On Approximation Complexity of Counting Matchings in Hypergraphs

Benjamin Cabrera

July 22, 2016

- Bachelorthesis -

First examiner: Prof. Dr. Marek Karpinski
Second examiner: PD Dr. habil. Elias Dahlhaus

Department of Computer Science (Chair V)

Faculty of Mathematics and Natural Sciences

University of Bonn

Contents

1 Introduction and Motivation

1.1 Motivation from statistical physics

In the late 19th century, mostly by the works of L. Boltzmann and J.W. Gibbs, the research field of statistical mechanics was born. Systems of interacting molecules and atoms were no longer studied in a deterministic way (which is almost impossible for the large systems of interest) but in a probabilistic way. One assumes a certain probability distribution pf the particles (usually depending also on the the temperature, pressure and similar properties of the system) and then studies their behaviour in probability. In particular, researchers are often interested if a system has *phase transitions*, that are abrupt, discontinuous changes in the properties of the system. The Ising model as a model for ferromagnetism, for example, has a phase transition at a certain critical temperature where the tiny ferromagnets suddenly lose their orientation and the magnet becomes non-magnetic.

Naturally the relation of interacting particles to each other can be formalized in graphs and hypergraphs. The problem analysed in this thesis is related to the study of monomer dimers (or their equivalent for bigger molecules). A *monomer-dimer system* is specified by a weighted graph (the weights often called *Boltzmann factors*) where so called dimers can be placed on edges such that no vertex of the graph is covered by more than one dimer. The uncovered vertices are called *monomers*. The graphs considered in statistical physics are mostly (infinite) d-dimensional lattices which represent the positions of particles in space. The dimers of the monomer-dimer system can be interpreted as diatomic molecules which occupy disjoint pairs of adjacent vertices of the lattice, the remaining vertices are the monomers. In a graph theoretical context the dimers form nothing else than a matching.

The question that is interesting in this context is: can a phase transition occur when varying the monomer concentration of a monomer-dimer system? O. Heilmann and E. Lieb [HL72] proved in 1972 that there is no phase transition in a monomer-dimer system. The authors also note (cf. [HL72, p. 194]) that because of the above mentioned relation between graphs and its line graphs (cf. Definition 2.15 and Lemma 3.4) their result implies that there is also no phase transition for the hard core lattice gas model (sampling independent sets instead of matchings). The latter

holds not only for line graphs but also for general claw-free graphs which was proved 2007 by M. Chudnovsky and P. Seymour in [CS07].
Monomer-dimer systems can be generalized to systems with bigger molecules (polymers) which gives monomer-trimer (molecules consisting of three similar parts) and monomer-polymer systems. In 1972 Heilmann [Hei72] proved that certain monomer-trimer systems have a phase transition. For monomer-polymers the underlying graph theoretical structure is not longer a graph but a hypergraph which motivates our effort to approximate the matching polynomial and the number of matchings in a hypergraph better.

1.2 Scope

The problem of counting matchings in hypergraphs has not gotten a lot of attention by researchers in the past. It was only recently that the topic became more actively researched, mainly because of the developments of some new techniques for constructing approximation algorithms (see Chapter 4) and because the hardness of exact counting was proved for regular graphs (see Theorem 3.7).

The first hardness result for the problem of exact counting matchings in graphs was proved by L. Valiant [Val79b], shortly after he had introduced the the #**P**-class for counting problems in 1979. As a result counting matchings in arbitrary hypergraphs is #**P**-complete.
When a problem is proven to be too hard to solve in polynomial time one introduces further restrictions on the underlying structures such that the problem becomes easier to solve but also less general. In the case of counting matchings in (hyper-)graphs it makes sense to restrict the problem to (hyper-)graphs of bounded degree, because then the number of edges intersecting in a vertex is bounded which makes matchings less "concentrated" on a vertex.
A useful observation in this context is the equality of matchings in a hypergraph H and independent sets in the intersection graph $L(H)$ (cf. Lemma 3.4). In 2000 C.Greenhill [Gre00] proved that counting independent sets in 3-regular graphs (graphs where each vertex has degree three) is already #**P**-complete. Using the above-mentioned relation between independent sets and matchings this implies that exact counting of matchings in hypergraphs is already #**P**-complete for the small subclass of hypergraphs with degree less or equal than two. In conclusion one has relatively small subclasses of hypergraphs for which exact counting is still #**P**-complete, but for which one can tackle the problem of approximate counting by using structural properties only existing in this subclasses.

Approximation algorithms can be divided into two classes: deterministic and randomized algorithms. Randomized algorithms are often used because there is a very general method for

constructing them, namely the usage of a Markov chain. In contrary, deterministic algorithms are often harder to construct but also more valuable because there is no probability of an unexpected error. We explain these differences in more detail in Chapter 4.
A first approximation algorithm for the number of matchings in a hypergraphs was a result of a construction by M. Luby and E. Vigoda [LV99]. Their FPRAS for the number of independent sets in graphs with degree bounded by four yields a FPRAS for the corresponding matching problem in hypergraphs. The last randomized approximation algorithm comes from M. Karpinski, A. Ruciński and E. Szymańska [KRS13]. Restricting to sparse hypergraphs without structures called 3-combs, they showed in 2013 the existence of a FPRAS for the number of matchings. The most recent developments started in 2007 with a paper by D. Weitz [Wei07] introducing the *correlation decay* technique, which results in deterministic approximation algorithms. His method was then generalized by D. Gamarnik and D. Katz [GK07] and used to construct a FPTAS for the number of matchings in graphs. In 2014 A. Dudek, M. Karpinski, A. Ruciński and E. Szymańska applied this method to the hypergraph case which resulted in a FPTAS for the subset of $(3, 3)$-hypergraphs.

1.3 Outline

In this thesis we analyse the problem of counting matchings in hypergraphs. We start in Chapter 2 with some preliminaries from graph theory, probability theory and computational complexity theory. In Chapter 3 we introduce the problem of counting matchings and cite some hardness and some first approximation results. In Chapter 4 we describe two general approaches for constructing approximation algorithms. In Chapters 5 and 6 we follow the proofs from [KRS13] and [DKRS14] that apply these methods to the problem of counting matchings. Finally in Chapter 7 we summarize our results and motivate further research.

2 Preliminaries

In this chapter we will introduce some basic concepts of mathematics and computational sciences that are required for understanding the rest of this thesis. Section 2.1 introduces some definitions of graph theory which are naturally the core of this whole thesis. In Section 2.2 we give some definitions and immediate results of probability theory which are mostly needed in Chapter 5. In Section 2.3 we introduce definitions from Computational Complexity Theory. These include Turing machines (Subsection 2.3.1), decision problems and the classes **P** and **NP** (Subsection 2.3.2) and also counting problems and the class #**P** (Subsection 2.3.3). Finally in Subsection 2.3.4 we define the types of approximation algorithms used in this thesis.

2.1 Graph Theory

This section introduces some graph theoretical definitions. Graphs are mathematical structures that model objects and especially their relationship to each other in a rigorous way. Graph theory belongs to the field of discrete mathematics, a relatively new part of mathematics, that deals with structures that are fundamentally discrete (instead of continuous structures used, for example, in Analysis). One of the first documented graph theoretical problems may be the 'Seven Bridges of Königsberg'-problem, in which the king of Königsberg wants to walk a loop, crossing all bridges of Königsberg exactly once. Leonard Euler(1707-1783) solved the problem in 1735 by proving that such a walk is not possible.

Today's interest in graph theory is often related to solving practically occurring problems efficiently with the help of modern computers. Current problems include: efficiently finding a solution for the *Travelling salesman problem (TSP)*, the *Hamiltonian path problem* and the problem of counting matchings in graphs efficiently.

Although we are interested in counting matchings in hypergraphs, we start by introducing the commonly used subset of graphs first.

Definition 2.1 (Undirected Graph)**.** *A* ***undirected graph*** $G = (V, E)$ *consists of a finite, nonempty set of* ***vertices*** *or* ***nodes*** $V = \{v_1, \ldots, v_n\}$ *and a set of* ***edges*** $E = \{e_1, \ldots, e_m\} \subset \{\{v, w\} : v, w \in V \text{ and } v \neq w\}$*. Two vertices* $v, w \in V$ *are* ***adjacent*** *in* G *if* $\{v, w\} \in E$*.*

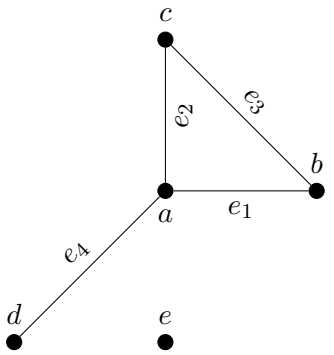
Definition 2.2 (Directed Graph)**.** *A **directed graph** $G = (V, E)$ consists of a finite, nonempty set of **vertices** $V = \{v_1, \dots, v_n\}$ and a set of **edges** $E = \{e_1, \dots, e_m\} \subset V \times V$. We say an edge e goes **from** v **to** w if $e = (v, w)$.*

Definition 2.3 (Degree of a vertex)**.** *The **degree** $deg(v)$ of a vertex v is the number of connected edges to that vertex. This means for undirected graphs*

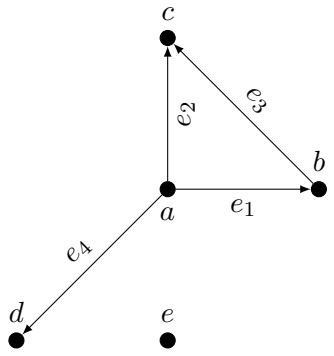
$$deg(v) = |\{e \in E : v \in e\}|,$$

and for directed graphs

$$deg(v) = deg_{in}(v) + deg_{out}(v) = |\{e = (v_1, v_2) \in E : v_2 = v\}| + |\{e = (v_1, v_2) \in E : v_1 = v\}|.$$



(a) Undirected graph $G = (V, E)$ where $V = \{a, b, c, d, e\}$ and $E = \{e_1 = \{a, b\}, e_2 = \{a, c\}, e_3 = \{b, c\}, e_4 = \{a, d\}\}$



(b) Directed graph $G = (V, E)$ where $V = \{a, b, c, d, e\}$ and $E = \{(a, b), (a, c), (b, c), (a, d)\}$

Figure 2.1: Drawings of graphs

From now on, unless stated otherwise, $n(G)$ will be the number of vertices of the graph G and $m(G)$ the number of edges. Also we always consider vertices and edges in a fixed order.

Definition 2.4 (Complete graphs)**.** *An undirected graph $G = (V, E)$ is **complete** if*

$$E = \{\{v, w\} \subset E \times E : v \neq w\}.$$

*We denote the complete graph with n vertices by $\mathbf{K_n}$. A **complete bipartite graph** is a bipartite graph (see Definition 2.11 with $k = 2$) with bipartition $V = V_1 \cup V_2$, where*

$$E = \{\{v, w\} \subset E \times E : v \in V_1, w \in V_2\}.$$

*We denote the complete bipartite graph with $|V_1| = r$, $|V_2 = s|$ by $\mathbf{K_{r,s}}$. For any k, $\mathbf{K_{1,k}}$ is called **star**, especially $\mathbf{K_{1,3}}$ is called **claw**.*

We are now able to state Euler's famous theorem that was already mentioned in the introduction of this section. A proof can be found, for example, in [KV07, Theorem 2.24].

Theorem 2.5. *Let $G = (V, E)$ be an undirected graph. An **Eulerian walk** is a sequence of vertices $v_1, \ldots, v_r$ such that $\{v_i, v_{i+1}\} \in E$ for all $i = 1, \ldots, r-1$ and each edge of G is crossed exactly once. Then G contains an Eulerian walk if and only if the degree of each vertex is even.*

Definition 2.6 (Cut, [KV07])**.** *A **cut** in an undirected graph G is an edge set of type $\delta(X)$ for some $\emptyset \neq X \subset V(G)$, where*

$$\delta(X) = \{e = (v, w) \in E(G) : v \in X, w \in V(G) \backslash X\}.$$

A hypergraph is similar to a graph, but its edges can contain more than two nodes.

Definition 2.7 (Hypergraph)**.** *A **hypergraph** $\mathbb{H} = (V, E)$ consists of a finite, nonempty set of **vertices** V and a set of **edges** $E \subset 2^V$. Two vertices are called **adjacent** if there exists an edge containing both of them. A vertex and an edge containing it are called **incident**.*

Definition 2.8 (k-Uniform Hypergraph)**.** *A **k-uniform hypergraph** $\mathbb{H} = (V, E)$ (also called **k-graph**) is a hypergraph where each edge consists of exactly k vertices (i.e. $|e| = k, \forall e \in E$).*

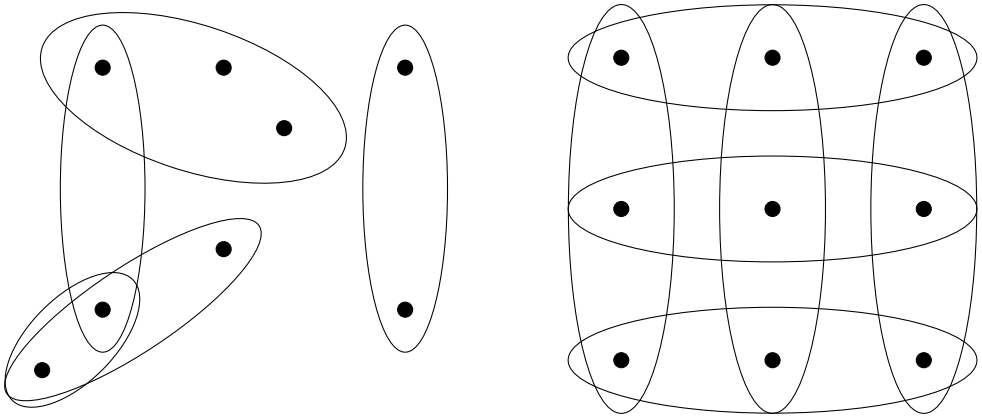
Definition 2.9 (Linear Hypergraph)**.** *A hypergraph is **linear** if no two edges share more than one vertex.*

Definition 2.10 (k-Regular Hypergraph)**.** *Analogue to the graph case the **degree of a vertex** v of a hypergraph H, denoted by $deg(v)$ is the number of edges incident to v. The **neighbors** $N_H(v)$ of a vertex v are defined as*

$$N_H(v) = \{w \in V(H) : v, w \in e \text{ for some } e \in E(H)\}.$$

*A hypergraph is **k-regular** if every node has degree k. We denote by $\Delta(H)$ the maximum degree of a vertex in H.*

Definition 2.11 (k-partite Hypergraph)**.** *A hypergraph is k-partite if V can be partitioned in k sets $V_1, \ldots, V_k$ such that nodes from the same set are never adjacent.*



(a) A hypergraph

(b) A 3-uniform, 2-regular hypergraph

Figure 2.2: Drawings of hypergraphs

Definition 2.12 (Subgraph)**.** *Let $H = (V, E)$ be a hypergraph. A **subgraph** $H' = (V', E')$ is a hypergraph where $V' \subset V$ and $E' \subset E$. It is an **induced subgraph** $H[V']$ (induced by subset of vertices V') if $V' \subset V$ and $E' = E \cap 2^{V'}$.*

Remark: *Let $H = (V, E)$ be a hypergraph and let $v \in V, e \in E$. We often write $H - v$ for the graph $H[V \backslash \{v\}]$ and $H - e$ for $(V, E \backslash \{e\})$ or the analogue if we replace v or e by sets of vertices or edges.*

Definition 2.13 (Paths and Cycles, cf. [KRS13])**.** *A **path** is a hypergraph P with edges $E(P) = \{e_1, \ldots, e_m\}, m \geq 1$ where for every $1 \leq i < j \leq m$, $e_i \cap e_j \neq \emptyset$ if and only if $j = i + 1$. If $m \geq 3$ and in addition $e_1 \cap e_m \neq \emptyset$, then such a k-graph is called a **cycle**.*

Remark: *Note that a pair of edges sharing at least two vertices is still path and not a cycle.*

Definition 2.14. *A hypergraph $H = (V, E)$ is **connected** if for each pair $\{v, w\} \subset V$ there exists a path between v and w. A subgraph of H that is connected is called **(connected) component** of H.*

Definition 2.15 (Intersection Graph)**.** *The **intersection graph** of a hypergraph $H = (V(H), E(H))$ is the graph $G = L(H)$ with vertex set $V(G) = E(H)$ and edge set $E(G)$ consisting of all intersecting pairs of edges of H:*

$$E(G) = \{\{e, f\} \in E(H) \times E(H) : e \cap f \neq \emptyset\}$$

*When H is a graph, the same definition is called **line graph**.*

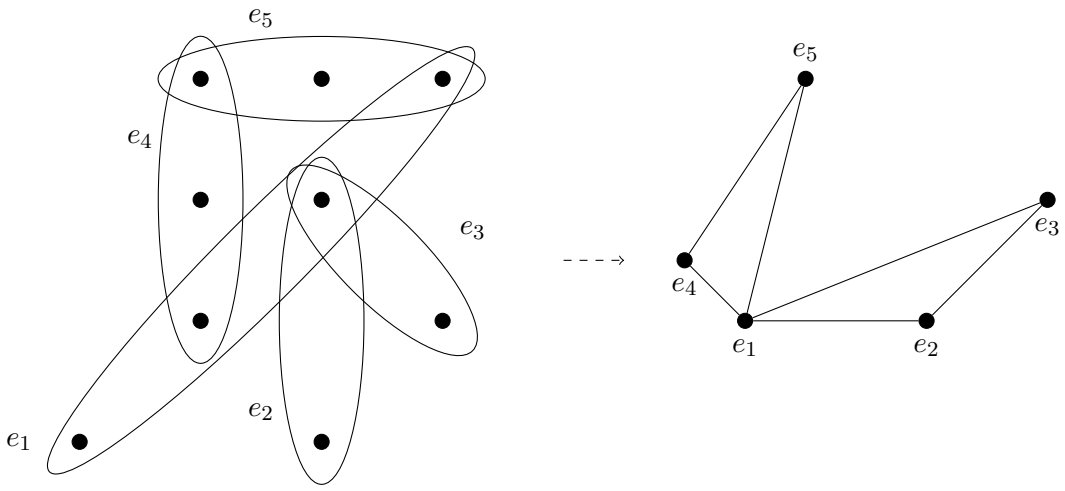


Figure 2.3: An example of a hypergraph (left) and its corresponding intersection graph (right).

