7**
2.1 Introduction . . . 7
2.1.1 State of the art FPRAS on arbitrary bipartite graphs . . . 7
2.1.2 Relating half-integral to integral matchings on the bipartite graph . . . 9
2.2 Weighing down half-integral matchings . . . 12
2.3 Limiting number of half-integral cycles and edges . . . 15
2.4 Limiting number of holes in integral matchings . . . 19
2.5 Discussions and future research . . . 20

3 Graph modifications with predictable changes to matching count **21**
3.1 Graph modification tools . . . 21
3.1.1 Forcing edge selection . . . 21
3.1.2 Inserting and removing pending paths . . . 21
3.1.3 Duplicating pending paths . . . 23
3.2 Reduction to graphs of constant minimum or maximum degree . . . 24
3.3 Reduction to tripartite graphs . . . 26

4 A polynomial-time algorithm for almost-nested-ear decomposable graphs **29**
4.1 Introduction . . . 29
4.2 Base algorithm on nested-ear decomposable graphs . . . 32
4.3 Extension to almost-nested-ear decomposable graphs . . . 42
4.4 Reduction from arbitrary factor-critical graphs to flat-ear decomposable graphs . . . 46

5 Further research **51**

6 Conclusion **55**

1 Introduction

We first introduce important notations and definitions we will need in this thesis, before describing the problem we are tackling in detail, the state of research on the subject, and what this document contains.

1.1 Notations and definitions

Graphs All graphs considered in this thesis will be undirected, and contain no parallel nor loop edge, unless otherwise stated. For any graph G, we will denote its vertex set by $V(G)$, and its edge set by $E(G)$. If G is bipartite, then we will refer to the two classes of its vertex set as $V_1(G)$ and $V_2(G)$. Because G is undirected, if u and v are vertices that form an edge in a graph G, then we will let (u, v) and (v, u) be equivalent notations for this edge.

Given two graphs G and G', we say that G' is a *subgraph* of G when $V(G') \subseteq V(G)$ and $E(G') \subseteq E(G)$. Further, we say that G' is an *induced subgraph* of G when for every pair $u, v \in V(G')$, $(u, v) \in E(G')$ if and only if $(u, v) \in E(G)$. We then say that $V(G')$ *induces* G' on G.

Given a graph G and a set of vertices $S \subseteq V(G)$, the graph $G \setminus S$ has vertex set $V(G) \setminus S$, and edge set all edges of $E(G)$ that are not incident to a vertex in S. Given two graphs G and G', the graph $G \setminus G'$ is equal to $G \setminus V(G')$.

Given a graph G and a vertex $v \in V(G)$, we denote $\delta(v) = \{w \in V(G) : (v, w) \in E(G)\}$ the set of all vertices in $V(G)$ that share an edge of $E(G)$ with v. We call it the *neighborhood* of v, each vertex of $\delta(v)$ a *neighbor* of v, and $|\delta(v)|$ the *degree* of v. We define the *minimum degree* of graph G as the degree of the vertex in $V(G)$ with smallest degree, and analogously the *maximum degree* of G.

Matchings Given a graph G, we define a *matching* of G as a subgraph M of G such that no two edges of M share a vertex. We note that other definitions define it as the edge set of this subgraph, which is effectively equivalent. A matching is called *perfect* when its vertex set equals that of G. A matching is called *near-perfect* when its vertex set contains all but one or two vertices of G, which we call its holes. More generally, all vertices of G that do not appear in a given matching are refered to as *holes* of that matching. We will let $\mathcal{M}_k(G)$ be the set of matchings of G with k holes. We also write simply $\mathcal{M}(G)$ for the set of perfect matchings $\mathcal{M}_0(G)$. When clear from context, we may omit the graph and write simply $\mathcal{M}_k$ and $\mathcal{M}$.

Given a matching M of G, we say M *matches* vertex u to v if $(u, v) \in E(M)$. We then say that edge (u, v) is *selected* by M.

Paths A *path* P is defined by a non-empty sequence of vertices $v_1, \ldots, v_l$ (with possible duplicates). Its vertex set is all vertices appearing in this sequence, and its edge set contains exactly (v_i, v_{i+1}) for all $i \in \{1, \ldots, l-1\}$. We then call $l - 1 = |E(P)|$ the *length* of path P. A path P is an *odd* (*even*) path if its length is odd (even). Given a graph G, we say P is a path of G if it is a subgraph of G.

Let P be a path defined by the sequence of vertices $v_1, \ldots, v_l$. Then v_1 and v_l are called its *endpoints*, and all other vertices of P are called its *inner-points*. P is said to be *self-intersecting* if an inner-vertex v of P appears more than once in the sequence of vertices that defines it. We say P is a *cycle* if its two endpoints are equal, and a self-intersecting cycle if an P is self-intersecting. Note therefore that if P is not a cycle, it is self-intersecting if and only if $E(P) \neq V(P) - 1$, and if it is a cycle, it is self-intersecting if and only if $E(P) \neq V(P)$.

Given a path P defined by the sequence of vertices $v_1, \ldots, v_l$ and two distinct vertices v_i and v_j in this sequence, we call the path defined by $v_i, v_{i+1}, \ldots, v_{j-1}, v_j$ the *subpath* from v_i to v_j on P. Note that if P is not self-intersecting, then this path is the unique path on P from v_i to v_j.

Finally, we call P a *pending* path on G if all inner-points of P have degree 2 in G.

Markov chains Formally, and for the purposes of this thesis, we define a Markov chain as a stochastic process determining the outcome of an infinite sequence of random variables $M_1, M_2, \ldots$ called *states* and taking values in a finite *state space* Ω, such that $\forall k > 1$ and for all $m_1, \ldots, m_k \in \Omega^k$, it holds that $\Pr(M_{k+1}|M_k = m_k, \ldots, M_1 = m_1) = \Pr(M_{k+1}|M_k = m_k)$, i.e. the probability distribution of state $k+1$ depends only on the value of the k^{th} state. The value of the initial state M_1 can be chosen arbitrarily.

We can thus characterize a Markov chain by its state space Ω, and for all pairs of states $m, m' \in \Omega$, the probability $\Pr(m'|m)$, which we call the *transition probability* from m to m'. We can thus simulate the evolution of a Markov chain given only a procedure which, given a state $m \in \Omega$, outputs a state $m' \in \Omega$ with probability $\Pr(m'|m)$.

If, for all $m \in \Omega$, the probability $\Pr(M_i = m)$ converges as i tends to infinity, then we call the resulting distribution the Markov chain's *stationary distribution*.

A Markov chain is *irreducible* if any state m can reach any other state m' through a sequence of transitions with non-zero probability. It is said to be *periodic* if there exists a state $m \in \Omega$ such that $\Pr(M_i = m|M_1) \geq 0$ only for values of i whose smallest common divisor is strictly greater than 1, and *aperiodic* otherwise. A Markov chain that is both *irreducible* and *aperiodic* is said to be *ergodic*.

If a Markov chain is *ergodic* and there exists a probability mass function π over its state space Ω such that, for all $m, m' \in \Omega$,

$$\pi(m)\Pr(m'|m) = \pi(m')\Pr(m|m'),$$

then it is said to be *time-reversible*, and π is a stationary distribution of the Markov chain. Moreover, π is unique. We call this condition the *detailed balance condition*.

Let now $\mathcal{D}$ and $\tilde{\mathcal{D}}$ be two distributions over a state space Ω, and $\pi_{\mathcal{D}}$ and $\pi_{\tilde{\mathcal{D}}}$ be their respective probability mass functions. We define the *total variation distance* between $\mathcal{D}$ and $\tilde{\mathcal{D}}$ as

$$\Delta(\mathcal{D}, \tilde{\mathcal{D}}) = \max_{S \in \Omega} |\pi_{\mathcal{D}}(S) - \pi_{\tilde{\mathcal{D}}}(S)|.$$

The *mixing time* $\tau(\delta, m_1)$ of a Markov chain on state space Ω is a function of a tolerated distance δ and an initial state m_1, and is defined as the minimum value t at which the distribution given by $\Pr(M_t|M_1 = m_1)$ has total variation distance to the stationary distribution at most δ.

Finally, the state space Ω of a Markov chain can be represented as an underlying graph G_Ω whose vertex set is Ω, and whose edge set contains all $(m, m') \in \Omega^2$ such that $\Pr(m'|m) > 0$. We note that this is a directed graph with potential parallel and loop edges, although in the context of a time-reversible Markov chain, we may represent it as undirected, as for any edge from m to m' there exists an edge from m' to m.

1.2 Counting perfect matchings

Finding a perfect matching in any given graph can be done in polynomial time with the Blossom algorithm due to Edmonds [3]. Despite this, the related problem of *counting* the number of distinct perfect matchings in a given graph G is known to be $\sharp P$-complete, since this number equals the

permanent of the 0-1 adjacency matrix of G, which was proved by Valiant [16] to be $\sharp P$-complete in 1979.

Because of this, research has turned to finding *fully polynomial randomized approximation schemes* (FPRAS) for the problem. As is described in [6], Such an algorithm finds, with high enough probability, an approximation of the result with arbitrarily small error, in time polynomial in the problem size *and* the error. Formally and in the context of the problem of counting perfect matchings, on input a graph G and a tolerated error $\varepsilon > 0$, the algorithm must output a number X such that

$$\Pr\Big((1-\varepsilon)X \leq |\mathcal{M}| \leq (1+\varepsilon)X\Big) \geq \frac{3}{4},$$

in time polynomial in $|V(G)|$ and ε^{-1}.

As in many combinatorial counting problems, an FPRAS can be constructed from a *fully-polynomial almost-uniform sampler* (FPAUS) of the objects to be counted, here perfect matchings[1]. Again in the context of perfect matchings, such an algorithm, given a graph G and a tolerated distance $\delta > 0$, can sample perfect matchings from $\mathcal{M}(G)$ following a distribution whose total variation distance to uniform over $\mathcal{M}(G)$ is at most δ. The algorithm must run in time polynomial in both $|V(G)|$ and δ^{-1}.

To the day of writing this thesis, the best known FPRAS applies to arbitrary bipartite graphs on $n+n$ vertices. It uses an FPAUS and is the result of an approach first proposed by Broder [1]. It is based on the simulation of a Markov chain whose state space Ω is the union of all perfect and near-perfect matchings of the graph, and whose stationary distribution is uniform over Ω. Jerrum and Sinclair [4] showed that the mixing time of this Markov chain was at most polynomial for any initial state, whenever the number of near-perfect matchings does not exceed that of perfect matchings by more than a polynomial factor. This can therefore be used to sample from $\mathcal{M}$ only, by sampling repeatedly from Ω until a perfect matching is obtained. They showed that this holds for bipartite graphs of minimum degree $\frac{n}{2}$, and it was recently proved by Ebrahimnejad, Nagda, and Gharan [2] that d-regular strong expander graphs also have this property, implying that an FPRAS exists for such graphs as well.

For general bipartite graphs, it may be that this bound on the number of near-perfect matchings is not verified, as illustrated in Figure 1 by the *chain of boxes* graph construction with k boxes that has exponentially many near-perfect matchings, and only one perfect matching. Indeed, while a perfect matching must match all edges of the form (v_{2i-1}, v_{2i}) for $i \in \{1, \ldots, k-1\}$ and is thus unique, near-perfect matchings can match each square induced by $\{v_{2i}, v_{2i+1}, v'_{2i}, v'_{2i+1}\}$ for $i \in \{0, \ldots, k\}$ as $\{(v_{2i}, v_{2i+1}), (v'_{2i}, v'_{2i+1})\}$, or as $\{(v_{2i}, v'_{2i}), (v_{2i+1}, v'_{2i+1})\}$, resulting in 2^k perfect matchings with holes x and y. This issue was resolved by Jerrum et al. [6], by a clever re-weighting of the Markov chain's states according to so called *hole-patterns*. It allows the probability in the stationary distribution of near-perfect matchings to exceed that of perfect matchings by no more than a polynomial, regardless of the relative sizes of each set. Their solution includes many technicalities that we will cover in more detail in later sections.

The natural next step is now to consider general graphs, and despite existing results for special cases such as the FKT algorithm for planar graphs [15, 7, 8] or the FPRAS on d-regular strong expander graphs [2], there is no known solution for the general case. Moreover, it has been shown by Štefankovič et al. [14] that the hole-pattern trick used by Jerrum et al. [6] – and any hole-pattern-based re-weighting – is not sufficiont in the general case. They do however describe an FPRAS that makes use of the Gallai-Edmonds decomposition of the graph to extend the solution by Jerrum et al.

[1] Such a relation is described in the context of counting perfect matchings by Jerrum, Sinclair, and Vigoda [6], and more generically by Jerrum and Sinclair [5].

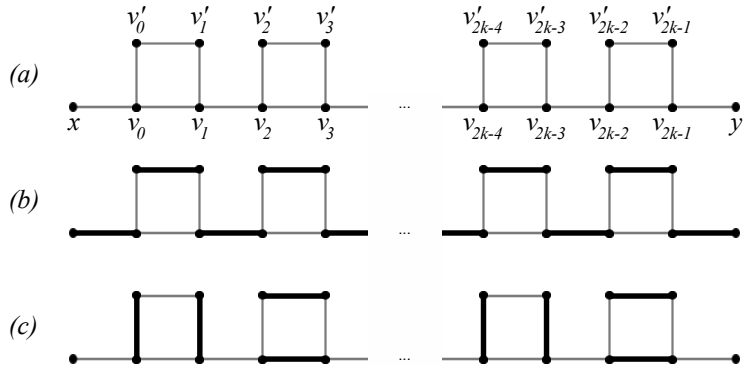


Figure 1: Chain of boxes graph with k boxes. (a) shows the construction of the graph with labels; (b) shows the unique perfect matching; (c) shows one near-perfect matching.

[6] to "almost bipartite" graphs. We will define this decomposition and the algorithm in chapter 4.

This thesis is divided into three main chapters. In chapter 2, we describe the state of the art solution on bipartite graphs by Jerrum et al. [6], and its relation with the half-integral equivalent problem on general graphs. Using this relation, we explore different strategies to extend their solution to the general graph, and provide a reduction from the general problem to one with at most $n^{\frac{k}{k+1}}$ local errors (such as half-integral edges or cycles, or unmatched vertices). In chapter 3, we describe tools that modify a given graph G into another with stronger properties, and whose number of (near-)perfect matchings is a known and invertible function of that of G. In particular, we show how to construct from arbitrary graph G a tripartite graph and a graph of maximum degree 3 whose number of perfect matchings equals that of G. In chapter 4, motivated by a reduction from the general graph to factor-critical graphs, due to Štefankovič, Vigoda, and Wilmes [14], we explore specific types of factor-critical graphs and provide exact algorithms for counting (near-)perfect matchings on them. We also show that arbitrary factor-critical graphs can be transformed into graphs with a flat ear decomposition with predictable effects on matching counts, using the tools we described in chapter 3.

2 Relating half-integral matchings on general graphs to the bipartite graph

In this chapter, we explore a relation that exists between the problem of counting perfect-matchings on bipartite graphs and its half-integral equivalent on general graphs. Essentially, a half-integral matching is one that may select only a $\frac{1}{2}$ fraction of each edge – we shall define this formally shortly. Our goal will be to take advantage of this relation to adapt the state of the art solution on bipartite graphs, due to Jerrum, Sinclair, and Vigoda [6], to the general graph. We first describe some important technicalities of this solution in section 2.1.1. We then introduce half-integral matchings and show the relation that exists between integral matchings in bipartite graphs and half-integral matchings in general graphs, in section 2.1.2. With this in mind, we start exploring different strategies that aim at giving to the FPAUS for bipartite graphs an awareness of this relation. Essentially, we wish to use it to sample from the set of half-integral and integral matchings, with a high enough probability of obtaining an integral solution. We first rule out the idea of simply weighing down matchings by an increasing function of the number of half-integral edges, in section 2.2. In section 2.3, we take a more binary approach by allowing a strictly bounded number of half-integral cycles in the solution, and describe how at most $n^{\frac{k}{k+1}}$ half-integral cycles can be tolerated. We note however some technical challenges arising from proving the validity of such a solution. Finally, we remark that the technique used to tolerate $n^{\frac{k}{k+1}}$ half-integral cycles can be generalized to other types of errors, such as holes in matchings, thus proving a reduction from counting perfect matchings to counting matchings with up to $n^{\frac{k}{k+1}}$ holes.

2.1 Introduction

This section provides a common ground that will be necessary for subsequent sections. We first provide a description of important points of the state of the art solution on bipartite graphs, and then describe how this problem relates to the half-integral equivalent problem on general graphs.

2.1.1 State of the art FPRAS on arbitrary bipartite graphs

The current best solution to the problem of counting perfect matchings is due to Jerrum, Sinclair, and Vigoda [6], who provide an FPRAS for arbitrary bipartite graphs. As the rest of this chapter will explore ways to adapt this solution to the general graph, we give here a description that should be sufficient for our purposes. For further and more technical details of their proof that we omitted here, please refer to the full paper [6].

The core of the solution by Jerrum et al. [6] is the simulation of a Markov chain, whose state space Ω is the union of all perfect and near-perfect matchings. The Markov chain is chosen to have the following properties: (1) it has a stationary distribution, (2) it is rapidly mixing, i.e. its mixing time is at most a polynomial in the size of the graph and the tolerated error, and (3) this stationary distribution gives to the set of perfect matchings at least an inverse polynomial probability. Any Markov chain verifying these conditions can then be used to obtain an FPRAS, by simulating its evolution until it reaches a distribution close enough to its stationary distribution, and simply rejecting near-perfect matchings. Point (2) allows a single simulation of the Markov chain to take at most polynomial time, while point (3) guarantees a high enough probability of eventually obtaining a perfect matching, after a polynomial number of simulations.

In order to achieve a stationary distribution that gives high enough probability to perfect matchings, they partition the state space Ω into *hole patterns*. For every pair of distinct vertices $u, v \in V(G)$, hole pattern $\mathcal{M}(u, v)$ is the set of all matchings with holes u and v. An additional

hole pattern is defined for all matchings with no holes, i.e. $\mathcal{M}$. There are thus $\binom{n}{2}+1$ hole patterns in total. The Markov chain is then constructed so as to be uniform over hole patterns (i.e. give equal probability to each hole pattern), and uniform within each hole pattern. As a result, perfect matchings receive an inverse polynomial probability, and a uniform distribution among perfect matchings.

For completeness and reference, we provide here the full description of the Markov chain's transition rules, as it is given in [6]. We will refer to it as MC_{JSV}. When in any state $M \in \Omega$, MC_{JSV} moves to state M' according to the following procedure.

1. (*Deletion*) If $M \in \mathcal{M}$, then select an edge $e \in E(M)$ uniformly at random, and let $M' = M \setminus \{e\}$.
2. If $M \in \mathcal{M}(u,v)$, then select a vertex $x \in V(M)$ uniformly at random.
   - (*Addition*) If x is u or v, and $(u,v) \in E(G)$, then let $M' = M \cup \{(u,v)\}$.
   - (*Exchange*) If x is such that $(x,u) \in E(G)$ and $\exists y \in V(G)$ such that $(x,y) \in E(M)$, then let $M' = M \setminus \{(x,y)\} \cup \{(x,u)\}$. Symmetrically when $(x,v) \in E(G)$.
   - Otherwise, let $M = M'$.
3. Let $p = \frac{1}{2}\min\{1, \frac{\Lambda(M')}{\Lambda(M)}\}$. Transition to M' with probability p; stay at M otherwise.

where Λ is proportional to a weight function w that equals 1 if $M \in \mathcal{M}$, and $\frac{|\mathcal{M}|}{|\mathcal{M}(u,v)|}$ if $M \in \mathcal{M}(u,v)$. Note that this last step 3 corresponds to a Metropolis rule: M' is called the *proposed* state, and the probability of transitioning to M' is made proportional to the size of the class to which M' belongs. This is what allows the stationary distribution (which can be easily seen to exist, as MC_{JSV} is ergodic and satisfies the detailed balance condition) to give each hole pattern the same probability. Note also that the factor $\frac{1}{2}$ in probability p makes the Markov chain "lazy", meaning that with probability $\frac{1}{2}$ it will not transition. This is a technicality to ensure that it is aperiodic. For the remainder of this section, we will let π denote the probability mass function of the stationary distribution of MC_{JSV}, $\Pr(M'|M)$ denote the transition probability from M to M', and G_{JSV} the Markov chain's underlying graph, whose vertex set and edge set are Ω, and $\{t = (M, M') : \Pr(M'|M) > 0\}$, respectively. Note that, by time reversibility, it is irrelevant whether G_{JSV} is chosen to be directed or undirected.[2]

The main technical challenge remaining is to prove that the Markov chain mixes fast enough, i.e. that its mixing time is at most polynomial. To this end, Jerrum et al. [6] take advantage of a relation that upper-bounds the mixing time by a function of the congestion of a flow on G_{JSV}. This flow is defined, for each pair $(I,F) \in \mathcal{M}_2 \times \mathcal{M}$, by a canonical path through G_{JSV}, with associated weight $\pi(I)\pi(F)$.[3]

These canonical paths must be defined in a way that no transition in G_{JSV} is over-utilized, i.e. such that the sum of weights of all canonical paths sharing a transition is no more than polynomial. In order to prove this, the following clever technique is used. For every transition t from M to M', an injective mapping associates an encoding $C \in \Omega' := \mathcal{M}_4 \cup \mathcal{M}_2 \cup \mathcal{M}$ to each pair of states $(I,F) \in \mathcal{M}_2 \times \mathcal{M}$ whose canonical path traverses t. One immediate implication is that no more than $|\Omega'|$ canonical paths can share one same transition.

[2] One technicality that we will not need in this thesis, but which we wish to mention for completeness, is the fact that weights w cannot be known exactly and must be estimated, which is done through an iterative initialization process that runs before the main sampling section of the FPAUS.

[3] The flow eventually needs to be defined for all pairs of states, which is done by extending the one presented here, at low cost to congestion. We refer to [6] for the complete description.

In addition, this encoding has the property that its weight $\Lambda(C)$ can be related to those of I, F, M and M' in such a way that the weight of each canonical path can be upper-bounded by a function of its corresponding encoding and t. The relation is the following.

$$\Lambda(I)\Lambda(F) \leq 8 \min\{\Lambda(M), \Lambda(M')\}\Lambda(C).$$

This is the crucial part that enables the flow's congestion to be upper-bounded by a polynomial. For each transition t, the total weight of all canonical paths that traverse t can be upper-bounded by a factor of the total weight of all states $C \in \Omega'$. Finally, the total weight of Ω' can be upper-bounded by that of Ω, which constitutes a good enough bound to get at most polynomial congestion. We note here again that we intend this subsection as an overview of the proof's idea with sufficient details for this chapter's later sections. For any omitted details and technicalities, we refer to [6].

One question that can come to mind is whether this algorithm, despite its proof relying on the bipartiteness of the graph, can be applied as is to general graphs, without modifications. This is what Štefankovič, Vigoda, and Wilmes [14] disproved by providing a counter-example graph on which *any* Markov chain of the form of MC_{JSV} – that is, potentially differing only in the choice of the weights assigned to each hole pattern – either does not mix in polynomial time, or gives inverse-exponential probability to perfect matchings in its stationary distribution. Their counter-example graph is constructed in such a way that there exists an exponential-sized subset of its matchings, of which only a constant number can transition out of it. One can think of it as the graph G_{JSV} being poorly connected in places, with an exponential-sized subgraph that is connected to the rest of G_{JSV} through only a constant number of edges. As a result, the conductance out of this subset is inverse-exponential in G_{JSV}, thus lower-bounding the mixing time by an exponential function. On the other hand, trying to improve this conductance by tweaking the hole-pattern weights would require assigning inverse-exponential probability to the set of perfect matchings, thus making it unusable as a perfect matching sampler.

While the result by Štefankovič et al. [14] disqualifies Markov chains of the form of MC_{JSV} for general graphs, it does not render this approach a dead end. As they discuss in their paper, modifying the transition rules to allow better communication throughout the graph G_{JSV} could be a potential solution. They propose a few such modifications, but note some challenges associated with them. In the remainder of this section, we will study a different approach to modifying MC_{JSV}. Rather than adapting it to run directly on a general graph, we will take advantage of a relation that exists between the problem on bipartite graphs, and the corresponding half-integral problem on general graphs. The next subsection describes this relation, and all subsequent sections explore ways to take advantage of it.

