THE UNIVERSITY OF WARWICK

THE LIBRARY
Tel: +44 1203 523523
Fax: +44 1203 524211

AUTHOR: **Michele A. A. Zito** DEGREE: **Ph.D.**

TITLE: **Randomised Techniques in Combinatorial Algorithmics**

DATE OF DEPOSIT: 16 JULY, 1999

I agree that this thesis shall be available in accordance with the regulations governing the University of Warwick theses.

I agree that the summary of this thesis may be submitted for publication.

I **agree** that the thesis may be photocopied (single copies for study purposes only).

AUTHOR'S SIGNATURE: ....

USER'S DECLARATION

1. I undertake not to quote or make use of any information from this thesis without making acknowledgement to the author.
2. I further undertake to allow no-one else to use this thesis while it is in my care.

DATE SIGNATURE ADDRESS

....

....

....

....

....

THE UNIVERSITY OF WARWICK, COVENTRY CV4 7AL

Randomised Techniques in Combinatorial Algorithmics

by

Michele A. A. Zito

Thesis

Submitted to the University of Warwick

for the degree of

Doctor of Philosophy

Department of Computer Science

November 1999

Contents

List of Figures

Acknowledgments

Although my work in Theoretical Computer Science has been mainly a solitary walk through Discrete Mathematics and Computational Complexity Theory, I would like to thank the many people that joined my walk from time to time or that helped my progress, first in Warwick University and then in the University of Liverpool.

First and foremost I would like to thank Alan Gibbons, my supervisor for his presence, his trust in me and his constant support. He has always been much more optimistic about my work than myself and he has often given me the strength to go on. Also I thank Mike Paterson and Martin Dyer for their careful reading of my thesis. Their comments and suggestions have contributed to improve the quality of my work.

I would also like to thank all the people that I met at Warwick. I feel particularly indebted to S. Muthukhrishnan (Muthu). His company, scientific, and social advice was a pleasure during the many days and evenings that we spent together at Warwick.

My thanks go also to all the people in the Department of Computer Science, at the University of Liverpool, especially Paul Dunne, Ken Chan and all the technical staff, for their friendship and support. Thanks to William (Billy) Duckworth, my office colleague during my staying in Liverpool (at least until he decided that the other hemisphere is more interesting than this one), for keeping me interested in spanners and teaching me the exact difference between “tree”, “three” and “free”! It was a great pleasure for me to work with him.

I thank the ‘Universitá di Bari’ in Italy for giving me the cultural means and the financial support to start my scientific journey. I am particularly grateful to Prof. Salvatore Caporaso, who introduced me to the Theory of Computation. I valued his conversations, the many arguments and the useful discussions we had. I also thank Nicola Galesi for the time we spent together, the work we did and the rock&roll music we played.

Finally, last but not least, I feel deeply grateful to my parents and my wife, for their constant love and support.

Michele A. A. Zito July, 1999

Declarations

This thesis is submitted to the University of Warwick in support of my application for admission to the degree of Doctor of Philosophy. No part of it has been submitted in support of an application for another degree or qualification of this or any other institution of learning. Parts of the thesis appeared in the following refereed papers in which my own work was that of a full pro-rata contributor:

- M. Zito, I. Pu, M. Amos, and A. Gibbons. RNC Algorithms for the Uniform Generation of Combinatorial Structures. *Proceedings of the 7th ACM-SIAM Annual Symposium on Discrete Algorithms*, pp. 429–437, 1996.
- P. E. Dunne and M. Zito. An Improved Upper Bound on the Non-3-colourability Threshold. *Information Processing Letters*, 65:17–23, 1998.
- M. Zito. Induced Matchings In Regular Graphs and Trees. *Proceedings of the 25th International Workshop on Graph-Theoretic Concepts in Computer Science*, 1999. Lecture Notes in Computer Science, vol 1665, Springer Verlag.
- M. Zito. Small Maximal Matchings in Random Graphs. Submitted LATIN'2000: Theoretical Informatics.

Unrefereed papers were also presented as follows:

- M. Zito, I. Pu, A. Gibbons. Uniform Parallel Generation of Combinatorial Structures. *11th British Colloquium on Theoretical Computer Science*, Swansea, April 1995. *Bulletin of the European Association of Theoretical Computer Science*, 58, 1996.
- I. Pu, M. Zito, M. Amos, A. Gibbons. RNC Algorithms for the Uniform Generations of Paths and Trees in Graphs. *11th British Colloquium on Theoretical Computer Science*, Swansea, April 1995. *Bulletin of the European Association of Theoretical Computer Science*, 58, 1996.
- P. E. Dunne and M. Zito. On the 3-Colourability Threshold. *13th British Colloquium on Theoretical Computer Science*, Sheffield, 1997. *Bulletin of the European Association of Theoretical Computer Science*, 64, 1998.

Michele A. A. Zito July, 1999

Abstract

Probabilistic techniques are becoming more and more important in Computer Science. Some of them are useful for the analysis of algorithms. The aim of this thesis is to describe and develop applications of these techniques.

We first look at the problem of generating a graph uniformly at random from the set of all unlabelled graphs with n vertices, by means of efficient parallel algorithms. Our model of parallel computation is the well-known parallel random access machine (PRAM). The algorithms presented here are among the first parallel algorithms for random generation of combinatorial structures. We present two different parallel algorithms for the uniform generation of unlabelled graphs. The algorithms run in $O(\log^2 n)$ time with high probability on an EREW PRAM using $O(n^2)$ processors.

Combinatorial and algorithmic notions of approximation are another important thread in this thesis. We look at possible ways of approximating the parameters that describe the phase transitional behaviour (similar in some sense to the transition in Physics between solid and liquid state) of two important computational problems: that of deciding whether a graph is colourable using only three colours so that no two adjacent vertices receive the same colour, and that of deciding whether a propositional boolean formula in conjunctive normal form with clauses containing at most three literals is satisfiable. A specific notion of maximal solution and, for the second problem, the use of a probabilistic model called the (young) coupon collector allows us to improve the best known results for these problems.

Finally we look at two graph theoretic matching problems. We first study the computational complexity of these problems and the algorithmic approximability of the optimal solutions, in particular classes of graphs. We also derive an algorithm that solves one of them optimally in linear time when the input graph is a tree as well as a number of non-approximability results. Then we make some assumptions about the input distribution, we study the expected structure of these matchings and we derive improved approximation results on several models of random graphs.

Chapter 1

Introduction

This chapter provides, in the first section, the algorithmic context of this thesis. The remainder of the chapter describes essential technical preliminaries for all subsequent chapters.

1.1 Algorithmic Background

Probabilistic techniques are becoming more and more important in Computer Science. Probabilistic paradigms can be grouped in two main classes: those concerned with the construction of randomised algorithms (under some reasonable model of computation) and those involved in the analysis of algorithms. Among the first, some have acquired wide popularity. Random sampling is often used to guess a solution in problems for which a large set of candidate solutions provably exists. A recent beautiful application of this technique is in the problem of computing the minimum spanning tree of a graph [KKT95]. Random re-ordering can be used to improve the performances of sorting algorithms [Knu73]. Montecarlo simulation of suitably defined Markov chains or (some other randomised dynamic process) finds wider and wider applications in generation and counting problems [DFK91] as well as in algorithmic analysis [FS96]. Finally what is called sometimes control randomisation (loosely speaking different algorithms or sub-routines are run on the particular problem instance depending on some random choices) is exploited to devise good hashing algorithms and for the complementary pattern matching problem [KR87]. In all cases the main advantages of the specific algorithmic solution over more traditional deterministic approaches are simplicity and good performance improvement It could be argued that this is achieved at the price of a more involved analysis process but since a good deal of discrete mathematics is involved in algorithmic analysis of traditional techniques anyway, this does not seem a major problem.

As mentioned at the beginning there is also another class of probabilistic paradigms. Although not directly related to algorithmic design, their use helps in understanding the combinatorial structure of several computational problems. Normally the set of all inputs for a specific problem is viewed as a probability space (see Section 1.2.3 for a formal definition of this concept) and this fact is exploited in either the performance analysis of specific algorithms or the understanding of structural properties of combinatorial problems. In the former case, sometimes called input randomisation [Bol85], the advantage is that the usual "worst-case approach" is abandoned and therefore worst-case instances only marginally influence the complexity of the different algorithmic solutions. In the latter case, sometimes, the probabilistic approach enables us to understand the behaviour of few parameters characterising the specific problem [ASE92].

The aim of this thesis is to describe and develop a few applications of several probabilistic techniques related to the second type of paradigm described above. The usual assumption about input randomisation is that input instances can indeed be generated with the desired distribution. In some cases this is an important problem in its own right. For example it is still an important open problem to find an efficient algorithm for generating a planar graph uniformly at random [HP73, DVW96, Sch97]. Several techniques have been developed to build sequential algorithms for generating combinatorial structures according to some predefined probability distribution [NW78]. In Chapter 2 we look at the issues involved in finding parallel algorithms for sampling combinatorial objects uniformly at random. In some cases trivial parallelisation of a sequential algorithm solves the problem quite efficiently. In some others the nature of the problem seems to prevent efficient solutions. The focus of this work is on "unlabelled" structures. All relevant definitions are given in Chapter 1.2 and 2. Loosely speaking the aim is to sample an object in a given set, disregarding a number of possible symmetries. For instance, if we were to sample the result of throwing two dice, we might be only interested in the sum of the two individual outcomes, not their ordered values. In this case the order among the two outcomes is irrelevant. In Chapter 2 we will study similar problems in the context of graph theory.