Next we define the *dual graph* as the hypergraph with interchanged vertices and edges.

Definition 2.16 (Dual Hypergraph)**.** *Let $H = (V, E)$ be a hypergraph. The **dual** $H^* = (V^*, E*)$ is defined by $V^* = E$ and $E* = \{\{e \in E | v \in e\} : v \in V\}$.*

Definition 2.17 (Matching)**.** *A **matching** M in a graph or hypergraph is a set (possibly empty) of disjoint edges (i.e. $e \cap f = \emptyset, \forall e, f \in M$).*

Remark: *Although we defined subgraphs, components, matchings and the dual only for hypergraphs, it should be noted that analogous definitions for undirected graphs result from the fact that hypergraphs can be seen as generalizations of undirected graphs.*

2.2 Probability Theory

This section introduces the most important definitions from probability theory which are mainly needed in Chapter 5. For this section basic knowledge in measure theory is required.

Definition 2.18 (Probability space)**.** *A probability space $(\Omega, \mathcal{F}, \mathbb{P})$ consist of*

- *a non-empty set called **sample space** Ω*
- *a σ-algebra $\mathcal{F} \subset \mathcal{P}(\Omega)$*
- *a **probability measure** $\mathbb{P}$, i.e. a positive measure $\mathbb{P} : \mathcal{F} \to [0, 1]$ such that*
  1. *$\mathbb{P}(\Omega) = 1$*
  2. *$\mathbb{P}$ is σ-additive: Let $(A_i)_{i \in \mathbb{N}}$*

$$\forall i \neq j : A_i \cap A_j = \emptyset \quad \Rightarrow \quad \mathbb{P}(\cup_{i \in \mathbb{N}} A_i) = \sum_{i \in \mathbb{N}} \mathbb{P}(A_i)$$

Definition 2.19 (Conditional probability)**.** *Let $(\Omega, \mathcal{F}, \mathbb{P})$ be a probability space and $A, B \in \mathcal{F}$ with $\mathbb{P}[B] > 0$, then we define the **probability of** A **conditioned on** B $\mathbb{P}[A|B]$ by*

$$\mathbb{P}[A|B] = \frac{\mathbb{P}[A \cap B]}{\mathbb{P}[B]}.$$

Because we are only considering finite, discrete spaces in this thesis we will from now on assume that $\Omega = \{\omega_1, \ldots, \omega_N\}$ is finite. Also we always chose $\mathcal{F} = 2^\Omega$. In this case the probability distribution of a probability measure can than be described by a mass function $p : \Omega \to [0, 1]$ such that $\mathbb{P}(\{\omega_i\}) = p(\omega_i) =: p_i$.

Definition 2.20 (Random variable)**.** *Let $(\Omega, \mathcal{F}, \mathbb{P})$ be a probability space and $(S, \mathcal{S})$ be a measurable space. A **random variable** X is a mapping $X : \Omega \to S$.*

Remark: *Note that we will later often talk about the distribution of a random X and mean the push-forward measure $\mathbb{P} \circ X^{-1}$.*

Definition 2.21 (Uniform distribution)**.** *Let $\left(\Omega, 2^\Omega\right)$ be a measure space, then the uniform distribution $\frac{1}{|\Omega|}$ is defined by*

$$\mathbb{P}[A] = \frac{|A|}{|\Omega|}, \quad \forall A \in 2^\Omega.$$

Because we will later need to compare different probability measures with each other we introduce the following definition.

Definition 2.22 (Total variation distance of probability measures)**.** *Let $(\Omega, \mathcal{F})$ be a measurable space and $\mathbb{P}$ and $\mathbb{Q}$ probability measures on it. Then we define the* ***total variation distance*** *of $\mathbb{P}$ and $\mathbb{Q}$ by*

$$d_{TV}(\mathbb{P}, \mathbb{Q}) = \sup_{A \in \mathcal{F}} |\mathbb{P}(A) - \mathbb{Q}(A)|.$$

Remark: *Note that for finite Ω, the total variation distance is just the l^1 norm of the corresponding mass functions p and q. Indeed*

$$d_{TV}(\mathbb{P}, \mathbb{Q}) = \frac{1}{2} \sum_{x \in \Omega} |p(x) - q(x)| = \frac{1}{2} \|p - q\|_{l^1}.$$

Definition 2.23 (Stochastic process)**.** *Let $(\Omega, \mathcal{F}, \mathbb{P})$ be a probability space and $(S, \mathcal{S})$ another measurable space. A* ***(discrete-time) stochastic process*** *on a probability space is a family $(X_i)_{i \in \mathbb{N}_0}$ of random variables $X_i : \Omega \to S$. The transition of such a process from X_i to X_{i+1} is called* ***step****.*

Definition 2.24 (Markov chain)**.** *Let $(\Omega, \mathcal{P}, \mathbb{P})$ be a probability space. A stochastic process $(X_n)_{n \in \mathbb{N}_0}$, with values in $(S, \mathcal{S})$, is called* ***Markov chain*** *if it can be described by a* ***transition kernel*** *P with*

$$\forall A \in \mathcal{S} : \ \mathbb{P}[X_{i+1} \in A | X_i, X_{i-1}, \ldots, X_1] = P(X_i, A) \quad \forall i \in \mathbb{N}_0.$$

This means, that that the next state of the process depends only on the current state and not the other previous ones. In the case of discrete S we use the notation $p_{i,j} := P(X_i, \{j\})$.

Definition 2.25 (Properties of Markov chains)**.** *Let $(X_t)_t$ be a discrete Markov chain on $(\Omega, \mathcal{F}, \mathbb{P})$. The chain is*

- ***irreducible*** *if it is possible to get from any state to any other state with positive probability.*
- ***aperiodic*** *if it holds that*

  $$\forall a \in \Omega : \quad gcd(i, j) = 1, \forall i \neq j \in \left\{ h : p^h(a, a) > 0 \right\}$$

  where $gcd(i, j)$ is the greatest common divisor of i and j.
- ***symmetric*** *if for all $i, j \in \Omega$ it holds $p_{i,j} = p_{j,i}$.*

Definition 2.26 (Stationary distribution)**.** *Let $(X_n)_n$ be a Markov chain on $(\Omega, \mathcal{F}, \mathbb{P})$, where Ω is discrete. Let P denote the transition kernel. A distribution with mass function $\pi : \Omega \to [0, 1]$ is called* ***stationary distribution of*** *$(X_n)_n$ if*

$$\sum_{x \in \Omega} \pi(x) P(x, y) = \pi(y), \quad \forall y \in \Omega$$

Theorem 2.27 (from [LPW06, Theorem 4.9]). *Let $(X_n)_n$ be a Markov chain on S. Let X be aperiodic and irreducible, then there exists a unique stationary distribution π and there exist constants $\alpha \in (0,1)$ and $C > 0$ such that*

$$\max_{x \in S} \|P^t(x, \cdot) - \pi\|_{TV} \leq C\alpha^t,$$

where P^t is the distribution of the chain after t steps.

Corollary 2.28. *Let $(X_n)_n$ be a symmetric, aperiodic, irreducible Markov chain, then its distribution converges exponentially fast to the uniform distribution.*

2.3 Computational Complexity Theory

The theory of computational complexity is based on *computational problems*, i.e. questions to be answered by intensive calculations, usually possessing several input parameters and a clear definition of what a solution to this problem is. If we provide all input parameters of a problem with concrete values we get an *instance* of the problem. A step-by-step procedure that produces a solution for every instance of a problem is called an *algorithm*.

To be able to compare different algorithms for the same problem with each other we will define some notions of complexity in this section. We start by defining the Turing machine in Section 2.3.1. It made comparing the complexity of algorithms possible in the first place. Then we introduce the well known decision problems and complexity classes for them in Section 2.3.2. In Subsection 2.3.3 we introduce counting problems (as in 'counting the matchings of a hypergraph'). Finally in contrast to the problem specific complexity classes we define some complexity classes for concrete approximation algorithms in Section 2.3.4.

This section is based on parts of [GJ79] and [Jer03].

Because it does not really fit in any subsection we start here with a mathematical definition that helps us compare different functions with each other. This, so called *Landau notation*, gives expressions like 'g is an upper bound for f' a rigorous meaning.

Definition 2.29 (Landau notation). *Let $f : \mathbb{N} \to \mathbb{N}$ be a function. Then we define the following function classes:*

$$O(f) = \{g : \mathbb{N} \to \mathbb{N} : \exists c > 0\ \exists N_0 \in \mathbb{N}\ \forall n \geq N_0 : g(n) \leq cf(n)\}$$
$$\Omega(f) = \{g : \mathbb{N} \to \mathbb{N} : \exists c > 0 : g(n) < cf(n) \text{ for infinitely many } n\}$$
$$\Theta(f) = O(f) \cap \Omega(f)$$

Remark: *It is common convention to write $f = O(g)$ instead $f \in O(g)$ and the analog for Ω and Θ.*

2.3.1 Turing machines

In this thesis we will often have to compare different algorithms with each other. For this matter we have to define some criteria in which we want to rate the algorithms. Usually one wants to have time-efficient (fast) and memory-efficient algorithms. The problem that arises is that the efficiency of an algorithm depends heavily on the system on which it is ran. If we use modern computers, a faster CPU or more memory it might lead to a faster running time, but also the architecture of the CPU, the voltage or the even the temperature may play a role. This makes results difficult to reproduce and to compare.

A solution to this problems came from Alan Turing (1912 - 1954) who introduced the *(universal) Turing machine*, a theoretical construct that describes a computational machine (a computer if you like), but stripped to its most basic minimum of functionality. It can be proved that the set of problems that can be solved using a Turing machine is equivalent to the one modern computers can solve (see Turing completeness, for example in [Her95]). Thus comparing the Turing machine implementations of different algorithms is a good way to rigorously compare their properties.

We start by introducing some notations that help to encode the problems we want to compute a solution for.

Definition 2.30 (Alphabets / Languages, from [GJ79])**.** *A finite set of symbols Σ is called an **alphabet**. Denote by Σ^* the set of all finite strings of symbols from Σ. A subset $L \subset \Sigma^*$ is called a **language over the alphabet** Σ.*

Remark: *Note that all the problems we are later considering can be encoded in alphabets and languages. However we will not in detail define how, for example, a graph can be encoded or how the subset of matchings looks like.*

We can now define what a *Turing machine* is.

Definition 2.31 (Turing machine, [GJ79, p. 22])**.** *A **(deterministic) Turing machine (DTM)** is specified by the following information:*

- *A finite set Γ of **tape symbols**, including a subset $\Sigma \subset \Gamma$ of **input symbols** and a distinguished **blank symbol** $b \in \Gamma \backslash \Sigma$,*
- *a finite Q of **states**, including a distinguished **start-state** q_0 and non-empty set $F \subset Q$ of **final- or accepting states**,*

- *a* ***transition function***

$$\delta : (Q \backslash F) \times \Gamma \to Q \times \Gamma \times \{-1, +1\}.$$

Remark: *Note that in [GJ79] the authors introduce what we call a DTM as a* ***program*** *for a DTM. For simplicity reasons we will not use this distinction in the following. We also replaced their two state accepting-state-set by the more general set F.*

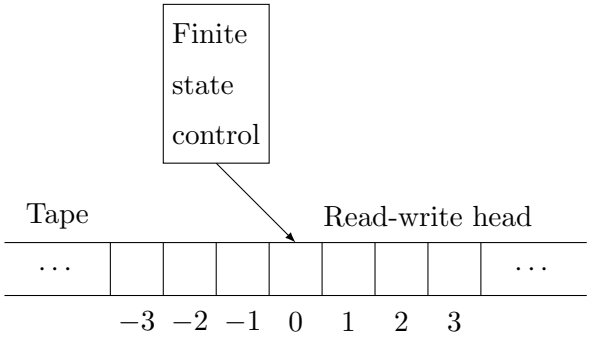


Figure 2.4: Schematic representation of a deterministic one-tape Turing machine (DTM) *(Source: [GJ79])*

The transition function of a DTM maps a state $q_t \in Q \backslash F$ and a tape symbol $\gamma \in \Gamma$ that is currently read by the machine, to a new state $q_{t+1} \in Q$, a symbol to be written to the tape, and an operation $o \in \{-1, +1\}$ (moving the head of the Turing machine either left or right). This way one can define the behaviour of an algorithm just by specifying the transition function. In conclusion a DTM represents a powerful computing machine, but with it's easy structure makes it possible to define what the complexity of an algorithm is. For this matter we define one *step* as one application of the transition function. The number of steps needed to get to an accepting state can be used as a measure for the time-complexity of the algorithm on a particular instance of the problem.

Definition 2.32 (Time-complexity of a Turing machine)**.** *Let M be a Turing machine, then we define the* ***time-complexity function*** *$T_M : \mathbb{N}_0 \to \mathbb{N}_0$ of M as*

$$T_M(n) = \max \left\{ m : \begin{array}{l} \text{there exists input } x \in \Sigma^*, \text{ with } |x| = n, \text{ such that} \\ \text{the computation of } M \text{ on input } x \text{ takes } n \text{ steps} \end{array} \right\}$$

Definition 2.33 (Polynomial-time Turing machine)**.** *We say a Turing machine is* ***polynomial-time*** *if there exists a polynomial p such that*

$$\forall n \in \mathbb{N}_0 : T_M(n) \leq p(n).$$

By modifying the transition function to a transition relation, the deterministic Turing machine becomes a *non-deterministic Turing machine (NDTM).*

Definition 2.34 (Non-deterministic Turing machine)**.** *A **non-deterministic Turing machine (NDTM)** is specified by the following information:*

- *A finite set Γ of **tape symbols**, including a subset $\Sigma \subset \Gamma$ of **input symbols** and a distinguished **blank symbol** $b \in \Gamma \backslash \Sigma$,*
- *a finite Q of **states**, including a distinguished **start-state** q_0 and non-empty set $F \subset Q$ of **final- or accepting states**,*
- *a **transition relation***

$$\delta : (Q \backslash F) \times \Gamma \times Q \times \Gamma \times \{-1, +1\}.$$

The difference between a deterministic and a non-deterministic Turing machine is that a NDTM basically decides randomly which next transition to take where in the deterministic case the transition function defined the behaviour exactly. A NDTM is much more powerful in the sense that it can solve problems much faster. This is because we define the time a NDTM takes so solve an instance of a problem as the time the machine takes in the best-case scenario, where the machine made perfect guesses on which transition to take to minimize the time for the whole computation. This interpretation immediately reveals that a NDTM is just a theoretical construct and nothing that can actually be build. A computer, for example, works deterministically in that it has to be precisely specified which next calculation to do for every situation possible.

2.3.2 Decision problems and NP-Completeness

In this section we introduce *decision problems* as the first fundamental class of problems. Also the famous complexity classes **P** and **NP** categorize decision problems.

Definition 2.35 (Decision problem)**.** *A **decision problem** $\mathcal{P}$ is a pair (Σ, L) where Σ is an alphabet and $L \subseteq \Sigma^*$ is a language over Σ. An instance of $\mathcal{P}$ is an element $x \in \Sigma^*$ and the solution of x is 'Yes', if $x \in L$ and 'No' else.*

The following two definitions give an example of a decision problem.

Definition 2.36. *A **Hamiltonian cycle** is a sequence of vertices that starts and ends at the same vertex, where two following vertices form an edge of G and where each vertex is exactly visited once.*

Problem 2.37 (HAMILTON − CYCLE)**.**

Input: *A graph G.*

Output: *Does G contain a Hamiltonian cycle?*

We will now make the connection between Turing machines and decision problems.

Definition 2.38. *A Turing machine M with $F = \{$ Yes , No $\}$ **accepts** a language $L \subset \Sigma^*$ if for all $x \in \Sigma^*$*

$$M(x) = \text{ Yes } \quad \Leftrightarrow \quad x \in L,$$

where $M(x)$ is the final state after the Turing machine calculated input x.

Now we can define polynomial algorithms and the complexity class **P**.

Definition 2.39 (Class P)**.** *We define the complexity class* **P** *as*

$$\mathbf{P} = \left\{ \mathcal{P} = (\Sigma, L) : \begin{array}{c} \mathcal{P} \text{ is a decision problem and there exists a} \\ \text{polynomial-time DTM which accepts } L \end{array} \right\}.$$

Analog to the deterministic case we define the time-complexity of a NDTM and it's polynomial running time. We can now define the complexity class **NP**.

Definition 2.40 (Class **NP**)**.** *The complexity class* **NP** *is defined as*

$$\mathbf{NP} = \left\{ \mathcal{P} = (\Sigma, L) : \begin{array}{c} \mathcal{P} \text{ is a decision problem and there exists a} \\ \text{polynomial-time NDTM which accepts } L \end{array} \right\}.$$

It's immediately clear that $\mathbf{P} \subset \mathbf{NP}$. The other inclusion forms one of the most important unsolved problems in modern mathematics, the so called $\mathbf{P} = \mathbf{NP}$ - problem. So far there are many problems, called **NP**-complete problems, for which there is still no polynomial algorithm. This could be because $\mathbf{P} \subsetneq \mathbf{NP}$, but a proof is still missing. What makes the $\mathbf{P} = \mathbf{NP}$ - problem even more interesting is, that because of the polynomial relation between **NP**-complete problems, finding a polynomial algorithm for one NP-complete problem would imply such an algorithm for all of them.

To formalize the notion of **NP**-completeness we need to first define *polynomial-time reduction.*

Definition 2.41 (Polynomial-time reduction)**.** *Let Σ_1, Σ_2 be alphabets and $L_1 \subset \Sigma_1^*, L_2 \subset \Sigma_2^*$ be languages. L_1 is **polynomial-time reducible** to L_2 if and only if there exists a DTM that calculates in polynomial time a function $f : \Sigma_1^* \to \Sigma_2^*$ for every $x \in \Sigma_1^*$ such that*

$$x \in L_1 \Leftrightarrow f(x) \in L_2.$$

In particular, this implies that a polynomial algorithm that accepts one of these languages can be easily transformed into one that accepts the other language.