2.1.2 Relating half-integral to integral matchings on the bipartite graph

Definition 2.1 (half-integral matching)**.** *Let G be an arbitrary graph on n vertices. A* half-integral matching *M is a function that maps each edge of G to a value in $\{0, \frac{1}{2}, 1\}$, such that for every vertex $v \in V(G)$,*

$$\sum_{e \in \delta(v)} M(e) \leq 1. \tag{1}$$

A half-integral perfect matching M' *makes this inequality tight for all* $v \in V(G)$*, i.e.*

$$\sum_{e \in \delta(v)} M'(e) = 1. \tag{2}$$

We will say that an edge is *selected* by a half-integral matching M if it is assigned value 1, that it is *half-selected*, or a *half-edge* of M if it is assigned $\frac{1}{2}$, and *not selected* otherwise. We will call *half-integral* any subgraph of G whose all edges are half-integral in M.

Remark 2.1. We note that perfect matchings can be defined as a special case of half-integral perfect matchings in which for all $S \subset V(G)$ of odd size, the sum $\sum_{e \in \partial S} M(e)$ is odd, where ∂S is the set of all edges with exactly one endpoint in S. In fact, with this added constraint (and that $M(e) \geq 0, \forall e \in E(G)$), we obtain a description of the perfect matching polytope, as given in [12].

We note also that all half-edges of a half-integral matching M form a collection of vertex-disjoint cycles and paths. This follows directly from (1), as the subgraph of half-edges must have maximum degree at most 2. Furthermore, if M is perfect, then its half-edges form only cycles, which again follows from (2), which restricts the subgraph of half-edges to be 2-regular.

We may now state the main claim of this subsection.

Claim 2.1. *For any graph* G*, there exists a bipartite graph* G_B *such that there is a mapping from matchings of* G_B *to half-integral matchings of* G*, that covers the set of half-integral matchings of* G*.*

Proof. Let G be an arbitrary graph on n vertices, and let us construct a bipartite graph G_B from G on $n + n$ vertices as follows. We let the two vertex classes V_1 and V_2 of G_B be copies of $V(G)$, and for convenience, if v is a vertex of $V(G)$, denote $v_1 \in V_1$ and $v_2 \in V_2$ the vertices corresponding to v in $V(G)$. For all $e = (u, v) \in E(G)$, let $e_1 = (u_1, v_2)$ and $e_2 = (u_2, v_1)$ be edges of $E(G_B)$. Thus G_B has twice the number of vertices and edges of G. Figure 2 shows an example graph G and the resulting graph G_B of this construction, along with a half-integral matching of G and a corresponding matching of G_B.

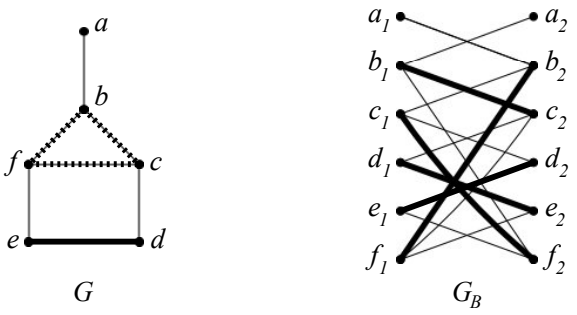


Figure 2: Example graph G and corresponding bipartite graph G_B per construction of Claim 2.1. A half-integral matching on G is shown with bold edges and dashed edges for selected and half-edges respectively. A corresponding matching on G_B is shown with bold edges.

We now proceed to describe the mapping from matchings of G_B to half-integral matchings of G. Consider an arbitrary matching M_B of G_B, and let M be the half-integral matching of G defined as follows.

$$\forall e \in E(G), \quad M(e) = |\{e_1, e_2\} \cap E(M_B)|/2$$

It should be clear that $M(e) \in \{0, \frac{1}{2}, 2\}$ for all $e \in E(G)$. Now let us consider, for an arbitrary vertex $v \in V(G)$, the sum $\sum_{e \in \delta(v)} M(e)$. Let v_1 and v_2 be the vertices corresponding to v in $V_1(G_B)$ and $V_2(G_B)$ respectively, and let S be the union of the neighborhoods of v_1 and v_2 in M_B. Thus S contains either 0, 1 or 2 edges of G_B, all incident to either v_1 or v_2.

Now observe that $\{e_1, e_2\} \cap E(M_B) = \{e_1, e_2\} \cap S$, so we can write

$$\begin{aligned}\sum_{e \in \delta(v)} M(e) &= \sum_{e \in \delta(v)} |\{e_1, e_2\} \cap S|/2 = \frac{1}{2}\Big| \bigcup_{e \in \delta(v)} (\{e_1, e_2\} \cap S) \Big| \\ &= \frac{1}{2}\Big|\Big(\bigcup_{e \in \delta(v)} \{e_1, e_2\}\Big) \cap S\Big| = \frac{|S|}{2} \leq 1\end{aligned}$$

where we used the fact that for any two edges of $E(G)$, the corresponding edges in G_B are distinct, and that $S \subseteq \cup_{e \in \delta(v)} \{e_1, e_2\}$. We note also that if M_B is perfect, then $|S| = 2$ and the final inequality becomes an equality, so that M is a half-integral perfect matching.

Finally, to see that this mapping covers the set of all half-integral matchings of G, consider an inverse mapping which given any half-integral matching M of G, returns a matching M_B of G_B that maps to M. For every edge $e \in E(G)$ that is not a half-edge of M, M_B is such that $\{e_1, e_2\} \cap E(M_B)$ equals $\emptyset$ if $M(e) = 0$, and $\{e_1, e_2\}$ if $M(e) = 1$. Let $\mathcal{C}$ be the collection of all cycles and paths formed by the half-edges of M. For each cycle or path P in $\mathcal{C}$, label its vertices in the order they appear when traversing P in an arbitrary direction (and starting from an arbitrary vertex if P is a cycle). Then, for any two consecutive vertices u and v of P, where u appears first in the order, let M_B select edge (u_1, v_2), and not (u_2, v_1).

We can see that M_B maps to M as required. Indeed, for every half-edge e of M, exactly one edge of $\{e_1, e_2\}$ is matched in M_B, so that $|\{e_1, e_2\} \cap E(M_B)|/2 = \frac{1}{2}$. The same argument applies to the cases where $M(e) = 1$ and $M(e) = 0$.

We now show that the resulting M_B is a valid matching of G_B. To this end, assume towards contradiction that there exists a vertex of G_B that is matched twice, and let $u \in V(G)$ be the corresponding vertex in G. We may assume by symmetry that the vertex that is matched twice is in $V_1(G)$, and we denote it u_1. Then there are two vertices v and w of G such that (u_1, v_2) and (u_1, w_2) are selected by M_B.

If u_2 is also matched in M_B, then either there is one of (u, v) or (u, w) that is selected by M, or there is a fourth vertex z such that (u, z) is half-selected by M. Either way, equality 1 is not verified for v and M is invalid.

If u_2 is not matched, then both (u, v) and (u, w) are half-matched by M in G. Moreover, they belong to a (potentially closed) half-integral path P of M, and u has degree 2 in P. If P is not a cycle, then u appears in-between v and w in the ordering of P's vertices, so that u_2 must be matched by construction, which is a contradiction. Similarly if P is a cycle and u is not first in the ordering of P's vertices. If P is a cycle and u is first in the ordering of P's vertices, and assuming by symmetry that v is second, then M_B must select (w_1, u_2) and u_2 is matched, a contradiction. □

Observe that, when constructing M_B from M in the above proof, only one arbitrary choice is made which has an impact on the resulting M_B, namely the direction of the ordering of each path P.

Moreover, if P is a cycle, only the choice of the direction around it (clockwise or counter-clockwise) has an impact on the resulting matching M_B – the choice of the first vertex in the ordering is irrelevant. Since half-integral cycles and paths are not self-intersecting, this shows the following corollary.

Corollary 2.1. *Given G and G_B defined as above, if M is a half-integral matching of G with k half-integral cycles or paths, then there exist 2^k matchings of G_B that map to M. In particular, if M is an integral matching of G, then there is exactly one matching of G_B that maps to M.*

This corollary implies that our construction for G_B does not constitute an immediate reduction from the problem of counting half-integral matchings on arbitrary graphs, to that of counting integral matchings on bipartite graphs. However, if possible to sample only matchings in G_B that correspond to integral matchings in G, then we can obtain a uniform sampler of integral matchings of G. This is what we will explore in the remainder of this chapter, where we will let G and G_B be defined as they were in this section.

2.2 Weighing down half-integral matchings

The first idea we describe is a modification of the Markov chain by Jerrum et al. [6] that weighs down matchings as a function of their number of half-integral edges. This weighting is intended to be used as part of the Markov chain's Metropolis rule to lower the probability of half-integral matchings in a "gradient" that gives higher priority to those with lower half-edge counts. We will describe this weight function and see that such a Markov chain either does not mix fast enough, or gives too low probability to integral matchings.

The weight function we consider in this section is such that it decreases by a multiplicative factor for each new half-edge. Formally, we will consider $\omega : \Omega \to \mathbb{R}^+$, where $\Omega := \mathcal{M}_2(G_B) \cup \mathcal{M}(G_B)$, such that it verifies the following condition: for any two matchings M_B and M'_B of Ω, and the corresponding half-integral matchings of G, M and M' respectively, if M' has exactly 1 more half-edge than M, then $\omega(M'_B)/\omega(M_B) = \gamma(n)$, where $\gamma(n)$ is some function of n that we will constrain later. Therefore, if M' has k more half-edges than M, then $\omega(M'_B)/\omega(M_B) = \gamma(n)^k$. Moreover, we require that $\omega(M_B)$ is the same for all M_B who share their number of half-edges. We will also assume that $\omega(M_B)$ depends only on n and the number of half-edges in M_B. The motivation for such a weight function is the following: on one hand we need the weight of some matchings to reach inverse-exponential, while on the other, some transition probabilities need to remain at least inverse-polynomial to ensure fast enough mixing. Therefore, if $\gamma(n)$ is small enough, then transition probabilities are not too low, yet we do get an exponentially decreasing gradient with the number of half-edges.

For instance, one such weight function could be defined as $\omega(M_B) = \prod_{e \in E(G)} \exp(\rho^{M(e)})$ for all $M_B \in \Omega$, for some parameter $\rho \geq 1$, and where M is the corresponding half-integral matching in G. In this case we would have $\gamma(n) = \exp(\sqrt{\rho} - \rho)$. Note that if $\rho = 1$ then $\gamma(n) = \omega(M_B) = 1, \forall M_B \in \Omega$ and we fall back to MC_{JSV} which is known to mix fast enough, while as ρ tends to infinity, the weight ratio of an integral to a half-integral matching tends to infinity, and we effectively disconnect the underlying graph G_{JSV} (because any path from one integral matching to another on G_{JSV} has to traverse a half-integral matching). We would thus like to find an in-between that mixes fast while decreasing probabilities of half-integral matchings enough.

The changes made to MC_{JSV} are effectively only in the Metropolis rule: when in state M and for proposed matching M', the metropolis rule now allows transition to M' with probability $\min\left(1, \frac{\Lambda(M')\omega(M')}{\Lambda(M)\omega(M)}\right)$. We recall that for any matching $M \in \mathcal{M}(u, v)$ (i.e. with holes u and v), $\Lambda(M)$ is proportional to $\frac{|\mathcal{M}|}{|\mathcal{M}(u,v)|}$.

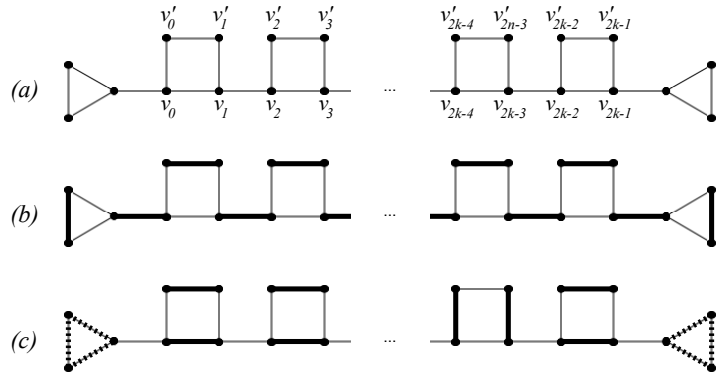


Figure 3: Example graph in (a) with exponentially many half-integral perfect matchings, and exactly one perfect matching in (b). One half-integral perfect matching is given in (c). Selected edges are shown in bold, while half-edges are dashed.

Let now $G = G_k$ be the graph shown in Figure 3 (a), and G_B its corresponding bipartite graph according to Claim 2.1. It is constructed from the chain of boxes with k boxes (introduced earlier in Figure 1), by appending a triangle to each of its two degree-1 vertices. Figure 3 (b) shows an integral perfect matching of G, and Figure 3 (c) shows a half-integral perfect matching of G. By noticing the relation of G with the chain of boxes of Figure 1, we can see that this integral perfect matching is unique, and that there are at least 2^k half-integral perfect matchings with exactly 2 half-triangles. By Corollary 2.1, a half-integral perfect matching of G with l distinct half-integral paths corresponds to 2^l distinct perfect matchings of G_B. Therefore, there is 1 perfect matching of G_B corresponding to an integral perfect matching of G, and at least $4 \cdot 2^k$ perfect matchings of G_B corresponding to a half-integral one of G with 6 half-edges.

Given any graph H and its bipartite equivalent H_B, we denote $\mathcal{M}_l^{(1/2)}(H_B)$ the set of all matchings of $\mathcal{M}_2(H_B) \cup \mathcal{M}(H_B)$ whose corresponding half-integral matching in H has exactly l half-edges. When clear from context, we may omit the graph and simply write $\mathcal{M}_l^{(1/2)}$. Thus here for G as given in Figure 3 (a), we have $|\mathcal{M}_0^{(1/2)}| = 1$ and $|\mathcal{M}_6^{(1/2)}| \geq 4 \cdot 2^k = 2^{k+2}$.

We claim that $\mathcal{M}_6^{(1/2)}$ contains exclusively perfect matchings of G_B, i.e. $\mathcal{M}_6^{(1/2)} \subseteq \mathcal{M}(G_B)$. To see this, consider a matching M_B of Ω on G_B such that its corresponding half-integral matching M of G has exactly 6 half-edges. We first show that M must have 2 half-integral triangles. Assume it does not. Note that M cannot have two half-integral open paths, as otherwise M_B would have 4 holes. Therefore, M must have at most one half-integral open path, and some number of half-integral cycles. If it has no cycle, then its open path P has length 6, so we can match vertices of P arbitrarily to obtain an integral near-perfect matching of G with 1 hole. Because $V(G)$ is even, this is a contradiction. If it has at least one half-integral cycle, then that cycle can have lengths either 3 or 4, as G contains no cycle of other lengths. If it has length 4, then its vertices can be matched arbitrarily within that cycle, and we are left with a half-integral matching with one path of length 2, yielding the same contradiction as before. If now the half-integral cycle has length 3 – and is therefore one of the two triangles of G – then because G has no parallel edges, there cannot be more than one other half-integral path P, whose length is therefore 3. If P is closed, it must be the second triangle of G and we are done. If it is not, then we can perfectly match its vertices to obtain a half-integral perfect matching of G with exactly one half-integral triangle. This matching contains an integral perfect matching of the graph obtained by removing all vertices of that triangle.

Because this graph has $|V(G)|-3$ vertices, it has an odd number of vertices and therefore no perfect matching, a contradiction. So any matching in $\mathcal{M}_6^{(1/2)}$ corresponds to a matching of G with 2 half-triangles. It now suffices to see that any such half-integral matching is perfect. Assume to this end that there is such a matching that is not perfect. It cannot have two half-matched vertices, as otherwise it would have more than 6 half-edges, so it must have exactly 1 unmatched vertex. Removing both triangles gives us a matching with exactly 3 holes on the chain of boxes, which is a contradiction since it has an even number of vertices. So we have that $\mathcal{M}_6^{(1/2)} \subseteq \mathcal{M}(G_B)$. Of course, we also have that $\mathcal{M}_0^{(1/2)} \subseteq \mathcal{M}(G_B)$.

We now compute the ratio of probabilities of integral to half-integral matchings with 6 half-edges. Let M_B and M'_B be arbitrary matchings of $\mathcal{M}_0^{(1/2)}$ and $\mathcal{M}_6^{(1/2)}$ respectively.

$$\begin{aligned}\frac{\Pr(\mathcal{M}_0^{(1/2)})}{\Pr(\mathcal{M}_6^{(1/2)})} &= \frac{\sum_{M_B \in \mathcal{M}_0^{(1/2)}} \Lambda(M_B)\omega(M_B)}{\sum_{M'_B \in \mathcal{M}_6^{(1/2)}} \Lambda(M'_B)\omega(M'_B)} = \frac{\omega(M_B)}{|\mathcal{M}_6^{(1/2)}|\omega(M'_B)} \\ &= \frac{1}{|\mathcal{M}_6^{(1/2)}|}\gamma(n)^{-6} \leq 2^{-k-2}\gamma(n)^{-6} \end{aligned} \tag{3}$$

where we used the fact that $\omega(M)$ depends only on n and the number of half-edges of M and is the same for all $M \in \mathcal{M}_l^{(1/2)}$ for any l, and that $\mathcal{M}_0^{(1/2)} \cup \mathcal{M}_6^{(1/2)}$ all belong to the same hole pattern, so that $\Lambda(M_B) = \Lambda(M'_B)$ for all $M_B \in \mathcal{M}_0^{(1/2)}$ and $M'_B \in \mathcal{M}_6^{(1/2)}$. As a consequence, in order for $\Pr(\mathcal{M}_0^{(1/2)})$ to be at least inverse-polynomial, it must be that $\gamma(n)^{-6} \geq 2^{k+2}\text{poly}^{-1}(n)$, for some polynomial function of n poly.

Consider now the n-cycle C, and let C_B be its corresponding bipartite graph according to Claim 2.1. There is exactly two integral perfect matchings on C, each corresponding to exactly one perfect matching of C_B by Corollary 2.1. Therefore, $|\mathcal{M}_0^{(1/2)}(C_B)| = 2$. Similarly, any matching with 1 half-edge corresponds to exactly two perfect matchings of C_B by Corollary 2.1. Because there are n possible choices for the half-edge, we have that $\mathcal{M}_1^{(1/2)}(C_B) = 2n$. Finally, there is exactly one half-integral matching on C with n half-edges, which by Corollary 2.1 corresponds to two matchings on C_B, so that $\mathcal{M}_n^{(1/2)}(C_B) = 2$.

We are now interested in the sizes of each hole pattern $\mathcal{M}(u_B, v_B)$ on C_B such that $(u_B, v_B) \in E(C_B)$. Let u and v be the vertices of C corresponding to u_B and v_B by the construction of Claim 2.1. Consider an arbitrary matching $M_B \in \mathcal{M}(u_B, v_B)$ and its corresponding matching M in C. One can see that M must have a half-integral path P from u to v, with P odd. There are exactly two possible such paths, each corresponding to one half-integral matching on C, which in turn corresponds to one matching on $\mathcal{M}(u_B, v_B)$. Therefore $\mathcal{M}(u_B, v_B)$ has size 2. Considering the hole pattern $\mathcal{M}$, we can see that it corresponds exactly to the union of $\mathcal{M}_0^{(1/2)}$ and $\mathcal{M}_n^{(1/2)}$, so that its size is 4.

We may now compute the probability of transitioning out of the unique integral perfect matching on C. First observe that given an integral perfect matching, the only transitions available to the Markov chain are edge removal, yielding a half-integral matching with exactly 1 half-edge. Therefore, it suffices to consider half-integral matchings with 1 half-edge as candidates for exiting the integral perfect matching. Let therefore M_B an M'_B be two matchings of $\mathcal{M}_0^{(1/2)}(C_B)$ and $\mathcal{M}_1^{(1/2)}(C_B)$ respectively, and let M and M' be their corresponding matchings in C. From state M_B, state M'_B is proposed to the Metropolis filter with probability $\frac{1}{4n}$ (factor $\frac{1}{2}$ for not staying in M_B by laziness,

and factor $\frac{1}{2n}$ for the choice of edge to remove). Thus we have

$$\Pr(M'_B|M_B) = \frac{1}{4n}\min\left(1, \frac{\Lambda(M'_B)\omega(M'_B)}{\Lambda(M_B)\omega(M_B)}\right) = \frac{1}{4n}\min\left(1, \frac{4}{2}\gamma(n)\right) \leq \frac{1}{2n}\gamma(n) \tag{4}$$

We now have two conflicting constraints on $\gamma(n)$. On one hand, in order for the ratio of Equation 3 to be at least inverse-polynomial, we need $\gamma(n)$ to be at most a polynomial factor of $(2^{1/6})^{-k}$, by Equation 3 (recall that k is the number of boxes in G, and is thus $\Theta(n)$). Plugging this inequality into Equation 4 would yield an inverse-exponential probability of transitioning to $M'_B \in \mathcal{M}_1^{(1/2)}(C_B)$. Since $\mathcal{M}_1^{(1/2)}(C_B)$ has size $2n$, the total probability of leaving the integral perfect matching $M_B \in \mathcal{M}_0^{(1/2)}(C_B)$ would remain inverse-exponential and the Markov chain would not mix. If on the other hand we want the transition probability $\Pr(M'_B|M_B)$ to be at least inverse-polynomial, then we need to choose $\gamma(n)$ to be at least inverse-polynomial. In this case, Equation 3 says that the probability of integral perfect matchings on G is at most inverse-exponential, thus too low.

This shows that any weight function of the same form as ω yields a Markov chain that either does not mix in polynomial time, or does not give inverse-polynomial probability to integral perfect matchings. If we wish to use a weight function that classifies matchings by their number of half-integral edges, it must therefore depend in some way on the graph. One difference between G and C is their number of cycles, so that we could think of making ω depend also on the number of cycles in the graph. A more advanced strategy would be similar to that of Jerrum et al. [6], which weigh each pattern by a function of their size. Assuming their procedure to estimate these sizes can extend to estimating the number of matchings with k half-edges for all $k \in \{0, \ldots, |E(G)|\}$, one could think of using weight $\frac{|\mathcal{M}_0^{(1/2)}|}{|\mathcal{M}_k^{(1/2)}|}$ for matchings with k half-edges. While this would bypass the problem described here, we will discuss in section 2.3 some difficulties in analyzing such a Markov chain, specifically its mixing time.

2.3 Limiting number of half-integral cycles and edges

Following the previous section, we consider a modification of MC_{JSV} that, instead of re-weighting according to the number of half-edges, simply sets a threshold on the allowed number of half-edges. Moreover, we will now consider the number of half-integral cycles, rather than that of half-edges. The main result of this section is a reduction from the problem of sampling integral perfect matchings on arbitrary graphs to that of sampling half-integral perfect matchings with at most $n^{\frac{k}{k+1}}$ half-integral cycles, for k a non-negative constant. We will start with proving this result, and then discuss the difficulties of restricting MC_{JSV} to states with any bounded number of half-integral cycles.

Theorem 2.1. *If $\sharp PM_{\gamma(n)}$ is the problem of sampling u.a.r. perfect matchings on an arbitrary graph on n vertices with at most $\gamma(n)$ half-integral cycles, then there exists a polynomial-time approximation-preserving reduction from $\sharp PM_{n^{\frac{k}{k+1}}}$ to $\sharp PM_0$, where k is a constant non-negative number.*

Note that this theorem can be adapted to various other local errors such as half-integral edges, or even holes in integral matchings. We will discuss this at the end of this section and the next section.

Proof of Theorem 2.1. Let G be an arbitrary graph on n vertices. We construct G' as the union of $\lceil n^k \rceil$ copies of G, for a constant k. Note that G' thus has n^{k+1} vertices.

We call A a fully-polynomial almost-uniform sampler of half-integral perfect matchings on arbitrary graphs with no more than $n^{\frac{k}{k+1}}$ half-integral cycles. Thus on input a graph H, A can sample from a collection Ω_H of half-integral perfect matchings with at most $n^{\frac{k}{k+1}}$ half-integral cycles, according to a distribution that has total variation distance at most δ to the uniform distribution on Ω_H for some $\delta > 0$, and in time polynomial in n and δ^{-1}.