Combinatorial and algorithmic notions of approximation are another important thread in this thesis. One of the generation algorithms described in Chapter 2 outputs a graph with a probability distribution that, in some sense, only approximates the uniform one. In Chapter 3 we look at possible ways of approximating the parameters that describe the phase transitional behaviour (similar in some sense to the transition in Physics between solid and liquid state) of two important computational problems: that of deciding whether a graph is colourable using only three colours so that no two adjacent vertices receive the same colour, and that of deciding whether a propositional boolean formula in conjunctive normal form with clauses containing at most three literals is satisfiable. A specific notion of maximal solution and, for the second problem, the use of a probabilistic model called (young) coupon collector allow us to improve the best known results for these problems.

Chapter 4 is even more about approximation, but the final part of it will describe a number of results obtained through the use of a number of probabilistic techniques. We look at two graph theoretic problems. A graph is given, as a collection of nodes and edges joining them, and we are interested in finding a set of disjoint edges satisfying some additional constraints. The goal in each case is to find an "optimal" set according to some criterion that is part of the specific problem definition. We first study the computational complexity of these problems and the algorithmic approximability of the optimal solutions, in particular classes of graphs. Then we make some assumptions about the input distribution, we study the expected structure of these matchings and we derive improved approximation results on several models of random graphs (see Section 1.2.5 for the formal definitions).

The thesis is mainly self-contained. All concepts used in it are defined. Original definitions are numbered whereas, normally, well-known concepts are introduced in a less formal way and normally referenced. All original results are proved in full details. All non-original results are clearly stated and their proof is normally either sketched or the reader is referred to an appropriate bibliographic reference. Chapter 2 contains some general definitions from the branches of mathematics and computability that are related to this thesis; the reader familiar with the specific field should be able to skip Chapter 2. However, to avoid conceptual discontinuities, a few specific technical concepts and results are introduced in the relevant chapters.

1.2 Technical Preliminaries

We recall some basic terminology and well-known results in the different areas of Computer Science and Mathematics which will be used later on. This section contains all those background definitions and results which are particularly useful in more than one of the following chapters, or simply too long to be put in the specific chapter, without distracting the reader's attention.

Some knowledge of basic set theory and elementary calculus is assumed [Giu83]. If f is a function on real numbers, for every $y \in \mathbb{R} \cup \{-\infty, +\infty\}$, $f(x) \to y$ (or simply $f \to y$ when the independent variable is clear from the context) is a shorthand for

$$\lim_{x \to \infty} f(x) = y$$

Also, the reader should be familiar with sequential computational models like Turing machines or Random Access Machines [AHU74] and basic complexity theoretic definitions [GJ79, BDG88]. Asymptotic notations like $O(n^2)$, $o(1)$, $\Omega(n)$, $\omega(2^n)$ and $\Theta(n)$ will denote function classes but we will normally write $f = \Omega(n)$ (instead of $f \in \Omega(n)$) with the intended meaning that there exists a constant c such that $f(n) \geq cn$ for n sufficiently large. The reader is referred to Section 2.1 in Cormen, Leiserson and Rivest [CLR90], for more formal definitions. In particular, given two functions f and g on integers, we will write $f \sim g$ and we will say that f is *asymptotic* to g if the ratio $f(n)/g(n) \to 1$ (the concept can be extended to functions on real numbers).

This section's content can be subdivided into two parts. The first two sections describe concepts from Computability Theory and Computational Complexity. The remaining sections present some relevant definitions and results from different areas of Mathematics.

More specifically, in Section 1.2.1 we recall some elementary definitions related to computational problems that will be used throughout this thesis. Section 1.2.2 defines the models of parallel computation which will be used in Chapter 2. Also the relevant complexity measures and complexity classes are defined. Section 1.2.3 introduces the basic terminology related to Probability Theory. Section 1.2.4 describes the relevant concepts in graph theory. Section 1.2.5 provides a glimpse into the beautiful and by now well established theory of random graphs. We describe several models of random graphs, each providing a different framework for the analysis of combinatorial and algorithmic properties of graphs. Finally Section 1.2.6 introduces all basic definitions and results in group and action theory that will be needed later, especially in Chapter 2.

1.2.1 Problems

Many computational problems can be viewed as the seeking of partial information about a relation (see [BDG88, Chapter 1] or [BC91]). More specifically suppose Σ is a finite alphabet and that problem instances and solutions are encoded as strings over Σ. A relation $\mathcal{SOL} \subseteq \Sigma^* \times \Sigma^*$ defines an association between *problem instances* and *solutions*. For every $x \in \Sigma^*$ the set $\mathcal{SOL}(x)$ contains all the $y \in \Sigma^*$ that encode a solution associated with x, with the implicit convention that if x does not encode a problem instance then $\mathcal{SOL}(x) = \emptyset$. For every $x \in \Sigma^*$, let $|x|$ denote the *length* of (the encoding) x. Notice incidentally that if S is a set, $|S|$ will be used, in the usual sense, as the *cardinality* of S. In this setting a *decision problem* is described by the relation $\mathcal{SOL}$ and a question which has a yes/no answer in terms of $\mathcal{SOL}$ [BDG88, p. 11]. If $x \in \Sigma^*$, a decision problem (also known as *existence problem*) answers the following question: is there a $y \in \Sigma^*$ such that $y \in \mathcal{SOL}(x)$? If $x, y \in \Sigma^*$ are given, another decision problem (which will be referred to as the *membership problem*) answers the question: does y belong to $\mathcal{SOL}(x)$?

There is a natural correspondence between a decision problem $\mathcal{Q}$ and the set Q of instances that have a "yes" answer. Thus no strong distinction between $\mathcal{Q}$ and Q will be kept: the "name" of the decision problem will also denote the set of instances with a solution. Notice that $Q \subseteq \Sigma^*$ so the words *language* or *property* will also be used as qualifiers.

Several other types of problems are definable in this setting. Informally, if $x \in \Sigma^*$ is (the encoding of) a problem instance then

1. a *construction problem* (or *search problem*) aims at exhibiting a $y \in \Sigma^*$ such that $(x, y) \in \mathcal{SOL}$;
2. an *optimisation problem*, given a cost function $c(x, y)$, aims at finding the $y \in \Sigma^*$ with $(x, y) \in \mathcal{SOL}$ such that $c(x, y)$ is maximised (respectively minimised);
3. a *uniform generation problem*, aims at generating a word $y \in \Sigma^*$ satisfying $(x, y) \in \mathcal{SOL}$ such that all y in $\mathcal{SOL}(x)$ come up equally often;
4. a *counting problem*, aims at finding the number of elements in $\mathcal{SOL}(x)$.

Example. In what follows we recall, in a rather informal way, a number of definitions related to boolean algebra. The reader is referred to [BDG88, Chapter 1] or [Dun88] for a more formal treatment of the subject.

A boolean formula is an expression like

$$\phi(x_3, x_{12}, x_{25}, x_{34}, x_{70}) =_{df} (x_{25} \wedge x_{12}) \vee \neg(\neg x_{70} \vee (\neg x_3 \wedge x_{34}))$$

built up from the elements x_i of a countable set $\mathcal{X}$ of *propositional variables*, a finite set of *connectives* (usually including $\wedge$, $\vee$ and $\neg$) and the brackets.

If variables are assigned values over a binary set of *truth-values* denoted by $\mathcal{V} = \{0, 1\}$ and connectives are interpreted in the usual way as operations on $\mathcal{V}$ then each formula represents a function on $\mathcal{V}$. It is evident that under this interpretation the formula $\phi(x_3, x_{12}, x_{25}, x_{34}, x_{70})$ is equivalent to $\phi(x_1, x_2, x_3, x_4, x_5)$ obtained by replacing x_3 with x_1, x_{12} with x_2 and so on. Thus, without loss of generality, a formula on n different variables can be regarded as containing exactly the variables $x_1, \ldots, x_n$. Let $\mathcal{X}\big|_n = \{x_1, \ldots, x_n\}$. Sometimes notation $\phi(\vec{x})$ will be used instead of $\phi(x_1, x_2, \ldots, x_n)$.

	$\wedge$	
1	1	1
1	0	0
0	0	1
0	0	0

	$\vee$	
1	1	1
1	1	0
0	1	1
0	0	0

	$\neg$
1	0
0	1

Any function $\alpha : \mathcal{X}\big|_n \to \mathcal{V}$ is called an n *variable-truth-assignment* (or simply a truth-assignment). Notation $\phi\{\alpha\}$ will be used for the truth-value of ϕ after each variable x_i has been replaced by $\alpha(x_i)$ and the tables above (called *truth-tables*) have been used to find the truth-value of conjunctions, disjunctions or negations of truth-values. Since truth-values are nothing but binary digits, the set of all n-variable-truth-assignments will normally be denoted by $\{0, 1\}^n$.