Remark: *If a problem* PROBLEM − A *is polynomial-time reducible to another problem* PROBLEM − B *we write*

$$\text{PROBLEM} - \text{B} \leq \text{PROBLEM} - \text{A}.$$

Definition 2.42 (NP-completeness)**.** *A problem $\mathcal{P} \in$* **NP** *is called* **NP-*complete***, *if every other problem in* **NP** *is polynomial-time reducible to $\mathcal{P}$.*

Remark: *If a problem is not necessarily in* **NP** *but all problems in* **NP** *can be polynomial-time reduced to it, then we call this problem* **NP-*hard***. *There are indeed problems that are even 'more complex' than* **NP***-complete, for example the* ***Halting problem***. *(for more information see [GJ79, Chapter 5])*

There is another class in between **P** and **NP** called **RP** which stands for *randomized polynomial* and which adds a probabilistic component to the Turing machine. Basically **RP** consists of decision problems for which there exists an algorithm of polynomial running time with the following property: It always returns 'NO' if the correct solution is 'NO', but it returns 'YES' only with probability $\frac{1}{2}$ if the correct solution is 'YES' otherwise it also returns 'NO'. As the rigorous definition with probabilistic Turing machine is quite long we refer the reader to [AB09] for a detailed introduction.

2.3.3 Counting problems and the class #P

In 1979 L. Valiant [Val79a] introduced the equivalent class to **NP** for counting problems which cannot be computed in polynomial time. The following definitions come mostly from [Val79a].

Definition 2.43 (Counting problem)**.** *A* ***counting problem*** *$\mathcal{P}$ is a pair (Σ, f) of an alphabet Σ and a function $f : \Sigma \to \mathbb{N}$. An algorithm solves the problem if it can compute $f(x)$ for all $x \in \Sigma^*$.*

Note that a counting problem is a function problem (i.e. for each input, the output has to be computed) and not a decision problem (only 'Yes' or 'No' possible). Thus counting problems are fundamentally different from decision problems. However, as the following example shows, many decision problem can be easily transformed into counting problems and vice versa.

Problem 2.44 (#HAMILTON − CYCLES)**.**
Input: *A graph G.*
Output: *How many Hamiltonian cycles does G contain?*

Remark: *In the following definitions we speak of Turing machines that can compute a value $f(x)$. Of course this is not possible with the Turing machine introduced in the previous section. That is because it has no output tape. Instead with a slight abuse of notation we are now talking about Turing machines which additionally have a (write-only) output tape to display the result of a calculation. All other operations of TMs stay the same.*

The class for counting problems analogue to **P** is denoted by **FP**.

Definition 2.45 (Class **FP**)**.** *A counting problem $\mathcal{P}$ is in* **FP***, if it is computable by a deter-*

ministic, polynomial-time Turing machine.

Of course if there is an analogue class to **P** one would also expect an analog class to **NP**.

Definition 2.46 (Class #**P**, cf. [Val79a])**.** *The class* #**P** *consists of the counting problems* (Σ, f) *for which there exists a non-deterministic Turing machine which computes* $f(x)$ *in polynomial time for all* $x \in \Sigma^*$*.*

Definition 2.47 (#**P**-completeness)**.** *A problem* $\mathcal{P}$ *is* #**P**-***complete*** *if it is in* #**P** *and every other problem of* #**P** *can be polynomial-time reduced to* $\mathcal{P}$*.*

Remark: *Although counting problems in* #**P** *are naturally related to decision problems in* **NP***, there are problems in P for which counting is* #**P***-hard. For example, counting perfect matchings in bipartite graphs is in* #**P** *while determining if the exists a perfect matching is in* **P***. [Val79a]*

2.3.4 Approximations

Similar to the decision problems, many counting problems of interest are #**P**-complete. Because their complexity grows exponentially with the size of the instance, exact counting is in practice often not possible, simply because it would take too long. The next best thing we can do is try to construct an algorithm that approximates the solution. To characterize the quality of an approximation, there also exist complexity classes for algorithms that distinguish good approximation algorithms from bad ones.

We start with the deterministic approximations.

Definition 2.48 (FPTAS, cf. [DKRS14])**.** *A* ***fully polynomial time approximation scheme (FPTAS)*** *for a function* f *on* Σ^* *is a deterministic algorithm which for every pair* (ε, x) *with* $\varepsilon > 0$ *and* $x \in \Sigma^*$*, return a number* $y(x)$ *such that*

$$|y(x) - f(x)| \leq \varepsilon f(x),$$

and runs in time polynomial in $1/\varepsilon$*, and* $|x|$*.*

Now we weaken this definition by only requiring a good approximation with a certain probability and we get the following definition.

Definition 2.49 ((ε, δ)-approximation, [KRS13])**.** *Given any fixed* $\varepsilon, \delta > 0$*, a random variable* Y *is a* (ε, δ)***-approximation*** *of a constant* C*, if*

$$\mathbb{P}\left[|Y - C| \geq \varepsilon C\right] \leq \delta$$

Definition 2.50 (FPRAS, [KRS13])**.** *An randomized algorithm is a* ***fully polynomial randomized approximation scheme (FPRAS)*** *for a function* f *on* Σ^**, if for every triple* (ε, δ, x)

with $\varepsilon > 0, \delta > 0$ *and* $x \in \Sigma^*$*, the algorithm returns an* (ε, δ)*-approximation* Y *of* f *and runs in time polynomial in* $1/\varepsilon$*,* $\log(1/\delta)$ *and* $|x|$*.*

The are also weaker definitions, where for example the approximation has to be only up to a certain constant precision (APX) or where its running time does not have to be polynomial in $1/\varepsilon$ (PTAS). Because we will not use them in this thesis we wont rigorously define these classes here. Also, in Chapter 4 we will go a bit more into the differences of randomized and deterministic approximations.

3 Counting Matchings in Hypergraphs

In this chapter we will introduce the counting problem we are analysing in this thesis: counting the number of matchings in hypergraphs. The definition of this problem and its relation to counting independent sets are described in section 3.1. In Section 3.2 we introduce some interesting subclasses of hypergraphs so that we break down the counting problem in smaller pieces which can be analysed separately. The next two Sections 3.3, 3.4 are the foundation for our further analysis. In Section 3.3 we will prove the hardness of exact counting of matchings. In Section 3.4 we introduce some preliminary results on the approximation hardness and clarify which approximation results we can hope for.

3.1 Counting matchings and its relation to independent sets

With Definition 2.17 we already introduced the notion of matchings in a hypergraphs. Naturally the problem of counting matchings in a hypergraph is then defined in the following way.

Problem 3.1 (#MATCHINGS)**.**
Input: *A hypergraph H.*
Output: *The number of matchings H does contain. We set*

$$\#\text{MATCHINGS}(H) = |\mathcal{M}(H)| = \left|\left\{ M \subset E(H) : M \text{ is a matching} \right\}\right|.$$

It turns out that counting matchings is closely related to another counting problem, namely the counting of independent sets.

Definition 3.2. *Let G be a graph or hypergraph. We call a set $I \subset V(G)$ an **independent set** (or **stable set**) if*

$$\forall v, w \in I \nexists e \in E(G) : v, w \in e.$$

This means that no two vertices in an independent set are allowed to be connected by an edge. Analog to the #MATCHINGS - problem we introduce the $\#\text{INDEPENDENT} - \text{SETS}$ - problem.

Problem 3.3 ($\#\text{INDEPENDENT} - \text{SETS}$)**.**
Input: *A graph or hypergraph G.*
Output: *The number of sets $I \subset V(G)$ such that I is an independent set.*

The relation between the two problems is shown in the following Lemma.

Lemma 3.4. *Let H be a hypergraph and $L(H)$ its corresponding intersection graph, then the number of matchings in H is equivalent to the number of independent sets in $L(H)$.*

Proof. Recalling that the nodes of $L(H)$ (representing edges of H) are only connected if the corresponding hyperedges intersect, we get that independent sets of $L(H)$ represent matchings in H. □

Indeed many following results in this chapter come from this relation between counting independent sets and counting matchings.

3.2 Hypergraph classes of interest

Because we are going to show in Sections 3.3 and 3.4, the #**P**-completeness and inapproximability of counting matchings in arbitrary hypergraphs, it makes sense to restrict ourselves to smaller subsets of hypergraphs. This way we can split the big problem into minor small ones. To be able to easily compare different results, in this section we introduce some general notations for graphs and hypergraphs.

First set $\mathcal{H}$ to be the set of all hypergraphs. The first thing we can do to narrow down the hypergraphs - in which we are counting the matchings in - is to only consider k-uniform hypergraphs. We denote this class by $\mathcal{H}(k)$. The restriction on this class gives more information about the structure of its intersection graph, namely that it has no induced copy of $K_{1,k+1}$.

Lemma 3.5. *Let $H \in \mathcal{H}(k)$ and denote by $G = L(H)$ its corresponding intersection graph. Then G does not contain an induced copy of $K_{1,k+1}$. We call G* ***k+1-claw-free****.*

Proof. Suppose G contains an induced copy of $K_{1,k+1}$, then there exists an edge in H intersecting $k+1$ other edges which are pairwise disjoint (do not intersect). This is a contradiction because, as H is k-uniform, each edge contains exactly k vertices and thus at least two of the $k+1$ other edges have to intersect each other. □

Restricting ourselves further gives the set $\mathcal{H}(k,r)$ of hypergraphs $H \in \mathcal{H}(k)$ with $\Delta(H) \leq r$ (maximum degree at most r). Hypergraphs in $\mathcal{H}(k,r)$ are called (k,r)-*graphs*. Furthermore in Chapter 5 we are going to define the set of hypergraphs without structures called *3-combs* by $\mathcal{H}_0$, and analogue $\mathcal{H}_0(k)$ for k-graphs without 3-combs and so on. As this restriction is of less general importance we give the precise definition only in Chapter 5 (see Definition 5.1).

For the special case of graphs we denote the set of all 2-uniform hypergraphs (graphs) by $\mathcal{G}$ and the set of graphs with degree at most r by $\mathcal{G}(r)$. For the sets of r-regular graphs or k-graphs we write $\mathcal{G}(=r)$, respectively $\mathcal{H}(k,=r)$.
For the restriction of problems to a certain class of hypergraphs we write, for example,

$$\#\text{MATCHINGS}\big(\mathcal{H}(k,r)\big)$$

for the problem of counting the number of matchings in (k,r)-graphs.

The above classes of hypergraphs set the scene for our analysis. We will try to step by step show #**P**-completeness, inapproximability or existence of an efficient approximation algorithm. Figure 3.1 is a visualization of the relations that we will later use to map our results.

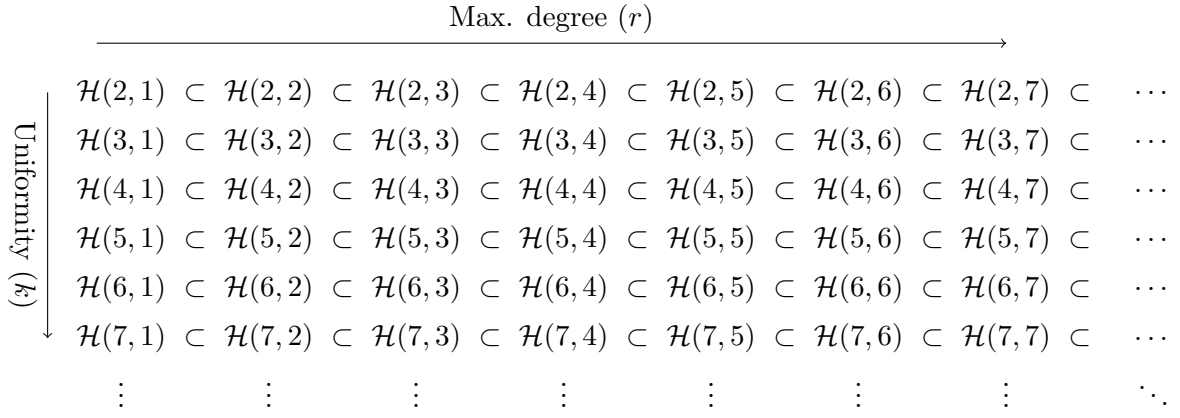


Figure 3.1: A map of hypergraph subclasses we will consider in this thesis

3.3 #P-completeness of exact counting

In this section we prove the #**P**-completeness for the #MATCHINGS - problem on almost all subclasses of hypergraphs we introduced in the previous section. This then satisfies why we are only considering approximation algorithms in the ongoing of this thesis. The main result of this section is from [Gre00] while a preliminary result comes from [Val79b].

The earliest hardness result related to our problem comes from [Val79b] by L. Valiant in 1979. It can be seen as a follow-up of his famous article 'The complexity of computing the permanent' (cf. [Val79a]), where he introduced the complexity class #**P**. He proved the #**P**-completeness for a bunch of problems, including the following.

Problem 3.6 (IMPERFECT MATCHINGS)**.**
Input: *A bipartite graph with* $2n$ *nodes.*
Output: *The number of matchings of any size.*

However, this implies that counting matchings in arbitrary graphs is #**P**-complete. Now since graphs are k-uniform hypergraphs (with $k = 2$), we have that #MATCHINGS is also #**P**-complete for arbitrary hypergraphs. As a result we restrict ourselves to the above mentioned subclasses of hypergraphs and to approximate counting instead of exact counting.

The #**P**-completeness for most of the these subclasses comes from a result of C. Greenhill in 2000. In [Gre00] she proved the following theorem.

Theorem 3.7 (cf. [Gre00, Theorem 3.1])**.** $\#\mathrm{INDEPENDENT} - \mathrm{SETS}\big(\mathcal{G}(= 3)\big)$ *is* #**P***-complete.*

We can now do a reduction to our problem.

Proposition 3.8.

$$\#\mathrm{INDEPENDENT} - \mathrm{SETS}\big(\mathcal{G}(= k)\big) \leq \#\mathrm{MATCHINGS}\big(\mathcal{H}(k, 2)\big)$$

Proof. Let $G \in \mathcal{G}(= k)$ and let $H := G^*$ be the dual hypergraph (cf. Definition 2.16). Then because each vertex of G has degree equal to k, each edge in H contains exactly k vertices and thus $H \in \mathcal{H}(k)$. Also because edges in G contain two vertices, the vertices in H have degree at most two, thus $H \in \mathcal{H}(k, 2)$. By construction we now have $V(H) = E(G)$ and each edge of the hypergraph $e_v \in E(H)$ corresponds to the edges incident to $v \in V(G)$ in G. In conclusion the number of independent sets in G is equal to the number of matchings in H. Thus $\#\mathrm{MATCHINGS}\big(\mathcal{H}(k, 2)\big)$ has to be at least as hard as $\#\mathrm{INDEPENDENT} - \mathrm{SETS}\big(\mathcal{G}(= k)\big)$. □

Now combining Theorem 3.7 and Proposition 3.8 we get the #**P**-completeness for

$$\#\mathrm{MATCHINGS}\big(\mathcal{H}(3, 2)\big).$$

In order to expand the result to $(k, 2)$-graphs with $k \geq 3$ we prove the following Lemma.

Lemma 3.9.

$$\#\mathrm{MATCHINGS}\big(\mathcal{H}(k, r)\big) \leq \#\mathrm{MATCHINGS}\big(\mathcal{H}(k + 1, r)\big)$$

Proof. Let $H = (V, E) \in \mathcal{H}(k, 2)$. Now to each edge $e_i \in E$ add a new vertex v_i^{new} such that $e_i^{new} = e \cup \{v_e\}$, set $E^{new} = \bigcup_{i=1}^{m} \{e_i^{new}\}$. Then $H^{new} = (V \cup \bigcup_{i=1}^{m} \{v_i^{new}\}, E^{new})$ is in $\mathcal{H}(k+1, r)$ and matchings in H are matchings in H^{new}. Thus $\#\mathrm{MATCHINGS}\big(\mathcal{H}(k + 1, r)\big)$ is at least as hard as $\#\mathrm{MATCHINGS}\big(\mathcal{H}(k, r)\big)$. □

Remark: *Because* $\mathcal{H}(k, r) \subset \mathcal{H}(k, r + 1)$ *we also have*

$$\#\mathrm{MATCHINGS}\big(\mathcal{H}(k, r)\big) \leq \#\mathrm{MATCHINGS}\big(\mathcal{H}(k, r + 1)\big).$$

Combining Theorem 3.7, Proposition 3.8 and Lemma 3.9 we get the final result.

Corollary 3.10. $\#\mathrm{MATCHINGS}\big(\mathcal{H}(k, r)\big)$ *is* #**P***-complete for* $k \geq 3$ *and* $r \geq 2$.

Note that $\mathcal{H}(k,1), k \geq 1$ are graphs without intersecting edges. Thus each such hypergraph is by itself a matching which makes counting them naturally easy, i.e. in polynomial time. A visual comprehension of the hardness results so far is displayed in Figure 3.2.