Running A on input G', we can thus sample half-integral perfect matchings of G' with at most $(n^{k+1})^{\frac{k}{k+1}} = n^k$ half-integral cycles. By the pigeonhole principle, for any matching $M \in \Omega_{G'}$, at least one copy G_i of G does not contain a half-integral cycle of M. We denote $G_i(M)$ the matching obtained by restricting M to G_i. Thus by running the algorithm N times for N large enough, we can have arbitrarily high probability of obtaining a matching $M \in \Omega_{G'}$ such that $G_0(M)$ is integral. We thus consider a sampler B that runs A on G' until it samples a matching M such that $G_0(M)$ is perfect, and outputs $G_0(M)$.

For completeness, let us explicitly compute the value of N. If we want a failure probability (i.e. of not obtaining a perfect matching after N samples from A on G') to be at most a parameter $\varepsilon > 0$, a value of $N = \log(\frac{1}{\varepsilon})n^k$ is sufficient. This can be seen by considering the binomial process that counts the number of integral matchings appearing on G_0 in N samples, which gives the probability of obtaining none to be

$$\left(1 - \frac{1}{n^k}\right)^N \leq \exp\left(-\frac{N}{n^k}\right)$$

by a standard inequality. If we let $N = \log\left(\frac{1}{\varepsilon}\right) n^k$, we thus have that

$$\left(1 - \frac{1}{n^k}\right)^N \leq e^{-\log\left(\frac{1}{\varepsilon}\right)} \leq \varepsilon$$

We now verify that B is a valid fully-polynomial almost-uniform sampler for integral perfect matchings on G that is also approximation-preserving with respect to A. The time complexity is easily seen to remain polynomial, as we simulate A on a graph that is only polynomially larger, for a polynomial number of iterations in both n and $\log(\varepsilon^{-1})$. Turning to the quality of the approximation, let $\tilde{\mathcal{U}}_{G'}$ be the distribution that A samples at on input G', and let $\mathcal{U}_{G'}$ be the uniform distribution on $\Omega_{G'}$. We know that $\tilde{\mathcal{U}}_{G'}$ and $\mathcal{U}_{G'}$ have total variation distance at most δ, by definition of A. If $\tilde{\pi}_{G'}$ and $\pi_{G'}$ denote the probability mass functions of $\tilde{\mathcal{U}}$ and $\mathcal{U}$ respectively, we thus have

$$\max_{S \subseteq \Omega} |\pi_{G'}(S) - \tilde{\pi}_{G'}(S)| \leq \delta.$$

Consider now an arbitrary subset $\mathcal{S} \subseteq \mathcal{M}(G_0)$ of perfect matchings of G_0. We denote π_{G_0} the probability mass function of the uniform distribution over $\mathcal{M}(G_0)$, and $\tilde{\pi}_{G_0}$ the probability mass function of our sampler B. Now observing that $\tilde{\pi}_{G_0}(\mathcal{S}) = \tilde{\pi}_{G'}(\{M' : G_0(M') = \mathcal{S}\})$, and $\pi_{G_0}(\mathcal{S}) = \pi_{G'}(\{M' : G_0(M') = \mathcal{S}\})$, we obtain

$$|\tilde{\pi}_{G_0}(\mathcal{S}) - \pi_{G_0}(\mathcal{S})| = |\tilde{\pi}_{G'}(\{M' : G_0(M') \in \mathcal{S}\}) - \pi_{G'}(\{M' : G_0(M') \in \mathcal{S}\})| \leq \delta,$$

by the total variation distance of distributions $\mathcal{U}_{G'}$ and $\tilde{\mathcal{U}}_{G'}$, which proves that our procedure is an almost-uniform sampler with total variation distance at most δ to the uniform distribution, thus making our reduction approximation-preserving. This completes the proof. □

We now discuss modifications to MC_{JSV} that limit the number of half-integral cycles to a bound $\gamma(n)$, where n is the size of the input graph. The way we do so is by adding a step at the end of

the transition process of MC_{JSV} that rejects the proposed state M' if it corresponds to a matching with too many half-integral cycles. Let $MC_{<\gamma(n)}$ be the resulting Markov chain, and $\Omega_{<\gamma(n)}(G)$ the collection of matchings that constitute its state space on input graph G.

One central problem we can see with this approach emerges when trying to bound the Markov chain's mixing time by a function of the congestion of a flow on its underlying graph, as is done in [6]. To recall, the flow is defined by a canonical path between any near-perfect matching I and any perfect matching F. This canonical path is then encoded into a state of an extended state space whose weight is only polynomially larger than that of the Markov chain's state space.

We briefly recall here the construction of the canonical path between I and F as described by Jerrum et al. [6], but refer to the paper for full details. It is first observed that $I \oplus F$, the symmetric difference of both matchings, consists of one path and some number of cycles, and that flipping all of them in I gives F (where *flipping* a path P in I is done by a sequence of exchanges, finished with an edge addition, and started with an edge deletion if P is a cycle – also called *unwinding* in [6]). The canonical path is defined by the sequence of transitions that flip the path first and then all cycles one by one, all in some arbitrary canonical order and direction.

Recall that, in our half-integral setting, the Markov chain $MC_{<\gamma(n)}$ runs on integral perfect matchings of the bipartite graph G_B corresponding to the general graph G. In the remainder of this subsection, when considering the symmetric difference between any two half-integral matchings, we will implicitly refer to the symmetric difference of two corresponding matchings of G_B.

Consider now the scenario depicted in Figure 4. Let G be the graph shown in (a), and let I and F be two half-integral matchings of G depicted in (b) and (c) respectively. Notice that the half-edges of I form a cycle and a path of length 1, and that those of F form a single cycle, distinct from that of I. The result of $I \oplus F$ consists of one path P_0, and two cycles C_1 and C_2, shown respectively in blue, yellow and red in (d). Figure 5 shows intermediate states on the canonical path defined above from I to F, after flipping each of P_0, C_1 and C_2, and for both choices of ordering of C_1 and C_2. We can see that, regardless of that choice, the path traverses a state with $\Theta(n) > O(n^{\frac{k}{k+1}})$ half-integral cycles, for any $k > 0$, so that it exits the Markov chain's state space.

Because there are other paths from I to F that do not exit the allowed state space, our argument does not imply that the Markov chain's underlying graph is disconnected. However, it does make the proof of [6] inapplicable as is. If we wish to follow a similar strategy, we need to design a canonical path from I to F that does not exit $\Omega_{<\gamma(n)}$, but still encodes enough information about I and F. In other words, we need to re-route through $\Omega_{<\gamma(n)}$ all paths that exit it. If such paths are too many,

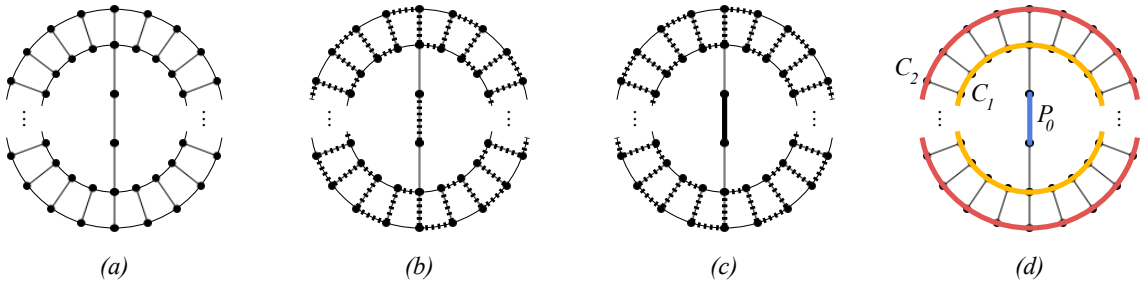


Figure 4: Example graph with pair of matchings and their symmetric difference. (b) and (c) show two matchings I and F respectively on the graph shown in (a), where half-selected edges are represented with dashed lines while selected edges are in bold. (d) highlights the path P_0 and the two cycles C_1 and C_2 that form $I \oplus F$, in blue, yellow and red respectively.

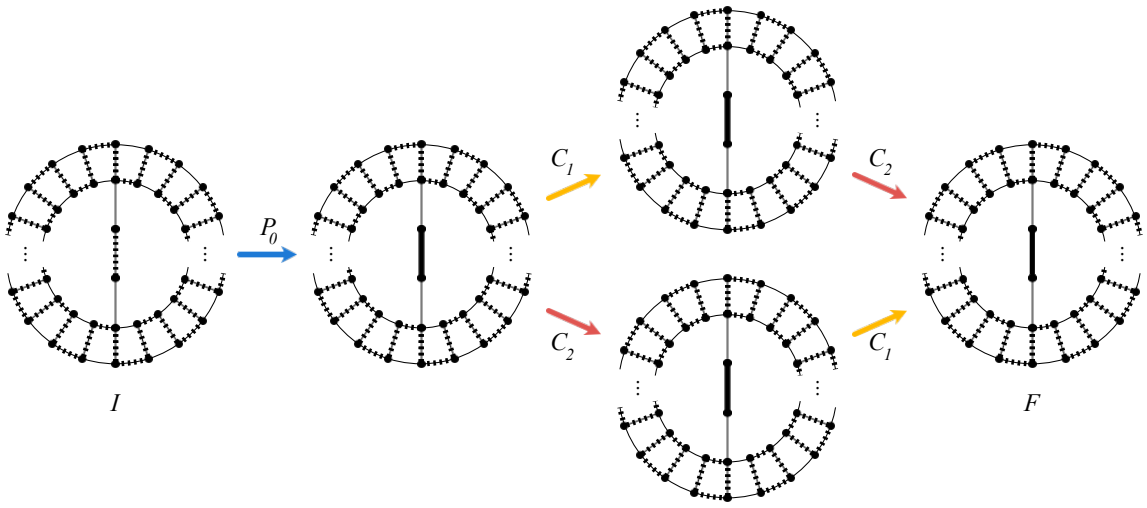


Figure 5: Intermediate results of flipping paths P_0, C_1 and C_2, where each transition is depicted by an arrow labelled with the corresponding path.

any such re-routing would induce exponential congestion on some transitions and consequently the Markov chain would not mix. So a question is whether there exists a canonical path construction that does not induce exponential congestion on the Markov chain's underlying graph. We emphasize also the fact that this construction needs to be somewhat invertible, i.e. so that given a canonical path P one can reconstruct the pair (I, F) – or a small enough set of candidate pairs, as we will discuss at the end of this chapter. It is interesting to recall now Corollary 2.1, as it can give a sense of the relative number of half-integral matchings with $O(n)$ half-integral cycles – thus outside of $\Omega_{<\gamma(n)}$ – compared to those with only $O(n^{\frac{k}{k+1}})$, which is exponential. See the discussions in section 2.5 for future research directions following this approach.

Limit number of half-edges The main result of this section, Theorem 2.1, essentially states that we can tolerate up to $n^{\frac{k}{k+1}}$ errors in a solution, for constant k. In this case, errors were half-integral cycles. The solution was to bypass the errors by duplicating the graph enough times, focusing on one of the copies G_0, and rejecting all solutions with an error in G_0.

This same strategy can be applied to other types of local errors, i.e. ones that can be defined by a subgraph of the input, and that do not propagate to other components of the input. For example, the result extends naturally to half-integral edges as errors.

Theorem 2.2. *Given a fully-polynomial almost-uniform sampler of half-integral perfect matchings of G with a maximum number of half-edges upper-bounded by $n^{\frac{k}{k+1}}$ for a constant k, there exists a fully-polynomial almost-uniform sampler of integral perfect matchings of G.*

We note that defining a Markov chain similar to $MC_{<\gamma(n)}$ that limits the number of half-integral edges to some bound $\gamma(n)$ would present the same problems as did $MC_{<\gamma(n)}$. A similar argument to the one made with the counter-example shown in Figure 4 can be made for such a Markov chain: consider the graph shown in Figure 4 (a) and two integral perfect matchings I and F that differ only in the cycle C_1, shown in red in Figure 4 (d). Because they are integral matchings, in order to go from I to F, a canonical path needs to flip C_1 twice: once to make it half-integral, and again to

reach F. This intermediate state thus has $n/3 > n^{\frac{k}{k+1}}$ half-edges for n large enough, and therefore exits the Markov chain's state space.

2.4 Limiting number of holes in integral matchings

Following the result of the previous section, we discuss here another type of error that can be bypassed using this strategy, in the context of integral solutions. We capture this in the following theorem, which essentially states that we can tolerate up to $n^{\frac{k}{k+1}}$ holes in any integral matching:

Theorem 2.3. *Given a fully-polynomial almost-uniform sampler of matchings of G with a maximum number of holes upper-bounded by $n^{\frac{k}{k+1}}$ for a constant k, there exists a fully-polynomial almost-uniform sampler of perfect matchings of G.*

This is a powerful result, as, in particular, it reduces the problem of sampling perfect matchings to that of sampling near-perfect and perfect matchings. In other words, it means that we need not worry about the ratio between near-perfect matchings and perfect matchings to use a sampler of $\mathcal{M}_2 \cup \mathcal{M}$ as a sampler of $\mathcal{M}$. Now this does not immediately invalidate all efforts made by Jerrum and Sinclair [4] and Jerrum, Sinclair, and Vigoda [6]. In [4], the ratio of near-perfect to perfect matchings is guaranteed to be no more than polynomial by restricting the solution to bipartite graphs G of minimum degree $V(G)/4$ (called "dense"). However, this restriction is also what allows them to prove that the Markov chain mixes in polynomially many steps. Similarly, the reason for the hole pattern strategy in [6] is to weigh down the probability of near-perfect matchings so that it does not over-weigh that of perfect matchings, in case they are exponentially more numerous. This does not only serve the purpose of making a polynomial fraction of the samples perfect; it again also allows the proof the Markov chain's mixing time to hold. One can therefore not say that these efforts could have been avoided, but it does, for future research, remove a reason for weighing down near-perfect matchings.

In fact, this result is more general even, as it allows not only near-perfect matchings, but any matching with a maximum number of holes bounded by $n^{\frac{k}{k+1}}$. For instance, it would suffice to have an algorithm capable of sampling matchings whose number of holes is at most $n^{\frac{k}{k+1}}$, at least a polynomial fraction of the time. We would then simply reject samples with too many holes. Another use of this result would be to find a type of graph whose number of matchings with more than $n^{\frac{k}{k+1}}$ holes is guaranteed to be at most a polynomial fraction of all matchings. Towards a general result, one could study ways to modify arbitrary graphs in ways that "duplicate" matchings with few holes, and not others, so as to reduce the proportion of matchings with too many holes. We would thus, rather than weighing down invalid solutions, weigh up valid ones by making them more numerous, in a controlled fashion. In the next chapter, we describe generic transformations to arbitrary graphs that have known consequences on matching counts, and that introduce other properties on the graph. Specifically, one of them multiplies by 4 the number of perfect matchings in the graph. In Chapter 5, where we discuss future research directions, we will address one way this transformation could be used to weigh up matchings with few holes, with the aim of obtaining a polynomial fraction of matchings with fewer than $n^{\frac{k}{k+1}}$ holes, for some k.

Proof of Theorem 2.3. The proof is almost identical to that of Theorem 2.1. The construction of G' and the choice of the number of samples N are identical, while the argument about the total variation distance of the resulting sampler only needs to refer to holes in place of half-integral cycles. □

2.5 Discussions and future research

For future reserach, we believe an interesting direction is to study the mixing time of the Markov chain $MC_{<\gamma(n)}$, either by finding a new canonical path definition, or through another technique that does not require them. If using congestion or conductance (as was done by Jerrum and Sinclair [4]), the technique of encoding canonical paths into new states through an injective mapping should be kept in mind. We note here that this technique does not require the mapping to be injective like it is in [4] and [6], but only for it to map at most a polynomial number of canonical paths to the same state – the polynomial would then propagate through to the final time complexity. We note also that, as is done in [6], the encoding needs not be part of the allowed state space of the Markov chain, as long as the weight of the state space it belongs to can be bounded by a polynomial in the allowed state space.

We note here also, as future research, a modification to the classes in which the state space is partitionned in [6]. We could consider dividing it further into groups that share the number of half-integral cycles, or edges. Note however that a similar argument to the one made in section 2.3 can be made here as well. Considering any class $\mathcal{C}_i$ with two holes, say u and v such that u and v form an edge, and some number k of half-edges or half-integral cycles. Then from any matching in $\mathcal{C}_i$, selecting (u, v) can either create or remove a half-edge, and similarly open or close a half-integral cycle. Therefore, the Markov chain can transition to more than one class from $\mathcal{C}_i$. This makes the analysis of the relative weights of different classes less practicle than in [6], where adding or removing two holes only ever moves to a single other class, whose weight can directly or indirectly be related to that of $\mathcal{M}$. This is what allows them to relate the weights of the pair I, F to the weight of their encoding C, and in turn bound the congestion.

Finally, the last result of this section – up to $n^{\frac{k}{k+1}}$ errors can be ignored – is important to keep in mind for future research on the problem. As we already discussed, it means that the requirement of countering an exponential ratio in the number of near-perfect and perfect matchings might be superfluous (that is, if it is not necessary for the remainder of the algorithm to function; for example for the underlying Markov chain to mix as in [4] and [6]). Furthermore, it generalizes to allow more than a constant number of holes, which could constitute a problem that is easier to tackle. For example, a re-weighting of matchings that gives low enough probability to solutions with more than $n^{\frac{k}{k+1}}$ holes would suffice.

3 Graph modifications with predictable changes to matching count

In this section, we introduce a set of tools for modifying a given graph's structure in ways that have predictable consequences on its matchings. These modifications are local, in the sense that they modify only subgraphs of constant size or pending paths of the given graph. While some of these tools will prove useful in the next chapter, we will also provide two applications them in sections 3.2 and 3.3, where we describe reductions for the problem of counting perfect matchings from arbitrary graphs to graphs of maximum degree 3, and to tripartite graphs, respectively.

3.1 Graph modification tools

3.1.1 Forcing edge selection

Given an arbitrary graph G and a fixed edge $e = (u, v) \in E(G)$, let us describe the construction of a graph G' such that there exists a bijection between all perfect matchings of G that select e, and all perfect matchings of G'. This will not be used in this thesis but is given for general interest. Starting from G, we remove e and add four vertices v_0, v_1, v_2, v_3 and five edges $(v_0, v_1), (v_1, v_2), (v_1, v_3), (v_2, u), (v_3, v)$. Let G' be the resulting graph. This construction is depicted in figure 6.

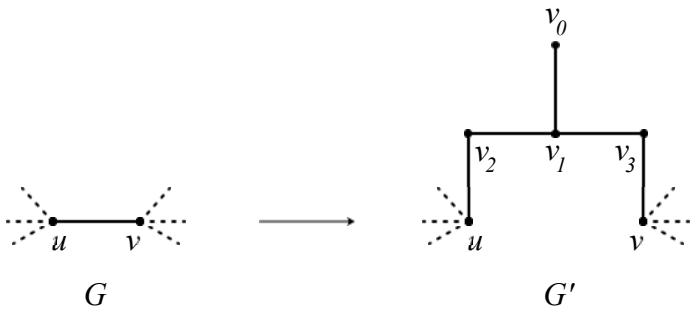


Figure 6: Modification of an edge (u, v) to force its selection by perfect matchings.

It should be clear that any perfect matching M' on G' contains edge (v_0, v_1), and therefore must match u and v to v_2 and v_3, respectively. As the rest of the graph is identical in G and G', the matching defined by the edge set $E(M') \setminus \{(v_0, v_1), (v_2, u), (v_3, v)\} \cup \{(u, v)\}$ is a valid perfect matching of G. Similarly, for any perfect matching M in G that selects (u, v), the matching defined by the edge set $E(M) \setminus \{(u, v)\} \cup \{(v_0, v_1), (v_2, u), (v_3, v)\}$ is a valid perfect matching of G'. Note that this construction conserves planarity, and that G' is k-partite if G is k-partite, for $k > 2$. We also mention the opposite graph modification consisting in removing edge e, that limits the set of matchings to those that do not contain e.

3.1.2 Inserting and removing pending paths

Inserting pending path *(or vertex splitting)* This construction will be useful in the next chapter to effectively allow a path to traverse the same vertex twice without self-intersecting. We will also see at the end of this chapter two applications of this tool that allow the reduction for the problem of

counting perfect matchings from arbitrary graphs to graphs of maximum degree 3 and to tripartite graphs.

Construction 3.1 (Vertex splitting)**.** *Given an arbitrary graph G, consider a vertex $v \in V(G)$ of degree at least 2, and partition its neighbors into two classes $N_1(v)$ and $N_2(v)$. Let G' be the graph obtained by removing v and its incident edges, and inserting three vertices v', v_1, v_2 along with edges (v', v_1), (v', v_2), (v_1, w_1) for all $w_1 \in N_1$ and (v_2, w_2) for all $w_2 \in N_2$. Note that, because G does not have parallel or loop edges, $\delta(v_1) \cap \delta(v_2) = \{v'\}$. This construction is depicted in Figure 7.*

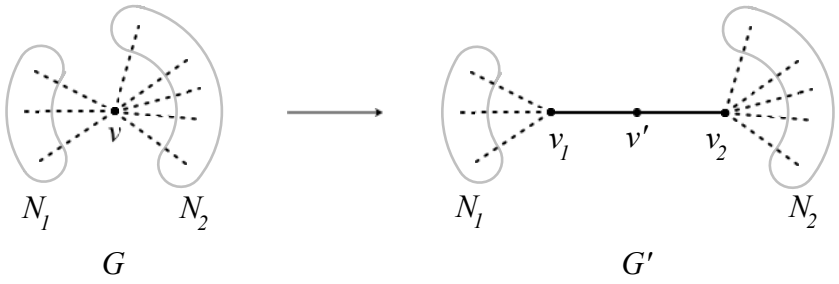


Figure 7: Pending path insertion on vertex v, given partition N_1, N_2.

In the rest of this thesis, we will refer to this transformation as *splitting* vertex v along partition (N_1, N_2).

Lemma 3.1. *Given an arbitrary graph G and graph G' obtained from G by splitting an arbitrary vertex $v \in V(G)$ according to Construction 3.1, there exists a bijection between $\mathcal{M}(G)$ and $\mathcal{M}(G')$.*

Proof. Consider any perfect matching M in G, and let $x \in V(G)$ be the vertex that it matches to v. Assume without loss of generality that $x \in N_1$. Then the subgraph of G' obtained from $E(M) \setminus \{(x, v)\} \cup \{(x, v_1), (v', v_2)\}$ is a perfect matching of G', as it matches v' to v_2, v_1 to x, and all other vertices identically to M. Conversely, consider any perfect matching M' in G' that matches v_1 to some $x \in V(G')$, and consequently v' to v_2. By construction, $x \in N_1(v)$. Furthermore, because v_2 is matched to v', all vertices of $\delta(v) \setminus \{x\}$ are matched outside of $\{v_1, v', v_2\}$, so that $M' \setminus \{x, v_1, v', v_2\}$ is a perfect matching of $G \setminus \{v, x\}$. Therefore, the subgraph M of G obtained from $E(M') \setminus \{(x, v_1), (v', v_2)\} \cup \{(v, x)\}$ is a valid perfect matching of G. □

This lemma extends naturally to near-perfect matchings with given hole.

Lemma 3.2. *Given an arbitrary graph G and vertex $v \in V(G)$, and graph G' obtained from G by splitting v according to Construction 3.1, there exists a vertex $v' \in V(G')$ such that there is a bijection between near-perfect matchings of G with hole v and near-perfect matchings of G' with hole v'.*

Proof. There are two cases to consider. If Construction 3.1 splits a vertex w different from v, then the argument of the proof of Lemma 3.1 holds here as well. If it splits v itself, then let v_1, v', v_2 be the path that v is replaced with. We claim that any near-perfect matching of G with hole v is equivalent to a near-perfect matching of G' with hole v_1.

To see this, observe that if v_1 is a hole in a near-perfect matching M' of G', then v_2 must be matched to v', so that all vertices of $N_2(v)$ are matched outside of $\{v_1, v', v_2\}$. As a result $M' \setminus \{(v', v_2)\}$ is a valid near-perfect matching of G with hole v. Similarly, for any near-perfect matching M of G with hole v, $M \cup \{(v', v_2)\}$ is a valid near-perfect matching of G' with hole v'. □

Removing pending path We describe here the inverse of vertex splitting. This tool will be used in the next section to reduce the problem of counting perfect matchings on arbitrary graphs with parallel and loop edges to graphs with minimum degree 3.

Construction 3.2. *Given a vertex $v \in V(G)$ of degree 2 with neighbors v_1 and v_2, let G' be obtained from G by removing edges (v_1, v) and (v, v_2), and contracting vertices v_1, v and v_2 into a single vertex v'.*

Note that this construction may generate loop edges on v', and parallel edges to all x such that $\{(x, v_1), (x, v_2)\} \in E(G)$. Thus by the same arguments as those of Lemma 3.1, there is a bijection between perfect matchings of G and those of G', if we allow them to contain parallel and loop edges.