The formula is said to be in *conjunctive normal form* (CNF for short) if it is in the form

$$C_1 \wedge C_2 \ldots \wedge C_m$$

where each *clause* C_i is the disjunction of some variables or negation of variables (expressions like x_i or $\neg x_i$ are called *literals*). Every boolean formula can be transformed by purely algebraic rules into a CNF formula (see for example [BDG88, p. 17–18]). A CNF formula ϕ is in k-CNF if the maximum number of literals forming a clause is k.

Every k-CNF formulae can be encoded over the finite alphabet $\Sigma = \{\wedge, \vee, \neg, (,), 0, 1\}$ (e.g. variable x_i is encoded by the binary representation of i). In this setting, k-SAT is the well known NP-complete problem [GJ79] of deciding whether a k-CNF ϕ is satisfiable, i.e. whether there exists an assignment α of values in $\mathcal{V}$ to all variables in ϕ such that the value of ϕ under this assignment is one.

Combinatorial structures associated with computational problems can be characterised by a number of *parameters*, describing specific features of the problem instances. For example a k-CNF formula is built on some n variables, m clauses and each clause has at most k literals. A graph (see Section 1.2.4 for relevant definitions) might have n vertices, m edges, maximum degree Δ. In most cases it will be possible to define two functions on natural numbers, the *order* and the *size*. $\mathcal{I}_n \subseteq \Sigma^*$ is the set of problem instances of order n.

The relationships between different parameters characterising the instances of two specific problems will be the object of the work described in Chapter 3. We conclude this section with a remark and a couple of useful definitions.

It is worth noticing that there might be no relationship between these parameters and the length of the encoding of the instances of a particular problem. For instance the natural encoding of a k-CNF formula of order (i.e. number of variables) n and size (i.e. number of clauses) m described in the example above has length at most $km \log n$.

A set $Q \subseteq \Sigma^*$ is a *monotone increasing property* (respectively *monotone decreasing property*) with respect to a partial order $<_Q$ if for every fixed n

$$x \in \mathcal{I}_n \cap Q, y \in \mathcal{I}_n, x <_Q y \text{ (respectively } y <_Q x) \Rightarrow y \in Q$$

A set $Q \subseteq \Sigma^*$ is a *convex property* if for every $x, y, z \in \mathcal{I}_n$ $x <_Q y <_Q z$ and $x, z \in Q$ imply $y \in Q$.

1.2.2 Parallel Computational Complexity

The algorithms described in Chapter 2 are designed for an idealised model of parallel computation called (see [GR88], for example) *parallel random access machine* (PRAM).

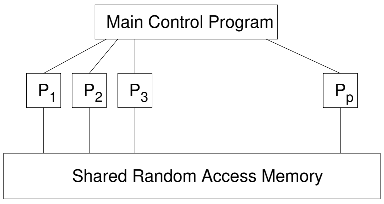

Figure 1.1: A p processor PRAM

There are p processors working synchronously and communicating through a common random-access memory. Each processor is a sequential random access machine (or RAM for short) in the spirit of [CR73]. The set of arithmetic operations available to each processor includes addition, subtraction, multiplication and division between arbitrary length integers at unit cost. Each RAM is also augmented with a facility to generate random numbers. The expressions rand$(0, 1)$ is a call to a function returning a random real number between 0 and 1. Tuples of values (also called *records*) will be used occasionally. For example, following the most standard notations, x.date is the identifier for the variable associated with the field “date” of the record structure x. Ordinal numerals are used as default field names. Assignments and the usual relational operations between field variables as well as whole record variables are allowed. Constant tuples will be represented by lists of value. For example $(4, 1, 3.14, 3)$ is a constant four-tuple. Constant time direct and indirect indexing allow us to handle (multidimensional) arrays of integers or real numbers. For example $A[i]$ is the i-th element of an array A, $A[i : j]$ is the portion of A between index i and index j, defined if $j \geq i$ and with $A[i : i] = A[i]$. The expression $A[i, \cdot]$ denotes the row vector formed by all elements in row i. In what follows the terms array and list will be used interchangeably.

In a single computation step each processor can perform any arithmetic operation on the proper number of operands stored in the shared memory. Notation $a \leftarrow b + c$ is a shorthand for the sequence

$$\mathrm{fetch}(R_1, b); \mathrm{fetch}(R_2, c); R_3 \leftarrow R_1 + R_2; \mathrm{store}(a, R_3);$$

including three memory transfer operations and one arithmetic manipulation. However given the asymptotic nature of the complexity results proved in Chapter 2, $a \leftarrow b + c$ will be counted as a single step. Operands R_1, R_2 and R_3 are registers local to each processor: each processor has available a constant number of them (the exact value of this constant being irrelevant). All the algorithms do not use shared access to the same memory location in a single time step. A PRAM enforcing this constraint is called *Exclusive Read Exclusive Write* PRAM (or EREW PRAM). Other models allowing some degree of concurrent access are the *Concurrent Read Exclusive Write* PRAM (or CREW PRAM) and the *Concurrent Read Concurrent Write* PRAM (or CRCW PRAM).

A PRAM algorithm will normally be specified as a sequence of instructions in a PASCAL-like pseudo-code. The most important syntactic constructs are listed below.

- Assignments such as $a \leftarrow b + c$ are the simplest instructions.
- Programs will contain typical control structures like **if-then-else** conditionals, **for** and **while** loops. In particular when similar tasks need to be performed on different data a parallel **for** loop statement might be used. The syntax will be as follows

  for all $x \in X$ **in parallel do** *instructions*

  Indentation will show nested instructions.

The complexity measures we use are (parallel) *running time* and *number of processors*, normally expressed in asymptotic notation and as a function of the input length. By efficient algorithms we mean algorithms that run in polylogarithmic (i.e. $O(\log^k n)$ for some constant k) time in the input length using a polynomial number of processors. Such problems define the complexity class

NC. A PRAM algorithm, in particular, is said to be *optimal* (see [GR88]) if the product of its parallel running time $t(n)$ with the number of processors used $p(n)$ is within a constant factor of the computation time of the fastest existing sequential algorithm. The quantity $w(n) = t(n) \cdot p(n)$ is called *work*.

Function and procedure names will often be used as macros. In particular the following pre-defined subroutines are assumed.

1. A function copying in $O(\log n)$ time using $O(n/\log n)$ processors an element x across all positions of an array of n elements. The syntax will be copy(x, n) and the result will be an array of n elements all equal to x.

2. A function computing an associative operation on a list of n elements in a certain domain, in $O(\log n)$ time using $O(n/\log n)$ processors. The function will have as parameters the list, its size n, the operation to be performed and will return a value of the appropriate type. The correct syntax for the function computing the sum of the n elements of a list L is tree$(L, n, +)$.

3. If $+$ is an associative operation over some domain D and $L[1], \ldots, L[n]$ is an array of elements of D the *prefix sums problem* is to compute the n prefix sums $S[i] = \sum_{j=1}^{i} L[j]$ for $i = 1, \ldots, n$. The algorithms in Chapter 2 will make use of a function prefix, computing the prefix sums of a list of n elements in parallel (this is also known as *parallel prefix computation*). If L is a list of n elements and $\times$ is the integer multiplication, then the instruction $R \leftarrow$ prefix$(L, n, \times)$ is carried out in $O(\log n)$ parallel steps using $O(n/\log n)$ processors, if n is the number of element of the list L. After this instruction it will be $R[i] = \prod_{j=1}^{i} L[j]$.

The following definition (essentially from [Joh90]) captures the class of PRAM algorithms which will be of interest in Chapter 2.

Definition 1 *A search problem belongs to the class RNC if there exists a randomised PRAM algorithm $\mathcal{A}$ whose running time is polylogarithmic which uses a polynomial number of processors and such that*

1. *if $\mathcal{SOL}(x) \neq \emptyset$ then $\mathcal{A}$ outputs $y \in \mathcal{SOL}(x)$ with probability more than 1/2;*

2. *if $\mathcal{SOL}(x) = \emptyset$ then the output of $\mathcal{A}$ is undefined.*

Indeed, in all cases, the parallel algorithms in this thesis will satisfy a stronger condition. If $\mathcal{S}\wr\updownarrow(x) \neq \emptyset$ then the probability that the algorithm does not produce an output is bounded above by an inverse polynomial function. In all such cases we say that the algorithms succeed *with high probability*.

Sometimes it is convenient to slow down part of a parallel algorithm in order to achieve optimal work over the whole algorithm. Consider a computation that can be done in t parallel steps with x_i primitive operations at step i. Trivial parallel implementation will run in t steps on $m = \max x_i$ processors. If we have $p < m$ processors the ith step will be simulated in $\lceil x_i/p \rceil \leq x_i/p + 1$ time and so the total parallel time is no more than $\sum_{i=1}^{t} x_i/p + t$. This is known as Brent's scheduling principle [Bre74]. This is assuming that processor allocation is not a problem: for specific problems we may need to provide the processor allocation scheme explicitly (i.e. redesigning the algorithm to work using p processors). Sometimes this principle can be used to find the number of processors which gives optimal work. For examples all library functions above have optimal work whenever $p(n) = O(n/\log n)$. Of course reducing the number of processors will slow down the computation. So if the parallel prefix operation is run on a list of n elements using $n/\log^5 n$ processors the resulting algorithm runs in $O(\log^5 n)$ parallels steps.