$\mathcal{H}(2,1)$	$\subset$	$\mathcal{H}(2,2)$	$\subset$	$\mathcal{H}(2,3)$	$\subset$	$\mathcal{H}(2,4)$	$\subset$	$\mathcal{H}(2,5)$	$\subset$	$\mathcal{H}(2,6)$	$\subset$	$\mathcal{H}(2,7)$	$\subset$	$\cdots$
$\mathcal{H}(3,1)$	$\subset$	$\mathcal{H}(3,2)$	$\subset$	$\mathcal{H}(3,3)$	$\subset$	$\mathcal{H}(3,4)$	$\subset$	$\mathcal{H}(3,5)$	$\subset$	$\mathcal{H}(3,6)$	$\subset$	$\mathcal{H}(3,7)$	$\subset$	$\cdots$
$\mathcal{H}(4,1)$	$\subset$	$\mathcal{H}(4,2)$	$\subset$	$\mathcal{H}(4,3)$	$\subset$	$\mathcal{H}(4,4)$	$\subset$	$\mathcal{H}(4,5)$	$\subset$	$\mathcal{H}(4,6)$	$\subset$	$\mathcal{H}(4,7)$	$\subset$	$\cdots$
$\mathcal{H}(5,1)$	$\subset$	$\mathcal{H}(5,2)$	$\subset$	$\mathcal{H}(5,3)$	$\subset$	$\mathcal{H}(5,4)$	$\subset$	$\mathcal{H}(5,5)$	$\subset$	$\mathcal{H}(5,6)$	$\subset$	$\mathcal{H}(5,7)$	$\subset$	$\cdots$
$\mathcal{H}(6,1)$	$\subset$	$\mathcal{H}(6,2)$	$\subset$	$\mathcal{H}(6,3)$	$\subset$	$\mathcal{H}(6,4)$	$\subset$	$\mathcal{H}(6,5)$	$\subset$	$\mathcal{H}(6,6)$	$\subset$	$\mathcal{H}(6,7)$	$\subset$	$\cdots$
$\mathcal{H}(7,1)$	$\subset$	$\mathcal{H}(7,2)$	$\subset$	$\mathcal{H}(7,3)$	$\subset$	$\mathcal{H}(7,4)$	$\subset$	$\mathcal{H}(7,5)$	$\subset$	$\mathcal{H}(7,6)$	$\subset$	$\mathcal{H}(7,7)$	$\subset$	$\cdots$
$\vdots$		$\vdots$		$\vdots$		$\vdots$		$\vdots$		$\vdots$		$\vdots$		$\ddots$
$\in \mathbf{P}$		#**P-complete** (Corollary 3.10)												

Figure 3.2: The complexity of counting matchings in certain subclasses of hypergraphs

3.4 (In-)approximability results

After in the previous section we showed the hardness of exact counting matchings for almost all hypergraphs, in this section we will state some results concerning the approximability of the exact number of matchings. Interestingly, contrary to the exact counting there are really some differences between the different subclasses, in the sense that, for example, for some hypergraphs there do not even exist FPRASs while for other there are even FPTASs. This also confirms that our splitting in subclasses of hypergraphs was a good choice. The results in this chapter come from [LV99], [Wei07], [Sly10], [SS12] and [KRS13].

A first result came from M. Luby and E. Vigoda [LV99] in 1999 when they constructed an FPRAS for $\#\mathrm{INDEPENDENT} - \mathrm{SETS}\big(\mathcal{G}(4)\big)$. In 2007 D. Weitz proved the existence of an FPTAS even for the more general set $\mathcal{G}(5)$. In his proof he used a new method called *correlation decay* which we will further explain in Chapter 4 and apply in Chapter 6.

Proposition 3.11 (cf. [Wei07])**.** *There exists an FPTAS for* $\#\mathrm{INDEPENDENT} - \mathrm{SETS}\big(\mathcal{G}(5)\big)$.

Again using this FPTAS on the intersection graph of a hypergraph and using Lemma 3.4 we get the following corollary.

Corollary 3.12. *There exists an FPTAS for* $\#\mathrm{MATCHINGS}\big(\mathcal{H}(k,2)\big)$ *for* $k \in \{3,4,5\}$.

Now recently this result was complemented by results in [Sly10], [SS12] by A. Sly resp. A. Sly and N. Sun. They proved that the number of independent sets in graphs with higher degree

than five are not efficiently approximable.

Proposition 3.13 (cf. [Sly10], [SS12])**.** *Unless* **NP** = **RP** *there exists no FPRAS for*

$$\#\text{INDEPENDENT} - \text{SETS}\big(\mathcal{G}(=6)\big).$$

Using the reduction from Proposition 3.8 this leads again to a corresponding result for the number of matchings.

Proposition 3.14. *For every $k \geq 6$ unless* **NP** = **RP**, *there is no FPRAS for the number of matchings in a 2-regular, linear k-graph.*

Remark: *Note that in Proposition 3.8 the dual hypergraph of a graph is linear.*

Remark: *Note that from the fact that no FPRAS can exists for a problem follows that no FPTAS can exists, but not the other way around.*

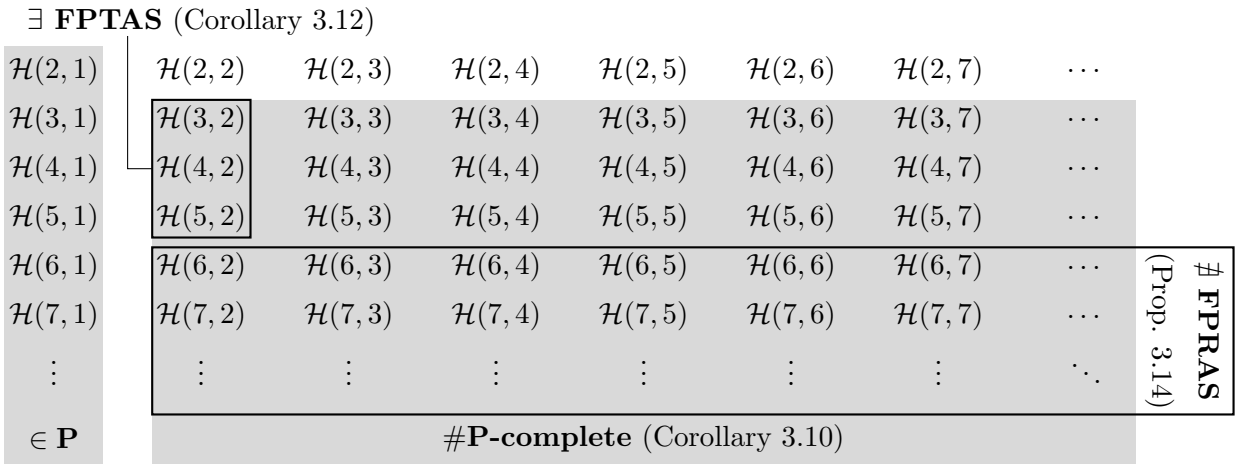


Figure 3.3: The extended complexity map with added (in-)approximability results from section 3.4

4 Methods for constructing approximation algorithms

In the two chapters after this one we are going to construct approximation algorithms for the problem of counting matchings in hypergraphs. The methods we are going to use are not limited to this special case but can also be used to construct approximation algorithms for other problems. In this chapter we will therefore describe two general methods on how deterministic and probabilistic approximation algorithms can be constructed or - to put it another way - how it can be proved that these algorithms are efficient (i.e. a FPTAS or FPRAS). First in Section 4.1 we explain the relation between counting and uniform sampling. In Section 4.2 we describe a method for constructing a FPRAS using Markov chains which will be applied in Chapter 5. In Section 4.3 we explain the recently developed [Wei07] deterministic *correlation decay method* used in Chapter 6.

4.1 The relation of approximate counting and uniform sampling

The underlying principle of many approximate counting algorithms is the fact that uniform sampling is almost as good as counting. This is meant in the sense that if we are able to efficiently sample according to the uniform distribution on the state space of the problem, this gives also an efficient approximate counting algorithm. Indeed this is not really surprising because uniform sampling means that each realization (possible value) of a random variable appears with the same probability. If now the space in which the random variable takes values is the space for which we want to count the elements, then naturally from the probability we can derive the number of items by taking the inverse. To formalize this idea consider the following definitions.

Definition 4.1 (ε-uniform distribution, cf. [KRS13])**.** *Let $(\Omega, 2^\Omega, \mathbb{P})$ be a probability space where Ω is the finite sample space. Let $\varepsilon > 0$ be fixed. We call the distribution of $\mathbb{P}$, ε-**uniform** if*

$$\forall S \subseteq \Omega, \quad \left| \mathbb{P}(S) - \frac{|S|}{|\Omega|} \right| < \varepsilon,$$

or equivalently $d_{TV}\left(\mathbb{P}, \frac{1}{|\Omega|}\right) < \varepsilon$.

Definition 4.2 (FPAUS, cf. [KRS13, Definition 2]). *A randomized algorithm is called a* ***fully polynomial almost uniform sampler (FPAUS)*** *for a counting problem $\mathcal{P} = (\Sigma, f)$ with $f(x) = |\Omega(x)|$, if for every pair (ε, x) with $\varepsilon > 0$ and $x \in \Sigma^*$, the algorithm samples $\omega \in \Omega(x)$ according to an ε-uniform distribution P and runs in polynomial time in $\frac{1}{\varepsilon}$ and $|x|$.*

A rigorous proof - at least for the problem of counting matchings - that indeed a almost uniform sampler can be used to count the items in Ω was done by M. Jerrum in [Jer03].

Proposition 4.3 (cf. [Jer03, Proposition 3.4]). *Let G be a graph with n vertices and m edges, where $m \geq 1$ to avoid trivialities. If there is an almost uniform sampler on the set of matchings $\mathcal{M}(G)$ with running time bounded by $T(n, m, \varepsilon)$, then there is a randomized approximation scheme for $|\mathcal{M}(G)|$ with running time bounded by $cm^2\varepsilon^{-2}T(n, m, \frac{\varepsilon}{6m})$, for some constant c. In particular, if there is an FPAUS for $\mathcal{M}(G)$ then there is an FPRAS for $|\mathcal{M}(G)|$.*

Although the proof is done only for the graph case, the hypergraph case follows immediately because the structure of the edges is nowhere used in the proof. An even more general proof for so called *self-reducible* counting problems can be found in [Jer85]. As a result we can now concentrate on uniform sampling instead of the counting problem itself.

4.2 Rapidly mixing Markov chains and the canonical paths method

Using randomized algorithms often makes sense if a problem can not be efficiently solved and not even efficiently approximated deterministically. Then the best approach can be to use a randomized algorithm which gives a good approximation only with a certain probability. In the many possible steps of relaxing the conditions required for a solution of a problem, allowing randomized approximations is often the last hope for getting any efficient algorithm at all. Nevertheless good randomized algorithms (FPRAS) are in practice often good enough. If, for example, the approximation quality (low probability of big errors) depends polynomially on the number of steps, it may be efficient to run the algorithm until we can be sure we get a good enough approximation.

4.2.1 Rapidly mixing Markov chains

In this section we motivate the use of Markov chain for the construction of a FPRAS. The definitions are again taken from [KRS13].

A common way to construct a randomized approximation algorithm is done with the help of an ergodic Markov chain (cf. Definition 2.24). The state space of the chain is usually the underlying configuration space of the problem. Also the chain is defined such that the stationary distribution is of interest for the problem (usually the uniform distribution). This can be done relatively

easy and the beauty of an ergodic Markov chain is that it converges towards this stationary distribution, giving an approximation for the problem. This principle is the base for many **NP** and #**P**-complete problems that are otherwise not efficiently solvable. However, to make this procedure an effective algorithm requires that the chain is approaching its stationary distribution fast (enough). In the following we define 'how fast a Markov chain is approaching its stationary distribution' as *mixing time*. Markov chains that converge fast to their stationary distribution are called *rapidly mixing*.

Definition 4.4 (Mixing time of a Markov chain)**.** *The* ***mixing time*** *of an ergodic Markov chain M on $(\Omega, \mathcal{F})$ is defined as*

$$t_{mix}(\varepsilon) = \min\{t : d_{TV}(P_t, P_\infty) \leq \varepsilon\}$$

where d_{TV} is the total variation distance, P_t the distribution of the Markov chain after t steps and P_∞ the stationary distribution of the Markov chain.

This definition does indeed make sense, because measures with little total variation distance can be considered 'close', in the sense that a certain configuration has similar probabilities in P_t and P_∞ (see Definition 2.22). The difficulty in using Markov chains to construct a FPRAS is usually not in the definition of the chain but in the fact that one has to prove a polynomial upper bound for the mixing time in order to really get a FPRAS. One way to prove this polynomial bound is the canonical paths method described in Subsection 4.2.2. It utilizes the *transition graph* of the Markov chain.

Definition 4.5 (Transition graph of a Markov chain)**.** *Let $\mathcal{MC}$ be a Markov chain on a probability space $(\Omega, \mathcal{F}, \mathbb{P})$ with finite Ω. The* ***transition graph*** *$G_{\mathcal{MC}} = (V, E)$ is defined by $V(G_{\mathcal{MC}}) = \Omega$ and*

$$E(G_{\mathcal{MC}}) = \{i, j \in \Omega : p_{i,j} > 0\}.$$

Remark: *At this point we want to especially emphasize that the transition graph should not be mixed up with the underlying graph or hypergraph in which we are counting the matchings. The transition graph exists for all kind of Markov chains. It is just a visualization of how the Markov chain behaves on its state space and which transitions might happen.*

Naturally one might feel that the speed by what a Markov chain converges to its stationary distribution is closely related to 'how well the transition graph is connected'. This is meant the sense of how likely it is to get from an arbitrary start-vertex to another arbitrary end-vertex. This is the core idea of the following definition.

Definition 4.6 (Conductance of symmetric Markov chain)**.** *Let $\mathcal{MC}$ be a symmetric Markov*

chain. The ***conductance*** *is defined by*

$$\Phi(\mathcal{MC}) = \min_{S\subset\Omega, 0<|S|<\frac{1}{2}|\Omega|} \frac{\sum_{\{i,j\}\in cut(S)} p_{i,j}}{|S|}.$$

Proposition 4.7 (cf. [JS89b, Theorem 2.2])**.** *It holds*

$$d_{TV}\left(P_t, \frac{1}{|\Omega|}\right) \leq |\Omega|^2 \left(1 - \frac{\Phi^2}{2}\right)^t, \tag{4.1}$$

and thus

$$t_{mix}(\varepsilon) \leq \frac{2}{\Phi^2}(2\log|\Omega| + \log\varepsilon^{-1}). \tag{4.2}$$

Idea of the proof. For the proof one can identify the transition probabilities with a stochastic matrix. The mixing time is closely related to the eigenvalues of this matrix. Now this eigenvalues can be bounded in terms of the conductance of the transition graph which then lets us conclude (4.1). A detailed proof can be found in [JS89a, Section 3]. □

In particular, this proposition implies that now we can focus on bounding the conductance from below in order to bound the mixing time from above. Define

$$p_{min} = \min\{p_{i,j} : \{i,j\} \in G_{\mathcal{MC}}, i \neq j\},$$

then because p_{min} is the smallest possible transition probability we have

$$\sum_{\{i,j\}\in\mathrm{cut}(S)} p_{i,j} \geq |\mathrm{cut}(S)| p_{min},$$

and thus

$$\Phi(\mathcal{MC}) \geq \min_{S\subset\Omega, 0<|S|<\frac{1}{2}|\Omega|} \frac{p_{min}|\mathrm{cut}(S)|}{|S|}. \tag{4.3}$$

If we assume that p_{min} can be bounded from below then, in order to get a bound for the conductance $\Phi(\mathcal{MC})$, we will only have to bound $|\mathrm{cut}(S)|$ from below. This is where the canonical paths method of Subsection 4.2.2 comes into play.

4.2.2 The canonical paths method in general

Recall that we are examining the transition graph $G_{\mathcal{MC}}$ and we want to polynomially bound $|\mathrm{cut}(S)|$ from below for every $S \subset V(G_{\mathcal{MC}})$. For this matter we want to use the *canonical paths method*. It was first introduced by M. Jerrum and A. Sinclair (cf. [JS89b]) and it consists of two steps:

1. Define a *canonical path* in $G_{\mathcal{MC}}$ for every pair (ω_I, ω_F) of elements in the state space Ω. That is a path deterministically defined only by (ω_I, ω_F) and which can always be reconstructed by just knowing ω_I and ω_F.

2. Bound from above the number of canonical paths containing a fixed edge of $G_{\mathcal{MC}}$ by $poly(n)|\Omega|$.

To understand why this helps bounding $|cut(S)|$, note that a path in $G_{\mathcal{MC}}$ from an element of S to an element of its complement $V(G_{\mathcal{MC}})\backslash S$, has to go through an edge of $cut(S)$. For

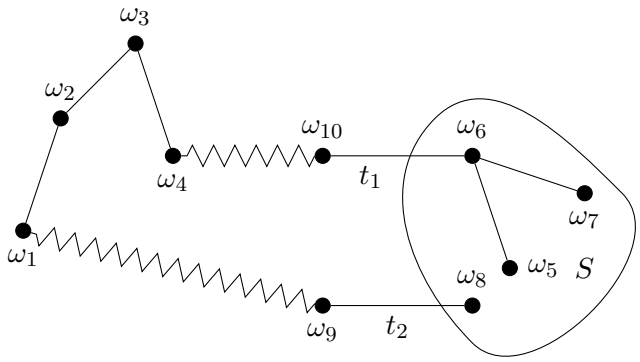


Figure 4.1: Example of a transition graph with canonical paths, where $\omega_1, \ldots, \omega_{10}$ are some states of the Markov chain. Here $S = \{\omega_5, \omega_6, \omega_7, \omega_8\}$ and $cut(S) = \{t_1, t_2\}$. In general we bound $|cut(S)| = |\{t_1, t_2\}|$ by bounding the number of paths that cross an arbitrary edge and using that $|S||\Omega\backslash S|$ paths have to cross an edge of $cut(S)$.

$|S| \leq \frac{1}{2}|\Omega|$ we then have

$$|cut(S)| \geq \frac{|S|(|\Omega| - |S|)}{poly(n)|\Omega|} \geq \frac{|S|}{2poly(n)}. \tag{4.4}$$

Indeed the first inequality follows because there are $|S|(|\Omega|-|S|)$ paths from S to $V(G_{\mathcal{MC}}) = \Omega\backslash S$ and no edge is allowed to have more than $poly(n)|\Omega|$ paths crossing it.

Using these two steps of the canonical paths method we will in Chapter 5 prove that the Markov chain defined in Section 5.2 is a FPAUS.

4.3 Correlation decay

A more recent method that can be used to construct an efficient approximation algorithm is *(spatial) correlation decay*. It was introduced by D. Weitz [Wei07] in 2007 for the problem of counting independent sets in a graph (see Proposition 3.11). Shortly after this, D. Gamarnik and D. Katz [GK07] slightly modified the method to be applicable to more general problems. In this section we summarize the important ideas of [Wei07] and [GK07] and use them to give an idea how correlation decay works in general. A rigorous application to the problem of counting matchings in hypergraphs follows in Chapter 6.