3.1.3 Duplicating pending paths

The following construction will be used in the next chapter to allow a path to traverse the same edge twice without overlapping itself.

Construction 3.3 (Duplicating pending path)**.** *Let G be an arbitrary graph, and let P be a pending path of G defined by the sequence of vertices $v_1, v_2, \ldots, v_k$. If P is even, then let G' be the graph obtained from G by adding vertices v'_1 and v'_2, and edges (v_1, v'_1), (v'_1, v'_2) and (v'_2, v_k). If P is odd, then call u and v its extremities and remove all its inner-points. Then insert the graph shown in Figure 8, such that vertices x and y of the figure correspond to u and v of G.*

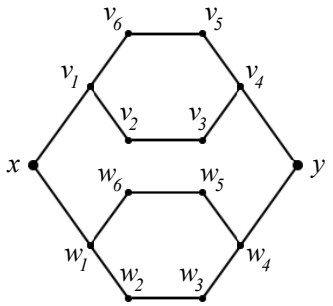


Figure 8: Subgraph with which to replace an odd path, for odd pending path duplication.

We call this construction *duplication* since it results in a graph that now has two edge-disjoint paths between u and v, instead of the original unique pending path.

Lemma 3.3. *Given graph G and pending path P of G, let G' be the graph obtained by Construction 3.3. If P is even, then G and G' have the same number of perfect matchings. If P is odd, then G' has exactly 4 times more perfect matchings than G.*

Proof. First assume that P is even, and refer to the added odd path v_1, v'_1, v'_2, v_k as Q. Let M be any perfect matching on G. Clearly, $M \cup \{(v'_1, v'_2)\}$ is a perfect matching of G'. Consider now any perfect matching M' of G'. In order for M to match all vertices of P in G', exactly one of v_1 or v_k must be matched to its neighbor in P. Assume without loss of generality that this vertex is v_k. It must then be that (v'_1, v'_2) is selected, and v_1 is not matched in Q. Therefore, $M' \setminus \{(v'_1, v'_2)\}$ is a perfect matching of G. There is thus a bijection between perfect matchings of G and those of G'.

Assume now that P is odd, and refer to the subgraph in Figure 8 as H. It suffices to show that the number of perfect matchings of H is the same as the number of near-perfect matchings with

holes x and y of H, which we will see is 4. To see why this suffices, consider any matching M of G. If M matches both endpoints of P inside of P, then we can obtain a perfect matching of G' by removing all edges of P from M and inserting a perfect matching of H in M. As there are 4 choices of perfect matchings for H, we can obtain 4 distinct perfect matchings of G' from M. Similarly, if M does not match any endpoint of P inside P, then replace edges of P in M with a near-perfect matching of H with holes x and y. Because the endpoints of P are matched outside of P, this results in a valid perfect matching of G'. Again, there are 4 distinct such matchings. Each matching of G thus corresponds to exactly 4 matchings of G'. The converse follows similarly: for any perfect matching M' of G', if it induces a perfect matching on H, then replace H with a perfect matching of P to obtain the corresponding perfect matching of G; if M' induces a near-perfect matching on H with holes x and y, then replace H with a near-perfect matching of P with holes its endpoints to obtain the corresponding perfect matching of G. Because any matching M' on G' induces on H either a perfect matching or a near-perfect matching with holes x and y – as otherwise it would not be perfect on G' – we have that the number of perfect matchings of G' is exactly 4 times that of G.

It remains only to show that H has exactly 4 perfect and 4 near-perfect matchings with holes x and y. First consider the number of near-perfect matchings on H with holes in x and y. Any such matching will induce a perfect matching on both the top and bottom 6-cycles of H. Because there are two possible perfect matchings on any even cycle, we have a total of 4 near-perfect matchings of H with holes x and y. Now for the perfect matchings on H, we first consider all those that match x to v_1. They must select (v_2, v_3) and (v_5, v_6), and thus also (v_4, y). Again, there is then two ways to match the bottom 6-cycle. By symmetry, we obtain the same number when x matches w_1, so that we have a total of 4 perfect matchings of H. This completes the proof.

□

This lemma naturally extends to near-perfect matchings with given hole.

Lemma 3.4. *Given graph G, vertex $z \in V(G)$ and pending path P of G, let G' be the graph obtained by Construction 3.3. If P is even, then there exists a vertex $z' \in G'$ such that G and G' have the same number of near-perfect with hole z and z', respectively. If P is odd, then G' has exactly 4 times more near-prefect matchings with hole z' than G with hole z.*

Proof. Consider graph G_0 obtained by adding to G a vertex x and the edge (x, z). Clearly, any perfect matching of G_0 must select (x, z), and thus contain a near-perfect matching of G with hole z. Similarly, any near-perfect matching of G with hole z corresponds to the perfect matching of G_0 that also selects (x, z). Therefore the number of perfect matchings of G_0 is equal to the number of near-perfect matchings of G with hole z. By Lemma 3.3, duplicating pending path P on G_0 results in a graph G_1 with 1 or 4 times more perfect matchings than G_0, depending on the parity of P. By the same argument as before, the number of perfect matchings of G_1 is the same as the number of near-perfect matchings of $G_1 - x$ with hole z. Consequently, the number of near-perfect matchings of $G_1 - x$ with hole z is exactly 1 or 4 times more than that of G with hole z, depending on the parity of P.

Finally, it is easy to see that $G_1 - x$ is the same graph as the one obtained by duplicating P directly in G, which completes the proof. □

3.2 Reduction to graphs of constant minimum or maximum degree

We describe here an example application of the tools described above, that shows how to modify a given graph into one whose maximum degree is bounded by 3, without changing the number of perfect matchings it contains.

Reduction to maximum degree 3 Recall Construction 3.1, which splits a vertex by inserting an odd pending path in its place. We describe here how this construction can be used to reduce the degree of a graph down to 3.

Construction 3.4. *Given a graph G, while it contains a vertex v of degree 4 or more, partition the neighborhood of v into two sets of size at least 2, and split v along this partition according to Construction 3.1.*

After splitting each vertex, either the number of vertices of maximum degree decreases by one, or the graph's maximum degree decreases by at least 1. As a consequence, this procedure terminates. Moreover, since it terminates, it must be that no vertex with degree 4 or more is left, so that the resulting graph must have maximum degree 3 or less. Seeing that any vertex v of G cannot contribute to more than $|\delta(v)|$ splits, the time complexity of this procedure is no more than $O(\sum_{v \in V(G)} |\delta(v)|) = O(|E(G)|)$.

Additionally, by an induction on the number of iterations of the procedure's loop and Lemma 3.1, there is a bijection between perfect matchings of G and perfect matchings of the resulting graph G'. Therefore, we have the following result.

Theorem 3.1. *If $\sharp PM_{\Delta \leq k}$ is the problem of counting perfect matchings on arbitrary graphs of size n and maximum degree at most k, then there exists a polynomial-time many-to-one reduction from $\sharp PM_{\Delta \leq n}$ to $\sharp PM_{\Delta \leq 3}$.*

Moreover, this theorem extends to the case of near-perfect matchings with given hole.

Construction 3.5. *Let G be a graph of odd degree n and v a given vertex of $V(G)$. Add to G a vertex x and the edge (x, v), and call G_0 the resulting graph. Obtain G' by applying Construction 3.4 to G_0. Let v' be the neighbor of x in G'. Remove x from G', and return the resulting graph, along with v'.*

Lemma 3.5. *Given graph G and vertex $v \in V(G)$, let G' and v' be the graph and vertex returned by Construction 3.5. Then G' has maximum degree 3 and there exists a bijection between all near-perfect matchings of G with hole v and all near-perfect matchings of G' with hole v'.*

Proof. First observe the graph G_0 obtained by adding vertex x and edge (x, v) to G. Since x has degree 1, any perfect matching M of G_0 matches (v, x), so that $M \setminus (v, x)$ constitutes a near-perfect matching of G. Similarly, any near-perfect matching of G with hole v can be mapped to that perfect matching of G_0 by simply selecting (v, x). Therefore, there is a bijection between near-perfect matchings of G with hole v, and perfect matchings of G_0.

Now by Construction 3.4 and Theorem 3.1, graph G' has maximum degree 3 and there exists a bijection between perfect matchings of G_0 and G'. Because Construction 3.4 modifies only those vertices of G_0 of degree greater than 3, vertex x added before is unchanged. Because v' is the neighbor of x in G', the same argument as between G and G_0 applies, and there is a bijection between perfect matchings of G' and near-perfect matchings of $G' \setminus \{x\}$ with hole v'.

Therefore, given graph G and vertex v, Construction 3.5 returns a graph G' and a vertex v' such that G' has maximum degree 3, and with a bijection between near-perfect matchings of G with hole v and near-perfect matchings of G' with hole v'. □

We restate this result as the following theorem, for completeness.

Theorem 3.2. *If $\sharp NPM_{\Delta \leq k}$ is the problem of counting near-perfect matchings on arbitrary graphs of odd size n, maximum degree at most k and given hole, then there exists a polynomial-time many-to-one reduction from $\sharp NPM_{\Delta \leq n}$ to $\sharp NPM_{\Delta \leq 3}$.*

Reduction to minimum degree 3 Although we are interested in graphs with no parallel or loop edges in this thesis, we include the following result here as well, as it constitutes an example application of another tool, namely the inverse of vertex splitting, Construction 3.2 that removes a pending path. We describe how it can be used to effectively delete all vertices of degree 2 (vertices of degree 1 being handled separately), and obtain a graph with minimum degree 3.

Theorem 3.3. *If $\sharp PM'_{\delta\geq k}$ is the problem of counting perfect matchings on arbitrary graphs with parallel and self edges, of size n and minimum degree at least k, and $\sharp PM_{\delta\geq k}$ is the same problem on graphs with no parallel or self edge, then there exists a polynomial-time many-to-one reduction from $\sharp PM_{\delta\geq 1}$ to $\sharp PM'_{\delta\geq 3}$.*

Proof. Since any problem in $\sharp PM_{\delta\geq k}$ is also a problem in $\sharp PM'_{\delta\geq k}$, we have an immediate reduction from the first to the second. It thus suffices to show a reduction from $\sharp PM'_{\delta\geq 1}$ to $\sharp PM'_{\delta\geq 3}$.

Consider any graph G with minimum degree 1, and let v be a vertex with exactly one neighbor x. Remove v and x from G, along with all the incident edges of x. Because any matching on G must select (x, v), it should be clear that the number of perfect matchings remains unchanged by this manipulation. Further, any matching on the obtained graph is equivalent to the matching in G obtained by selecting (x, v). We repeat this iteratively until the resulting graph G' has minimum degree either 0 or at least 2.

If G' has minimum degree 0, then we are in one of two cases. If G' is empty, then we know that G has exactly one matching, consisting of all edges formed by the pairs of vertices removed at each step of the iteration. On the other hand, if G' is not empty, then G' contains singletons that can thus not be matched in G' so that G' has zero matchings. Because each step of the iteration maintains the number of perfect matchings, G has zero perfect matchings as well.

In the case where G' has minimum degree at least 2, we can apply Construction 3.2 repeatedly to remove all pending paths from G'. Because contracting vertices cannot decrease the degree of any vertex, and because each contraction removes one vertex of degree 2, G'' must be reached and have minimum degree 3.

Finally, observing that the full construction of G'' from G takes $O(|E(G)|)$ steps completes the proof. □

3.3 Reduction to tripartite graphs

We present here another application of the vertex splitting tool to reduce the problem of counting perfect matchings in arbitrary graphs to that of counting perfect matchings on tripartite graphs, i.e. graphs whose vertex set can be partitionned into 3 independent sets, which we will denote $V_1(G)$, $V_2(G)$ and $V_3(G)$ for tripartite graph G.

Construction 3.6. *Let G be an arbitrary graph with n vertices. Order all vertices of G arbitrarily into a sequence $v_1, v_2, \ldots, v_n$. Partition the neighborhood of each vertex v_i into classes $N_1(v_i) = \{v_j : j > i\}$ and $N_2(v_i) = \{v_j : j < i\}$. Split each v_i along the partition of its neighborhood according to Construction 3.1. Call G' the resulting graph.*

Figure 9 shows an example graph G and the result G' of applying this construction to G.

Lemma 3.6. *Given graph G, the graph G' obtained from Construction 3.6 is tripartite.*

Proof. For every vertex v_i in $v_1, v_2, \ldots, v_n$, we denote $v_{i,1}$, $v_{i,2}$ and $v_{i,3}$ the three vertices obtained after splitting v_i along partition $N_1(v_i), N_2(v_i)$, such that $\delta(v_{i,1}) = N_1(v_i) \cup \{v_{i,2}\}$, $\delta(v_{i,2}) = \{v_{i,1}, v_{i,3}\}$, and $\delta(v_{i,3}) = N_2(v_i) \cup \{v_{i,2}\}$. We claim that

$$V_1 = \{v_{i,1} : 1 \leq i \leq n\}, \quad V_2 = \{v_{i,2} : 1 \leq i \leq n\}, \quad V_3 = \{v_{i,3} : 1 \leq i \leq n\}$$

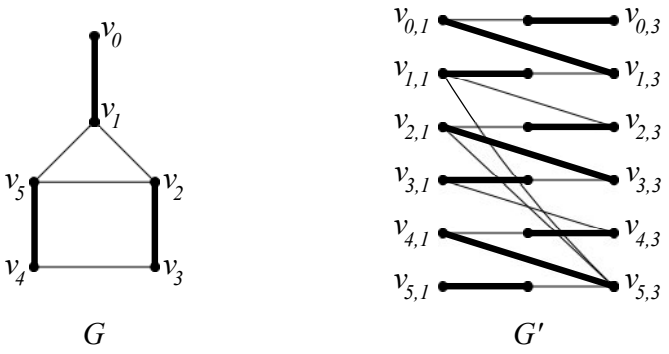


Figure 9: Example graph G and its corresponding tripartite graph G'. A perfect matching on G and its corresponding perfect matching on G' are shown in bold edges. Vertices $v_{0,2}, v_{1,2}, v_{2,2}, v_{3,2}, v_{4,2}, v_{5,2}$ are not labelled on G' for legibility.

is a valid partition of $V(G')$ such that V_i, $i \in \{1, 2, 3\}$ is an independent set of G', i.e. no pair of vertices of V_i spans an edge in G'. We can immediately see that V_1, V_2 and V_3 partition the set $V(G')$, so it remains to prove that they are independent sets.

First observe that for any $i \in \{1, \ldots, n\}$, the neighborhood of $v_{i,2}$ is exactly $\{v_{i,1}, v_{i,3}\}$, with does not intersect with V_2 so that V_2 is an independent set of G'.

Consider now an arbitrary vertex $v_{i,1} \in V_1$, and assume towards contradiction that there exists another vertex $v_{j,1} \in V_1$ such that $v_{j,1} \in \delta(v_{i,1}) = N_1(v_i)$. By construction, it must be that $j > i$. However, by symmetry, it must also be that $i < j$. This is a contradiction, which proves that no such $v_{j,1}$ exists. A similar argument shows that V_3 is also an independent set of G', proving that G' is a tripartite graph with classes V_1, V_2 and V_3. □

Because this construction splits each vertex once, it takes $O(n)$ steps. Furthermore by Lemma 3.1, splitting vertices does not impact the number of perfect matchings. Therefore, we have the following result.

Theorem 3.4. *If $\sharp PM$ is the problem of counting perfect matchings on arbitrary graphs, and $\sharp PM_{trip}$ is the same problem on tripartite graphs, then there exists a polynomial-time many-to-one reduction from $\sharp PM$ to $\sharp PM_{trip}$.*

Remark 3.1.

- It is interesting to note here that the counter-example graph provided by Štefankovič et al. [14] is tripartite, as can be seen by the coloring shown in Figure 10, indicating that their argument holds on tripartite graphs as well, so that the hole-pattern approach by Jerrum et al. [6] does not apply to tripartite graphs.
- Moreover, we should note that the solution by Jerrum et al. [6] heavily relies on the bipartiteness property, and specifically on the non-existence of odd cycles on such graphs. This further indicates that their solution cannot be applied on tripartite graphs without substantial modifications.

Finally, we note that we can further modify the tripartite graph returned by Construction 3.6 to obtain a tripartite graph of maximum degree 3, using Construction 3.4. Recall that it splits vertices of degree 4 or more repeatedly until no such vertex remains. As vertices of V_2 have degree 2, only vertices in V_1 and V_3 are affected. Consider a vertex $v \in V_1$, and the three vertices v_1, v', v_2 obtained

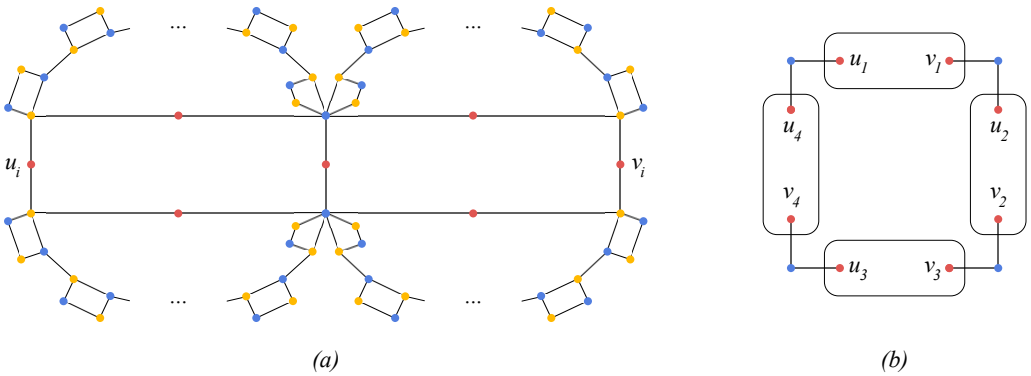


Figure 10: 3-coloring of the counter-example graph by Štefankovič et al. [14]. (a) shows the torpid mixing gadget H_i, which is copied 4 times into the counter-example graph in (b).

by splitting it. Recall that the neighborhoods of v_1 and v_2 are subsets of that of v, and v' is adjacent to exactly v_1 and v_2. Therefore, v_1 and v_2 are independent of each other, and of all vertices of V_1. We can thus place v_1 and v_2 in V_1, and v' in V_2 or V_3 arbitrarily. The same argument holds when v is in V_3. Therefore, the resulting graph is tripartite and of maximum degree 3 by Theorem 3.1. This result is formally stated in the following theorem.

Theorem 3.5. *If $\sharp PM_{\Delta \leq k}$ is the problem of counting perfect matchings on arbitrary graphs of size n and maximum degree at most k, and $\sharp PM_{\Delta \leq k,trip}$ is the same problem on tripartite graphs of size n and maximum degree at most k, then there is a polynomial-time many-to-one reduction from $\sharp PM_{\Delta \leq n}$ to $\sharp PM_{\Delta \leq 3,trip}$.*

4 A polynomial-time algorithm for almost-nested-ear decomposable graphs

In this chapter, we make use of the tools described in the previous chapter to study a specific type of graph, called factor-critical, whose interest in the context of counting perfect matchings can be seen in the results by Štefankovič, Vigoda, and Wilmes [14]. They introduced an algorithm that takes advantage of the Gallai-Edmonds decomposition of arbitrary graphs to reduce the problem on general graphs to the problem on factor-critical graphs – all new terms will be defined in the introductory section 4.1. To this end, they used a weighted version of the bipartite solution by Jerrum et al. [6]. In this chapter, we study those factor-critical graphs. We first introduce necessary definitions and briefly present this reduction. We then describe a polynomial time algorithm that finds the exact number of perfect matchings in a specific type of factor-critical graphs which we call almost-nested-ear decomposable, in sections 4.2 and 4.3. This gives us, using the algorithm by Štefankovič et al. [14], an FPRAS for specific types of graphs – namely those whose factor-critical components in the Gallai-Edmonds decomposition are almost-nested-ear decomposable. Finally, we make a step towards general factor-critical graphs by describing a reduction from the problem on arbitrary factor-critical graphs to a more constrained version called flat-tree-ear decomposable.

4.1 Introduction

We first introduce some necessary definitions and theorems that will be useful in this chapter.

Definition 4.1 (Factor-critical graph)**.** *A graph G is* factor-critical *when for any $v \in V(G)$, $G - v$ has a perfect matching.*

Definition 4.2 (Ear decomposition)**.** *Given a graph G, a pending path P of G and the subgraph H of G obtained by removing edges and inner-vertices of P from G, P is called an* ear *of G with respect to H. An ear is odd (resp. even) when its corresponding pending path is odd (resp. even).*

An ear decomposition *of a graph G is an ordered sequence of paths $(P_1, \dots P_k)$ whose union forms G, and such that P_1 is a cycle, and for all $i \in \{2, \dots, k\}$, P_i is an ear of $\cup_{j \leq i} P_j$ with respect to $C_0 \cup (\cup_{j<i} P_j)$. The endpoints of P_1 are the same vertex z which we call the* root *of the decomposition. We also call P_1 its* root ear*. An ear decomposition is odd (resp. even) when all its ears are odd (resp. even). If a graph G has an ear decomposition, it is called* ear decomposable*.*

Figure 11 (a) shows an example of an ear decomposable graph.

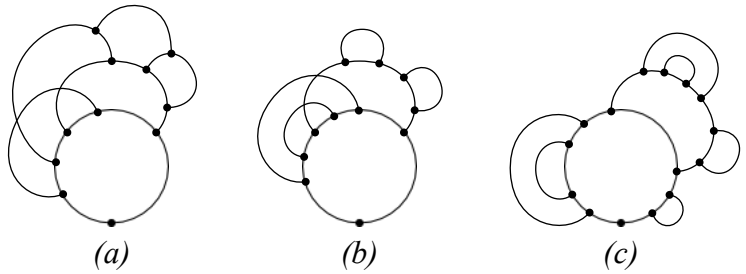


Figure 11: Examples of ear decomposable graphs. All three are ear decomposable, (b) and (c) and tree-ear decomposable, and (c) is nested-ear decomposable.

Definition 4.3 (Tree ear decomposition)**.** *Let G be a graph and $\mathcal{D} = (P_1, \ldots, P_k)$ be an ear decomposition of G with root vertex z. $\mathcal{D}$ is a* tree ear decomposition *of G if for every ear P_i with $i > 1$, both endpoints of P_i belong to at least one same ear P_j with $j < i$, which we call P_i's* parent*. P_i is then a* child *of P_j. An ear with no children is called a* leaf *ear. If a graph G has a tree-ear decomposition, it is called* tree-ear decomposable*.*

If h is the maximum number of ears that must be traversed to reach a leaf ear from z, we say h is the depth *of $\mathcal{D}$. A tree ear decomposition with depth at most 2 is called* flat*.*

Note that if G has maximum degree greater than 3, it may be that an ear has both its endpoints belong to more than one other ear. For instance, if P_i and P_j both have endpoints u and v on parent P_r, then the parent of P_i can be chosen to be either P_j or P_r. We define the tree ear decomposition such that P_i has at most one parent (none if and only if it is the root ear), chosen arbitrarily if there exist multiple candidates. Figure 11 (b) shows an example of a tree-ear decomposable graph.

Definition 4.4 (Nested ear decomposition)**.** *An ear decomposition $\mathcal{D} = (P_1, \ldots, P_k)$ is called nested when $\mathcal{D}$ is a tree ear decomposition, and for every ear P_i with $i > 1$ and its parent P_j, if p is the path on P_j joining the two endpoints of P_i, then any child of P_j has either both or none of its endpoints in p. We call this path p the path* induced *by P_i on its parent, and denote it p_{P_i}. If a graph G has a nested-ear decomposition, it is called* nested-ear decomposable*.*

Figure 11 (c) shows an example of a nested-ear decomposable graph.