1.2.3 Probability

Many of the results in this thesis are probabilistic. In this section some terminology and general results are given.

Following [GS92], a *probability space* is a triple $(\Omega, \Sigma, \Pr)$, where Ω is a set called a *sample space*, $\Sigma = \{E : E \subseteq \Omega\}$ is the set of *events* and $\Pr$ is a non-negative real valued measure on Σ with $\Pr[\Omega] = 1$. The elements of Ω are particular events called *elementary events*. Unless otherwise stated Ω will be a finite set and Σ will be the set of all subsets of Ω. For every $E \in \Sigma$ the *probability* of the event E, $\Pr[E] =_{df} \sum_{\omega \in E} \Pr[\omega]$.

Theorem 1 *The probabilities assigned to the elements of a sample space Ω satisfy the following properties (for every $E, F \in \Sigma$):*

1. $\Pr[E] \geq 0$.
2. *(Monotonicity) If $E \subseteq F$ then* $\Pr[E] \leq \Pr[F]$.
3. $\Pr[E \cup F] = \Pr[E] + \Pr[F] - \Pr[E \cap F]$.
4. $\Pr[\bar{E}] = 1 - \Pr[E]$.

Theorem 2 (Total probability) *If* $E_1, \ldots, E_n$ *is a partition of* Ω *with* $E_i \in \Sigma$ *for all* i *and* $E \in \Sigma$ *then* $\Pr[E] = \sum_{i=1}^n \Pr[E \cap E_i]$.

Proof. Immediate from Theorem 1.*3* since the events $E \cap E_i$ are all disjoint. $\square$

$\Pr[E|F]$ will denote the probability of the event E given that the event F has happened. If $\Pr[F] > 0$, we define $\Pr[E|F] =_{df} \Pr[E \cap F]/\Pr[F]$. A sequence of events E_i are *mutually independent* if $\Pr[E_1 \cap \ldots E_n] = \prod_{i=1}^n \Pr[E_i]$. Mutual independence between pairs of events is called *pairwise independence*.

Example. Let $\Omega = \{a, b, c, d\}$ with $\Pr[a] = q$ and $\Pr[b] = \Pr[c] = \Pr[d] = p$. Let $E_1 = \{a, b\}$, $E_2 = \{a, c\}$ and $E_3 = \{a, d\}$. $\Pr[E_i] = p + q$ and $\Pr[E_1 \cap E_2 \cap E_3] = \Pr[\{a\}] = q$. Solving $q = (p + q)^3$ with the constraint $3p + q = 1$ we get

$$\Pr[E_1 \cap E_2 \cap E_3] = \Pr[E_1] \cdot \Pr[E_2] \cdot \Pr[E_3]$$

for $p = (3 - \sqrt{3})/4$ and $q = (3\sqrt{3} - 5)/4$. On the other hand $\Pr[E_i|E_j] = q/(p + q) \neq \Pr[E_i]$. Similarly sample spaces can be built in which it is possible to construct events that are pairwise independent but not mutually independent.

A real valued *random variable* X on a probability space $(\Omega, \Sigma, \Pr)$ is a function from Ω to the set of real numbers such that for every real number x the set $\{\omega \in \Omega : X(\omega) \leq x\} \in \Sigma$. The *distribution function* of a random variable X is the function $F : \mathbb{R} \to [0, 1]$ with $F(x) = \Pr[X \leq x]$.

Moments. If h is any real-valued function on the set of real numbers $\mathbb{R}$ then the *expectation* of $h(x)$ is

$$\mathrm{E}(h(X)) =_{df} \sum_{\Omega} h(x) \Pr[X = x]$$

In particular the *mean* of a random variable X, usually denoted by μ, is $\mathrm{E}(X)$ and the *k-th moment* of X is $\mathrm{E}(X^k)$ (of course, if Ω is not finite, these quantities might not exist). The *k-th binomial moment* of X is $\mathrm{E}(\binom{X}{k})$ whereas the *k-th factorial moment* of X is $\mathrm{E}_k(X) =_{df} \mathrm{E}(X \cdot (X-1) \ldots \cdot (X-k+1))$. It follows that $\mathrm{E}_k(X) = k!\,\mathrm{E}(\binom{X}{k})$.

Theorem 3 *If* $X = \sum X_i$ *then* $\mathrm{E}(X) = \sum \mathrm{E}(X_i)$.

This result, known as *linearity of expectation*, (the proof follows immediately from the definition of expectation) is a very useful tool for computing the mean of a random variable. For example if the value of X is the sum of a number of very simple random variables X_i then the mean of X is easily defined in terms of the means of the X_i.

The *variance* of X, usually denoted by σ^2, is defined by $\mathrm{Var}(X) =_{df} \mathrm{E}((X-\mu)^2) = \mathrm{E}(X^2) - \mu^2$.

Theorem 4 *If* $X = \sum X_i$ *and the* X_i *are pairwise independent then* $\mathrm{Var}(X) = \sum \mathrm{Var}(X_i)$.

Theorem 5 *If* X *is a positive random variable then* $\Pr[X \geq \lambda] \leq \mathrm{E}(X^k)/\lambda^k$ *for every* $\lambda > 0$ *and integer* $k > 0$.

Proof. By definition $\mathrm{E}(X^k) \geq \sum_{x \geq \lambda} x^k \Pr[X = x]$. If $x \geq \lambda$ then the sum above is lower bounded by $\lambda^k \Pr[X \geq \lambda]$ and the result follows. □

Theorem 5 has many useful special cases, depending on the choices of λ and k.

Theorem 6 (Markov inequality) $\Pr[X > 0] \leq \mathrm{E}(X)$.

Theorem 7 (Chebyshev inequality) $\Pr[|X - \mathrm{E}(X)| \geq \lambda \cdot \sigma] \leq \lambda^{-2}$.

An important use of Chebyshev inequality is in proving that a positive random variable takes a value larger than zero with "high" probability (this property will be used repeatedly for example in Chapter 4).

Corollary 1 *If* $X \geq 0$ *then* $\Pr[X = 0] \leq \mathrm{Var}(X)/\mathrm{E}(X)^2$.

Proof. $\Pr[X = 0] \leq \Pr[|X - \mathrm{E}(X)| \geq \lambda \cdot \sigma]$ if $\lambda = \mu/\sigma$. □

So assuming that a natural number n can be associated as a parameter with the elements of the sample space under consideration, and that $\mathrm{E}(X)$ and $\mathrm{Var}(X)$ are thus functions of n, if $\lim_{n\to\infty} \mathrm{E}(X) = \infty$ and $\mathrm{Var}(X) = o(\mathrm{E}(X)^2)$ then the last corollary implies that $\Pr[X = 0]$ becomes smaller and smaller as a function of n.

Distributions. We now briefly review the discrete probability distributions that will be used in later chapters. The *discrete uniform distribution* on a finite sample space Ω containing n elements is defined by

$$\Pr[\omega_i] = \frac{1}{n} \qquad\qquad \forall i \in \{1, \ldots, n\}$$

and in this case we say that ω_i is generated uniformly at random. The random variable $X : \Omega \to \{1, \ldots, n\}$ defined by $X(\omega_i) = i$ has discrete uniform distribution (or equivalently that its values are distributed uniformly over Ω). Using the following simple identities, which can be easily proved by induction on n,

$$\sum_{i=1}^{n} i = \frac{n(n+1)}{2} \qquad\qquad \sum_{i=1}^{n} i^2 = \frac{n(n+1)(2n+1)}{6}$$

it is possible to derive

$$\mathrm{E}(X) = \frac{1}{n}\sum_{i=1}^{n} i = \frac{1}{n}\cdot\frac{n(n+1)}{2} = \frac{n+1}{2}$$

$$\begin{aligned}
\mathrm{Var}(X) &= \frac{1}{n}\sum_{i=1}^{n}\left(i-\frac{n+1}{2}\right)^2 \\
&= \frac{1}{n}\left[\sum_{i=1}^{n} i^2 - \sum_{i=1}^{n} i(n+1) + \sum_{i=1}^{n}\frac{(n+1)^2}{4}\right] \\
&= \frac{1}{n}\sum_{i=1}^{n} i^2 - \sum_{i=1}^{n} i - \frac{n+1}{2} + \frac{(n+1)^2}{4} \\
&= \frac{1}{n}\sum_{i=1}^{n} i^2 - \frac{n(n+1)}{2} + \frac{n^2-1}{4} \\
&= \frac{1}{n}\sum_{i=1}^{n} i^2 - \left(\frac{n+1}{2}\right)^2 \\
&= \frac{(n+1)(2n+1)}{6} - \left(\frac{n+1}{2}\right)^2 = \frac{n^2-1}{12}
\end{aligned}$$

If $X : \Omega \to \{0,1\}$ and $\Pr[X = 1] = p$ then X is called a *0–1-random variable* or *random indicator*. 0–1-random variables model an important class of random processes called *Bernoulli trials*. During one of these trials an experiment is performed which *succeeds* with a certain positive probability p. In particular from now on we will always abbreviate $\Pr[X = 1]$ by $\Pr[X]$ and $\Pr[X = 0]$ by $\Pr[\overline{X}]$. We have

$$\mathrm{E}(X) = 0\cdot(1-p) + 1\cdot p = p$$

$$\mathrm{Var}(X) = (0-p)^2\cdot(1-p) + (1-p)^2\cdot p = (1-p)(p^2+p-p^2) = p(1-p)$$

Random indicators have many applications in probability. For example they can be used to estimate the variance of a random variable.