4.3.1 Weitz's approach using spatial correlation decay

For some *activity* $0 < \lambda < \lambda_c$ Weitz considers weighted independent sets I in a graph $G = (V, E)$ with weights proportional to $\lambda^{|I|}$. The *partition function* is the sum over all these weights of independent sets: $Z \equiv Z_G^\lambda = \sum_I \lambda^{|I|}$. Thus for $\lambda = 1$, Z is just the number of independent sets. The interest lies in calculating Z or sampling independent sets according to $\frac{\lambda^{|I|}}{Z}$.

The core idea of correlation decay is to look at the marginal distribution of a single vertex v (i.e. the probability that a vertex is part of a uniformly drawn independent set)

$$p_v \equiv p_{G,v}^\lambda = \frac{\sum_{v \in I} \lambda^{|I|}}{Z_G^\lambda}.$$

We then want to show that it is more or less independent of the condition that another set $\Lambda \subset V$ is in the independent set, unless Λ is close to v. Weitz formalizes this idea in his definitions of (weak/strong) spatial mixing.

Definition 4.8 (Weak spatial mixing, [Wei07, Definition 2.1])**.** *Let $\delta : \mathbb{N} \to \mathbb{R}^+$. We say that the distribution over independent sets of $G = (V, E)$ with activity parameter λ exhibits **weak spatial mixing** with rate $\delta(\cdot)$ if and only if for every $v \in V, \Lambda \subset V$, and any two configurations σ_Λ, τ_Λ specifying independent sets of Λ,*

$$|p_v^{\sigma_\Lambda} - p_v^{\tau_\Lambda}| \leq \delta(\mathrm{dist}(v, \Lambda)),$$

where $\mathrm{dist}(v, \Lambda)$ *stands for the graph distance (the length of the shortest path) between the vertex v and the subset Λ.*

Naturally this property seems useful for the construction of a fast approximation algorithm because, as Weitz [Wei07] points out:
"In statistical physics the graph G is usually an infinite graph [...] and weak [spatial] mixing with rate δ that goes to zero is equivalent to the uniqueness of the Gibbs measure, i.e. to the existence of a unique macroscopic equilibrium."
Weitz's approach to show this spatial correlation decay for independent sets uses a clever transformation of the graph problem to the tree of *self-avoiding walks*. This transformation implies that the probability of a vertex to be in an independent set in G is the same as the probability for the root of the tree to be in an independent set. However for the tree it can be easily show that the correlation between two vertices decays exponentially.

4.3.2 Using a recursive computation tree

A disadvantage of Weitz's approach is the fact that his transformation of the graph to a tree works only on two-valued models (f.e. a vertex can only be in an independent set or not). Also the approximations of the marginal probabilities are not efficient enough, for example, for a

approximate counting algorithm for the number of matchings (cf. [BGK+07, Chapter 5]). As a result, D. Gamarnik and D. Katz [GK07] use a slightly different approach where the 'tree-like-structure' does not come as a result of a transformation of the graph. Instead it comes from a recursive relation of the probabilities similar to recursions that appear in dynamic programming.

We take an example application of this method from [BGK+07] where M. Bayati et al. constructed a simple deterministic approximation algorithm for counting matchings in graphs and which was the motivation for the proof in the hypergraph case explained in Chapter 6. Bayati et al. first derive the recursive relation for the probability that a vertex v is in a random matching $\mathbf{M}$, sampled uniformly from the set of all matchings.

Proposition 4.9 (cf. [BGK+07, Proposition 3.1])**.** *The following holds for every vertex v:*

$$\mathbb{P}_G(v \notin \mathbf{M}) = \frac{1}{1 + \lambda \sum_{u \in N_G(v)} \mathbb{P}_{G \setminus \{v\}}(u \notin \mathbf{M})}.$$

[...]

Then one can define a similar looking, approximating function Φ_G depending on an additional (time) parameter t.

Definition 4.10 ([BGK+07])**.** *For every subgraph $\hat{G}$ of the graph G, every vertex $v \in \hat{G}$ and every $t \in \mathbb{Z}_+$ we introduce a quantity $\Phi_{\hat{G}}(v,t)$ computed inductively as follows.*

1. *$\Phi_{\hat{G}}(v,0) = 1$ for all $\hat{G}$, $v \in V$.*
2. *For every $t \geq 1$,*

$$\Phi_{\hat{G}}(v,t+1) = \frac{1}{1 + \lambda \sum_{u \in N_{\hat{G}}(v)} \Phi_{\hat{G} \setminus \{v\}}(u,t)}.$$

At first this approximation seems bad in the sense that, because of the sum over recursive function calls, we still have to evaluate the expression for exponentially many subgraphs of G. The solution to this problem comes from correlation decay. After we have iterated the recursion a few steps, the correlation between the initial vertex v and the neighbours of its neighbours of its neighbors (and so on) is so low that we can stop and still get a good enough approximation of the probability $\mathbb{P}(v \notin \mathbf{M})$. Bayati et al. prove the following statement.

Theorem 4.11 (cf. [BGK+07, Theorem 3.2])**.** *The following holds for every vertex v and every possible even value t:*

$$\left| \log \mathbb{P}_G(v \notin \mathbf{M}) - \log \Phi_G(v,t) \right| \leq \left(1 - \frac{2}{\sqrt{1+\lambda\Delta}+1}\right)^{\frac{t}{2}} \log(1+\lambda\Delta)$$

where $\Delta = \Delta(G)$ the max. degree of a vertex in G.

We will carry out the details in Chapter 6 where we will do the proof for the hypergraph case.

5 A FPRAS for counting matchings in hypergraphs without 3-combs

After in the previous chapter we described the idea of how a Markov chain can be used for an approximation algorithm, in this chapter we will give a detailed application for the problem of counting matchings in hypergraphs.
First in Section 5.1 we will describe the further assumption on the set of hypergraphs that is needed for us to prove that the algorithm is indeed fast (enough) approximating the exact number of matchings. In Section 5.2 we will define the Markov chain that is the heart of our algorithm. Next in Section 5.3 we apply the canonical paths method described in Subsection 4.2.2 to show that the define Markov chain is rapidly mixing. Finally in section 5.4 we draw conclusions and again summarize where we stand at this point.

This chapter is mostly based on [KRS13] by M. Karpinski, A. Ruciński and E. Szymańska where the proof is taken from.

5.1 Further assumptions on the hypergraphs

In this section we introduce some additional assumptions on the hypergraphs. These assumptions are needed to show that the Markov chain we are going to use is rapidly mixing.

Definition 5.1 (3-comb, cf. [KRS13])**.** *We call a hypergraph a **3-comb** if consists of a matching* $\{e_1, e_2, e_3\}$ *and one extra edge* e_4 *such that* $|e_4 \cap e_i| \geq 1$ *for* $i = 1, 2, 3$.

Definition 5.2 (Wide edge, cf. [KRS13])**.** *Call an edge **wide** if it intersects a matching in* H *of size at least three.*

Remark: *Note that every 3-comb contains a wide edge. That is how the two definitions are related.*

We denote hypergraphs without 3-combs by $\mathcal{H}_0$ and hypergraphs with at most s wide edges by $\mathcal{H}_s$. Analog, in combination with our previous notation, we define $\mathcal{H}_0(k), \mathcal{H}_s(k, r)$ and so on.

The reason why such a restriction is needed in the first place can probably be best explained

by looking at the transition graph of the Markov chain. As we mentioned in the last chapter, it is important that the transition graph is 'well connected', i.e. that it is easy to get from one end to the other. The Markov chain we are going to define in the next section will only allow transitions (from one matching to another) that add, remove or exchange exactly one edge. Now if a hypergraph contains many 3-combs the transition graph of the Markov chain is not well connected in the sense that there are areas that are connected with each other only by very few transitions. For example, let M be the matching consisting of the edges e_1, e_2, e_3 of a 3-comb (cf. Definition 5.1) and let M' be the matching consisting only of e_4. Then in order to get from M to M' using transitions of our Markov chain, first all edge e_1, e_2, e_3 have to be removed before e_4 can be added to the matching. The probability for this to happen is very low and thus naturally it feels like the chain might take a while to reach the stationary distribution (in this case visit each matching equally often).

However because these restricted hypergraphs are rather sparse (of size $O(n^{k-1})$ [KRS13]), we have to check that this counting problem is still #**P**-complete. To show this, we use a result from [Vad02] where S. Vadhan proved several #**P**-completeness results for sparse graphs (graphs with bounded degree).

Lemma 5.3 (cf. [Vad02, Theorem 4.1])**.** *Let $G \in \mathcal{G}(4)$ be bipartite. Then* #MATCHINGS$\big(G\big)$ *is* #**P***-complete.*

We use this to prove the following proposition.

Proposition 5.4 (cf. [KRS13, Proposition 1])**.** *Let $H \in \mathcal{H}_0(k, 4)$ be linear and k-partite. Then* #MATCHINGS$\big(H\big)$ *is* #**P***-complete for every $k \geq 3$.*

Remark: *Note that the proof is very similar to the proof of Lemma 3.9.*

Proof. We use the result of Lemma 5.3 by doing a polynomial reduction of the old problem to the current problem, which then implies #**P**-completeness and finishes the proof.
Let $G = (V, E) \in \mathcal{G}(4)$ be bipartite and denote the bipartition by V_1, V_2 (i.e. $V = V_1 \cup V_2$). Let $k \geq 3$ be fixed. We now transform G to a k-graph $H_{new} = (V_{new}, E_{new}) \in \mathcal{H}(k, 4)$ by the following steps (see Fig. 5.1):

1. For every edge $e \in E$ add $k-2$ vertices, i.e. $V_{new} = V \cup \bigcup_{e \in E} \{v_1^e, \ldots, v_{k-2}^e\}$
2. Transform each $e \in E$ to a new edge $e_{new} = (v, v_1^e, \ldots, v_{k-2}^e)$ in E_{new}

As a result we have

$$|V_{new}| = |V| + (k-2)|E| \quad \text{and} \quad |E_{new}| = |E|.$$

Also note that H_{new} is linear, k-partite and has maximum degree at most 4 and contains no 3-combs. It is also clear that each matching in G has a corresponding matching in H_{new} and thus

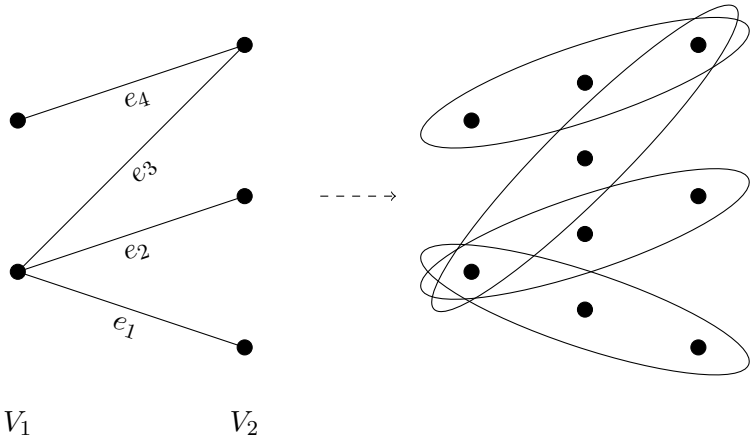


Figure 5.1: Example transformation of a bipartite graph to its 3-graph counterpart

a polynomial algorithm for counting matchings in H yields a polynomial algorithm for counting matchings in G. Now using the result from Lemma 5.3 finishes the proof. □

We are now in the position to state our main theorem.

Theorem 5.5 (cf. [KRS13, Theorem 1])**.** *For every $k \geq 3$ and $s \geq 0$ there exists an FPRAS for the problem of counting matchings in a hypergraph $H \in \mathcal{H}_s(k)$.*

Remark: *Note that we will first prove the Theorem for the case $H \in \mathcal{H}_0(k)$. In the end we will explain how the result can be generalized.*

5.2 Construction of the Markov chain

In this subsection we will first define the Markov chain taking values in the set of matchings. A way to define a Markov chain that has a certain stationary distribution is using a *Metropolis(-Hastings)* chain, named after N. Metropolis and W. Hastings. The idea behind their method is to sample the next state of the Markov chain in 2 steps: First proposing a next state according to an arbitrary but fixed proposal kernel Q and then in a second step accepting the state transition with a certain acceptance probability. By choosing the acceptance probability correctly one can always attain a certain stationary distribution. We will do something similar to construct the Markov chain on the set of matchings of a hypergraph.

Definition 5.6. *Let $H = (V, E)$ be a k-graph and $\Omega(H)$ the set of all matchings of H. Define a Markov chain $\mathcal{MC}(H) = (X_t)_{t=0}^{\infty}$ on $\Omega(H)$ by the following procedure.*

1. *Sample an edge with respect to the uniform distribution, i.e. $h \in E \sim \frac{1}{|\Omega(H)|}$.*
2. *Let $X_t = M = \{h_1, \dots h_s\}$ and $I_h := \{i : h \cap h_i \neq \emptyset,\ i = 1, \dots, s\}$ the set of edges intersected by h. Define a new matching M' considering the following cases:*

(-) *if* $h \in M$, *then* $M' := M - h$

(+) *if* $h \notin M$ *and* $|I_h| = 0$, *then* $M' := M + h$

(+/-) *if* $h \notin M$ *and* $I_h = \{j\}$, *then* $M' := M + h - h_j$

(0) *if* $h \notin M$ *and* $|I_h| \geq 2$, *then* $M' := M$

3. *With probability* $\frac{1}{2}$ *set* $X_{t+1} = M'$, *else set* $X_{t+1} = X_t$.

In the remainder of this section we show some properties of this Markov chain.

Lemma 5.7. *The Markov chain* $\mathcal{MC}(H)$ *is ergodic and symmetric.*

Proof. For the symmetry consider two different matchings $M, M' \in \Omega(H)$, then by the definition of our Markov chain the transition probability is given by

$$P_{M,M'} = \begin{cases} \frac{1}{2|H|} & \text{if } |M \oplus M'| = 1 \\ \frac{1}{2|H|} & \text{if } M \oplus M' = \{e, f\}, e \cap f \neq \emptyset \\ 0 & \text{otherwise} \end{cases}$$

Thus $P_{M,M'} = P_{M',M}$, which proves the symmetry. For the ergodicity note that $\mathcal{MC}(H)$ is aperiodic (because the chain can stay in a matching) and irreducible (because we can get from any initial matching to any matching target matching). Then by Theorem 2.27 the chain is ergodic. □

As a direct result of Corollary 2.28 we get the following statement.

Corollary 5.8. *The stationary distribution of* $\mathcal{MC}(H)$ *is the uniform distribution.*

Moreover we have that p_{min} is bounded from below (this was needed for the canonical paths method to work). Indeed

$$p_{min} = \min\left\{P_{M,M'} : \{M, M'\} \in G_{\mathcal{MC}}, M \neq M'\right\} = \frac{1}{2|H|} \geq n^{-k},$$

where the last inequality holds because there are n^k possibilities to take a k-tuple from a set of n distinct objects. This corresponds to the maximal number of of hyperedges in a k-graph.

We have now everything ready to go into the proof that this Markov chain is indeed rapidly mixing.

5.3 Proof that the Markov chain is rapidly mixing using the canonical paths method

5.3.1 Definition of the canonical paths

In the following subsection we will carry out step one of the above described canonical paths method: the definition of the canonical paths. For this matter we will use a structural property of hypergraphs without 3-combs.

Let $H = (V, E) \in \mathcal{H}_0(k)$ be the hypergraph for which we want to count the number of matchings. Let the set of nodes be enumerated, $V(H) = \{1, 2, \ldots, n\}$ and let $\min S = \min\{i : i \in S\}$ for any $S \subset V(H)$. Also a fix an initial matching $I \in V(G_{\mathcal{MC}})$ and a final matching $F \in V(G_{\mathcal{MC}})$. We will in the following define a canonical path $\gamma(I, F)$ in $G_{\mathcal{MC}}$ from I to F.

Note that for the symmetric difference

$$I \oplus F = \left\{ e \in E(G_{\mathcal{MC}}) : (e \in I \wedge e \notin F) \vee (e \notin I \wedge e \in F) \right\}$$

we have $\Delta(I \oplus F) \leq 2$. Indeed because I and F are matchings, vertices in $I \oplus F$ can not have more than two incident edges. Due to the assumption that $H \in \mathcal{H}_0(k)$ we also have $\Delta(L(I \oplus F)) \leq 2$. This means that no edge of $I \oplus F$ intersects more than two edges. Hence each component of $I \oplus F$ is either a path or a cycle (cf. Definition 2.13) and each cycle has an even number of edges. We can use this fact to define the canonical path from I to F by traversing these paths or cycles while adding or removing edges and in this way transforming I to F.
Let $Q_1, \ldots, Q_q$ be the ordered components of $I \oplus F$ so that

$$\min V(Q_1) < \cdots < \min V(Q_q). \tag{5.1}$$

Let $M_0, \ldots, M_t$ be the steps of the canonical path (i.e. $\gamma(I, F) = (M_0, \ldots, M_t)$). Because the path starts from I, we set $M_0 = I$. Recall that consecutive matchings M_i, M_{i+1} can only differ by a transition **(+)**, **(-)**, **(+/-)** (see Definition 5.6). We will now traverse the components Q_1 up to Q_q in a well-defined way, so that we transform the initial matching I 'step-by-step' into the final matching F.