Definition 4.5 (Inner- and outer-ears)**.** *Let $\mathcal{D} = (P_1, \ldots, P_k)$ be a nested-ear decomposition, and let P_i and P_j be two ears of $\mathcal{D}$ that share their parent. If P_j has both endpoints in p_{P_i}, we say that they are* nested*, that P_j is an* inner-ear *of P_i, and P_i is an* outer-ear *of P_j. If P_i and P_j share both endpoints, we arbitrarily define one as the inner-ear of the other. If there exists no third ear that is an inner-ear of P_i and an outer-ear of P_j, then we call P_i and P_j* closest *inner- and outer-ears of each other. Given a collection of children of an ear P_i, we call* outer-most *and* inner-most *the ears of the collection that have no outer-ear or inner-ear in that collection, respectively.*

Definition 4.6 (Descendants)**.** *Given an ear decomposition $\mathcal{D} = (P_1, \ldots, P_k)$ and an ear $P_i \in \mathcal{D}$, we define recursively the descendants of an ear as the union of its children and, if any, their descendants.*

We now give a characterization of factor-critical graphs that is due to Lovász [10][4].

Theorem 4.1. *A graph G is factor-critical if and only if it has an odd ear decomposition $(P_1, \ldots P_k)$. Moreover, the number of ears k is an invariant of G.*

We also note that Korte et al. [9] provide a linear time algorithm to obtain an ear decomposition for any factor-critical graph.

We now introduce the Gallai-Edmonds decomposition. The following definition is standard and is taken from [11].

Definition 4.7. *Let G be an arbitrary graph. The following three subsets of $V(G)$ form the* Gallai-Edmonds decomposition *of G.*

- *$D(G)$ contains all vertices which are not covered by at least one maximum matching of G.*
- *$A(G)$ contains all vertices of $V(G) - D(G)$ that are adjacent to a vertex of $D(G)$.*
- *$C(G)$ contains all other vertices of $V(G)$.*

[4] A proof of this result can also be found at [9].

We note that this decomposition can be constructed in polynomial time by the Blossom Algorithm of Edmonds [3]. The following theorem provides some fundamental properties of this decomposition.

Theorem 4.2 (Gallai-Edmonds Structure Theorem (abridged) [11])**.** *Let G be an arbitrary graph, and $D(G)$, $A(G)$ and $C(G)$ be its Gallai-Edmonds decomposition. Then*

1. *The components of the subgraph induced by $D(G)$ are all factor-critical,*
2. *The subgraph induced by $C(G)$ has a perfect matching,*
3. *Given any maximum matching M of G, M forms a near-perfect matching on each component of $D(G)$, a perfect matching on each component of $C(G)$, and matches every vertex in $A(G)$ to a vertex in a distinct component of $D(G)$.*

Using this decomposition, Štefankovič, Vigoda, and Wilmes [14] describe an FPRAS for counting perfect matchings on what they call "bipartite enough" graphs. For completeness and easy reference, we provide their algorithm in 1. It assumes the existence of an FPRAS `Permanent` that computes the permanent of a bipartite graph with weighted edges and given tolerated multiplicative error ε. Recall that the number of perfect matchings of a weighted bipartite graph is equal to the permanent of a matrix representation of the graph [6]. Thus `Permanent` can be thought of as the algorithm described by Jerrum et al. [6].

Algorithm 1: Approximate counter of number of perfect matchings of a graph within tolerated error ε

1 **Procedure** `Recursive-Count`(G, ε)**:**
2 **if** $V(G) = \emptyset$ **then**
3 **return** 1.
4 Arbitrarily select $u \in V(G)$, and let $G' = G - u$.
5 Let $D(G')$, $A(G')$ and $C(G')$ form the Gallai-Edmonds decomposition of G'.
6 **for** H **in** $D(G')$ **do**
7 **for** $v \in H$ **such that** $\delta(v) \cap A(G') \neq \emptyset$ **do**
8 Let $m_v =$ `Recursive-Count`$(H - v, \frac{\varepsilon}{2n})$
9 Let B be the bipartite graph obtained from G by removing $C(G)$, contracting each connected component H of $D(G)$ into a vertex v_H, removing all edges spanned by $A(G) \cup \{u\}$, and removing all parallel and loop edges.
10 For each $x \in A(G) \cup \{u\}$ and connected component H of $D(G)$, assign to edge (x, v_H) weight

$$\sum_{v \in \delta(x) \cap H} m_v.$$

return `Recursive-Count`$(C(G'), \frac{\varepsilon}{3})$ $\cdot$ `Permanent`$(B, \frac{\varepsilon}{3})$.

We note that the time complexity of this algorithm as it is presented here is exponential, due to the recursive call on line 8. If this call is replaced by one to a brute-force algorithm for counting perfect matchings on factor-critical graphs of at most a constant number of ears, then the procedure runs in polynomial time on all "bipartite enough" graphs, i.e. those that guarantee that no component H in any call to the procedure will have more than a constant number of ears.

We observe also that, if G is such that there is only ever one vertex $v \in H$ for all $H \in D(G')$ such that $\delta(v) \cap A(G') \neq \emptyset$, then Algorithm 1 runs in polynomial time. However, there is no known way

to modify G such that this can be guaranteed – even with modifications presented in the previous chapter.

Regardless, we note that if there exists an FPRAS for counting perfect matchings on factor-critical graphs of arbitrary size, then this algorithm forms an FPRAS for arbitrary graphs. This chapter aims at making a step in this direction.

4.2 Base algorithm on nested-ear decomposable graphs

In this section, we describe a polynomial-time algorithm that computes exactly the number of near-perfect matchings on arbitrary nested-ear decomposable graphs with given hole. It can therefore be used to compute the number of perfect matchings of graph $H - v$, for H a factor-critical graph and $v \in V(H)$, as is needed in Algorithm 1. We note that nested-ear decomposable graphs are easily seen to be planar, implying that one could use the FKT algorithm [15, 7, 8] to solve the same problem. However, this algorithm runs in $O(n^3)$ as it requires the computation of the Pfaffian of a matrix, while ours runs in quadratic time, and linear time for graphs of maximum degree 3. Moreover, the algorithm we describe here is meant as a foundation to be extended, in the following section, to graphs with an ear decomposition that is not fully nested, and thus not necessarily planar. We start by providing the main result of this section.

Theorem 4.3. *Let G be a nested-ear decomposable graph on n vertices. There exists a $O(n^2)$ algorithm for computing the exact number of near-perfect matchings in G with given hole $v \in V(G)$.*

Before proving this result, we will need the following few lemmas. The first lemma will be a convenience that allows us to work on a nested-ear decomposable graph with maximum degree 3. Specifically, this removes any scenario where two distinct ears share an endpoint, simplifying the definition and analysis of the section's main algorithm.

Lemma 4.1. *Given any nested-ear decomposable graph G and vertex $z \in V(G)$, there exists a nested-ear decomposable graph G' and vertex $z' \in V(G')$ such that the number of near-perfect matchings of G with hole z is the same as that of G' with hole z'. Moreover, G' and z' can be found in $O(|E(G)|)$ time.*

Proof. Recall Construction 3.4 which transforms a graph into one of maximum degree 3, by repeatedly splitting vertices of degree 4 or more along a given partition. We also saw that this process requires at most $O(|E(G)|)$ steps. We will here describe a similar procedure where the order in which vertices are split and the partition of their neighborhood is no longer arbitrary. Let G be the given graph, z a vertex of $V(G)$, and $\mathcal{D} = (P_1, \ldots, P_k)$ an arbitrary nested-ear decomposition of G.

To this end, we will describe a smaller procedure that takes as argument a nested-ear decomposition $(Q_1, \ldots, Q_k)$ and one of its ears Q_i, and modifies the graph and the ear decomposition so that all vertices of Q_i have degree at most 3, and the ear decomposition remains nested. We will refer to this procedure as *separating* ear Q_i. It requires that all vertices of all descendants of Q_i have degree at most 3. It also requires that all vertices of all inner-ears of Q_i have degree at most 3. Finally, it guarantees to not increase the maximum degree of any child or inner-ear of Q_i, or any of their descendants.

This procedure will be used as follows. We let $\mathcal{S} = (P_1, \ldots, P_k)$ be the given ear decomposition of the given graph G. We first apply the procedure to all leaf ears with no inner-ear, and remove them from $\mathcal{S}$. We can do this because they verify the procedure's pre-conditions. Moreover, $\mathcal{S}$ remains a valid nested-ear decomposition of the subgraph it decomposes. Again, there must be at least one ear in $\mathcal{S}$ that has no child and no inner-ear in $\mathcal{S}$. We apply the procedure to all such ears and remove them from $\mathcal{S}$. This is repeated until $\mathcal{S}$ is empty. Because an ear is removed from $\mathcal{S}$ only

after the procedure has run on it, all vertices of any ear P_i not in $\mathcal{S}$ must have degree at most 3. Furthermore, because all descendants and inner-ears of P_i are also not in $\mathcal{S}$, all their vertices have degree at most 3. Therefore, when $\mathcal{S}$ is empty, $(P_1, \ldots, P_k)$ is a nested-ear decomposition of a graph G' with maximum degree 3.

It now remains to describe this procedure. Let P_i be an ear of $\mathcal{S}$ and assume for now that $i > 1$. Because P_i has no child and no inner-ear in $\mathcal{S}$, it must be that all its children and inner-ears of $\mathcal{D}$ (if any) have maximum degree 3. Consequently, if P_i has a vertex of degree 4 or more, it must be one of its endpoints. Let $(v_1, \ldots, v_l)$ be an ordered labeling of the vertices along P_i, in an arbitrary direction. Thus v_1 and v_l are its endpoints.

First, if $v_1 = v_l$, then P_i has length at least 3. Furthermore, the parent P_j of P_i must have length at least 1, so there exists a vertex w of P_j in the neighborhood of $v_1 = v_l$. We define $N_1(v_1) = \{v_2, w\}$, and $N_2(v_1) = \delta(v_1) \setminus N_1(v_1)$. We then split v_1 along N_1 and N_2, i.e. remove v_1 and add x, v_1' and y, such that $\delta(x) = N_1(v_1) \cup \{v_1'\}$, and $\delta(y) = N_2(v_1) \cup \{v_1'\}$. In P_i, we replace v_1 and v_l with x and y respectively, in P_j we replace v_1 with the path (x, v_1', y), and in all edges incident to a vertex of $N_2(v_1)$, we replace v_1 with y. This is depicted in Figure 13, where paths P_i and P_j are highlighted in yellow and blue, respectively, both before and after vertex splitting. Clearly, P_i remains a child ear of P_j, all other children of P_j remain so, and all ears remain nested. We now continue with the next step, with v_1 and v_l now defined as x and y, the new endpoints of P_i.

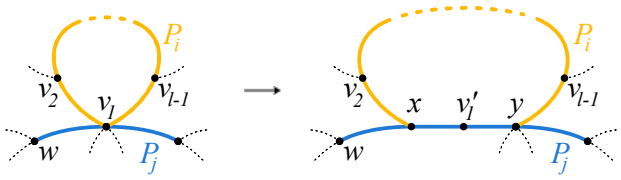


Figure 12: Separating closed ear P_i on parent P_j, in yellow and blue respectively.

If no endpoint of P_i has degree 4 or more, we are done. Otherwise, assume that v_1 has such degree. Because $v_1 \neq v_l$, there exists a vertex w in the path p_{P_i} induced by P_i on its parent P_j that is a neighbor of v_1 (note that w may be v_l). We thus define $N_1(v_1) = \{v_2, w\}$, and $N_2(v_1) = \delta(v_1) \setminus N_1(v_1)$, and split v_1 along $(N_1(v_1), N_2(v_1))$. We thus have removed v_1 and added x, v_1' and y such that $\delta(x) = N_1(v_1) \cup \{v_1'\}$ and $\delta(y) = N_2(v_1) \cup \{v_1'\}$. In P_i, we replace v_1 with x, in P_j we replace v_1 with the path (x, v_1', y), and in all edges incident to a vertex of $N_2(v_1)$, we replace v_1 with y. This is depicted in Figure 13, where paths P_i and P_j are highlighted in yellow and blue respectively, both before and after vertex splitting. Again, P_i remains a child of P_j and all children of P_j remain so. Moreover, no ear becomes non-nested after this transition, so the resulting ear decomposition remains nested. Finally, the new endpoint x of P_i has now degree 3. If v_l has degree 4, we apply the symmetric strategy on it. As a result, both endpoints of P_i have degree at most 3. Because all vertices of all its children also have degree at most 3, so do their endpoints, and so do all inner-points of P_i.

Finally, assume that P_i is the root ear P_1. By the procedure's pre-conditions all vertices of all children of P_1 have degree at most 3. Therefore vertices of P_1 have degree either 2 or 3, and we are done.

This describes the procedure of separating an ear that we apply to all ears of $\mathcal{S}$, as described earlier. As we mentioned, when all ears of $\mathcal{S}$ have been separated, the resulting ear decomposition is nested and corresponds to a graph of maximum degree 3. Moreover, by Lemma 3.2, given a vertex z, splitting any vertex v of a graph results in a new graph with a vertex v', such that the number of

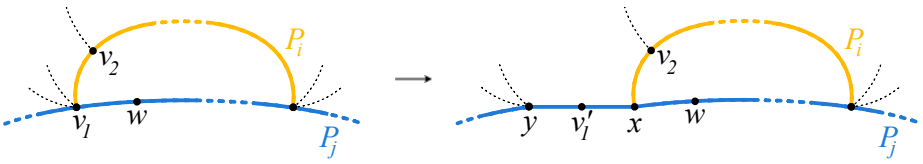


Figure 13: Separating open ear P_i on parent P_j, in yellow and blue respectively.

near-perfect matchings on the original graph with hole v is the same as that of the resulting graph with hole v'. Because the only transformations we do to G are vertex splitting, there exists a vertex z' of G' such that the number of near-perfect matchings of G' with hole z' is the same as that of G with hole z. Finally, noting that separating each ear P_i takes constant time, and that it is applied $O(k)$ times, where $k = O(|E(G)|)$ is the number of ears of G, completes the proof. □

Second, our algorithm will make use of a nested-ear decomposition of the given graph, so we need to prove that one can find such a decomposition in polynomial time. We do this with the following two lemmas.

Lemma 4.2. *Let G be a nested-ear decomposable graph with maximum degree 3. Then for any $v \in V(G)$, there exists a nested-ear decomposition of G with root v.*

Proof. We first describe how, given an arbitrary nested-ear decomposition $(Q_1, \ldots, Q_k)$ of G and an ear Q_i whose parent is Q_1, one can obtain a new nested-ear decomposition whose root ear contains Q_i. We can assume without loss of generality that $i = 2$ – indeed, the ordering of the decomposition requires only that Q_i appears after its parent, and before all its children. If (u_2, v_2) are the endpoints of path Q_2 and z is the root vertex of the decomposition, let q_L and q_R be the two edge-disjoint paths joining z to u_2 and z to v_2, respectively. Note that their intersection is exactly z, by the maximum degree 3 of G. Figure 14 shows ears Q_1 and Q_2, along with root vertex z and paths q_L and q_R.

We now construct a new nested-ear decomposition whose root ear is the path $Q_1' = q_L \cup Q_2 \cup q_R$, with root z. We define Q_2' as the path p_{Q_2} induced by Q_2 on Q_1. For all $j \in \{3, \ldots, k\}$, we define ear Q_j' to be equal to Q_j. One can now see that Q_2' is a child of Q_1'. Furthermore, because the original decomposition was nested, any ear Q_j' with $j > 2$ whose parent was Q_1 is now a child of either Q_1' or Q_2'. The sequence $(Q_1', \ldots, Q_k')$ is thus a valid nested-ear decomposition with $Q_2 \subseteq Q_1'$. We will refer to this procedure as "stretching" the root ear Q_1 around its child Q_2.

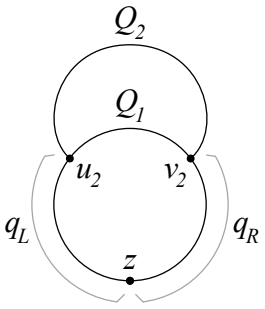


Figure 14: Ear Q_1, its child Q_2, and paths q_L and q_R. Not all vertices are shown in this figure.

We can now prove the lemma. Let $(P_1, \ldots, P_k)$ be a nested-ear decomposition of G with root $z \in V(G)$. If $z = v$, we are done. Otherwise, let P_r be the ear that v belongs to and whose both endpoints differ from v. Note that P_r is unique. By the nested structure of the decomposition, there exists a sequence of ears $P_{i_1}, \ldots P_{i_l}$ such that $P_{i_1} = P_1$ and $P_{i_l} = P_r$, and such that P_{i_j} is the parent of $P_{i_{j+1}}$ for all $j \in \{1, l-1\}$. We can see this as a path through the ears of the decomposition from the root to P_r.

Starting from the ear decomposition $(P_1, \ldots, P_k)$, we can stretch ear $P_1 = P_{i_1}$ around its child P_{i_2} to obtain a new nested ear decomposition whose root ear P'_1 contains P_{i_2}. Any child of P_{i_2} is therefore now a child of P'_1. For each $j \in \{1, \ldots, l\}$, we can thus repeatedly update P'_1 by stretching it around ear P_{i_j}, so that in the end, P'_1 contains ear $P_{i_l} = P_r$. Consequently, P'_1 also contains vertex $v \in V(P_r)$.

It now suffices to rotate P_1 so that its root is v – i.e. we replace $P'_1 = (z, v_1, \ldots, v, \ldots, z)$ with $P''_1 = (v, \ldots, z, v_1, \ldots, v)$. Note that this rotation may modify the induced subpath p_{P_i} of some ear P_i on P''_1. If it occurs, this path becomes the unique other path joining the endpoints of P_i on the cycle P''_1. As a consequence, if there exists an ear P_i that is not nested after the rotation (i.e. only one of its endpoints belongs to the induced subpath of another ear), then it was also not nested before the rotation. So rotating ear P'_1 into P''_1 is valid and we have a nested-ear decomposition with root v, which completes the proof. □

Theorem 4.4. *Let G be an arbitrary graph of maximum degree 3. G is nested-ear decomposable if and only if all ear decompositions of G are nested.*

Proof. By definition, if G has at least one nested ear decomposition, then it is nested-ear decomposable. On the other hand, assume G is nested-ear decomposable, and let $\mathcal{S} = \{P_1, \ldots, P_k\}$ and $\mathcal{T} = \{Q_1, \ldots, Q_k\}$ be two arbitrary ear decomposition of G. Let $z_\mathcal{S}$ and $z_\mathcal{T}$ be their respective root vertices. We will show that if $\mathcal{S}$ is nested then so is $\mathcal{T}$, by applying a series of transformations to $\mathcal{S}$ that result in $\mathcal{T}$, and such that none breaks the nested property of the decomposition.

Assume therefore that $\mathcal{S}$ is a nested ear decomposition. Regardless of the root vertex of $\mathcal{S}$, we know we can obtain a nested-ear decomposition with root $z_\mathcal{T}$. Let $\mathcal{S}' = \{P'_1, \ldots, P'_k\}$ be this decomposition.

We now make the following observation. Assume $P'_i = Q_i$ for all $i \leq r$, for some index $r < k$. That is, $\mathcal{S}'$ and $\mathcal{T}$ agree on their first r ears. Let P'_j with $j > r$ be a child of P'_r. Then there exists an ear of $\mathcal{T}$ whose endpoints are equal to those of P'_j. To see this, consider the subgraph H of G consisting of P'_j and all its descendent. Because $\mathcal{S}'$ is nested, only the two endpoints of P'_j connect outside of H, as otherwise there would be an ear with two distinct parents. Assume now we are given the graph $G' = \cup_{i \leq r} P_i$. Because $\mathcal{T}$ and $\mathcal{S}'$ agree up to P'_r, there must be a subset of the ears of $\mathcal{T}$ whose union forms H. If we wish to construct $G' \cup H$ from this set by adding them one by one to G', we must start with an ear Q whose both endpoints are in P'_r. Because H has exactly two vertices that connect outside of H and that those are the endpoints of P'_j, the endpoints of Q are the endpoints of P'_j. We also note that the same immediately holds for root ears P'_1 and Q_1, since the two decomposition share their root vertex.

Now iteratively, for each $i \in \{1, \ldots, k\}$ starting at 1 and in increasing order, we update $\mathcal{S}'$ so that P'_i becomes equal to the ear of $\mathcal{T}$ that shares its endpoints, and such that all P'_j with $j < i$ remain unchanged. We will show that in each step, the updated ear decomposition is valid and nested. As a result, we will have that $\mathcal{S}' = \mathcal{T}$ at the end of the iteration.

Whether at the base when $i = 1$, or in a step when $i > 1$, we follow the same process and use the same argument. In the case where $i > 1$, we assume that $P'_j = Q_j$ for all $j < i$. Let now P'_j with $j < i$ be the parent of P'_i, if any, and find an ear of $\mathcal{T}$ with the same endpoints as P'_i. If that ear is not Q_i, we insert it before Q_i and relabel all ears of $\mathcal{T}$ so that Q_i is the one whose endpoints

correspond to those of P'_i. We can do this because this only happens when $i > 1$, and because Q_i's parent is P'_j, with $j < i$.

While Q_i and P'_i differ, we do the following. Let p be a path of $P'_i \setminus Q_i$. It must be that the endpoints of p are endpoints of an ear of $\mathcal{S}'$. To see this, assume that they are not. Clearly, they must be the endpoints of ears of $\mathcal{S}'$, so we assume those are distinct ears P and P'. Now this implies that Q_i can jump from (a descendant of) P to (a descendant of) P', which cannot be as they are part of a nested-ear decomposition, and this would imply an ear exists with two distinct parents. Thus, let P_l be the ear of $\mathcal{S}'$ whose endpoints are those of p. We update P'_i to be equal to $(P'_i \setminus p) \cup P_l$, and update P_l to p. By the same argument as in the proof of the previous lemma, the updated $\mathcal{S}'$ is a valid nested-ear decomposition. Figure 15 shows an example of this process: a target path Q_i is highlighted in yellow on the left, and the sequence of updates to P'_i from its initial to its final value is shown on the right.

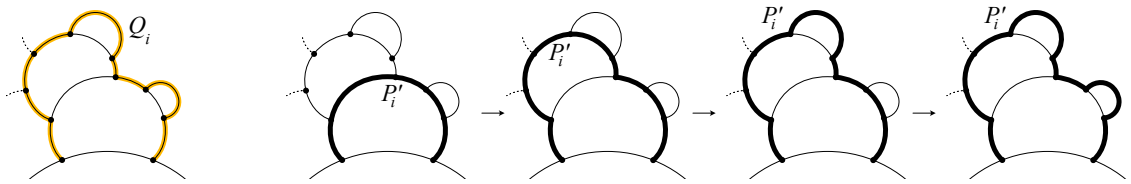


Figure 15: Sequence of updates to path P'_i until equal to ear Q_i highlighted in yellow on the left.

At each step of this loop, the size of the intersection between Q_i and P'_i increases by at least 2, so that it terminates after a linear number of steps. Moreover, at each step, $\mathcal{S}'$ remains a valid nested-ear decomposition. Finally, when this has run for all $i \in \{1, \ldots, k\}$, we have that $\mathcal{S}' = \mathcal{T}$, and that $\mathcal{S}'$ is a nested-ear decomposition. Therefore, $\mathcal{T}$ is a nested-ear decomposition. □

Note that this theorem implies that, given a graph G and any root vertex z, any algorithm for constructing ear decompositions produces a nested-ear decomposition, which is the reason why we needed this result. Moreover, a linear algorithm that computes an ear decomposition was presented by Schmidt [13], so given a nested-ear decomposable graph G, one can obtain a nested-ear decomposition of G in linear time. Consequently, one can also test whether G is nested-ear decomposable in linear time by simply generating an ear decomposition of G.

We are now ready to prove the main theorem of this section.

Proof of Theorem 4.3. We describe an algorithm which, for a given graph G and vertex $z \in V(G)$, computes in time $O(n^2)$ the number of near-perfect matchings with hole z in G.

The first step is to produce from G and z a graph G' and a vertex $z' \in V(G)$ such that G' has maximum degree 3, and the number of near-perfect matchings of G with hole z is the same as the number of near-perfect matchings of G' with hole $z' \in V(G')$. This is what Construction 3.5 provides by Theorem 3.2, in time $O(|E(G)|)$. We may thus assume from now on, for legibility, that G has maximum degree 3.