Theorem 8 *If X can be decomposed in the sum of n not necessarily independent random indicators then*

1. $\mathrm{Var}(X) \le 2\sum_{\{i,j\}}\Pr[X_i \wedge X_j] + \mathrm{E}(X)$ *where the sum is over all 2-sets on* $\{1,\ldots,n\}$.

2. $\mathrm{Var}(X) \le \mathrm{E}_2(X) + O(\mathrm{E}(X))$.

Proof. It follows from the definition that $\mathrm{Var}(X) \leq \mathrm{E}(X^2)$. For every real number $x > 0$ we can write $x^2 = x + 2\binom{x}{2}$. Hence $\mathrm{E}(X^2) = \mathrm{E}(X) + 2\mathrm{E}(\binom{X}{2})$. If $X = \sum_i X_i$ then $\binom{X}{2}$ is the number of ways in which two different X_i can assume the value one, disregarding the ordering. So $\mathrm{E}(\binom{X}{2}) = \sum_{\{i,j\}} \mathrm{E}(\{X_i, X_j\}) = \sum_{\{i,j\}} \Pr[X_i \wedge X_j]$ over all $\{i,j\} \subset \{1,\ldots,n\}$.

The second inequality is trivial since $\mathrm{E}_2(X) = 2\mathrm{E}(\binom{X}{2})$. □

The proof of Theorem 8 gives a combinatorial meaning to $\mathrm{E}(\binom{X}{2})$ in terms of the random indicators X_i. If $X = \sum_{i=1}^n X_i$ where X_i are random indicators also the k-moment and the k-th factorial moment of X have an interpretation in terms of the X_i. $\mathrm{E}(X^2)$ is the sum over all pairs of (not necessarily distinct) i and j of $\Pr[X_i \wedge X_j]$ where $\mathrm{E}_2(X)$ is the sum over all ordered pairs of distinct i and j of $\Pr[X_i \wedge X_j]$.

If X_i are n independent random indicators with common success probability equal to p then $X = \sum_{i=1}^n X_i$ has *binomial distribution* with parameters n and p. Simple calculations (using Theorem 3 and 4) imply

$$\mathrm{E}(X) = np \qquad\qquad \mathrm{Var}(X) = np(1-p)$$

If a sequence of identical independent random experiments is performed with common success probability equal to p then the random variable Y_1 counting the number of trials up to the first success has *geometric distribution* with parameter p. $\Pr[Y_1 = k] = p(1-p)^{k-1}$ hence using the binomial theorem and some easy properties of power series

$$\mathrm{E}(Y_1) = 1/p \qquad\qquad \mathrm{Var}(Y_1) = \frac{1-p}{p^2}$$

In the same setting as above Y_k counting the number of trial up to the k-th success has the *Pascal distribution* (or *negative binomial distribution*). $\Pr[Y_k = n] = \binom{n-1}{k-1}p^k(1-p)^{n-k}$. Since each trial is independent $Y_k = \sum_{j=1}^k Y_1^j$ where the Y_1^j have a geometric distribution. Hence by Theorem 3 and Theorem 4 and the results for the geometric distribution we have

$$\mathrm{E}(Y_k) = k/p \qquad\qquad \mathrm{Var}(Y_k) = \frac{k(1-p)}{p^2}$$

Improved Tail Inequalities. The beauty of Theorem 5 resides in the fact the only assumption made on X is on the existence of $\mathrm{E}(X^k)$. If more accurate information is available it is possible to improve considerably the quality of the results. The following Theorem states a couple of inequalities proved in [Hm90].

Theorem 9 *Let $n \in \mathbb{N}$ and let $p_1, \ldots, p_n \in \mathbb{R}$ with $0 \leq p_i \leq 1$, $i = 1, \ldots, n$. Put $p = (1/n) \sum p_i$ and $m = np$ and let $X_1, \ldots, X_n$ be independent 0-1 random variables with $\Pr(X_i) = p_i$, $i = 1, \ldots, n$. Let $S = \sum X_i$. Then*

$$\Pr(S \geq (1+\epsilon)m) \leq \mathrm{e}^{-\epsilon^2 m/3},\ 0 \leq \epsilon \leq 1$$

and

$$\Pr(S \leq (1-\epsilon)m) \leq \mathrm{e}^{-\epsilon^2 m/2},\ 0 \leq \epsilon \leq 1.$$

In most cases the *Chernoff bounds* stated above will be used on a sequence of n independent identically distributed 0-1 random variables. Under these assumptions, S has binomial distribution and some improved bounds are possible (see [Bol85, Ch. I]).

1.2.4 Graphs

Most of the graph-theoretic terminology will be taken from [Har69] and [Bol79]. A (*simple undirected*) *graph* $G = (V, E)$ is a pair consisting of a finite nonempty set $V = V(G)$ of *vertices* (or *nodes* or *points*) and a collection $E = E(G)$ of distinct subsets of V each consisting of two elements called *edges* (or *lines*). If $e = \{u, v\} \in E$ then the vertices u and v are *adjacent*, vertex u and the whole edge e are *incident* (or else we say that u belongs to e, sometimes using the set-theoretic notation $u \in e$). Also if $f = \{v, w\} \in E$ then e and f are incident. If $F \subseteq E(G)$ then $V(F)$ is the set of vertices incident to some $e \in F$. For every $U \subseteq V(G)$, $N(U)$ will denote the set of vertices adjacent to some $v \in U$ and not belonging to U. If $U = \{v\}$ we write $N(v)$ instead of $N(\{v\})$. If $U, W \subseteq V$ then cut(U, V) is the set of edges having one endpoint in U and the other in W.

The *degree* of a vertex v is defined as $\deg_G v =_{df} |N(v)|$. The *minimum* (resp. *maximum*) *degree* of G is $\delta = \delta(G) = \min_{v \in V} \deg_G v$ (resp. $\Delta = \Delta(G) = \max_{v \in V} \deg_G v$). For all $i \in \{0, \ldots, n-1\}$ let $V_i(G) = \{v \in V : \deg_G v = i\}$. A *multiset* is a collection of objects in which a single object can appear several time. A *multigraph* is a pair $H = (U, E)$ in which U is the set of vertices and E is a multiset of edges. If e appears $x_e > 1$ times in E then each of its occurrences is a *parallel edge*. The *skeleton* of a multigraph $H = (U, E)$ is a graph G with $V(G) = U$ and $E(G)$ containing a single copy of every parallel edge in H plus all the $e \in E$ with $x_e = 1$. A graph is *directed* if the edges are ordered pairs. Round brackets will enclose vertices belonging to a directed edge.

A graph is *labelled* if its vertices are distinguished from one another by names. Figure 1.2 shows the 64 different labelled graphs on four vertices. Some of these graphs only differ for the