Lets assume that so far we already traversed the components $Q_1, \ldots, Q_{r-1}$ and constructed corresponding matchings $M_0, \ldots, M_j$. Let $Q = Q_r$ be the next component to be traversed. Q is either an even path (a path where the number of transitions is even), an odd path or an even cycle and we will give a construction for each of this cases.
Case 1 (Q is an even path):
Because Q is a path, there are two possible start points. Chose e_1 (a vertex of $Q \subset G_{\mathcal{MC}}$ and

thus an edge of H) as the edge to start with, such that $e_1 \in F$. Follow the path, enumerating $Q = \{e_1, \ldots, e_s\}$, then because Q is even, $e_s \in I$. Now recursively iterate the sequence of already defined matchings $M_0, \ldots, M_j$ by

$$\begin{aligned} M_{j+1} &= M_j + e_1 - e_2, \\ M_{j+2} &= M_{j+1} + e_3 - e_4, \\ &\vdots \\ M_{j+\frac{s}{2}} &= M_{j+\frac{s}{2}-1} + e_{s-1} - e_s. \end{aligned}$$

Case 2 (Q is an odd path):

As in case 1, because Q is a path, there are two possible start points. Chose e_1 such that $\min(e_1 \cap e_2) < \min(e_{s-1} \cap e_s)$. Because the path is odd, we have either $e_1, e_s \in I$ or $e_1, e_s \in F$. If $e_1, e_s \in I$ define

$$\begin{aligned} M_{j+1} &= M_j - e_1, \\ M_{j+2} &= M_{j+1} + e_2 - e_3, \\ M_{j+3} &= M_{j+1} + e_4 - e_5, \\ &\vdots \\ M_{j+\frac{s+1}{2}} &= M_{j+\frac{s-1}{2}} + e_{s-1} - e_s, \end{aligned}$$

if $e_1, e_s \in F$ set

$$\begin{aligned} M_{j+1} &= M_j + e_1 - e_2, \\ M_{j+2} &= M_{j+1} + e_3 - e_4, \\ &\vdots \\ M_{j+\frac{s-1}{2}} &= M_{j+\frac{s-3}{2}} + e_{s-2} - e_{s-1}, \\ M_{j+\frac{s+1}{2}} &= M_{j+\frac{s-1}{2}} + e_s. \end{aligned}$$

Case 3 (Q is a cycle):

If Q is a cycle component we could start in any point and in two directions. To make the selection of a starting point unique, we chose $e_1, e_2, \ldots, e_s$ such that

$$e_1 = \min(V(Q) \cap V(I)) \quad \text{and} \quad \min(e_2 \cap e_3) > \min(e_{s-1} \cap e_s).$$

The sequence of transitions is then constructed as

$$\begin{aligned} M_{j+1} &= M_j - e_1, \\ M_{j+2} &= M_{j+1} + e_2 - e_3, \\ M_{j+3} &= M_{j+2} + e_4 - e_5, \\ &\vdots \\ M_{j+\frac{s}{2}} &= M_{j+\frac{s}{2}-1} + e_{s-2} - e_{s-1}, \\ M_{j+\frac{s}{2}+1} &= M_{j+\frac{s}{2}} + e_s. \end{aligned}$$

Remark: *Note that for each component $Q_i \subset G_{\mathcal{MC}}$ we chose a starting point e_1 (which corresponds to an edge of H) and sometimes a direction. These choices are quite arbitrary, but for the uniqueness of the canonical path they are important.*

To make it easier to spot the differences resulting from a transition (M_j, M_{j+1}), we call a component $Q_r \subset H$ the *venue* of the transition (M_j, M_{j+1}) if $M_j \oplus M_{j+1} \subseteq E(Q_r)$.

Lemma 5.9. *By construction the obtained path $\gamma(I, F) = (M_0, \ldots, M_t)$ is unique and has the following properties:*

1. *$M_0 = I$ and $M_t = F$,*
2. *$\{M_j, M_{j+1}\}$ is always an edge of $G_{\mathcal{MC}}$ for $0 \leq j < t$,*
3. *$I \cap F \subseteq M_j \subseteq I \cup F$ for every $0 \leq j \leq t$,*
4. *for every $0 \leq j \leq t$, we have $F \cap \bigcup_{i=1}^{r-1} Q_i \subseteq M_j$ and $I \cap \bigcup_{i=r+1}^{q} Q_i \subseteq M_j$, where Q_r is the venue of (M_j, M_{j+1})*

Proof. The uniqueness follows from the fact that we precisely specified: the order in which we traverse the components (in (5.1)), the starting point and the direction for each component (see cases 1 to 3). Together with the fact that the components can only be paths or cycles, this results in an unique way of traversing $I \oplus F$ and thus the path $\gamma(I, F)$ is unique for each $I, F \in \Omega(H)$.
The properties 1 and 2 directly follow from the construction of the canonical paths. Property 3 holds because during the construction we only add and remove edges in $I \oplus F$. Edges in $I \cap F$ are in $M_0 = I$ and are not touched afterwards.
Finally property 4 perfectly summarizes the construction process. After the traversing of $Q_i, i = 1, \ldots, r-1$ all edges of $F \cap \bigcup_{i=1}^{r-1} Q_i$ are already in the matching M_j and are never modified again. Also edges $I \cap \bigcup_{i=r+1}^{q} Q_i$ have not yet been looked at and they are still in M_j. □

5.3.2 Bounding the Cuts

In this subsection we carry out step two of the canonical paths method. We want to bound $|cut(S)|$ using the canonical paths defined in the previous subsection. We will do so by bounding the number of canonical paths going through an arbitrary fixed transition and then using (4.4). Call this fixed transition (M, M') and let

$$\Pi_{M,M'} = \{(I, F) : (M, M') \in \gamma(I, F)\}$$

be the starting and ending matchings for which the connecting canonical path goes through (M, M'). Also define

$$\Omega_0(H) = \{H' \subseteq H : \exists e \in H' \text{ such that } H' - e \in \Omega(H)\}$$

as the hypergraphs which can be transformed into an matching of H by removing one edge. Note that

$$|\Omega_0(H)| \leq |\{(M,e) : M \in \Omega(H), e \in H\}| \leq n^k |\Omega(H)| \tag{5.2}$$

and $\log|\Omega(H)| = O(n \log n)$. We now want to show, that $|\Pi_{M,M'}| \leq |\Omega_0(H)|$. We will do so by defining a function $\eta_{M,M'} : \Pi_{M,M'} \to \Omega_0(H)$ and showing that it is injective. Then $\Omega_0(H)$ has to have at least as many elements as $\Pi_{M,M'}$. For fixed $(I,F) \in \Pi_{M,M'}$ define

$$\eta_{M,M'}(I,F) = (I \oplus F) \oplus (M \cup M'), \tag{5.3}$$

then the following Lemma holds.

Lemma 5.10. *For all $(I,F) \in \Pi_{M,M'}$ we have $\eta_{M,M'}(I,F) \in \Omega_0(H)$.*

Proof. Because $(I,F) \in \Pi_{M,M'}$, the canonical path $\gamma(I,F) = (M_0, \ldots, M_t)$ contains a consecutive pair $M_j = M$, $M_{j+1} = M'$ for some $j \in \{0, \ldots, t\}$. Let Q_r be the component which is the venue of (M,M') on $\gamma(I,F)$. Now note that by the construction of the canonical paths (similar to property 4 of Lemma 5.9) we have that

$$\eta_{M,M'}(I,F) \cap \bigcup_{i=1}^{r-1} Q_i = I \cap \bigcup_{i=1}^{r-1} Q_i \quad \text{and} \quad \eta_{M,M'}(I,F) \cap \bigcup_{i=r+1}^{q} Q_i = F \cap \bigcup_{i=r+1}^{q} Q_i.$$

Thus $\eta_{M,M'}(I,F) \cap \bigcup_{i=1}^{r-1} Q_i$ and $\eta_{M,M'}(I,F) \cap \bigcup_{i=r+1}^{q} Q_i$ are matchings and the only part of $\eta_{M,M'}(I,F)$ that may not be a matching is $\eta_{M,M'}(I,F) \cap Q_r$. The only way that $\eta_{M,M'}(I,F) \cap Q_r$ is not a matching is if Q_r is a cycle and $M' = M + e_l - e_{l+1}$ for some $l \in \{2,4,\ldots,s-2\}$, because then $e_1, e_s \notin M_j \cup M_{j+1}$ and thus two incident edges are in $\eta_{M,M'}(I,F) \cap Q_r$. But in this case we can just delete edge e_1 and get $\eta_{M,M'}(I,F) - e_1 \in \Omega(H)$, hence $\eta_{M,M'}(I,F) \in \Omega_0(H)$. □

Lemma 5.11. *The mapping $\eta_{M,M'}(I,F) : \Pi_{M,M'} \to \Omega_0(H)$ is injective.*

Proof. This lemma can be proved by showing that from a given η in the image of $\eta_{M,M'} : \Pi_{M,M'} \to \Omega_0(H)$¡ we can recover the pair (I,F) such that $\eta = \eta_{M,M'}(I,F)$. This is only possible if $\eta_{M,M'} : \Pi_{M,M'} \to \Omega_0(H)$ is injective.

Reversing the definition of $\eta_{M,M'}$ in (5.3) yields

$$I \oplus F = \eta \oplus (M \cup M').$$

From property 3 of Lemma 5.9 we have that $I \cap F = M \backslash (I \oplus F)$ and thus so far we can restore $I \cup F = (I \oplus F) \cup (I \cap F)$. It remains to distinguish between the edges of $I \oplus F$ which belong to I and to F. Recall that we ordered the Q_i such that

$$\min V(Q_1) < \cdots < \min V(Q_q).$$

This ordering can be restored by performing the same calculation again. Also note that the venue component Q_r can be found by locating $M \oplus M'$. Now since we know by property 4 of Lemma 5.9 that $Q_i \cap M \subseteq F$ for every $i < r$ and $Q_i \cap M \subseteq I$, we just have to reconstruct I and F on Q_r. Note that because of the alternating pattern of I and F on Q_r identifying one edge with I or F is enough to traverse the whole component. To this end notice that

$$|M \backslash M'| \leq 1 \quad \text{and} \quad |M' \backslash M| \leq 1.$$

Now the construction of the canonical paths implies that if $M \backslash M' = \{e\}$ then $e \in I$ and if $M \backslash M' = \emptyset$ then the unique edge in $M' \backslash M$ is in F. □

Now to put everything together recall that we had to polynomially bound the mixing time of the Markov chain defined in Section 5.2. For this matter in (4.2) we showed that it is bounded by the conductance of the transition graph which itself was bounded in (4.3) by $|cut(S)|$. Finally in this section we defined canonical paths between each pair of matchings I and F and bounded the number of this paths crossing an arbitrary edge by a polynomial term (5.2). Now using the inequality 4.4 we have a polynomial lower bound for $|cut(S)|$ and thus a polynomial upper bound for the mixing time. This finishes the proof for the case where there are no 3-combs in H. In the following subsection we will explain how this proof can be generalized to the case where there are at most s wide edges.

5.3.3 The general case $s > 0$

The presence of wide edges in H makes the proof more complicated, because the proof for the special case relied on the fact that the intersection graph $L(I \oplus F)$ has an easy structure. Indeed we heavily used that components of $I \oplus F$ are either paths or cycles. Now with wide edges, $L(I \oplus F)$ can have vertices with degrees up to k. Nevertheless with a fixed threshold for the number of wide edges we can modify the proof to still result in a FPRAS.

For this matter consider the same Markov chain $\mathcal{MC}(H)$ as before and again order the components Q_i of $I \oplus F$ in the same way. Recall that in the special case without 3-combs the components were either paths or cycles which made it easy to think of an exact way to define the canonical paths. Now in the general case again focus on a certain component Q_r of $I \oplus F$. We artificially construct a similar situation as before by replacing each edge e_k of Q_r with a (graph) cycle C_k. The resulting component S_r is called the *skeleton graph* of component Q_r. Again, to be clear, S_r is a graph (not only a hypergraph) and we will only use it to define the order in which to construct the canonical path of edges of the hypergraph H. Now note that because I and F are matchings and because we are looking at $I \oplus F$, each vertex of S_r has either degree two (if only one of the matchings I, F contains this vertex) or four (if the two matchings

are intersecting in this vertex). This means we can apply Euler's theorem (Theorem 2.5) and obtain an Eulerian tour $E_r = (e_1^{S_r}, \dots, e_s^{S_r})$ consisting of edges of S_r.

To construct the canonical path $\gamma(I, F)$ in the transition graph $G_{\mathcal{MC}}$ we will trace the tours E_r, $r = 1, \dots, q$ and add and / or remove the hyperedges we are passing through to the matching until we reach the final matching. To this end, for a fixed component Q_r define the start vertex v_0 of S_r as the vertex with the smallest indicator. Next we chose a direction in the following way.

1. If $\deg_{S_r}(v_0) = 4$ then there exist intersecting edges $g \in I$ and $f \in F$ such that $v_0 \in f \cap g$. Then as the first edge of the Eulerian tour E_r take (v_0, w), where w is the smaller of the two neighbors on S_r which are in g.

2. If $\deg_{S_r}(v_0) = 2$ and $v_0 \in g \in I$, then chose (v_0, w) as in case 1.

3. If $\deg_{S_r}(v_0) = 2$ and $v_0 \in g \in F$, then chose the first edge of E_r as (v_0, w), where w is the smaller of the two neighbors of v_0 on S_r (which are in f).

Now that the start vertex and the direction of each Eulerian tour $E_1, \dots, E_s$ is specified, we can define the canonical path $\gamma(I, F)$. Again fix E_r and assume we already traversed the first $l-1$ edges of E_r and the current matching of the transition path $\gamma(I, F)$ is M_{j-1}. We go on by considering one of two cases.

1. When $e_l \subseteq g \in I$, then if $g \in M_{j-1}$ set $M_j = M_{j-1} - g$, otherwise if $g \notin M_{j-1}$ do nothing.

2. For the case $e_l \subseteq f \in F$ denote by $I_f = \{h_1, \dots, h_m\}$ the edges in M_{j-1} intersecting with f. Now if already $f \in M_{j-1}$ do nothing, but if $f \notin M_{j-1}$, then set

$$\begin{aligned} M_j &= M_{j-1} - h_1, \\ M_{j+1} &= M_j - e_2, \\ &\vdots \\ M_{j+m-2} &= M_{j+m-3} - h_{m-1}, \\ M_{j+m-1} &= M_{j+m-2} + f - h_m. \end{aligned}$$

We will go on with bounding $|\Pi_{M,M'}| \leq poly(n)|\Omega(H)|$ by using again the function $\eta_{M,M'}(I, F)$ defined in (5.3). The difference to the case where we had no wide edges is, that $\eta_{M,M'}(I, F)$ might now be further away from being a matching, in the sense that we can only show

$$\eta_{M,M'}(I, F) : \Pi_{M,M'} \to \Omega_s(H), \tag{5.4}$$

where

$$\Omega_s(H) = \{H' \subset H : \exists e_0, \dots, e_s \in H' \text{ such that } H' - \{e_0, \dots, e_s\} \in \Omega(H)\}.$$

But for a fixed number of wide edges s we are then still be able to conclude that

$$|\Omega_s(H)| \leq |\{(M, e_0, e_1, \ldots, e_s) : M \in \Omega(H), e_0, \ldots, e_s \in H\}| \leq n^{(s+1)k}|\Omega(H).$$

Thus we still get a polynomial bound on $|cut(S)|$.

To show (5.4) we have to deal with the problem that because there are wide edges in H there may be cases where $e_1, e_2, e_3 \in I$, $e_4 \in F$ and $e_4 \cap e_i \neq \emptyset, i = 1, 2, 3$. Then, in order to add e_4 to the current matching M_j, we have to remove e_1 and e_2 and at least one of them, say e_2 by using the transition **(-)**. Removing e_2 might lead to a path of length three in $\eta_{M,M'}(I, F)$, because e_2 can intersect at most two other edges of F. Since there are s wide edges, this problem might at most occur s times. Thus there are always s edges such that removing them will result in a matching. This proves (5.4).

5.4 Conclusion

In this chapter we saw a first example of an approximation algorithm which turned out to be efficient enough to be a FPRAS. Figure 5.2 visualizes this result in the known way.

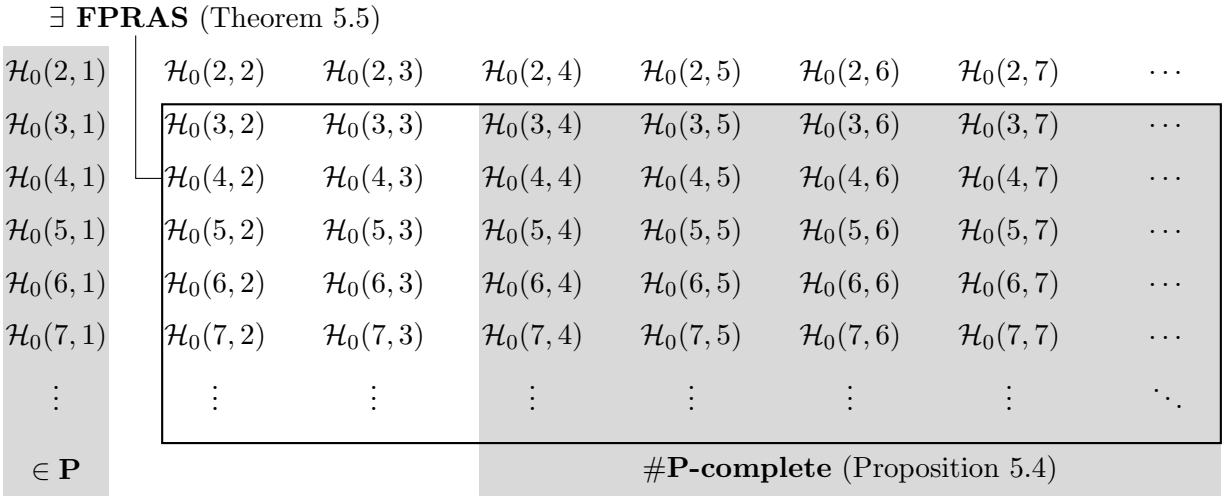


Figure 5.2: (In-)approximability and complexity map for hypergraphs without 3-combs

However in order to show that the used Markov chain is rapidly mixing we had to restrict on a subset of rather sparse hypergraphs. Although we proved that counting the matchings is still #**P**-complete for this class of hypergraphs, there is still the question, what kind of hypergraphs satisfy the 'no 3-comb condition'? We refer the reader to the original source [KRS13] where the authors give some examples of non-trivial hypergraphs without 3-combs.