For this proof, we introduce the following notation. For each $i \in \{1, \ldots, k\}$, we let u_i and v_i be the endpoints of P_i. We label them such that when traversing any ear P_i from u_i to v_i, we encounter the endpoints of any child P_j of P_i in the order u_j, v_j. Moreover, we assume that the decomposition $(P_1, \ldots, P_k)$ is such that if P_i and P_j share their parent P_r and $i < j$, then u_i appears before u_j when traversing P_r from u_r to v_r. For every $i \in \{1, \ldots, k\}$, let H_i be the subgraph defined as the union of P_i and all its descendants, the path p_{P_i}, all inner-ears of P_i and all their descendants. Recall

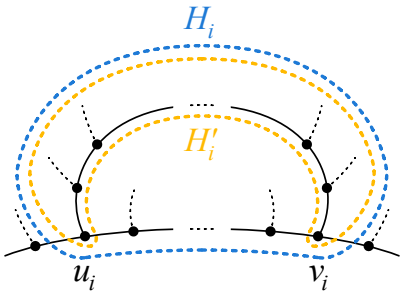


Figure 16: Subgraphs H_i and H'_i, highlighted in blue and yellow, respectively.

that p_{P_i} is the path induced by P_i on its parent. We also define H'_i as the union of P_i and all its descendants only. Figure 16 highlights subgraphs H_i and H'_i in blue and yellow, respectively. Note that u_i and v_i are the only vertices of H_i with neighbors outside of H_i, and similarly for H'_i.

For each $i \in \{1, \ldots, k\}$ the algorithm will construct the mapping $m_{i,\text{final}} : \mathcal{P}(\{u_i, v_i\}) \to \mathbb{N}$, where $\mathcal{P}(\{u_i, v_i\})$ is the power set of $\{u_i, v_i\}$, such that $m_{i,\text{final}}(W)$ equals the number of (near-)perfect matchings of H_i with holes exactly W. For example, $m_{i,\text{final}}(\emptyset)$ is the number of perfect matchings of H_i, and $m_{1,\text{final}}(\{z\})$ is the number of near-perfect matchings of $H_1 = G$ with hole z. To construct $m_{i,\text{final}}$ for all i, we will also construct a mapping $m_{i,\text{partial}} : \mathcal{P}(\{u_i, v_i\}) \to \mathbb{N}$ such that $m_{i,\text{partial}}(W)$ equals the number of (near-)perfect matchings of H'_i with holes exactly W.

These mappings will initially be undefined. For each ear P_i, $m_{i,\text{partial}}$ will be computed before $m_{i,\text{final}}$, starting from inner-most leaf ears and ending in the root ear. To this end, we will use the following procedures, which we describe as black boxes for now and will detail later.

- `merge_inner_ears` takes ear an P_i and the set of closest inner-ears of P_i. Assuming that $m_{i,\text{partial}}$ is known and $m_{j,\text{final}}$ is known for each closest inner-ear P_j of P_i, it computes $m_{i,\text{final}}$.
- `merge_children` takes an ear P_i and all its outer-most children. Assuming that $m_{j,\text{final}}$ is known for each outer-most child P_j of P_i, it computes $m_{i,\text{partial}}$.

With these two procedures, we can compute $m_{i,\text{final}}$ for all ears P_i of the decomposition by following algorithm 2, which takes a decomposition $\mathcal{D} = (P_1, \ldots, P_k)$ as input.

Algorithm 2: Compute $m_{i,\text{final}}$ of each ear P_i of the decomposition.

Procedure `compute_mappings`$(\mathcal{D} = (P_1, \ldots, P_k))$**:**
  Let $\mathcal{S}_{\text{init}} = \{P_1, \ldots, P_k\}$.
  Let $\mathcal{S}_{\text{partial}} = \emptyset$.
  while $\mathcal{S}_{\text{init}} \cup \mathcal{S}_{\text{partial}} \neq \emptyset$ **do**
    while $\exists P_i \in \mathcal{S}_{\text{init}}$ such that P_i is a leaf in $\mathcal{S}_{\text{init}} \cup \mathcal{S}_{\text{partial}}$ **do**
      Let $\mathcal{C}_{P_i}$ be all outer-most children of P_i in $\mathcal{D}$.
      Run `merge_children`$(P_i, \mathcal{C}_{P_i})$.
      Remove P_i from $\mathcal{S}_{\text{init}}$ and add it to $\mathcal{S}_{\text{partial}}$.
    while $\exists P_i \in \mathcal{S}_{\text{partial}}$ such that P_i is a leaf inner-most ear of $\mathcal{S}_{\text{init}} \cup \mathcal{S}_{\text{partial}}$ **do**
      Let $\mathcal{I}_{P_i}$ be all closest inner-ears of P_i in $\mathcal{D}$.
      Run `merge_inner_ears`$(P_i, \mathcal{I}_{P_i})$
      Remove P_i from $\mathcal{S}_{\text{partial}}$.

Claim 4.1. *At the end of any iteration of a loop of Algorithm 2, $\mathcal{S}_{init}$ contains all ears P_i whose neither $m_{i,partial}$ nor $m_{i,final}$ is known, and $\mathcal{S}_{partial}$ contains all P_i whose only $m_{i,partial}$ is known.*

Proof. We prove this claim by induction on the number of loop iterations completed. Clearly, before execution starts, no mapping is known, so that $\mathcal{S}_{\text{init}}$ and $\mathcal{S}_{\text{partial}}$ are as described in the claim. We now verify that they remain so after each iteration of a loop of the algorithm. It suffices to consider their states at the end of each inner loop (i.e. after lines 8 and 12), assuming they verify the claim's property when the loop starts.

Line 8 runs after the execution of `merge_children`. Because P_i is a leaf of $\mathcal{S}_{\text{init}} \cup \mathcal{S}_{\text{partial}}$, all its children are in $\mathcal{D} \setminus \mathcal{S}_{\text{init}} \setminus \mathcal{S}_{\text{partial}}$. Thus by the induction hypothesis, $m_{j,\text{final}}$ is known for each child P_j of P_i. Therefore, $m_{i,\text{partial}}$ becomes known on line 8, where it is moved to $\mathcal{S}_{\text{partial}}$, so that the two sets verify the claim's property.

Similarly, line 12 runs after the execution of `merge_inner_ears`. Because P_i is an inner-most leaf of $\mathcal{S}_{\text{init}} \cup \mathcal{S}_{\text{partial}}$, every closest inner-ear P_j of P_i is in $\mathcal{D} \setminus \mathcal{S}_{\text{init}} \setminus \mathcal{S}_{\text{partial}}$ and by the induction hypothesis, $m_{j,\text{final}}$ is known for them. Moreover, P_i is in $\mathcal{S}_{\text{partial}}$, so that $m_{i,\text{partial}}$ is known. So $m_{i,\text{final}}$ becomes known on line 12, where it is removed from $\mathcal{S}_{\text{partial}}$, so that the two sets still verify the claim's property. □

Because the size of $\mathcal{S}_{\text{partial}} \cup \mathcal{S}_{\text{init}}$ reduces by one at the end of each of the two inner loops, this procedure terminates. By the claim, the result of the algorithm is thus that $m_{i,\text{partial}}$ and $m_{i,\text{final}}$ is known for every ear $P_i \in \mathcal{D}$.

It now remains to describe the two procedures `merge_children` and `merge_inner_ears`. As they share some behavior, we will start by introducing a procedure `merge_neighbors` that both will make use of. For any two vertices x and y that are non-endpoints of one same ear P_i, we denote $p(x, y)$ the path on P_i that joins x to y. If $P_i = P_1$, then this path is one that does not contain z as an inner-vertex. In particular, for any two ears P_i and P_j with $i < j$ that share their parent, $p(u_i, v_j)$ is the path on their parent that contains all four endpoints of P_i and P_j.

Given two ears P_i and P_j with $i < j$ that share their parent and are not nested, `merge_neighbors` will compute a temporary mapping $m_{i,j} : \mathcal{P}(\{u_i, v_j\}) \to \mathbb{N}$ such that $m_{i,j}(W)$ equals the number of (near-)perfect matchings with holes exactly W on the graph obtained by the union of $p(u_i, v_j)$, all ears whose both endpoints belong to $p(u_i, v_j)$ and all their descendants. Figure 17 highlights this subgraph for two such ears P_i and P_j. `merge_neighbors` requires that its two input ears P_i and P_j share the same parent, are such that $i < j$ and are not nested, that no ear has only one endpoint in $p(u_i, v_j)$, and that $m_{r,\text{final}}$ is known for every ear P_r whose both endpoints belong to $p(u_i, v_j)$. Procedure `merge_neighbors` is given in Algorithm 3.

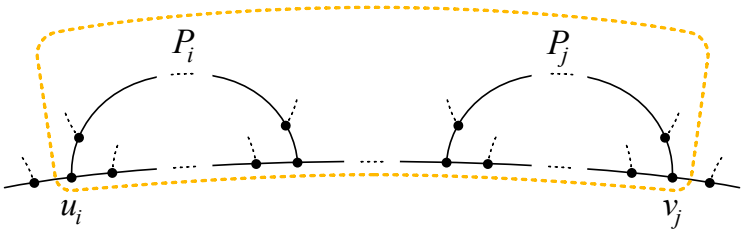


Figure 17: Subgraph considered by mapping $m_{i,j}$, for two ears P_i and P_j with $i < j$.

Algorithm 3: Compute $m_{i,j}$ given ears P_i, P_j.

1 **Procedure** `merge_neighbors(`P_i, P_j`)`**:**

2 Let $\mathcal{S}_{\text{init}} = \{P_1, \ldots, P_k\}$.

3 Let $\mathcal{S}_{\text{partial}} = \emptyset$.

4 **if** $i = j$ **then**

5 Let $m_{i,j} = m_{i,\text{final}}$

6 **else**

7 Let P_s be the ear on $p(u_i, u_j)$ that ends closest to P_j, i.e. the ear P_s that minimizes the length of $p(v_s, u_j)$.

8 Let $m_{i,s} =$ `merge_neighbors(`P_i, P_s`)`

9 Let $m_{i,j}(W) = \begin{cases} m_{i,s}(W - v_j + v_s)m_{j,\text{final}}(W - u_i + u_j) \\ \quad +m_{i,s}(W - v_j)m_{j,\text{final}}(W - u_i) & \text{if } p(v_s, u_j) \text{ is odd,} \\ m_{i,s}(W - v_j)m_{j,\text{final}}(W - u_i + u_j) \\ \quad +m_{i,s}(W - v_j + v_s)m_{j,\text{final}}(W - u_i) & \text{if } p(v_s, u_j) \text{ is even.} \end{cases}$

Claim 4.2. *Given two paths P_i and P_j with $i < j$ that share their parent and are not nested, if for every ear P_s whose both endpoints belong to $p(u_i, v_j)$, $m_{s,final}$ is known, then after running Algorithm 3 on P_i and P_j, $m_{i,j}$ is known and correct by its above definition.*

Proof. We prove this claim by induction on the number of ears whose both endpoints belong to $p(u_i, v_j)$. Clearly, if $i = j$, then $m_{i,j}$ and $m_{i,\text{final}}$ are equal by definition, so the claim holds in the base case.

Assume now it holds for any two ears P_i, P_s verifying the claim's assumptions, and let P_j be such that, when running `merge_neighbors` on P_i and P_j, P_s is the one found on line 7. That is, P_j is the very next ear encountered after P_s when traversing their parent ear from P_i. By the induction hypothesis, $m_{i,s}$ is known and valid after line 8.

It now remains to show that the computation on line 9 yields the correct value of $m_{i,j}$. This can be seen by exhaustion: when $p(v_s, u_j)$ is odd, consider any matching M on the subgraph considered by $m_{i,j}$ (i.e. the union of $p(u_i, v_j)$, all ears with both endpoints in $p(u_i, v_j)$ and their descendants, highlighted in figure 17). If M cannot have a hole outside of $\{u_i, v_j\}$, then M matches either both v_s and u_j inside $p(v_s, u_j)$, or both outside $p(v_s, u_j)$. The first term counts those that match them inside, while the second counts those that match them outside. On the other hand, if $p(v_s, u_j)$ is even, then exactly one of the two must be matched on $p(v_s, u_j)$, hence the two terms of the sum.

Finally, the recursive call is made between two ears P_i and P_s such that $p(u_i, v_s)$ is shorter than $p(u_i, v_j)$, and P_s must be either an ear that starts after P_i and ends before P_j, or P_i itself, so that the recursion terminates after $O(|p(u_i, v_j)|)$ steps. □

We can now present the two building blocks we used in Algorithm 2. Algorithm 4 describes `merge_children`, which takes as inputs an ear P_i and all its outer-most children. Recall that it requires that $m_{j,\text{final}}$ is known for each outer-most child P_j of P_i.

Algorithm 4: Compute $m_{i,\text{partial}}$ given ear P_i and its outer-most children $\mathcal{C}_{P_i}$.

1 **Procedure** `merge_children`(P_i, $\mathcal{C}_{P_i}$)**:**

2 **if** $\mathcal{C}_{P_i} = \emptyset$ **then**

3 Let $m_{i,\text{partial}}(W) = \begin{cases} 1 & \text{if } P_i \text{ is even and } |W| = 1, \text{ or } P_i \text{ is odd and } |W| \neq 1 \\ 0 & \text{otherwise} \end{cases}$

4 **else**

5 Let $(P_{i_1}, \ldots, P_{i_l})$ be an ordering of $\mathcal{C}_{P_i}$, so that $i_j \leq i_{j+1}$ for all $j \in \{1, \ldots, l-1\}$.

6 Run `merge_neighbors`(P_{i_1}, P_{i_l}).

7 **for** $W \in \mathcal{P}(\{u_i, v_i\})$ **do**

8 Let $W' \subseteq \{u_{i_1}, v_{i_l}\}$ be such that

$$\begin{aligned} u_{i_1} \in W' \quad &\text{iff} \quad p(u_i, u_{i_1}) \text{ is even and } u_i \in W, \text{ or } p(u_i, u_{i_1}) \text{ is odd and } u_i \notin W, \\ v_{i_l} \in W' \quad &\text{iff} \quad p(v_{i_l}, v_i) \text{ is even and } v_i \in W, \text{ or } p(v_{i_l}, v_i) \text{ is odd and } v_i \notin W. \end{aligned}$$

9 Let $m_{i,\text{partial}}(W) = m_{i_1,i_2}(W')$.

Claim 4.3. *Given ear P_i such that for all children P_j of P_i, $m_{j,final}$ is known, Algorithm 4 computes $m_{i,partial}$.*

Proof. When P_i has no children, then $m_{i,\text{partial}}$ is trivial. When P_i has outer-most children $P_{i_1}, \ldots, P_{i_l}$, $m_{i_s,\text{final}}$ is known by assumption for every child P_{i_s}. Therefore, m_{i_1,i_l} is known after we run `merge_neighbors` on P_{i_1} and P_{i_l}, by Claim 4.2.

Then the correctness of the value assigned on line 9 can be verified by exhaustion. Any matching on H_i' (i.e. the union of P_i and all its descendants) with no hole allowed outside of $\{u_i, v_i\}$ matches both u_i and u_{i_1} on $p(u_i, u_{i_1})$ if and only if $p(u_i, u_{i_1})$ is odd. Similarly, it matches exactly one of u_1 and u_{i_1} on $p(u_i, u_{i_1})$ if and only if $p(u_i, u_{i_1})$ is even. The same argument holds for v_i and v_{i_l} by symmetry. Line 8 follows from this observation. □

Algorithm 5 describes `merge_inner_ears`. Recall that it takes as input an ear P_i and all its closest inner-ears, and it requires that $m_{i,\text{partial}}$ and $m_{j,\text{final}}$ for every closest inner-ear P_j of P_i be known.

Claim 4.4. *Given ear P_i such that for all closest inner-ears P_j of P_i, $m_{j,final}$ is known, and $m_{i,partial}$ is known, Algorithm 5 computes $m_{i,final}$.*

Proof. The argument is very similar to that of the proof of Claim 4.3. The call to `merge_neighbors` is valid by assumptions on the input of `merge_inner_ears`. For the rest, all cases can be verified by exhaustion. The case where P_i has no inner-ear requires considering all combinations of holes in $\{u_i, v_i\}$ on H_i that are allowed by the parity of path $p(u_i, v_i)$. For the general case where $\mathcal{I}_{P_i} \neq \emptyset$, we must consider all combinations of holes in $\{u_i, v_i\}$ on H_i, and in $\{u_{i_1}, v_{i_l}\}$ on the union of $p(u_{i_1}, v_{i_l})$ and its descendants, that are allowed by the parity of paths $p(u_i, u_{i_1})$ and $p(v_{i_l}, v_i)$ □

We now have completed the full description of the main Algorithm 2 and shown that it computes the mapping $m_{i,\text{final}}$ for every ear P_i of the given decomposition. It should be clear that $m_{1,\text{final}}(\{z\})$ is the number of near-perfect matchings on G with hole z, where z is the root of the decomposition.

Turning to the complexity of the algorithm, we see that `merge_neighbors` is never called twice with the same second argument, so that there are at most $O(k)$ mappings of the form $m_{i,j}$ that are computed. Because one such mapping is computed for each call to `merge_neighbors`, it is not

Algorithm 5: Compute $m_{i,\text{final}}$ given ear P_i and its closest inner-ears $\mathcal{I}_{P_i}$.

Procedure `merge_inner_ears`$(P_i, \mathcal{I}_{P_i})$**:**

  if $\mathcal{I}_{P_i} = \emptyset$ **then**

    Let

$$m_{i,\text{final}}(W) = \begin{cases} m_{i,\text{partial}}(W) & \text{if } p(u_i, v_i) \text{ is odd and } |W| > 0 \\ m_{i,\text{partial}}(\emptyset) + m_{i,\text{partial}}(\{u_i, v_i\}) & \text{if } p(u_i, v_i) \text{ is odd and } |W| = 0 \\ m_{i,\text{partial}}(\{u_i, v_i\}) & \text{if } p(u_i, v_i) \text{ is even and } |W| = 1 \\ m_{i,\text{partial}}(\{u_i\}) + m_{i,\text{partial}}(\{v_i\}) & \text{if } p(u_i, v_i) \text{ is even and } |W| = 0 \\ 0 & \text{if } p(u_i, v_i) \text{ is even and } |W| = 2 \end{cases}$$

  else

    Let $\{P_{i_1}, \ldots, P_{i_l}\} = \mathcal{I}_{P_i}$, so that v_{i_j} always appears before $u_{i_{j+1}}$ when traversing P_i from u_i to v_i.

    Run `merge_neighbors`(P_{i_1}, P_{i_l}).

    if $p(u_1, u_{i_1})$ and $p(v_{i_l}, v_i)$ are both even **then**

$$\begin{aligned} m_{i,\text{final}}(\{u_i, v_i\}) = \;& m_{i_1,i_l}(\{u_{i_1}, v_{i_l}\}) m_{i,\text{partial}}(\{u_i, v_i\}) \\ m_{i,\text{final}}(\{u_i\}) = \;& m_{i_1,i_l}(\{u_{i_1}, v_{i_l}\}) m_{i,\text{partial}}(\{u_i\}) + m_{i_1,i_l}(\{u_{i_1}\}) m_{i,\text{partial}}(\{u_i, v_i\}) \\ m_{i,\text{final}}(\emptyset) = \;& m_{i_1,i_l}(\emptyset) m_{i,\text{partial}}(\{u_i, v_i\}) + m_{i_1,i_l}(\{u_{i_1}, v_{i_l}\}) m_{i,\text{partial}}(\emptyset) \\ & + m_{i_1,i_l}(\{u_{i_1}\}) m_{i,\text{partial}}(\{u_i\}) + m_{i_1,i_l}(\{v_{i_l}\}) m_{i,\text{partial}}(\{v_i\}) \end{aligned}$$

    else if $p(u_1, u_{i_1})$ and $p(v_{i_l}, v_i)$ are both odd **then**

$$\begin{aligned} m_{i,\text{final}}(\{u_i, v_i\}) = \;& m_{i_1,i_l}(\emptyset) m_{i,\text{partial}}(\{u_i, v_i\}) \\ m_{i,\text{final}}(\{u_i\}) = \;& m_{i_1,i_l}(\{v_{i_l}\}) m_{i,\text{partial}}(\{u_i\}) + m_{i_1,i_l}(\emptyset) m_{i,\text{partial}}(\{u_i, v_i\}) \\ m_{i,\text{final}}(\emptyset) = \;& m_{i_1,i_l}(\emptyset) m_{i,\text{partial}}(\emptyset) + m_{i_1,i_l}(\{u_{i_1}, v_{i_l}\}) m_{i,\text{partial}}(\{u_i, v_i\}) \\ & + m_{i_1,i_l}(\{u_{i_1}\}) m_{i,\text{partial}}(\{v_i\}) + m_{i_1,i_l}(\{v_{i_l}\}) m_{i,\text{partial}}(\{u_i\}) \end{aligned}$$

    else if $p(u_1, u_{i_1})$ is even and $p(v_{i_l}, v_i)$ is odd **then**

$$\begin{aligned} m_{i,\text{final}}(\{u_i, v_i\}) = \;& m_{i_1,i_l}(\{u_{i_1}\}) m_{i,\text{partial}}(\{u_i, v_i\}) \\ m_{i,\text{final}}(\{u_i\}) = \;& m_{i_1,i_l}(\{u_{i_1}, v_{i_l}\}) m_{i,\text{partial}}(\{u_i, v_i\}) + m_{i_1,i_l}(\{u_{i_1}\}) m_{i,\text{partial}}(\{u_i\}) \\ m_{i,\text{final}}(\emptyset) = \;& m_{i_1,i_l}(\{v_{i_l}\}) m_{i,\text{partial}}(\{u_i, v_i\}) + m_{i_1,i_l}(\{u_{i_1}\}) m_{i,\text{partial}}(\emptyset) \\ & + m_{i_1,i_l}(\{u_{i_1}, v_{i_l}\}) m_{i,\text{partial}}(\{v_i\}) + m_{i_1,i_l}(\emptyset) m_{i,\text{partial}}(\{u_i\}) \end{aligned}$$

    else if $p(u_1, u_{i_1})$ is odd and $p(v_{i_l}, v_i)$ is even **then**

      Define $m_{i,\text{final}}$ symmetrically to the case where $p(u_1, u_{i_1})$ is even and $p(v_{i_l}, v_i)$ is odd.

    Define $m_{i,\text{final}}(\{v_i\})$ symmetrically to $m_{i,\text{final}}(\{u_i\})$.

called more than k times. Similarly, no mapping of the form $m_{i,\text{partial}}$ or $m_{i,\text{final}}$ is computed twice, and because every call to `merge_children` and `merge_inner_ears` results in computing one new such mapping, they are not called more than k times. These observations limit the total number of iterations of any loop in Algorithm 2 to $O(k)$. Taking into account the initial modifications to the input graph to reduce its maximum degree to 3, which took $O(|E(G)|)$ time, and recalling that $k = O(|E(G)|)$, we have that our algorithm runs in time $O(|E(G)|)$ for arbitrary nested-ear decomposable graphs.

□

We recall that, in the algorithm proposed by Štefankovič et al. [14] and given in Algorithm 1, the recursive call on line 8 can be replaced by our algorithm to obtain an FPRAS on non-bipartite graphs whose all factor-critical subgraphs are nested-ear decomposable. Again, this result is only minor, as the existing FKT algorithm could be applied to such graphs and more generally those with

planar factor-critical subgraphs, only at a multiplicative cost n to the asymptotic time complexity. This is what we will improve on in the next section.

4.3 Extension to almost-nested-ear decomposable graphs

We now proceed to describe how to extend our main algorithm of the previous section to apply to specific ear decomposable graphs that are not nested and, in particular, not planar, thus exiting the realm of graphs handled by the FKT algorithm.

Definition 4.8 (Almost-nested ears)**.** *Let G be an ear decomposable graph, and let $\mathcal{D} = (P_1, \ldots, P_k)$ be an ear decomposition of G. We say that two ears P_i and P_j of $\mathcal{D}$ are* almost-nested *if they share their parent, and are such that the paths p_{P_i} and p_{P_j} they respectively induce on their parent are such that $p_{P_i} \oplus p_{P_j}$ consists of two disconnected paths of length at least 1 that contain the endpoint of no other ear of $\mathcal{D}$.*

Figure 18 shows an example of two almost-nested ears P_i and P_j.

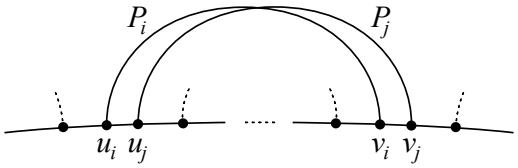


Figure 18: Example of two almost-nested ears P_i and P_j.