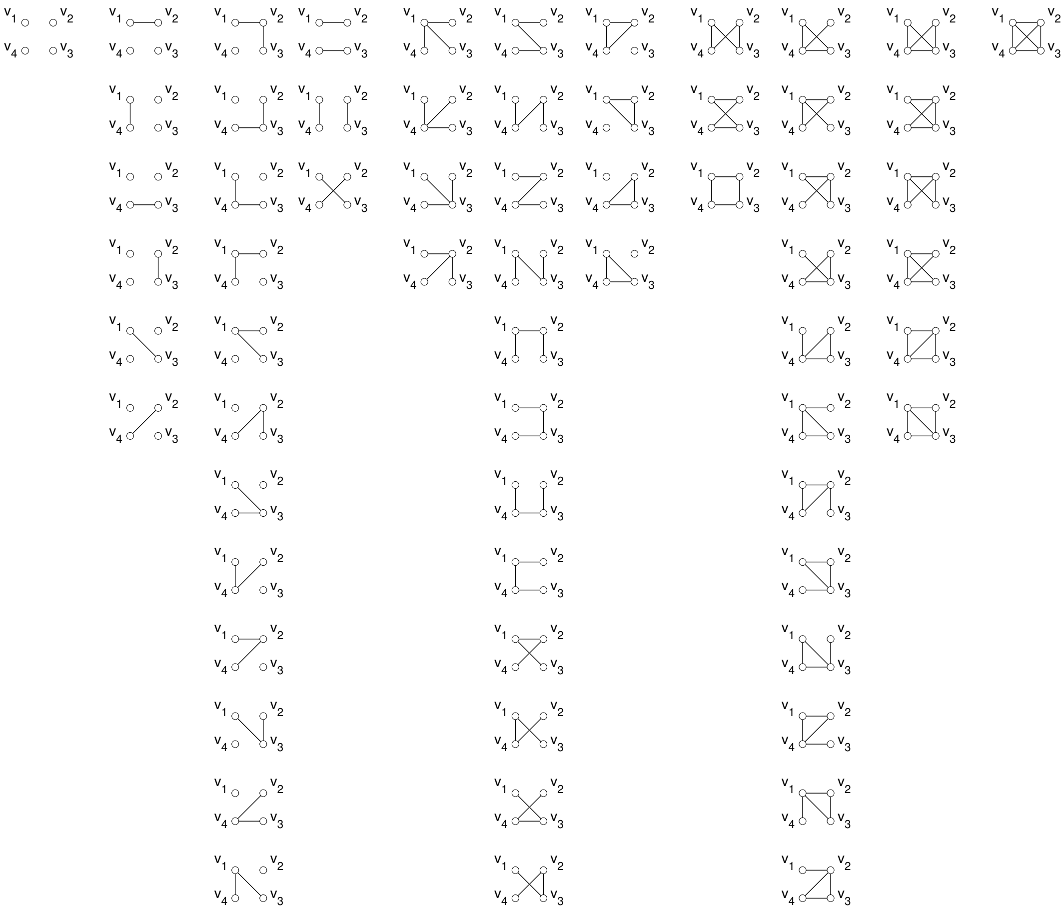

Figure 1.2: The 64 distinct labelled graphs on 4 vertices.

labelling of their vertices, their topological structure is the same. More formally, two graphs G_1 and G_2 are *isomorphic* if there is a one-to-one correspondence between their labels which preserves adjacencies. A graph is *unlabelled* if it is considered disregarding all possible labelling of its vertices that preserve adjacencies. Figure 1.3 shows the eleven unlabelled graphs on four vertices.

A graph is completely determined by either its adjacencies or its incidences. This information can be conveniently stated in matrix form. The *adjacency matrix* of a labelled undirected (resp. directed) graph $G = (V, E)$ with n vertices, is an $n \times n$ matrix A such that, for all $v_i, v_j \in V$, $A_{i,j} = 1$ if v_i is adjacent to v_j (resp. if $(v_i, v_j) \in E$) and $A_{i,j} = 0$ otherwise.

A *subgraph* of $G = (V, E)$ is a graph $H = (W, F)$ with $W \subseteq V$ and $F \subseteq E$. H is a *spanning subgraph* if $W = V$ and it is an *induced subgraph* if whenever $u, v \in W$ with $\{u, v\} \in E$ then $\{u, v\} \in F$. If $W \subseteq V(G)$ we will denote by $G[W]$ the induced subgraph of G with vertex set W. K_n is the *complete* simple graph on n vertices. It has $n(n-1)/2$ edges. Every graph on n vertices is a subgraph of K_n.

A graph $G = (V, E)$ is *bipartite* if V can be partitioned in two sets V_1 and V_2 such that every line of G joins a vertex in V_1 with a vertex in V_2. K_{n_1,n_2} is the complete bipartite graph on $n = n_1 + n_2$ vertices. A graph is *planar* if it can be drawn on the plane so that no two edges intersect.

If G is a graph and $v \in V$ then $G - v$ is the graph obtained from G by removing v and all edges incident to it; if $v \notin V$ then $G + v = (V \cup v, E)$. If $e = \{u, v\} \in E$ then $G - e = (V, E \setminus \{e\})$ and $G + e = (V \cup \{u, v\}, E \cup e)$. These operations extend naturally to sets of vertices and edges.

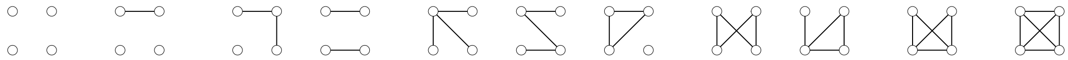

Figure 1.3: The 11 distinct unlabelled graphs on 4 vertices.

A *path* in a graph $G = (V, E)$ is an ordered sequence of vertices formed by a *starting vertex* v followed by a path whose starting vertex belongs to $N(v)$. The path is *simple* if all vertices in the sequence are distinct. The *length of a path* $P = (v_1, \ldots, v_k)$ is $k - 1$. A *cycle* is a simple path $P = (v_1, \ldots, v_k)$ such that $v_1 = v_k$. A single vertex is a cycle of length zero. Since $v \notin N(v)$ there is no cycle of length one. An edge $\{u, v\} \in E$ belongs to a path $P = (v_1, \ldots, v_k)$ if there exists $i \in \{1, \ldots, k - 1\}$ such that $\{u, v\} = \{v_i, v_{i+1}\}$. Two vertices u and v in a graph are *connected* if there is a path $P = (v_1, \ldots, v_k)$ such that $\{u, v\} = \{v_1, v_k\}$. The *distance* $dst_G(u, v)$ between them is the length of a shortest path between them. The subscript G will be omitted when clear from the context. A *connected component* is a subgraph whose vertex set is $U \subseteq V$, such that all $u, v \in U$ are connected and no $v \in V \setminus U$ is connected to some $u \in U$.

1.2.5 Random Graphs

Let $\mathcal{G}^{n,m}$ be the set of all (labelled and undirected) graphs with n vertices and m edges. If $N = \binom{n}{2}$ and $\mathcal{G}^n = \bigcup_{m=0}^{N} \mathcal{G}^{n,m}$ then $|\mathcal{G}^{n,m}| = \binom{N}{m}$ and $|\mathcal{G}^n| = 2^N$. Informally, a *random graph* is a pair formed by an element G of $\mathcal{G}^n$ along with a non-negative real value p_G such that $\sum_{G \in \mathcal{G}^n} p_G = 1$. In other words random graphs are elements of a probability space associated with $\mathcal{G}^n$, called the *random graph model*. There are several random graph models. In most cases the set of events is the set of all subsets of $\mathcal{G}^n$ and the definition is completed by giving a probability to each $G \in \mathcal{G}^n$. If Γ is a random graph model we will write $G \in \Gamma$ to mean that $\Pr[G]$ is defined according to the given model.

The probability space $\mathcal{G}(n, m)$ is obtained by assigning the same probability to all graphs on

n vertices and m edges and assigning to all other graphs probability zero. For each $m = 0, 1, \ldots, N$, $\mathcal{G}(n, m)$ has $\binom{N}{m}$ elements that occur with the same probability $\binom{N}{m}^{-1}$. Sometimes the alternative notation $\mathcal{G}(K_n, m)$ is used instead of $\mathcal{G}(n, m)$, where K_n is called the *base* graph since the elements of the sample space are all subgraphs of the complete graph. Variants of $\mathcal{G}(n, m)$ are thus obtained by changing the base graph. For example the sample space of $\mathcal{G}(K_{n,n}, m)$ is the set of all bipartite graphs on $n + n$ vertices and m edges. This is made into a probability space by giving the same probability to all such graphs.

In the model $\mathcal{G}(n, p)$ (sometimes denoted by $\mathcal{G}(K_n, p)$) we have $0 < p < 1$ and the model consists of all graphs with n labelled vertices in which edges are chosen independently and with probability p. In other words if $G \in \mathcal{G}(n, p)$ and $|E(G)| = m$ then $\Pr[G] = p^m(1-p)^{N-m}$. A variant of $\mathcal{G}(n, p)$ is $\mathcal{G}(K_n, (p_{i,j}))$ in which edge $\{i, j\}$ is selected to be part of the graph or not with probability $p_{i,j}$. So for example $\mathcal{G}(K_{n,n}, p)$, whose sample space is the set of bipartite graphs on $n + n$ vertices in which each edge is present with probability p, is indeed an instance of $\mathcal{G}(K_{2n}, (p_{i,j}))$.

To avoid undesired inconsistencies it is important that under fairly general assumptions results obtained on one model translate to results in another model. A property Q holds *almost always* (or a.a.), for *almost all graphs* or *almost everywhere* (a.e.) if $\lim_{n\to\infty} \Pr[G \in Q] = 1$. The following theorem, reported in [Bol85, Ch.II], relates $\mathcal{G}(n, p)$ and $\mathcal{G}(n, m)$.

Theorem 10 *(i) Let Q be any property and suppose that $\lim_{n\to\infty} p(1-p)N = +\infty$. Then the following two assertions are equivalent.*

1. *Almost every graph in $\mathcal{G}(n, p)$ has Q.*
2. *Given $x > 0$ and $\epsilon > 0$, if n is sufficiently large, there are $l \geq (1-\epsilon)2x\sqrt{p(1-p)N}$ integers $M_1, \ldots, M_l$ with*

$$pN - x\sqrt{p(1-p)N} < M_1 < M_2 < \ldots < M_l < pN + x\sqrt{p(1-p)N}$$

such that $\Pr_{M_i}[Q] > 1 - \epsilon$ for every $i = 1, \ldots, l$.

(ii) If Q is a convex property and $\lim_{n\to\infty} p(1-p)N = +\infty$, then almost every graph in $\mathcal{G}(n, p)$ has Q, where $M = \lfloor pN + x\sqrt{p(1-p)N} \rfloor$.