6 A FPTAS for counting matchings in (3,3)-graphs

After we derived a randomized approximation algorithm in the last chapter, in this chapter we will construct a deterministic FPTAS for the number of matchings in one of the subclasses we introduced in chapter 3, namely the class of $(3,3)$-graphs denoted by $\mathcal{H}(3,3)$.

The proof in this chapter is structured as follows. Section 6.1 contains structural properties for the class of $(3,3)$-graphs. This properties are strongly used in the proof, which is why it can not trivially generalized to other subclasses of hypergraphs. To be more precise, we again reduce the problem of counting matchings to the one of counting independent sets in the corresponding intersection graph. These intersection graphs of $(3,3)$-graph have special properties that let us split the graph into disjoint parts which can be used to formulate a recursive relation on the number of independent sets. This relation is explained in section 6.2. Finally in section 6.4 we state the main algorithm using an approximation for this recursive relation. By using the *correlation decay method* we prove that the approximation error becomes small even for a polynomial number of steps which implies that the algorithm is a FPTAS.

This chapter is based on the paper [DKRS14] by A. Dudek, M. Karpinski, A. Ruciński and E. Szymańska who refined some ideas from [BGK+07].

The main theorem we will prove is the following.

Theorem 6.1. *Let $\varepsilon > 0$ be an arbitrary approximation precision. Then the algorithm Count-Matchings (see Algorithm 1) provides an FPTAS for the number of matchings in $(3,3)$-hypergraphs and runs in time*

$$O\left(n^2\left(\frac{n}{\varepsilon}\right)^{\log_{50/49} 144}\right).$$

6.1 Structural properties of $(3,3)$-Hypergraphs

First recall that in Lemma 3.5 we already proved that the intersection graph $L(H)$ of a hypergraph $H \in \mathcal{H}(k)$ can not contain an induced $K_{1,k+1}$. More properties of the intersection graphs

of $(3,3)$-graphs are gathered in the following Lemma.

Lemma 6.2. *Let H be a $(3,3)$-graph and denote by $G = L(H)$ its corresponding intersection graph. Then*

1. *G has maximum degree at most 6 (i.e. $\Delta(G) \leq 6$),*
2. *and the neighborhood $N_G(v)$ of any vertex v with $\deg(v) \geq 5$ induces a subgraph with at most $6 - d$ isolated vertices.*

Proof. Property 1 follows from the fact that because in a $(3,3)$-graph an edge can at most intersect with 6 other edge (because the vertex degree is bounded by 3 and each edge consists of 3 vertices). Concerning property 2 notice that if a vertex v in G has degree greater or equal to 5 the edges corresponding to $N_G(v)$ span a matching of size $\lfloor \frac{d}{2} \rfloor$. □

Now combining the results from Lemmas 3.5 and 6.2 and using Lemma 3.4, which stated that counting matchings in H equals counting independent sets in $L(H)$, yields that the following technical lemma implies Theorem 6.1.

Lemma 6.3. *Let G be a graph with the following properties:*

- *It is 4-claw-free,*
- *has maximum degree at most 6 (i.e. $\Delta(G) \leq 6$),*
- *the neighbourhood of every vertex of degree $d \geq 5$ induces a subgraph with at most $6 - d$ isolated vertices.*

Then there exists an FPTAS for the problem of counting independent sets in G.

This means that we can now concentrate on constructing a FPTAS for the number of independent sets in graphs with the above properties. From now on in this chapter, unless it is stated otherwise, let G always denote the intersection graph of a $(3,3)$-graph (which then has the properties of Lemma 6.2).

In graph theory a set of vertices that induces a complete subgraph is called a *Clique*. In this proof we will need a more general definition of a set of vertices which is 'almost' a clique.

Definition 6.4 (Simplicial 2-clique)**.** *A set $K \subset V(G)$ is a **2-clique** if the size of the largest independent set of $G[K]$ is at most 2. A 2-clique is **simplicial** if for every $v \in K$, $N_G(v) \backslash K$ is a 2-clique in $G - K$.*

We define a special kind of 2-clique.

Definition 6.5 (Simplicial block)**.** *A 2-clique K in a graph G is called a **block** if $|K| \leq 4$ and, if $|K| = 4$, the minimal degree of a vertex in G is at least 1 (i.e. there are no isolated vertices in K). A block is **simplicial** if for every $v \in K$ the set $N_G(v) \backslash K$ is a block in $G - K$.*

Remark: *Note that the empty set is also considered a block.*

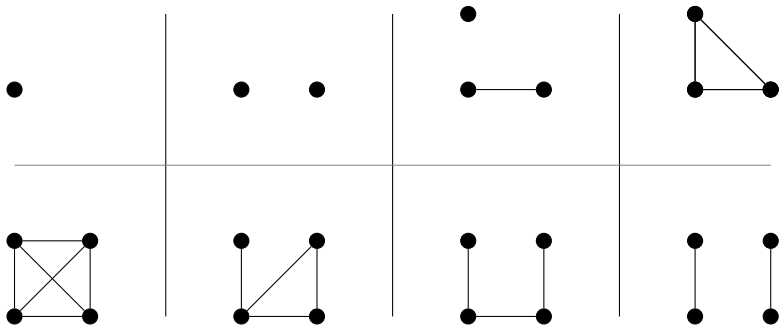


Figure 6.1: Examples of blocks

From this definition we immediately get the following property.

Corollary 6.6. *If K is a (simplicial) block in G, then for every $V' \subset V(G)$ the set $K \cap V'$ is a (simplicial) block in the induced subgraph $G[V']$ of G.*

The following Lemma will give us a 'self-reproducing' property of simplicial blocks. This is meant in the sense, that if we start with one simplicial block, we can always remove this from the graph and still get a new simplicial block in the remainder of the graph.

Lemma 6.7. *If K is a simplicial block in G, then for every $v \in K$ the set $N_G(v) \backslash K$ is a simplicial block in $G - K$.*

Proof. Let $v \in K$ be an arbitrary vertex, set $K_v = N_G(v) \backslash K$. We first show that K_u is indeed again simplicial. Let $u \in K_v$ and $K_u = N_G(u) \backslash (K \cup N_G(v))$. Now suppose that K_u is not a 2-clique, then there is an independent set I of size $|I| = 3$. This implies that u, v and I form an induced $K_{1,4}$ in G with u in the center which is a contradiction to Lemma 3.5 and thus K_u is a 2-clique and K_v is simplicial.
In order to show that K_u is a block, we have to use a property coming from the fact that H is a $(3,3)$-graph. Recall that from Lemma 6.2 we have that $\Delta(G) \leq 6$ and thus $|K_u| \leq 5$. But if $|K| = 5$ then v has to be an isolated vertex in $G[N_G(u)]$. This is a contradiction to the second property of Lemma 6.2. Also if $|K_u| = 4$ by the same reasoning there can be no isolated vertex in $G[K_u]$. □

Now we can state the result - we mentioned in the introduction of this chapter - that a graph, containing at least one simplicial block, can be completely split into disjoint simplicial blocks.

Lemma 6.8. *Let K be a non-empty simplicial block in G. If, in addition, G is connected, then there exists a partition $V(G) = K_1 \cup \cdots \cup K_m$ such that $K_1 = K$ and for every $i = 2, \ldots, m$, K_i is a non-empty, simplicial block in $G_i = G - \bigcup_{j=1}^{i-1} K_j$.*

Proof. The proof is straight forward. Suppose we already constructed disjoint $K_1 \cup \cdots \cup K_s$ such that $K = K_1$ and K_i is a simplicial block in $G_i = G - \bigcup_{j=1}^{i-1} K_j$ for all $i = 2, \ldots, s$. Call the remainder $R_s = V(G) \setminus \bigcup_{j=1}^{i-1} K_j$ and assume $R \neq \emptyset$ (otherwise we are already finished). Because G is connected, there exists an edge between R_s and a vertex $v \in K_i$ for some i. Set $V' = K_i \cup R_s$. Now we can use Corollary 6.6 and get that K_i is not only a simplicial block in G_i but also in the subgraph $G_i[V']$. This means that we can apply Lemma 6.7, where in the notation of the Lemma we set $G = G_i[V']$, $K = K_i$. Thus we get that for each $v \in K_i$, $N_G(v) \cap R_s$ is a simplicial block in $G_{s+1} = G - \bigcup_{i=1}^{s} K_i$. Thus we have a new simplicial block in G_{s+1} and we can repeat this procedure until there are no vertices left. □

We introduced the notion of blocks because we want to split the intersection graph of a hypergraph in disjoint parts. These parts then help defining a recursive relation on the probabilities that a a block is part of a uniformly sampled independent set. So far, we have only proven that this splitting is possible for connected graphs, and we assumed that there we already have a simplicial block to start with. The restriction to connected graphs is not a problem as the counting of matchings in not connected graphs breaks down to counting of matchings in its connected components. However so far there is no reason to believe that we can always find a simplicial block to start with. We will address this problem in the next section.

6.2 Using blocks to count the number of independent sets

Denote the number of matchings in a (hyper-)graph H by $Z_M(H)$ and the number of independent sets in a graph G by $Z_I(G)$. In Lemma 3.4, we already proved their relation to each other to be

$$Z_M(H) = Z_I(L(H)).$$

Denote by $K_1, \ldots, K_m$ the blocks as in Lemma 6.8. Then because $G_{m+1} = \emptyset$ and thus $Z_I(G_{m+1}) = 1$, we have

$$Z_I(G) = \frac{Z_I(G_1)}{Z_I(G_2)} \frac{Z_I(G_2)}{Z_I(G_3)} \cdots \frac{Z_I(G_i)}{Z_I(G_{i+1})} \cdots \frac{Z_I(G_m)}{Z_I(G_{m+1})}.$$

Observe that these quotients can actually be interpreted as probabilities. In rigorous terms this means

$$\mathbb{P}[K_i \cap \mathbf{I} = \emptyset] = \frac{Z_I(G_i - K_i)}{Z_I(G_i)}, \tag{6.1}$$

where $\mathbf{I}$ is a random variable, uniformly distributed on the set of independent sets of G_i (see Definition 2.21). Our goal now has to be to approximate this probability within a given precision

$1 \pm \frac{\varepsilon}{n}$ in polynomial time. After reading chapter 5, one would of course instantly guess that this should be done by using a Markov chain. Instead we will choose a different approach using the correlation decay method.

First we have to deal with the problems mentioned at the end of the previous section. Concerning the problem of not connected graphs, note that

$$Z_I(G) = \prod_{i=1}^{c} Z_I(G_i),$$

where G_i are the connected components of G.
On the other hand, S. Fadnavis [Fad12] proposed a solution for the problem that there might not be a simplicial block to start with. First fix $v \in V(G)$ such that $G - v$ is connected (such a vertex exists because of the assumptions we made on G, cf. Lemmas 3.5 and 6.2), then the following holds:

$$Z_I(G) = Z_I(G - v) + Z_I(G^v),$$

where $G^v = G - N_G[v]$ and $N_G[v] = N_G(v) \cup \{v\}$. Indeed this is true because the missing independent sets not counted in $Z_I(G - v)$ are the ones containing v and thus can not contain $N_G(v)$. Now G^v dissolves into connected components G_i^v. In the following Lemma we finally state the crucial property that each G_i contains a simplicial block.

Lemma 6.9. *Each of the G_i^v contains a simplicial block K_i.*

Proof. Because $G - v$ is connected we can for each i find vertices $u_i \in N_G(v)$ such that $N_G(u_i) \cap V(G_i^v) \neq \emptyset$. Now set $K_i = N_G(u_i) \cap V(G_i^v)$. The proof that K_i is indeed a simplicial block (i.e. for all $w \in K_i$, $N_{G_i}(w) \backslash K_i$ is a block in $G_i^v - K_i$) is the same as the proof of Lemma 6.7. To prove that K_i is a block, suppose there is an independent set I in $G[K_i]$ of size $|I| = 3$. Then because u_i is a neighbour of v, v, u_i and the vertices of I form a $K_{1,4}$. This is a contradiction and thus K_i is a 2-clique. Notice that there is no edge between v and K_i thus if $|K_i| = 5$, v would be an isolated vertex in $G[N(u_i)]$ which is a contradiction to the assumptions on G. If $|K_i| = 4$ is the only isolated vertex in $G[N(u_i)]$ and thus minimal degree of a vertex in $G[K_i]$ is greater or equal to 1. In conclusion K_i is also a block. □

To shorten notation set $N_v = N_G(v)$. Also, when we are writing $uv \notin G[K]$ we mean the ordered pairs $\{u, v\}$ which are not an edge in $G[K]$.
Now we can derive the recursive relation between the number of independent sets in the complete graph, and the number of independent sets in graphs where we have taken out a simplicial block. Let K be a simplicial block, then the following equation describes the useful recurrence relation.

$$Z_I(G) = Z_I(G - K) + \sum_{v \in K} Z_I(G - (N_v \cup K)) + \frac{1}{2} \sum_{uv \notin G[K]} Z_I(G - (N_u \cup N_v \cup K))$$

Since the first term on the right-hand side stands for the number of independent sets in $G-K$, the others have to somehow add the number of independent sets in G that got lost in $G-K$. Indeed the second term sums over all independent sets that contained a vertex $v \in K$. The neighbors of v can then not be in the independent set and thus we are summing over $Z_I(G-(N_v \cup K))$. The last term on the right-hand side represents all independent sets that contain two vertices $u, v \in K$. Of course then u and v can not be connected in K. That is why we sum over $uv \notin G[K]$.

Dividing by $Z_I(G-K)$ yields

$$\frac{Z_I(G)}{Z_I(G-K)} = 1 + \sum_{v \in K} \frac{Z_I(G-(N_v \cup K))}{Z_I(G-K)} + \frac{1}{2} \sum_{uv \notin G[K]} \frac{Z_I(G-(N_u \cup N_v \cup K))}{Z_I(G-K)} \tag{6.2}$$

and another representation of the last term on the right-hand side is given by

$$\frac{Z_I(G-(N_u \cup N_v \cup K))}{Z_I(G-K)} = \frac{Z_I(G-(N_u \cup N_v \cup K))}{Z_I(G-(N_v \cup K))} \frac{Z_I(G-(N_v \cup K))}{Z_I(G-K)}. \tag{6.3}$$

Now by Lemma 6.7, $N_v \backslash K$ is a simplicial block in $G-K$. Because we want to apply this equation recursively on itself, it is left to show that $N_u \backslash (N_v \cup K)$ is a simplicial block in $G-(N_v \cup K)$.

Lemma 6.10. *Let K be a simplicial block in G and let $u, v \in K$ be such that $u \neq v$ and $uv \in G[K]$. Further let $G' = G-(N_G(v) \cup K)$. Then $N_{G'}(u)$ is a simplicial block in G'.*

Proof. We know from Lemma 6.7 that $N_G(u) \backslash K$ is a simplicial block in $G-K$. Now use Corollary 6.6 on $N_G(u) \backslash K$ and $G-K$ with $V' = V(G')$. □

6.3 The recursive relation of probabilities

After we derived the structural properties in the previous sections we can now start bounding the probability in (6.1). For this matter set

$$\Pi_G(K) = \mathbb{P}(K \cap \mathbf{I} = \emptyset) = \frac{Z_I(G-K)}{Z_I(G)}, \tag{6.4}$$

where $\mathbf{I}$ is a uniformly distributed random variable on the independent sets of G.

The recurrence relation in (6.2) together with (6.3) can be written in terms of the new defined probabilities (6.4). First set $K_v = N_v \backslash K$ and $K_{uv} = N_u \backslash (N_v \cup K)$. Then we get $G-(N_v \cup K) = G-K-K_v$. Finally we can rewrite (6.2) to

$$\Pi_G^{-1}(K) = 1 + \sum_{v \in K} \Pi_{G-K}(K_v) \left(1 + \frac{1}{2} \sum_{uv \in G[K]} \Pi_{G-K-K_v}(K_{uv}) \right). \tag{6.5}$$

Remark: *Note that we used the inverse of* (6.3) *to take the factor $\sum_{v \in K} \Pi_{G-K}(K_v)$ out of the last two summands.*

So far we have rewritten our problem of counting the independent sets into a problem of calculating a probability. We also derived a recursive relation similar to the one in Section 4.3. Following the method of Section 4.3 we define an approximating function Φ_G similar to the one in Definition 4.10.

Definition 6.11. *For every graph G, every simplicial block K in G and an integer $t \in \mathbb{Z}_+$, the function $\Phi_G(K,t)$ is defined recursively as follows: For $t = 1$ of $K = \emptyset$ set*

$$\Phi_G(K,0) = \Phi_G(K,1) = 1 \quad \text{and} \quad \Phi_G(\emptyset,t) = 1,$$

for $t \geq 2$ and $K \neq \emptyset$

$$\Phi_G^{-1}(K,t) = 1 + \sum_{v \in K} \Phi_{G-K}(K_v, t-1) \left(1 + \frac{1}{2} \sum_{uv \in G[K]} \Phi_{G-K-K_v}(K_{uv}, t-2)\right).$$

As a result of this definition we get the following technical observation.

Lemma 6.12. *Both quantities $\Pi_G(K)$ and $\Phi_G(K,t)$ always fall into the interval $[\frac{1}{9}, 1]$.*

Proof. The upper bound holds trivially because $\Pi_G(K)$ and $\Phi_G(K,t)$ are probabilities and in their definition they are the inverse of a number always greater or equal than one. The lower bound on $\Pi_G(K,t)$ holds because in the worst case $\sum_{v \in K} \Pi_{G-K}(K_v) \leq 4$, because we already bounded $\Pi_{G-K}(K_v)$ by one from above and a block does at most contain four vertices. Also because vertices in $G[K]^c$ have degree at most 2 (because of the definition of a block), the total number of summands in the last sum of (6.2) does not exceed 8 and each of them is lower or equal to $\frac{1}{2}$. □

Now that we have all definitions and preliminary results together we can define the actual algorithm 'CountMatchings' (see Algorithm 1) with its subroutine 'CountIS' (see Algorithm 2). The running time of the algorithm now depends heavily on the speed at which Φ approximates Π. We will prove in the next section that the following proposition (the analogue statement to Theorem 4.11) holds.