Definition 4.9 (Sequence of almost-nested ears)**.** *We will refer to any sequence of ears whose all pairs of consecutive ears are almost-nested as a* sequence of almost-nested ears*. We also allow such a sequence to have length 1. We call this sequence* maximal *if no ear outside the sequence is almost-nested with an ear of the sequence.*

Definition 4.10 (Almost-nested-ear decomposition)**.** *We say that $\mathcal{D}$ is an* almost-nested-ear decomposition *of a graph G when removing all but one ear from every maximal sequence of almost-nested ears of $\mathcal{D}$ results in a nested-ear decomposition. We then say G is* almost-nested-ear decomposable*.*

We will assume again from now on that whenever we refer to a nested-ear decomposition $(P_1, \ldots, P_k)$, for any two ears P_i and P_j that share their parent P_r, if $i < j$ then u_i appears before u_j when traversing P_r from u_r to v_r.

Lemma 4.3. *Let G be an almost-nested-ear decomposable graph, and $\mathcal{D}$ be an almost-nested-ear decomposition of G. Then for any two ears P_i and P_j from two distinct maximal sequences of almost-nested ears of $\mathcal{D}$, the two the paths p_{P_i} and p_{P_j} they induce on their parents are edge-disjoint.*

Proof. Assume towards contradiction that there exist two maximal sequences S_1 and S_2 that contain ears P_i and P_j respectively, such that $p_{P_i} \cap p_{P_j}$ is non-empty. We then remove all ears but P_i in S_1, all ears but P_j in S_2, and all but one ear from all other maximal sequences of almost-nested ears. In the resulting decomposition, P_i still has exactly one endpoint on an inner-point of p_{P_j}, so this decomposition is not nested. Therefore $\mathcal{D}$ does not verify the definition of an almost-nested-ear decomposition, a contradiction.

Note that if G has maximum degree 3, then the edge-disjoint term in the theorem can be replaced by vertex-disjoint. □

Theorem 4.5. *Let G be an almost-nested-ear decomposable graph on n vertices of maximum degree 3. There exists a $O(|E(G)|)$ algorithm for computing the exact number of near-perfect matchings in G with given hole $v \in V(G)$.*

Proof. Consider the algorithm given in the previous section for nested-ear decomposable graphs. We will generalize the different mappings we defined there so that instead of ears, they depend on sequences of almost-nested ears. The core logic will remain the same however. We thus first need to extend some important definitions on ears to sequences. We will say a sequence S_1 is an *inner-sequence* of another S_2 if for every ear P_i in S_1, the endpoints of P_i appear in the paths induced by all ears of S_2 on their parent. We then call S_2 an *outer-sequence* of S_1, and say S_1 and S_2 are nested. Inner-most and outer-most sequences are defined as those that have no inner- or outer-sequence in a given collection, respectively. A *child* sequence of an ear P_i is one whose all ears are children of P_i.

We now redefine the analogues of the mappings defined earlier, on sequences of nested-ears.

- For any sequence of almost-nested ears $S = (P_{i_1}, \ldots, P_{i_l})$, the mapping $m_{S,\text{final}} : \mathcal{P}(\{u_{i_1}, v_{i_l}\})$ is defined such that $m_{S,\text{final}}(W)$ equals the number of (near-)perfect matchings with holes exactly W on the subgraph H_{i_1,i_l} obtained by the union of $p(u_{i_1}, v_{i_l})$, all ears whose both endpoints belong to $p(u_{i_1}, v_{i_l})$ and all their descendants. Figure 19 highlights this subgraph in blue. Notice that all ears in S are contained in this subgraph. Also, we note that $m_{i,\text{final}} = m_{(P_i),\text{final}}$ for every ear P_i of the decomposition.
- For any two P_i, P_j that share a parent and whose paths induced on their parent are vertex-disjoint, the mapping $m_{i,j} : \mathcal{P}(\{u_i, v_j\}) \to \mathbb{N}$ remains as defined previously, i.e. such that $m_{i,j}(W)$ is the number of near-perfect matchings with holes exactly W on the union of $p(u_i, v_j)$, all ears whose both endpoints belong to $p(u_i, v_j)$ and all their descendants.
- For any sequence of almost-nested ears $S = (P_{i_1}, \ldots, P_{i_l})$, we define the mapping $m_{S,\text{partial}} : \mathcal{P}(\{u_{i_1}, u_{i_l}, v_{i_1}, v_{i_l}\}) \to \mathbb{N}$ such that $m_{S,\text{partial}}(W)$ is the number of (near-)perfect matchings with holes exactly W on the subgraph H'_{i_1,i_l} obtained by the union of $p(u_{i_1}, u_{i_l})$, $p(v_{i_1}, v_{i_l})$, all ears with both endpoints in $p(u_{i_1}, u_{i_l}) \cup p(v_{i_1}, v_{i_l})$ and all their descendants. Figure 19 highlights an example of such a subgraph in yellow. Note that all ears with both endpoints in on $p(u_{i_1}, u_{i_l}) \cup p(v_{i_1}, v_{i_l})$ are exactly those in S, by the definition of a sequence of almost-nested ears. Note also that $m_{i,\text{partial}} = m_{(P_i),\text{partial}}$ for all P_i.

Finally, note that for a sequence of almost-nested ears $S = (P_{i_1}, \ldots, P_{i_l})$, the subgraph H_{i_1,i_l} is connected to the rest of the graph through exactly u_{i_1} and v_{i_l}, while H'_{i_1,i_l} is connected to the rest

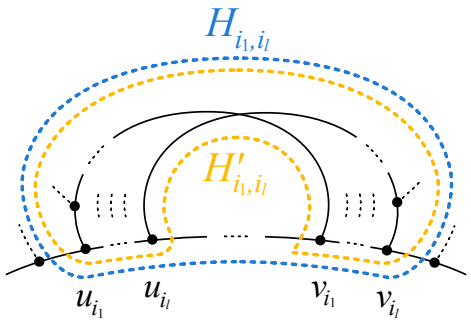


Figure 19: Subgraphs H_{i_1,i_l} and H'_{i_1,i_l} for sequence of almost-nested ears $S = (P_{i_1}, \ldots, P_{i_l})$, highlighted in blue and yellow, respectively.

of the graph through $u_{i_1}, u_{i_l}, v_{i_1}$ and v_{i_l}.

We then introduce a new procedure that can compute $m_{S,\text{partial}}$ for any sequence of almost-nested ears $S = (P_{i_1}, \ldots, P_{i_l})$. Procedure `merge_almost_nested` requires that $m_{(P_{i_j}),\text{partial}}$ be known for all $j \in \{1, \ldots, l\}$. We denote S_r the sequence containing the first r ears of sequence S, i.e. $S_r = (P_{i_1}, \ldots, P_{i_r})$, for all $r \in \{1 \ldots, l\}$. This procedure recursively computes $m_{S_{l-1},\text{partial}}$ (where at the base $m_{(P_{i_1}),\text{partial}}$ is known by the procedure's pre-condition), and then uses this result to compute $m_{S_l,\text{partial}}$ through expressions similar to those used in `merge_inner_ears`, depending on the parities of $p(u_{i_{l-1}}, u_{i_l})$ and $p(v_{i_{l-1}}, v_{i_l})$. We do not describe the details of this procedure here, as they can be found easily by exhaustion in the same way as were found the expressions used in `merge_inner_ears`.

In every procedure used in the original algorithm 2, all references to paths are replaced with sequences of almost-nested ears. Formulas are replaced accordingly: wherever path P_i is replaced with sequence $S = (P_{i_1}, \ldots, P_{i_l})$, any use of $m_{i,\text{final}}$ with u_i and v_i is replaced with $m_{S,\text{final}}$ with u_{i_1} and v_{i_l}. In `merge_neighbors` with inputs S, S' such that S starts with P_{i_1} and S' ends with $P_{i'_l}$, the recursion stops when $S = S'$. P_s is replaced with a sequence of almost-nested ears with analogue properties, and $m_{i,j}$ is replaced with m_{i_1,i'_l}. In `merge_children` and `merge_inner_ears`, the sets $\mathcal{C}_{P_i}$ and $\mathcal{I}_{P_i}$ are replaced with collections of all outer-most child sequences and outer-most inner-sequences, respectively.

Finally, more substantial changes are necessary in `merge_inner_ears`. All formulas are modified so that if `merge_inner_ears` is called with $S = \{P_{i_1}, \ldots, P_{i_l}\}$ and a collection $\mathcal{I}_S = \{S_1, \ldots, S_r\}$ of outer-most inner-sequences, such that P_i is the first in S_1 and P_j is the last in S_r, $m_{S,\text{final}}$ is computed as a function of $m_{i,j}$. We provide here the formula when $p(u_{i_l}, u_i)$ and $p(v_j, v_{i_1})$ are both even.

$$\begin{aligned} m_{S,\text{final}}(W) = {} & m_{i,j}(\{u_i, v_j\}) m_{S,\text{partial}}(W) + m_{i,j}(\emptyset) m_{S,\text{partial}}(W \cup \{u_{i_l}, v_{i_1}\}) \\ & + m_{i,j}(\{v_j\}) m_{S,\text{partial}}(W \cup \{u_{i_l}\}) + m_{i,j}(\{u_i\}) m_{S,\text{partial}}(W \cup \{v_{i_1}\}). \end{aligned}$$

Note the resemblance between this and the value assigned to $m_{i,\text{final}}(\emptyset)$ in `merge_inner_ears` when both paths are even. This is because, since all holes are among $\{u_{i_1}, v_{i_l}\}$, no hole can be in u_{i_l} or v_{i_1}, so that the constraints become similar to those allowing no holes when S is a singleton. All other cases can be constructed in an identical fashion, so we omit them here for legibility.

Finally, we provide in Algorithm 6 the result of adapting the main algorithm of last section, Algorithm 2, to our new generalized context. The major addition is the call to `merge_almost_nested` on all maximal sequences of almost-nested ears between the call to `merge_children` and the call to `merge_inner_ears`. Indeed, once $m_{(P_i),\text{partial}}$ is known for every child P_i, we need to "merge" them into $m_{S,\text{partial}}$ for all maximal sequences of almost-nested ears S, before using them in `merge_inner_ears` for the final computation of $m_{S,\text{final}}$.

Claim 4.5. *At the end of any iteration of a while loop of Algorithm 6, $\mathcal{S}_{init}$ contains all sequences of almost-nested ears S such that neither $m_{S,final}$ nor $m_{S,partial}$ is known, and $\mathcal{S}_{partial}$ contains all sequences of almost-nested ears S such that only $m_{S,partial}$ is known.*

Proof. First observe that at the start and end of any while loop of the algorithm, all ears appearing in $\mathcal{S}_{\text{init}}$ and $\mathcal{S}_{\text{partial}}$ form the decomposition $\mathcal{D}_{\text{init}}$ and $\mathcal{D}_{\text{partial}}$, respectively.

We now prove the claim by induction on the number of iterations of any while loop. Clearly, it suffices to consider only the two inner while loops. At the beginning of the execution, it is clear that $\mathcal{S}_{\text{init}}$ and $\mathcal{S}_{\text{partial}}$ verify the properties of the claim.

Assume now that they do at the beginning of an arbitrary iteration of the first inner while loop. After the for loop has executed, every ear P_i in the sequence S is such that $m_{(P_i),\text{partial}}$ is known.

Algorithm 6: Compute $m_{S,\text{final}}$ of every maximal sequence of almost-nested ears S.

Procedure `compute_mappings`($\mathcal{D} = \{P_1, \ldots, P_k\}$)**:**
  Let $\mathcal{D}_{\text{init}} = \{P_1, \ldots, P_k\}$.
  Let $\mathcal{D}_{\text{partial}} = \emptyset$.
  Let $\mathcal{S}_{\text{init}}$ be the collection of all maximal sequences of almost-nested ears.
  Let $\mathcal{S}_{\text{partial}} = \emptyset$.
  while $\mathcal{D}_{\text{init}} \cup \mathcal{D}_{\text{partial}} \neq \emptyset$ **do**
    while $\exists S \in \mathcal{S}_{\text{init}}$ such that $\forall P_i \in S$, P_i is a leaf in $\mathcal{D}_{\text{init}} \cup \mathcal{D}_{\text{partial}}$ **do**
      for $P_i \in S$ **do**
        Let $\mathcal{C}_{P_i}$ be all children of P_i in $\mathcal{D}$, grouped in maximal sequences of almost-nested ears.
        Run `merge_children`(P_i, $\mathcal{C}_{P_i}$).
        Remove P_i from $\mathcal{D}_{\text{init}}$ and add it to $\mathcal{D}_{\text{partial}}$.
      Run `merge_almost_nested`(S).
      Remove S from $\mathcal{S}_{\text{init}}$ and add it to $\mathcal{S}_{\text{partial}}$.
    while $\exists S \in \mathcal{S}_{\text{partial}}$ such that S has no inner-sequence in $\mathcal{D}_{\text{init}} \cup \mathcal{D}_{\text{partial}}$ **do**
      Let $\mathcal{I}_S$ be all maximal inner-sequences of almost-nested ears of S in $\mathcal{D}$.
      Run `merge_inner_ears`(S, $\mathcal{I}_S$)
      Remove all P_i in S from $\mathcal{D}_{\text{partial}}$, and remove S from $\mathcal{S}_{\text{partial}}$.

Thus the pre-condition of `merge_almost_nested` is verified and it results in $m_{S,\text{partial}}$ being known as well. All $P_i \in S$ are moved to $\mathcal{D}_{\text{partial}}$ and S is moved to $\mathcal{S}_{\text{partial}}$ so that they still verify the required properties.

Assume now that they do at the beginning of an arbitrary iteration of the second inner while loop. Because sequence S is taken from $\mathcal{S}_{\text{partial}}$, it must be that $m_{S,\text{partial}}$ is known. Additionally, because S has no inner-sequence in $\mathcal{D}_{\text{init}} \cup \mathcal{D}_{\text{partial}}$, it must be that all sequences of $\mathcal{I}_S$ are in $\mathcal{D} \setminus \mathcal{D}_{\text{init}} \setminus \mathcal{D}_{\text{partial}}$, so that by the induction hypothesis, $m_{S,\text{final}}$ is known for every sequence S of $\mathcal{I}_S$. The pre-conditions of `merge_inner_ears` are thus verified and $m_{S,\text{final}}$ is known after its execution. The last line updates $\mathcal{D}_{\text{partial}}$ and $\mathcal{S}_{\text{partial}}$ so that they still verify the claim's properties. □

It then immediately follows that, at the end of Algorithm 6, $m_{S,\text{final}}$ is known for all maximal sequences of almost-nested ears. Because (P_1) forms such a sequence, if z is the root of the decomposition, then $m_{(P_1),\text{final}}(\{z\})$ is known and equals the number of near-perfect matchings of the graph with hole z. Finally, the time complexity can be seen to be polynomial by the following argument. For any maximal sequence of almost-nested ears S, `merge_almost_nested` is called exactly once, and thus $O(k)$ in total. Similarly, `merge_children` and `merge_inner_ears` are called only once per ear or sequence, respectively. Consequently, the total time complexity of this algorithm is $O(k) = O(|E(G)|)$. □

We now have an algorithm that can find the exact number of near-perfect matchings in almost-nested ear decomposable graphs with given hole in polynomial time. Again, replacing the recursive call of Algorithm 1 with Algorithm 6 results in an FPRAS for arbitrary graphs whose all factor-critical components are almost-nested ear decomposable. This represents a category of graphs that could not be covered by the FKT algorithm, which we recall applies only to planar graphs.

4.4 Reduction from arbitrary factor-critical graphs to flat-ear decomposable graphs

We now turn to general factor-critical graphs, and describe directions towards applying our solution for (almost-)nested-ear decompositions to general factor-critical graphs (i.e. with arbitrary odd ear decompositions).

There are two key differences between nested-ear decompositions and general ear decompositions which allowed our algorithm to work. The first is that every ear must have a single parent (i.e. its both endpoints must be on one same ear), and the second is that two ears with the same parent must not "cross" each other, i.e. it cannot be that one has exactly one endpoint in the path induced by the other on its parent – that is, in the case of nested-ear decompositions; almost-nested ears do cross but in a tractable way. Ears having a single parent creates a tree structure that simplifies reasoning and restricts dependencies among ears, while preventing crossing makes it possible to count from leaves to root without necessity of considering of number of hole combinations that grows exponentially as we progress towards the root ear.

In the main result of this section, we show that, perhaps surprisingly, the first restriction – namely ears having a single parent – does not render the problem easier. This will also have consequences on the existence of an extension of our above algorithm to ear decompositions with arbitrary crossings and a single parent. We recall that a tree ear decomposition is called flat when it has depth 2, i.e. when all non-root ears are children of the root ear.

Theorem 4.6. *There exists a polynomial time, many-to-one reduction from the problem of counting the number of near-perfect matchings on arbitrary ear decomposable graphs with given hole to the same problem on graphs with a flat tree ear decomposition.*

Towards proving this result, we need the following lemma. Note that this is a generalization of Lemma 4.1 for arbitrary ear decompositions.

Lemma 4.4. *Given any ear decomposable graph G and vertex $v \in V(G)$, there exists an ear decomposable graph G' and vertex $v' \in V(G')$ such that G' has maximum degree 3, and the number of near-perfect matchings of G' with hole v' is the same as that of G with hole v. Moreover, G' can be found in time $O(|E(G)|)$.*

Proof. Recall Construction 3.5, which repeatedly splits vertices of degree at least 4 until none remains. We prove that there exists a way to split each such vertex, that does not break the ear decomposition property of the given graph. To this end, consider a vertex v of degree at least 4. It suffices to describe a partition of its neighborhood $\delta(v)$ such that splitting v along that partition yields an ear decomposable graph.

Let $(P_1, \ldots, P_k)$ be an ear decomposition of G with root z distinct from v. Observe that v is an inner-point of exactly one ear, and an endpoint of all other ears that contain v. Thus exactly two neighbors of v, say u and w, are such that u, v, w forms a subpath of an ear P_i, and all other neighbors are part of other ears that share endpoint v. Let y be one such neighbor. We now partition v into $N_1(v) = \{u, y\}$, and $N_2(v) = \delta(v) \setminus N_2(v)$. This gives us three vertices v_1, v', v_2 such that $\delta(v_1) = N_1(v) \cup \{v'\}$, $\delta(v_2) = N_2(v) \cup \{v'\}$, and $\delta(v') = \{v_1, v_2\}$. Figure 20 illustrates this. In ear P_i, we replace v with the path v_1, v', v_2. As a result, ear P_i remains one, all ears with an endpoint on P_i still have one on P_i, and all other ears are unchanged. Therefore, the resulting graph is ear decomposable. Moreover, by Lemma 3.2, given a graph and vertex z on it, after splitting any vertex of the graph, there exists a known vertex z' such that the number of near-perfect matchings with hole z in the original graph equals that with hole z' in the resulting graph. As a consequence, we

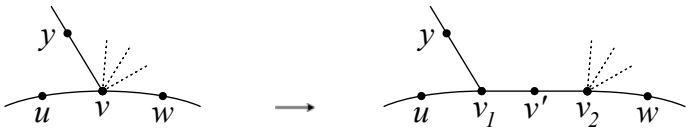


Figure 20: Splitting vertex v of degree 4 or more without breaking ear decomposition.

can obtain vertex v' of the lemma such that the number of near-perfect matchings with hole v in G is the same as the number with hole v' in G'.

□

Proof of Theorem 4.6. To prove this result, we consider an arbitrary ear decomposable graph G and vertex z, and describe a polynomial-time procedure for constructing a graph G' with a flat tree ear decomposition and a vertex $z' \in V(G')$, such that the number of near-perfect matchings on G' with hole z' is a known function of the number of near-perfect matchings of G with hole z.

The first step is the construction of a graph G' and vertex $z' \in V(G')$ such that G' has maximum degree 3 and the number of near-perfect matchings with hole z' in G' is the same as that of G with hole z. This is possible by Lemma 4.4. We will thus assume from now on, for better legibility, that G already has maximum degree 3.

Find an arbitrary spanning tree T of G with root z. Let P_T be an arbitrary self-intersecting closed path on T that covers T, starts and ends in z, and such that it traverses each edge of T exactly twice. We do this by traversing T in the order a depth-first recursive algorithm would traverse T. Figures 21 (a) and (b) show two drawings of the same example ear decomposable graph with an arbitrary spanning tree on it in bold edges. Figure 21 (b) also indicates in yellow arrows an example order in which T is traversed by P_T.

We start the construction of a graph G' that will allow us to map P_T to a closed path P'_T on G', such that P'_T traverses each vertex of G' at most once. We will then use P'_T as the root ear of a flat tree ear decomposition of G'.

Let $\mathcal{S}_{\text{pending}}$ be the collection of all paths of T that are pending in G. For all $P \in \mathcal{S}_{\text{pending}}$, duplicate P according to Construction 3.3. This replaces P with a subgraph that contains at least two edge-disjoint paths that join the endpoints of P, which we denote P_1 and P_2, and call *duplicated*

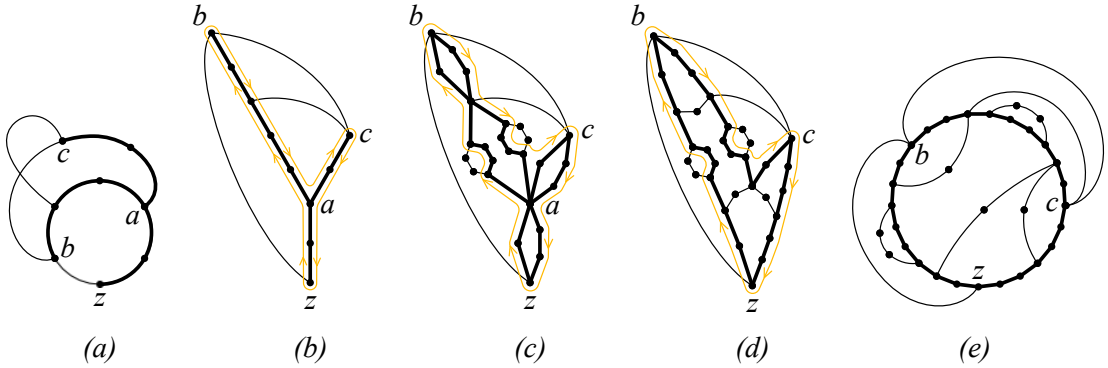


Figure 21: Intermediate steps of the reduction from arbitrary ear decomposition to flat-tree-ear decomposition, on example graph (a).

paths of P. For example, in Figure 8 on page 23, these paths could be $(x, v_1, v_2, v_3, v_4, y)$ and $(x, w_1, w_2, w_3, w_4, y)$. We call G_1 the resulting graph. Note that we can now have an edge-disjoint closed path on G_1 that covers all vertices of G as follows: for all $P \in \mathcal{S}_{\text{pending}}$, replace in P_T the first occurrence of P with P_1, and the second with P_2. Figure 21 (c) shows, for the same example graph, the result of this transformation and one such edge-disjoint cycle in bold edges.

Consider now every vertex v in G such that there are exactly 2 pending paths P and Q in $\mathcal{S}_{\text{pending}}$ with endpoint v (i.e. v has degree 2 in T). We partition the neighborhood of v in G_1 into the two classes $N_1(v) = \delta(v) \cap (P_1 \cup Q_1)$ and $N_2(v) = \delta(v) \setminus N_1(v)$. We then split v along this partition, by replacing it with path v_1, v', v_2, such that $\delta(v_1) = N_1(v) \cup \{v'\}$, $\delta(v_2) = N_2(v) \cup \{v'\}$ and $\delta(v') = \{v_1, v_2\}$. We can thus now define a closed path from P_T that overlaps itself only on vertices of degree 3 on T by the same process as above of replacing every $P \in \mathcal{S}_{\text{pending}}$ with P_1 and P_2 in P_T.

Finally, we consider vertices v of degree 3 in T. Let P, Q and R be the three paths in T that are pending in G and whose shared endpoint is v, such that they are first encountered in the order P, Q, R when traversing T through P_T. We first partition the neighborhood of v into the two classes $N_1(v) = \delta(v) \cap (P_1 \cup Q_1)$ and $N_2(v) = \delta(v) \setminus N_1(v)$, and split it along this partition. We then have three vertices v_1, v', and v_2 in place of v, with again $\delta(v_1) = N_1(v \cup \{v'\}$, $\delta(v_2) = N_2 \cup \{v'\}$ and $\delta(v') = \{v_1, v_2\}$. In the resulting graph, we partition the neighborhood of v_2 into $N_1(v_2) = \delta(v_2) \cap (V(Q_2) \cup V(R_1) \cup \{v'\})$ and $N_2(v_2) = \delta(v) \setminus N_1(v_2)$, and split it along this partition. We label the three new vertices w, v'' and v_3 such that $\delta(w) = N_1(v_2) \cup \{v''\}$, $\delta(v_3) = N_2(v_2) \cup \{v''\}$ and $\delta(v'') = \{w, v_3\}$. Figure 22 shown such a vertex v and the results of each of these two vertex splits. We call G' the resulting graph. Still on the same example graph, Figure 21 (d) shows the result of splitting the endpoints of pending paths with degrees 2 and 3 in T.