(iii) If Q is a property and $0 < p = M/N < 1$ then

$$\Pr_M[Q] \leq \Pr_p[Q] \mathrm{e}^{1/6M} \sqrt{2\pi p(1-p)N} \leq 3\sqrt{M} \Pr_p[Q]$$

The success of a random graph model depends on many factors. From a practical point of view the model must be reasonable in terms of real world problems and it must be computationally easy to generate graphs according to the specific distribution assigned by the model. From the theoretical point of view the choice of one model over another depends on the specific problem at hand and it is often a matter of trading-off the simplicity of combinatorial calculations performed under the assumption that a given graph was sampled according to a certain model, for the tightness of the desired results. $\mathcal{G}(n,m)$ often gives sharper results but it is sometimes more difficult to handle than $\mathcal{G}(n,p)$. In Chapter 3 a slightly different model will be used which keeps the good features of $\mathcal{G}(n,m)$ and is easier to analyse. Let $\mathcal{M}^{n,m}$ be the set of all (labelled and undirected) multigraphs on n vertices and m edges; let $\mathcal{M}^n = \bigcup_{m=0}^{\infty} \mathcal{M}^{n,m}$. $\mathcal{M}(n,m)$ is the probability space whose sample space is the set of pairs (M,σ) where $M \in \mathcal{M}^{n,m}$ and σ is a permutation of m objects (see Section 1.2.6 for further details on permutation groups) giving an ordering on the m edges of M. The probability measure on the sample space assigns the same probability N^{-m} to all elements of $\mathcal{M}^{n,m} \times S_m$. Strictly speaking, $\mathcal{M}(n,m)$ is a *random multigraph* model. Figure 1.4 shows

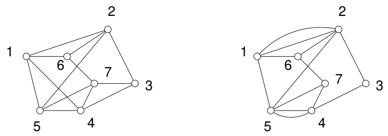

Figure 1.4: Examples of random graphs

a graph on 7 vertices and 13 edges and a multigraph with the same number of vertices and edges. In particular the multigraph shown on the right corresponds to $m!$ elements of the sample space of $\mathcal{M}(n,m)$, one for each possible ordering. The model is somehow intermediate between $\mathcal{G}(n,m)$ and the *uniform model* or *multigraph process model* as defined in [JKŁP93] in which m ordered pairs of (not necessarily distinct) elements of $[n] = \{1, 2, \ldots, n\}$ are sampled.

The practical significance of $\mathcal{M}(n,m)$ is supported by the very simple process which enables us to generate an element in this space: for m times select uniformly at random an element in $[n]^{(2)}$, the set of unordered pairs of integers in $[n] = \{1, \ldots, n\}$.

Again a result that relates properties of $\mathcal{M}(n,m)$ to those of $\mathcal{G}(n,m)$ is needed. The following suffices for the purposes of Chapter 3.

Theorem 11 *Let X and Y be two random variables defined respectively on $\mathcal{G}(n,m)$ and $\mathcal{M}(n,m)$. If $X(G) = Y(G)$ for every $G \in \mathcal{G}^{n,m} \cap \mathcal{M}^{n,m}$ and $m = cn$ then* $\mathrm{E}(X) \leq O(\mathrm{E}(Y))$.

Proof.

$$\begin{aligned}
\mathrm{E}(X) &= \sum_{G \in \mathcal{G}^{n,m}} X(G) \frac{m!\,(N-m)!}{N!} \\
&\leq \sum_{G \in \mathcal{G}^{n,m}} X(G) \frac{m!}{(N-m)^m} \\
&= \sum_{G \in \mathcal{G}^{n,m}} X(G) \frac{m!}{N^m} \left(1 - \frac{m}{N}\right)^{-m}
\end{aligned}$$

If $m = cn$ since $N = n(n-1)/2$ we have

$$\left(1 - \frac{m}{N}\right)^{-m} \leq \mathrm{e}^{\frac{c^2 n}{n-1}}$$

which is asymptotic to e^{c^2}. Hence

$$\mathrm{E}(X) \leq O(1) \sum_{G \in \mathcal{G}^{n,m}} X(G) \frac{m!}{N^m}$$

For every simple graph G with m edges there are exactly $m!$ elements of $(G, \sigma) \in \mathcal{M}^{n,m} \times S_m$ Since $X(G) = Y(G)$ for every $G \in \mathcal{G}^{n,m} \cap \mathcal{M}^{n,m}$ we can write $\mathrm{E}(X) \leq O(1) \cdot \mathrm{E}(Y)$. □

1.2.6 Group Theory

Most of the definitions and the results in this section are taken from [Rot65].

Basic Definitions. A *group* is an ordered pair $(\mathcal{G}, *)$ where G is a set and $*$ is a binary operation on G satisfying the following properties:

g1 $g_1 * (g_2 * g_3) = (g_1 * g_2) * g_3$ for all $g_1, g_2, g_3 \in \mathcal{G}$.

g2 There exists $id \in \mathcal{G}$ (the *identity*) such that $g * id = g = id * g$ for all $g \in \mathcal{G}$.

g3 For all $g_1 \in \mathcal{G}$ there exists $g_2 \in \mathcal{G}$ (the *inverse* of g_1, often denoted by g_1^{-1}) such that $g_1 * g_2 = id = g_2 * g_1$.

If X is a nonempty set, a *permutation* of X is a bijective function g on X. Let S_X denote the set of permutations on X. Although most of the definitions are general, in all our subsequent discussion X will be the set $[n]$.

There are many ways to represent permutations. We will normally use the *cycle notation* defined as follows:

(1) Given g, draw n points labelled with the numbers in $[n]$.

(2) Join the point labelled i to the point labelled j by an edge with an arrow pointing towards j if $g(i) = j$. (This will form a number of cycles).

(3) Write down a list $(i_1, i_2, \ldots, i_k)$ for each cycle formed in step (2).

(4) Remove all lists formed by a single element.

So for example $g \in S_6$ with $g(1) = 3$, $g(2) = 2$, $g(3) = 4$ $g(4) = 1$, $g(5) = 6$ and $g(6) = 5$ will be represented as $(1\ 3\ 4)(5\ 6)$. It is possible to associate a unique multiset $\{1 : k_1, 2 : k_2, \ldots, n : k_n\}$ (sometimes represented symbolically as $x_1^{k_1} x_2^{k_2} \ldots x_n^{k_n}$ or simply $[k_1, k_2, \ldots, k_n]$) to every permutation $g \in S_n$ describing its *cycle structure* (or *cycle type*): g has k_i cycles of length i. In particular k_1 is the number of elements of $[n]$ that are *fixed* by g, i.e. such that $g(i) = i$. If $g(i) \neq i$ we say that g *moves* i.

If $g_1, g_2 \in S_X$ then $g_1 \circ g_2$ is a new function on X such that $(g_1 \circ g_2)(x) =_{df} g_1(g_2(x))$. It is easy to verify that $g_1 \circ g_2 \in S_X$. The pair $(S_X, \circ)$ is indeed a group called the *symmetric group* on X. S_n will denote both the set $S_{[n]}$ and the group $(S_{[n]}, \circ)$.

Subgroups and Lagrange Theorem. If $(\mathcal{G}, *)$ is a group, a nonempty subset H of $\mathcal{G}$ is a *subgroup* of $(\mathcal{G}, *)$ if

sg1 $g_1 * g_2 \in H$ for all $g_1, g_2 \in H$.

sg2 The identity of $(\mathcal{G}, *)$ belongs to H.

sg3 $g^{-1} \in H$ for all $g \in H$.

Theorem 12 *If H is a subgroup of a group $\mathcal{G}$ then there exists $m \in \mathbb{N}^+$ such that $|\mathcal{G}| = m|H|$.*

Proof. (Sketch, see [Rot65] for details) Given H and $g \in \mathcal{G}$, define the set $gH = \{g * h : h \in H\}$. It follows from **g1-g3** and **sg1-sg3** that

1. $|gH| = |H|$ for all $g \in \mathcal{G}$.

2. If $g_1 \neq g_2 \in \mathcal{G}$ then either $g_1 H = g_2 H$ or $g_1 H \cap g_2 H = \emptyset$.

3. For all $g_1 \in \mathcal{G}$ there exists $g_2 \in \mathcal{G}$ such that $g_1 \in g_2 H$.

So there exists an m such that $\mathcal{G} = g_1 H \cup \ldots \cup g_m H$ and the sets $g_i H$ form a partition of $\mathcal{G}$. $\square$

The study of permutation groups is strictly related to the study of graphs because a graph provides a picture of a particular type of subgroup in S_n.

Definition 2 [HP73] *Given a graph $G = (V, E)$ the collection of all permutations $g \in S_V$ such that $\{u, v\} \in E$ if and only if $\{g(u), g(v)\} \in E$ for all $u, v \in V$ is the* automorphism group *of G and is denoted by* $\mathrm{Aut}(G)$.

The structure and the properties of the automorphism group of a graph are of particular importance in the study of unlabelled graphs and isomorphisms between labelled graphs.