Proposition 6.13. *For*

$$t = 2 \left\lceil \frac{\log\left(\frac{18n}{\varepsilon}\right)}{\log\left(\frac{50}{49}\right)} \right\rceil, \tag{6.6}$$

which is of order $\Theta(\log n)$, $\Phi_G(K,t)$ approximates $\Pi_G(K)$ within a multiplicative factor of $1 \pm \frac{\varepsilon}{n}$.

As a conclusion we can now prove Lemma 6.3 which implies Theorem 6.1 and thus the existence of an FPTAS for the number of matchings in a $(3,3)$-graph.

Proof of Lemma 6.3. In the recursion of Step 4 of 'CountIS' (see Algorithm 2) at most 12 recursive expressions have to be evaluated (cf. Definition 6.11). Thus calculating $\Phi(K,t)$ in Step 4 takes a running time of 12^t. As a result the for-loop in 'CountIS' takes at most $n12^t$ steps.

Algorithm 1 CountMatchings(H, t)

Input: A $(3,3)$-graph H, a time $t \in \mathbb{Z}_+$

Output: The number of matchings in H

1: $G := L(H)$
2: $Z_M := 1, F := G$
3: **while** $F \neq \emptyset$ **do**
4: Pick $v \in V(F)$ such that $F - v$ is connected
5: $F^v := F - N_F[v]$
6: If $F^v = \emptyset$ then $Z_M = Z_M + 1$ and goto Line 3
7: $F^v = \bigcup_{i=1}^{c} F_i^v$, where F_i^v are the connected components of F^v
8: **for** $i := 1$ to c **do**
9: Find K_i as in Lemma 6.9
10: **end for**
11: $Z_M := Z_M + \prod_{i=1}^{c}$ **CountIS**(F_i^v, K_i, t)
12: $F := F - v$
13: **end while**
14: **return** Z_M

Algorithm 2 CountIS(G, K, t)

Input: A graph G, a simplicial block K, a time $t \in \mathbb{Z}_+$ that determines how well Φ approximates Π

Output: The number of independent sets in G

1: Let $V(G) = \bigcup_{i=1}^{m} K_i$ be a partition of $V(G)$ as in Lemma 6.8 with $K_1 = K$
2: $Z_I := 1, F := G$
3: **for** $i = 1$ to m **do**
4: $Z_I := \frac{Z_I}{\Phi_F(K_i, t)}$
5: $F := F - K_i$
6: **end for**
7: **return** Z_I

The subroutine 'CountIS' is invoked at most n times in Step 11 by 'CountMatchings' (see Algorithm 1). Now using that Proposition 6.13 states that we get a good enough approximation of $\Pi(K)$ by $\Phi(K,t)$ already for t as in (6.6). In total we get a running time of the algorithm 'CountMatchings' of

$$O\left(n^2\left(\frac{n}{\varepsilon}\right)^{\log_{50/49}144}\right).$$

□

What is still missing is a proof for Proposition 6.13 which we give in the next section.

6.4 Using correlation decay to prove Proposition 6.13

As a result of Lemma 6.12 the condition that the approximation is within a multiplicative factor $1 \pm \frac{\varepsilon}{n}$ breaks down to

$$|\Pi_G(K) - \Phi_G(K,t)| \leq \frac{\varepsilon}{9n}.$$

To show this bound we will use the correlation decay in the computation tree of (6.5). The idea is to express the difference $|\Pi_G(K) - \Phi_G(K,t)|$ in terms of a function f such that

$$|\Pi_G(K) - \Phi_G(K,t)| = |f(\mathbf{x}) - f(\mathbf{y})|,$$

and then use the mean value theorem to bound this recursively. The construction of such a function f is done by defining differentiable functions $g : [0,1] \to \mathbb{R}$ and $h : \mathbb{R} \to [0,1]$ such that they are inverse to each other (i.e. $g \circ h \equiv 1$). The function f is then a function of $|K| + 2e(G^c[K])$ variables mapping to $\mathbb{R}$. Here $e(G^c[K])$ stands for the number of ordered pairs of vertices in K which are not an edge in $G[K]$. We denote those variables by

$$\mathbf{z} = (z_1, \ldots, z_{|K|}, z_{uv} : uv \notin G[K]),$$

and the index set by

$$J = K \cup \{(u,v) : \{u,v\} \notin G[K]\}.$$

The function f itself is then defined by

$$f_K(\mathbf{z}) = f(\mathbf{z}) = g\left(\left[1 + \sum_{v \in K} h(z_v)\left(1 + \frac{1}{2}\sum_{uv \notin G[K]} h(z_{uv})\right)\right]^{-1}\right)$$

This function already looks very similar to the recursion for Π and Φ. If we apply this function to the following input values it becomes even clearer why the definition in this form was a good idea. Set

$$x = g(\Pi_G(K)), \quad x_v = g(\Pi_{G-K}(K_v)), \quad x_{uv} = g(\Pi_{G-K-K_v}(K_{uv})),$$

and

$$y = g(\Phi_G(K,t)), \quad y_v = g(\Phi_{G-K}(K_v, t-1)), \quad y_{uv} = g(\Phi_{G-K-K_v}(K_{uv}, t-2)).$$

Now we have $f(\mathbf{x}) = x$ and $f(\mathbf{y}) = y$ which means that the difference can be expressed as

$$|x - y| = |f(\mathbf{x}) - f(\mathbf{y})|.$$

The great advantage of this rewriting is the fact, that because f is differentiable, we can apply the mean value theorem and thus bound the difference of Π to Φ in terms of the difference of $\mathbf{x}$ to $\mathbf{y}$. Indeed we have for some $\alpha \in [0,1]$, a $\mathbf{z}_\alpha = \alpha\mathbf{x} + (1-\alpha)\mathbf{y}$ such that,

$$|f(\mathbf{x}) - f(\mathbf{y})| = |\nabla f(z_\alpha)(\mathbf{x} - \mathbf{y})| \leq |\nabla f(z_\alpha)| \max_{k \in J} |x_k - y_k|.$$

Because of the 'self-reproducing' property of simplicial blocks in Lemma 6.7 we can iterate this argument, until we have the following final situation: we have G', an induced subgraph of G such that we have a block K' in G' and $t = t' \in \{0, 1\}$. Now set $\mu_g = |g(1)| + |\max_{s \in [0,1]} g(s)|$ and we get

$$|x - y| \leq \gamma^{\frac{t}{2}} |g(\Pi_{G'}(K')) - g(1)| \leq \gamma^{\frac{t}{2}} \mu_g \leq \frac{\varepsilon}{9n},$$

for

$$t \geq 2 \frac{\log\left(\frac{9\mu_g n}{\varepsilon}\right)}{\log\left(\frac{1}{\gamma}\right)}, \quad \text{where} \quad \max_z |\nabla f(z)| < \gamma < 1. \tag{6.7}$$

We used for this that we have to iterate at most $\frac{1}{2}t$ times.

Now the thing left to do is bound $\max_z |\nabla f(z)|$ by a $\gamma < 1$. For this matter we chose $g(s) = s^{\frac{1}{4}}$ and $h(s) = s^4$. Then $\mu_g = 2$ and

$$|\nabla f(\mathbf{z})| \leq \sum_{k \in J} \left| \frac{\partial f(\mathbf{z})}{\partial z_k} \right| = \frac{\sum_{v \in K} \left[z_v^3 + \frac{1}{2} \sum_{uv \notin G[K]} (z_v^3 z_{uv}^4 + z_v^4 z_{uv}^3) \right]}{\left[1 + \sum_{v \in K} z_v^4 \left(1 + \frac{1}{2} \sum_{uv \notin G[K]} z_{uv}^4 \right) \right]^{\frac{5}{4}}}$$

Note that this bound depends only on the (isomorphism) type of $G[K]$.

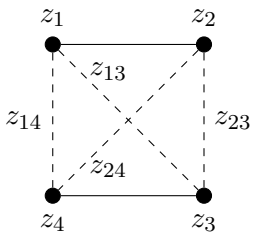


Figure 6.2: The essential block graph, Source: [DKRS14]

By definition there are only finitely many block graphs K (up to isomorphic changes). Thus $\|\nabla f(\mathbf{z})\|$ is definitely bounded, the question remains if it is bounded by a constant smaller than one. But indeed using a kind of 'worst case block graph' (see Figure 6.2), the authors of [DKRS14] calculated that $\|\nabla f(\mathbf{z})\| < 0.971$ for $0 \leq z_i \leq 1$ and $0 \leq z_{ij} \leq 1$. Thus we can chose $\gamma = 0.98 = \frac{49}{50}$ and if we plug this in (6.7) with $\mu_g = 2$ we get the bound in the statement of Proposition 6.13.

6.5 Conclusion

As a result of this chapter we can cross one more class of hypergraphs of the map because we now know that a FPTAS exists for $(3,3)$-graphs. Than the complexity map looks as in Figure 6.3.

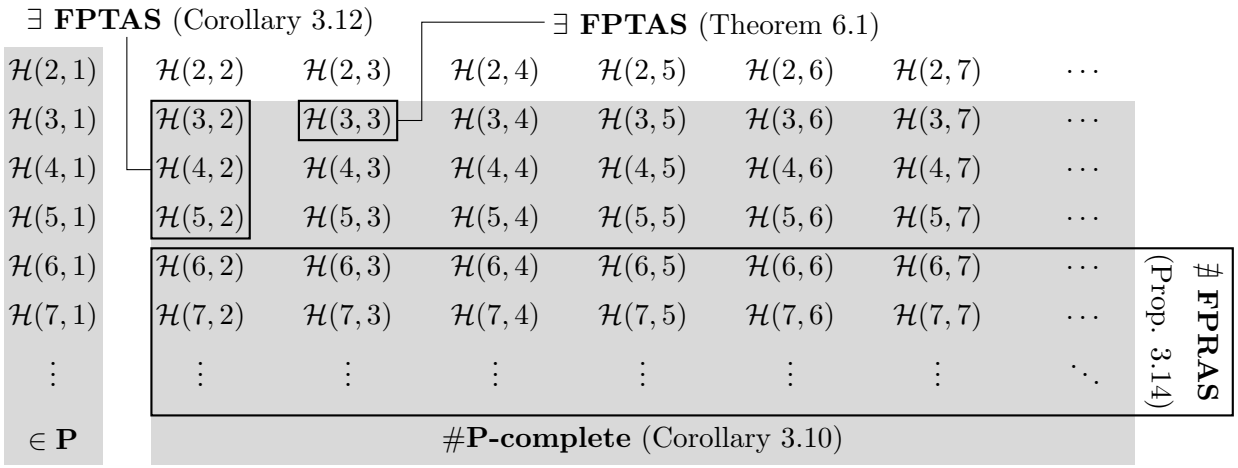


Figure 6.3

7 Further research

First let us note that, in the end, our result from Chapter 5 was complemented by an even stronger result. As the authors of [DKRS14] also point out, the intersection graphs of the hypergraphs without 3-combs we considered in chapter 5 are claw-free. Thus one can apply the result on counting independent sets from [BGK+07; Fad12] on these intersection graphs and get a FPTAS for the number of matchings in (k, r)-graph without 3-combs. This kind of improves the result from Chapter 5.

At the end of this thesis there are still open problems. By looking at the maps in Figures 5.2 and 6.3 it is easy to see that the next open hypergraph subclasses - for which there is to show either inapproximability or the existence of an approximation algorithm - are $\mathcal{H}(3, 4)$ or $\mathcal{H}(4, 3)$. As an intermediate step the authors of [DKRS14] suggest looking at (k, r)-graphs without 4-combs. Then the first open instance is that of $(4, 3)$-graphs without 4-combs. Also there is hope that the correlation method used in Chapter 6 can be applied to $\mathcal{H}(3, 4)$ or $\mathcal{H}(4, 3)$.

Additionally because of its fast spread in recent publications, the correlation decay method seems to be a very useful tool for constructing fast approximation algorithms, not only for the problem of counting matchings but also for other approximation problems.

Bibliography

[AB09] Arora, S. and Barak, B. *Computational Complexity: A Modern Approach*. 1st. New York, NY, USA: Cambridge University Press, 2009 (cit. on p. 16).

[BGK+07] Bayati, M., Gamarnik, D., Katz, D., Nair, C., and Tetali, P. "Simple Deterministic Approximation Algorithms for Counting Matchings". In: STOC '07. ACM, 2007, pp. 122–127 (cit. on pp. 31, 45, 57).

[CS07] Chudnovsky, M. and Seymour, P. "The Roots of the Independence Polynomial of a Clawfree Graph". In: *J. Comb. Theory Ser. B* 97.3 (May 2007), pp. 350–357 (cit. on p. 2).

[DKRS14] Dudek, A., Karpinski, M., Ruciński, A., and Szymańska, E. "Approximate Counting of Matchings in (3, 3)-Hypergraphs". In: *Proceedings of 14th Scandinavian Symposium and Workshops on Algorithm Theory (SWAT)* (2014) (cit. on pp. 3, 17, 45, 54, 57).

[Fad12] Fadnavis, S. *Approximating independence polynomials of claw-free graphs*. 2012. URL: `http://www.math.harvard.edu/~sukhada/IndependencePolynomial.pdf` (cit. on pp. 49, 57).

[GJ79] Garey, M. and Johnson, D. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. B&T, 1979 (cit. on pp. 11–13, 16).

[GK07] Gamarnik, D. and Katz, D. "Correlation Decay and Deterministic FPTAS for Counting List-colorings of a Graph". In: *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms*. SODA '07. Society for Industrial and Applied Mathematics, 2007, pp. 1245–1254 (cit. on pp. 3, 29, 31).

[Gre00] Greenhill, C. "The complexity of counting colourings and independent sets in sparse graphs and hypergraphs". In: *Computational complexity* 9.1 (2000), pp. 52–72 (cit. on pp. 2, 21, 22).

[Hei72] Heilmann, O.J. "Existence of phase transitions in certain lattice gases with repulsive potential". In: *Lettere Al Nuovo Cimento Series 2* 3.3 (1972), pp. 95–98 (cit. on p. 2).

[Her95] Herken, R. *The universal Turing machine : a half-century survey*. Springer-Verlag, 1995 (cit. on p. 12).

[HL72] Heilmann, O.J. and Lieb, E. "Theory of monomer-dimer systems". In: *Communications in Mathematical Physics* 25.3 (1972), pp. 190–232 (cit. on p. 1).

[Jer03] Jerrum, M. *Counting, Sampling and Integrating: Algorithms and Complexity*. Springer, 2003 (cit. on pp. 11, 26).

[Jer85] Jerrum, M. "Random Generation of Combinatorial Structures from a Uniform Distribution (Extended Abstract)". In: *Proceedings of the 12th Colloquium on Automata, Languages and Programming*. London, UK, UK: Springer-Verlag, 1985, pp. 290–299 (cit. on p. 26).

[JS89a] Jerrum, M. and Sinclair, A. "Approximate Counting, Uniform Generation and Rapidly Mixing Markov Chains". In: *Information and Computation* 82.1 (July 1989), pp. 93–133 (cit. on p. 28).

[JS89b] Jerrum, M. and Sinclair, A. "Approximating the Permanent". In: *SIAM Journal on Computing* 18.6 (Dec. 1989), pp. 1149–1178 (cit. on p. 28).

[KRS13] Karpinski, M., Ruciński, A., and Szymańska, E. "Approximate Counting of Matchings in Sparse Uniform Hypergraphs". In: *Proceedings of the 13th SIAM Meeting on Analytic Algorithmics and Combinatorics* (2013), pp. 71–78 (cit. on pp. 3, 8, 17, 23, 25, 26, 33–35, 43).

[KV07] Korte, B. and Vygen, J. *Combinatorial Optimization: Theory and Algorithms*. 4th. Springer Publishing Company, Incorporated, 2007 (cit. on pp. 6, 7).

[LPW06] Levin, D., Peres, Y., and Wilmer, E. *Markov chains and mixing times*. AMS, 2006 (cit. on p. 11).

[LV99] Luby, M. and Vigoda, E. "Fast Convergence of the Glauber Dynamics for Sampling Independent Sets". In: *Random Structures & Algorithms* 15.3-4 (Oct. 1999), pp. 229–241 (cit. on pp. 3, 23).

[Sly10] Sly, A. "Computational Transition at the Uniqueness Threshold". In: *Proceedings of the 2010 IEEE 51st Annual Symposium on Foundations of Computer Science*. FOCS '10. Washington, DC, USA: IEEE Computer Society, 2010, pp. 287–296 (cit. on pp. 23, 24).

[SS12] Sly, A. and Sun, N. "The Computational Hardness of Counting in Two-Spin Models on d-Regular Graphs". In: *Proceedings of the 2012 IEEE 53rd Annual Symposium on Foundations of Computer Science*. FOCS '12. IEEE Computer Society, 2012, pp. 361–369 (cit. on pp. 23, 24).

[Vad02] Vadhan, S. "The Complexity of Counting in Sparse, Regular, and Planar Graphs". In: *SIAM Journal on Computing* 31.2 (Feb. 2002), pp. 398–427 (cit. on p. 34).

[Val79a] Valiant, L. "The complexity of computing the permanent". In: *Theoretical Computer Science* 8 (1979), pp. 189–201 (cit. on pp. 16, 17, 21).

[Val79b] Valiant, L. "The Complexity of Enumeration and Reliability Problems." In: *SIAM Journal on Computing* 8.3 (1979), pp. 410–421 (cit. on pp. 2, 21).

[Wei07] Weitz, D. "Counting Independent Sets Up to the Tree Threshold". In: *Proceedings of the Thirty-eighth Annual ACM Symposium on Theory of Computing*. STOC '06. ACM, 2007, pp. 140–149 (cit. on pp. 3, 23, 25, 29, 30).

Ehrenwörtliche Erklärung

Hiermit versichere ich, diese Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen benutzt zu haben. Wörtliche und sinngemäße Zitate sind kenntlich gemacht.

Bonn, July 22, 2016 Benjamin Cabrera