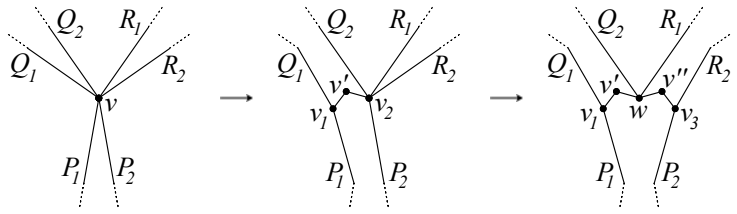


Figure 22: Transformation of a vertex v of degree 3 in spanning tree T.

We can now construct a closed path P'_T that does not intersect itself except in z, by the same process as above (we replace each $P \in \mathcal{S}_{\text{pending}}$ in P_T with P_1 and P_2 in its first and second occurrence respectively). We now show that P'_T is not self-intersecting outside of z. To see this, first observe that P'_T does not intersect itself on any duplicated path (i.e. path of the form P_i for $P \in \mathcal{S}_{\text{pending}}$ and $i \in \{1, 2\}$). Moreover, P'_T only traverses duplicated path, so that it must be edge-disjoint. Now it remains to see that there is no 3 duplicated paths that share a vertex. To this end, notice that after the step of duplicating all paths, for any endpoint v of any duplicated path, exactly 6 duplicated paths share v if v has degree 3 in T, exactly 4 if it has degree 2 in T, and exactly 2 if it has degree 1 in T. In the latter case, we are done. If v has degree 2 in T, v is split into v_1, v' and v_2 so that exactly 2 duplicated paths share v_1, and exactly two share v_2. Similarly when v has degree 3 in T, then v is in the end replaced by 5 vertices, three of which are the endpoints of exactly two duplicated paths.

We now show that P'_T is the first path of a tree ear decomposition of depth 2. Because T is a spanning tree of G, all edges of G not in T have both endpoints in T, and thus form odd ears of our decomposition. Furthermore, when duplicating an odd pending path P with endpoints u and v, we

create 4 distinct paths from u to v, and in our construction kept only 2 that are edge-disjoint, called P_1 and P_2, which are then traversed by P'_T. It is easy to see from Figure 8 that there is exactly one pending path of length 3 with both endpoints on P_1, and similarly on P_2. These are also ears of our decompositions. Finally, paths added when splitting a vertex are even pending paths between vertices that belong to P_T, so that they also form ears. Thus we have an ear decomposition where P_T is the first ear and such that all other ears are children of P_T. Figure 21 (e) shows an alternate drawing of (d) that makes this decomposition clearer.

If k is the number of ears in G, one can chose T such that at most k edges do not belong to T. To see this, let $(P_1, \ldots, P_k)$ be an ear decomposition of G with root vertex z. T starts in z and covers all vertices of P_1. Then until T is not a spanning tree, select an ear P_i with both endpoints in T, create a branch from T through one endpoint of P_i and through all its vertices. We can prove by induction on the number of ears in the decomposition that there are at most k edges uncovered by T. The base case where there is only one ear is immediate. Then if G has an ear decomposition $\mathcal{D}$ with k ears and T covers all but at most k edges of G, consider G' obtained by adding one new ear P_{k+1} at the end of $\mathcal{D}$. Both endpoints of this ear must be in T because T is a spanning tree of G, so that we can branch off of T into P_{k+1} and through all its vertices. Clearly, this leaves one edge of P_{k+1} uncovered, so G' has $k+1$ ears and at most $k+1$ edges uncovered by T. Therefore, in the tree ear decomposition of depth 2 of our graph G', at most k ears are due to edges originally in G.

Furthermore, each duplicated path can have up to one ear, so that every path of T pending in G causes at most 2 new odd ears in G'. Finally, for each split vertex, a new even ear is created. Thus in total, we have $O(k + |E(G)| + |V(G)|) = O(n^2)$ many ears in G', where $n = |V(G)|$.

Finally, let us relate the number of near-perfect matchings of G with hole z to that of graph G'. We know by Lemmas 3.1 and 3.2 that vertex splitting does not modify the number of perfect and near-perfect matchings with given hole. However, we know by Lemmas 3.3 and 3.4 that duplicating an odd pending path maps every perfect matching and near-perfect matching with given hole of the original graph to exactly 4 of the resulting graph. Therefore, if the above construction duplicates r odd pending paths, the number of perfect and near-perfect matchings with hole z of G' will be 4^r times the number on G.

Thus assume there exists an algorithm for counting perfect and near-perfect matchings with given hole for tree ear decomposable graphs of degree 2 that run in polynomial time in the size of the graph. On input an arbitrary ear-decomposable graph G (for example, a factor-critical graph), we can construct G' as described here and keep track of the number r of duplicated odd pending paths, obtain the number of near-perfect matchings of G' and return that number divided by 4^r. We also note that, because duplicating an odd pending path quadruples *all* perfect and near-perfect matchings with given hole, the problem of sampling perfect and near-perfect matchings with given hole on G also reduces to that of sampling them on G'.

□

Although this result shows that ear decompositions with ears whose endpoints are on two distinct parents are no more than polynomially harder than tree ear decompositions, it does reduce the problem of arbitrary ear decomposable graphs to those with a much more constrained structure, that is tree ear decomposable, and moreover with constant depth. We also note that the earlier result of this thesis that reduces the problem to graphs with maximum degree 3 also applies here without increasing the depth of the graph or breaking its structure, by appropriate splitting of vertices.

We also note that an additional structure of the graph G' constructed from G is that all children ears can be partitioned into subsets that form nested-ear decompositions with the root ear. One

future direction to research is whether there exists a way to merge the results of counting the number of matchings on each such subset. One first difficulty along this direction is the following: consider a case where there exists a vertex v on the root ear P_1 such that v is an endpoint of an ear in a subset S, but of no ear in S'. Then, in order to sample all matchings on S' that can be contained in a matching of the original graph, we need to allow S' to sample matchings that have v as a hole. Otherwise all matchings that match v to the ear in S would not be found. If this can be solved, then we also need to be able to merge the samples (or counts) of each subset into a sample (or the count) of the full graph. Naively rejecting conflicting combinations of samples is not viable, as we expose ourselves to exponential runtimes if there are exponentially many conflicting combinations for only a polynomial number of valid ones.

5 Further research

Leverage redundancy between hole patterns Recall the algorithm by Štefankovič et al. [14], which uses the Gallai-Edmonds decomposition of arbitrary graphs and an FPRAS on bipartite graphs to count perfect matchings on general graphs. For each factor-critical component H of the Gallai-Edmonds decomposition, and every vertex $v \in V(H)$ that has a neighbor outside of H, they need to obtain the number of perfect matchings in $H - v$ (or equivalently the number of near-perfect matchings with hole v). As we briefly stated earlier, if there is only one such v per factor-critical component H, then the time complexity of their algorithm becomes polynomial. While we have not found a way to modify the input graph so as to guarantee this, we have not proved it impossible, and studying graphs that have this property could help understand what structure of the graph implies this property (if any), and whether it can be added to the graph without unknown impacts on matching counts, through transformations similar to those introduced in Chapter 3.

One other observation on this algorithm is that there may exist redundancy between perfect matchings of $H - v$ and perfect matchings of $H - w$, for distinct vertices $v, w \in V(H)$. When computing the number of perfect matchings on $H - v$ for multiples vertices $v \in V(H)$, we may thus do some work more than once. It is therefore natural to think of leveraging this redundancy to avoid doing unnecessary work. One approach to this is to sample from $H - v$ for some v using a standard, costly method (here we mean, by a recursive call to the algorithm by Štefankovič et al. [14]), and then from this result, obtain a sample of $H - w$ for some distinct w. Let us discuss this idea.

We focus mostly here on sampling rather than counting, as it appears to us more promising than to obtain the number of perfect matchings of $H - w$ from only a number. We assume therefore that we have a (almost-)uniform sampler for perfect matchings of $H - v$. The goal is to obtain, from that sampler, a (almost-)uniform sampler of perfect matchings of $H - w$. We observe that, from any perfect matching M of $H - v$, we can obtain one of $H - w$ by flipping an alternating even path between v and w on M – an alternating path is one which alternates between selected and non-selected edges of a given matching.

Assuming there exists a uniform sampler of alternating paths between two vertices and given a matching, sampling a matching of $H - w$ from M by simply flipping an alternating path obtained uniformly at random fails to yield a uniform sampler over $H - w$. To see this, consider Figure 23. It shows a factor-critical graph H (i.e. with an odd ear decomposition), and the only two matchings of $H - v$, on the left. In the middle, each of the two possible alternating paths is shown for both perfect matchings of $H - v$, and on the right are the three perfect matchings of $H - w$. Arrows show which perfect matching of $H - v$ has the corresponding alternating path, and what the resulting matching of $H - w$ is. In such a context, a uniform sampler of alternating paths is insufficient, as it would result in a distribution of perfect matchings over $H - w$ with probability $\frac{1}{2}$ for the second perfect matching of $H - w$, against $\frac{1}{4}$ for the two others.

We would therefore need a sampler of alternating paths that is biased towards those perfect matchings of $H - w$ that have less matchings of $H - v$ that can reach them through an alternating path. This is likely to be a more difficult problem, as it needs to somehow encapsulate the information of the dependencies we depicted in Figure 24 for the input graph G.

Furthermore, there is a stronger argument against the possibility of finding such a sampler. There are factor-critical graphs H on which there exist two vertices v and w such that $|\mathcal{M}(H - v)|$ is exponential, while $|\mathcal{M}(H - w)|$ is constant. Consider Figure 24, which shows a graph H constructed, again, from the chain of boxes with k boxes. The number of perfect matchings on $H - v$ is one, while that on $H - w$ is $O(2^k)$. Therefore, the problem of sampling on $H - w$ is not made easier by having access to sampler for $H - v$, as there is essentially no redundancy.

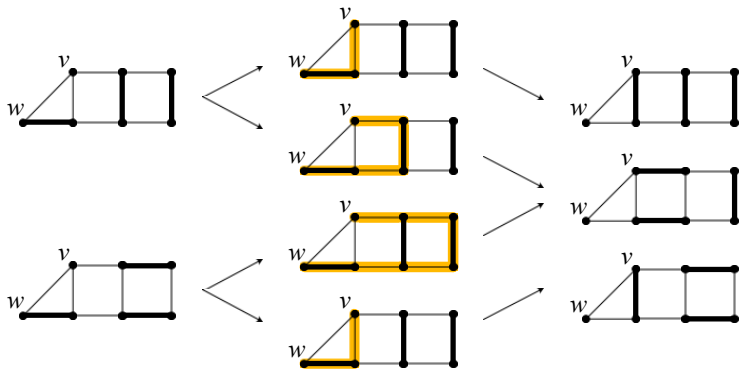


Figure 23: Near-perfect matchings on factor-critical graph with holes v on the left and w on the right, and which alternating paths can go from one to the other, highlighted in yellow.

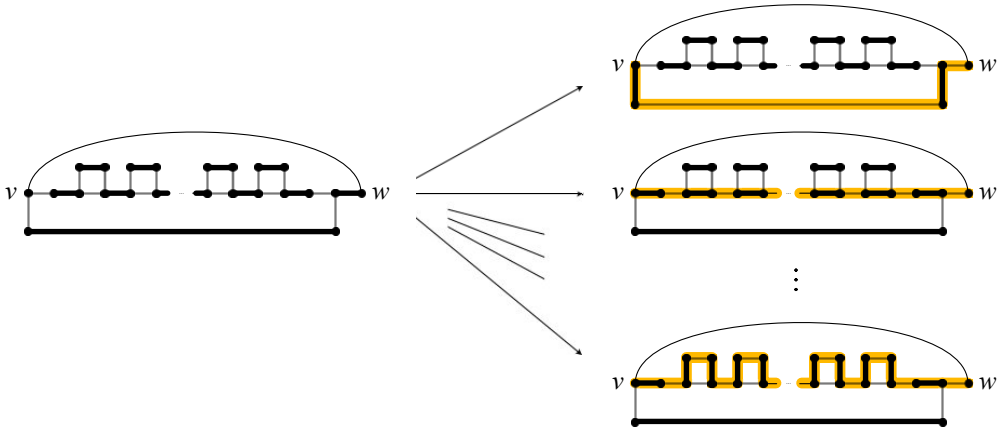


Figure 24: Near-perfect matchings on factor-critical graph with holes v on the left and w on the right, with the alternating path that can go from one to the other, highlighted in yellow.

Study graph modifications We believe that graph modifications of the form of those we introduced in Chapter 3 are powerful tools for counting perfect matchings, whose further study can both improve our understanding of matchings, and manipulate their form, number and distribution, in ways that we can predict. Specifically, we think that this has potential as an alternative approach to impact the distribution of matchings of a given graph. Indeed, the techniques used to extend the FPRAS on dense bipartite graphs [4] to arbitrary bipartite graphs [6] was done by adding complicated weights to the Markov chain, which rendered its analysis more difficult. Instead, one could impact the distribution of matchings through careful modifications of the input graph, similar to those we described here. This is what we explore in the next paragraph.

Make perfect matchings more numerous We describe here an example of a graph modification using tools we introduced in section 3, that aims at making matchings with few holes more numerous than those with lots of holes, thus effectively weighing them up in the distribution of matchings in the graph. This approach therefore also means to use our Theorem 2.3 that states that we may allow up to $n^{\frac{k}{k+1}}$ holes in a sample, for some constant k.

Recall the pending path duplication described in Construction 3.3 of section 3.1.3, that multiplies all matchings by exactly 4 when the pending path is odd. Recall also Construction 3.1 that inserts a pending path of length 2, described in section 3.1.2. We propose the following usage of these two tools. First, every edge of the given graph G is extended to length $n = V(G)$ by insertions of pending paths. Then, we split each resulting edge, which we may do since an edge is simply an odd pending path of length 1. This results in a graph G' with $O(n|E(G)|)$ edges, and $O(n + n|E(G)|)$ vertices.

Observe now that, by Lemma 3.1, the number of perfect matchings of G has been multiplied by exactly $4^{n|E(G)|}$. However, consider a matching with only one selected edge (i.e. with $n - 2$ holes). In G, there are $|E(G)|$ such matchings, and $|E(G')| = O(n|E(G)|)$ in G', so that it has been multiplied by only n. We thus have a graph G' whose relative number of perfect matchings is greater than in G. The question is: by how much? More specifically, how does this transformation impact matchings with an arbitrary number of holes, and how favorable is it to matchings with fewer than $n^{\frac{k}{k+1}}$ holes, for some constant k? Recall that we have shown in this report that given an almost-uniform sampler of matchings with at most $n^{\frac{k}{k+1}}$ holes, we can obtain an almost-uniform sampler of perfect matchings. Therefore, we only need that the number of matchings with less than $n^{\frac{k}{k+1}}$ holes represents at least a polynomial fraction of the total number of matchings of G'. If this holds, we can reject matchings with too many holes, and return the first with few enough holes. By Theorem 2.3, this could then be used to sample perfect matchings of G'. Knowing that there are exactly $4^{n|E(G)|}$ times more perfect matchings in G' than in G, having the count on G' (within error $1 \pm \varepsilon$) could give us the count on G (within error $1 \pm \varepsilon$ again). Therefore, proving that in G', the number of matchings with at most $n^{\frac{k}{k+1}}$ is at least a polynomial fraction of all matchings for some k, would reduce the problem of uniformly sampling perfect matchings to that of sampling arbitrary matchings. We note also that the choice of replacing each edge with a sequence of n is arbitrary in this example, and could be done with smaller or larger values as well. We will discuss the problem of sampling arbitrary matchings shortly.

We would like to remark here that this can be seen as approaching the problem of re-weighting matchings from the opposite direction. Indeed, by re-weighting the Markov chain's states, we weigh down all matchings that we wish to avoid so that perfect matchings are not over-shadowed in the sampling. One difficulty with this approach we often encountered is that by weighing down a given set of matchings, we are decreasing the conductivity of the Markov chain's underlying graph. While [6] proved that this can be done carefully enough that it does not become too low for bipartite graphs, we do not know to this day whether this is also true for general graphs: it may be that in order to weigh down near-perfect matchings enough, it is necessary to bring the underlying graph's conductivity too low for the Markov chain to mix. Contrast this now with our proposed approach: rather than weighing down unwanted solutions, we weigh up those we are looking for. From the point of view of the Markov chain, we are adding new states to it, so that its underlying graph is only more connected. Whether our proposed idea specifically is found to have potential or not, we believe that, in general, an approach that modifies the graph so as to effectively "weigh up" desired solutions could be an interesting direction for future approaches.

Finally, following our proposed idea, let us discuss the problem of uniformly sampling arbitrary matchings (i.e. those with any number of holes). We observe that, in [4] and [6], a source of complexity emerges from the strict limit of 2 holes (i.e. the Markov chain has only perfect and near-perfect matchings as allowed sates). When describing a canonical path from any two matchings I and F on the Markov chain's underlying graph, this implies that the potential open path in $I \oplus F$ must be flipped (or *unwound*) first, before flipping the cycles of $I \oplus F$, as otherwise the path would traverse states with 4 holes. However, deciding to flip the path first implies that the encoding of a

canonical path may have 4 holes. In other words, either the canonical path traverses a state with 4 holes, or its encoding may have 4 holes. This is due to the choice of the encoding: at transition t, a canonical path traversing t is encoded by essentially undoing all it has done until t, and doing what it will to after t. Thus if the path was flipped first, it will be flipped back in the encoding, thus yielding an encoding with 4 holes, and symmetrically if it is flipped last. To remedy this, [4] did not include the path in the encoding (which cost them the use of the density assumption), while [6] did include it, and needed non-trivial relations between weights of different hole patterns to show that the cost of "exiting" the Markov chain's state space – by allowing 4 holes – was at most polynomial.

Now without any restriction on the number of holes, we may flip the path of $I \oplus F$ arbitrarily, since creating holes has no incidence. Similarly, the encoding may have any number of holes while remaining in the Markov chain's state space. We also note the resemblance of this remark with the observation we made when considering sampling half-integral matchings: if limiting the allowed number of half-integral edges or cycles in the state space, the encoding could exit it by having any number of half-integral edges or cycles. Thus in this case as well, by not restricting the number of half-integral edges or cycle, the encoding would always be a valid state of the Markov chain's state space.

As a conclusion for this section, we believe that studying the construction we proposed here (or one similar) for increasing the number of matchings with few holes is a direction with potential, especially given our result that tolerates up to $n^{\frac{k}{k+1}}$ holes. Moreover, because sampling arbitrary matchings appears to be missing one central difficulty that the state of the art approaches needed to tackle, and similar to one we encountered when trying to tackle the general problem, we see this direction for future research as only more interesting towards solving the problem on general graphs.

6 Conclusion

In this thesis, we explored several directions for solving the problem of approximately counting perfect matchings on general graphs, and introduced new tools and results that we believe can be regarded as potential building blocks for a future solution. We described a relation between the problem of counting perfect matchings on bipartite graphs and that of counting half-integral perfect matchings on the general graph, as an approach to generalizing the solution by Jerrum, Sinclair, and Vigoda [6] that does not require it to run directly on the general graph. We studied two main strategies to weigh down half-integral matchings in the sampler's distribution, and shed light on key difficulties that future research should tackle in order to make progress in this direction. One important result we obtained in this regard is the ability to tolerate up to $n^{\frac{k}{k+1}}$ local errors in a sample, where a local error can be a half-integral edge, cycle or hole in the matching. The latter is one we believe is powerful, as it implies that it suffices to sample matchings with no more than $n^{\frac{k}{k+1}}$ holes in order to be able to sample perfect matchings.

One other finding we believe is important in this thesis, is the introduction of a collection of tools that modify a given graph, with the property that they result in predictable changes in the number of perfect matchings. Specifically, they allowed us to construct reductions for the problem of counting perfect matchings on arbitrary graphs to that on graphs with more defined structure, such as tripartiteness, or maximum degree 3. We also discussed how such tools could be used for more intricate goals, such as modifying the distribution of matchings on arbitrary graphs so as to make perfect matchings more frequent, by a known exponential factor. We emphasize again our belief that tools with this property can constitute an interesting direction for future research.

Finally, motivated by a known result by Štefankovič, Vigoda, and Wilmes [14] that implies that the problem on general graphs can be solved by solving it on factor-critical graphs, we studied one key property of such graphs, namely ear decomposability. We described algorithms that count exactly the number of near-perfect matchings with given hole on ear decomposable graphs whose decomposition is nested and almost-nested, thus resulting in an FPRAS for graphs whose all factor-critical components in the Gallai-Edmonds decomposition are almost-nested ear decomposable. Towards a solution for arbitrary factor-critical graphs – and by extension, for general graphs – we described a reduction from factor-critical graphs to flat-tree-ear decomposable graphs, that is, those whose ear decomposition has depth 2. This implies that the study of the ear decomposition of a factor-critical graph can be simplified to one in which all ears share the same parent.

References

[1] Andrei Z Broder. How hard is it to marry at random?(on the approximation of the permanent). In *Proceedings of the eighteenth annual ACM symposium on Theory of computing*, pages 50–58, 1986. [5]

[2] Farzam Ebrahimnejad, Ansh Nagda, and Shayan Oveis Gharan. Counting and sampling perfect matchings in regular expanding non-bipartite graphs. *arXiv preprint arXiv:2103.08683*, 2021. [5]

[3] Jack Edmonds. Paths, trees, and flowers. *Canadian Journal of mathematics*, 17:449–467, 1965. [4 and 31]

[4] Mark Jerrum and Alistair Sinclair. Approximating the permanent. *SIAM journal on computing*, 18(6):1149–1178, 1989. [5, 19, 20, 52, 53, and 54]

[5] Mark Jerrum and Alistair Sinclair. The markov chain monte carlo method: an approach to approximate counting and integration. *Approximation Algorithms for NP-hard problems, PWS Publishing*, 1996. [5]

[6] Mark Jerrum, Alistair Sinclair, and Eric Vigoda. A polynomial-time approximation algorithm for the permanent of a matrix with nonnegative entries. *Journal of the ACM (JACM)*, 51(4): 671–697, 2004. [5, 6, 7, 8, 9, 12, 15, 17, 19, 20, 27, 29, 31, 52, 53, 54, and 55]

[7] Pieter W Kasteleyn. The statistics of dimers on a lattice: I. the number of dimer arrangements on a quadratic lattice. *Physica*, 27(12):1209–1225, 1961. [5 and 32]

[8] Pieter W Kasteleyn. Graph theory and crystal physics. In F. Harary, editor, *Graph Theory and Theoretical Physics*, pages 43–110. Academic Press, London, 1967. [5 and 32]

[9] Bernhard H Korte, Jens Vygen, B Korte, and J Vygen. *Combinatorial optimization*, volume 1, chapter 10.4, pages 223–229. Springer, 2011. [30]

[10] László Lovász. A note on factor-critical graphs. *Studia Sci. Math. Hungar*, 7(279-280):11, 1972. [30]

[11] László Lovász and Michael D Plummer. *Matching theory*, volume 367, chapter 3.2, pages 93–102. American Mathematical Soc., 2009. [30 and 31]

[12] László Lovász and Michael D Plummer. *Matching theory*, volume 367, chapter 7.3, pages 280–281. American Mathematical Soc., 2009. [10]

[13] Jens M Schmidt. A simple test on 2-vertex-and 2-edge-connectivity. *Information Processing Letters*, 113(7):241–244, 2013. [36]

[14] Daniel Štefankovič, Eric Vigoda, and John Wilmes. On counting perfect matchings in general graphs. In *Latin American Symposium on Theoretical Informatics*, pages 873–885. Springer, 2018. [5, 6, 9, 27, 28, 29, 31, 41, 51, and 55]

[15] Harold NV Temperley and Michael E Fisher. Dimer problem in statistical mechanics-an exact result. *Philosophical Magazine*, 6(68):1061–1063, 1961. [5 and 32]

[16] Leslie G Valiant. The complexity of computing the permanent. *Theoretical computer science*, 8(2):189–201, 1979. [5]