Action Theory. A group $(\mathcal{G}, *)$ *acts on* a set Ω if there is a function (called *action*) $\cdot : \mathcal{G} \times \Omega \to \Omega$ such that

1. $id \cdot \alpha = \alpha$ for each $\alpha \in \Omega$.

2. $g_1 \cdot (g_2 \cdot \alpha) = (g_1 * g_2) \cdot \alpha$ for all $g_1, g_2 \in \mathcal{G}$ and $\alpha \in \Omega$.

The action of $\mathcal{G}$ on Ω induces an equivalence relation $\sim$ on Ω ($\alpha \sim \beta$ if and only if $\alpha = g\beta$ for some $g \in \mathcal{G}$). The equivalence classes are called *orbits*. For each $g \in \mathcal{G}$, define $Fix(g) = \{\alpha \in \Omega : g \cdot \alpha = \alpha\}$ and conversely for each $\alpha \in \Omega$ define the *stabilizer* of α to be the set $\Gamma_\alpha = \{g \in \mathcal{G} : g \cdot \alpha = \alpha\}$.

Lemma 1 *Γ_α is a subgroup of $\mathcal{G}$.*

In particular S_n can be acting on itself: $f \cdot g = f \circ g \circ f^{-1}$. In this case $\sim$ is called *conjugacy relation* and the orbits are called *conjugacy classes*. In what follows C will denote a conjugacy class in S_n.

Theorem 13 *Conjugacy classes in S_n are formed by all permutations with the same cycle type.*

Proof. If $g = \ldots(\ldots ij \ldots)\ldots$ then $h \circ g \circ h^{-1}$ has the same effect of applying h to the elements of g hence g and $h \circ g \circ h^{-1}$ have the same cycle type. Let f and g belong to the same conjugacy class. Then $f = h \circ g \circ h^{-1}$ for some $h \in S_n$. But this implies that f has the same cycle type of g.

Conversely if f and g have the same cycle type, align the two cycle notations, define h and it is easy to prove that f and g are conjugate. $\square$

Thus the number of different conjugacy classes is the same as the number of different cycles types. From now on, a conjugacy class C will be identified with the decomposition of n defining the cycle type of the permutations in C. The following result is well known (see for example [Kri86]).

Theorem 14 *The number of permutations with cycle type $[k_1, \ldots, k_n]$ is $\frac{n!}{\prod_{i=1}^{n}(i^{k_i}\, k_i!)}$.*

Proof. Given the form of the cycle notation

$$\underbrace{(-)(-)\ldots(-)}_{k_1}\underbrace{(-,-)(-,-)\ldots(-,-)}_{k_2}\ldots\underbrace{\underbrace{(-,-,\ldots,-)}_{x}}_{k_x}$$

it is possible to count the number of ways to fill it.

- There are $n!$ ways to fill the n places.
- The first k_1 unary cycles can be arranged in $k_1!$ ways.
- The k_2 cycles of length 2 can be arranged in $k_2!$ ways times for each of the k_2 cycles the possible ways to start (two). So $k_2! \cdot 2^{k_2}$ overall.
- Similarly for k_i, there are i ways to start one of the i-cycles. Hence $k_i! \cdot i^{k_i}$ ways to put ik_i chosen items in cycles of length i.

□

The following theorem states a couple of well known results which will be useful.

Theorem 15 *Let $\mathcal{G}$ be a finite group acting on a set $\Omega \neq \emptyset$.*

1. *(Orbit-Stabilizer Theorem) For each orbit ω, $|\{(g,\alpha) : \alpha \in \omega \cap Fix(g)\}| = |\mathcal{G}|$.*
2. *(Fröbenius-Burnside Lemma) The number of orbits is*

$$m = \frac{1}{|\mathcal{G}|}\sum_{\alpha\in\Omega}|\Gamma_\alpha| = \frac{1}{|\mathcal{G}|}\sum_{g\in\mathcal{G}}|Fix(g)|.$$

Proof. For each orbit ω the elements of $\{(g,\alpha) : \alpha \in \omega \cap Fix(g)\}$ are pairs with $g \in \Gamma_\alpha$ and $\alpha \in \omega$. There are $|\Gamma_\alpha| \cdot |\omega|$ of these pairs. The first result follows from Theorem 12 applied to Γ_α since there is a bijection ϕ between ω and the collection $g_i\Gamma_\alpha$: if $\beta \in \omega$ then $\beta = g \cdot \alpha$ for some $g \in \mathcal{G}$; define $\phi(\beta) = g\Gamma_\alpha$.

The first part of the second result follows from the first result. Assume there are $\omega_1, \ldots, \omega_m$ different orbits. Summing over all $\alpha \in \omega_i$ we have

$$\sum_{\alpha\in\omega_i}|\omega_i|\cdot|\Gamma_\alpha| = \sum_{\alpha\in\omega_i}|\mathcal{G}|$$

and from this

$$\sum_{\alpha\in\omega_i}|\omega_i|\cdot|\Gamma_\alpha| = |\omega_i|\cdot|\mathcal{G}|$$

and finally, simplifying on both sides

$$\sum_{\alpha \in \omega_i} |\Gamma_\alpha| = |\mathcal{G}|$$

Finally, adding over all orbits

$$\sum_{i=1}^{m} \sum_{\alpha \in \omega_i} |\Gamma_\alpha| = m|\mathcal{G}|$$

To understand the second equality observe that the sum on the left in the expression above is counting pairs (g, α) for $\alpha \in \Omega$ and $g \in \Gamma_\alpha$. This is equivalent to count pairs (g, α) for $g \in \mathcal{G}$ and $\alpha \in Fix(G)$. Hence

$$\sum_{i=1}^{m} \sum_{\alpha \in \omega_i} |\Gamma_\alpha| = \sum_{g \in \mathcal{G}} |Fix(g)|$$

□

Pair Group and Combinatorial Results. Let $S_n^{(2)}$ be the permutation group on the set of unordered pairs of numbers in $[n]$. Every permutation $g \in S_n$ induces a permutation $g^* \in S_n^{(2)}$ defined by $g^*(\{i, j\}) = \{g(i), g(j)\}$.

Theorem 16 *Let $f, g \in C \subseteq S_n$ and assume the cycle type of C is $[k_1, \ldots, k_n]$. Then*

1. $f^* \sim g^*$;
2. $|Fix(g)| = 2^{q(C)}$ *where $q(C)$ is the number of cycles of $g^* \in S_n^{(2)}$ definable in terms of g;*
3. *If $\varphi(n) =_{df} |\{x : 1 \leq x < n, \gcd(n, x) = 1\}|$ is the Euler totient function and $l(i) = \sum_{j=1}^{\lceil n/i \rceil} k_{ij}$ then*
$$q(C) = \frac{1}{2}\left\{\sum_{i=1}^{n} l(i)^2 \varphi(i) - l(1) + l(2)\right\}$$
4. $|Fix(f) \cap \omega| = |Fix(g) \cap \omega|$ *for every orbit ω.*

Proof. For every $g \in S_n$ the cycle type of g^* only depends on the cycle type of g (see for example [HP73, p. 84]). The first statement is then immediate. The second statement follows from Theorem 15.*2* and the formula for the number of unlabelled graphs given by the Pólya enumeration theorem (see [HP73, Section 4.1]). The third and fourth results are mentioned in [DW83]. □

1.3 Concluding Remarks

This chapter has provided both the algorithmic context and the necessary technical background for this thesis. A few more specific concepts will be defined in the relevant chapters. We are now in a position to apply a number of randomised techniques to several combinatorial problem areas.

Chapter 2

Parallel Uniform Generation of Unlabelled Graphs

In this chapter we look at some of the issues involved in the construction of parallel algorithms for sampling combinatorial objects uniformly at random. The focus is on the generation of *unlabelled* structures. After giving some introductory remarks, providing the main motivations and defining our notion of parallel uniform generator, in Section 2.2 we describe the main features of Dixon and Wilf's [DW83] algorithmic solution to the problem of generating an unlabelled undirected graph on a given number of vertices. We present its advantages and drawbacks and comment on the limits of some simple parallelisations. In Section 2.3 we present the major algorithmic technique that, combined with some of the features of Dixon and Wilf's solution, allows us to define our parallel generators. Section 2.4 represents a detour from the main chapter's goal. The focus is shifted to the problem of devising efficient parallel algorithms for listing integer partitions. Such an algorithm will be used as a subroutine in the uniform generation algorithms presented in the following sections. The last four sections of the chapter present the main parallel algorithmic solutions. In Section 2.5 we describe how to implement efficiently in parallel the second part of Dixon and Wilf's algorithm. The initial parallel generation problem is thus reduced to the problem of sampling correctly into an appropriate set of permutations. We then present three increasingly good methods to achieve this. Section 2.6 describes a first algorithm which shares some of the drawbacks of Dixon and Wilf's solution and, moreover does not produce an exactly uniform output. With an appropriate choice of some parameters, given the number of unlabelled graphs on n vertices, the algorithm generates one of them in $O(\log n)$ steps with optimal $O(n^2)$ work on an EREW PRAM, with high probability. Unfortunately some unlabelled graphs are much more likely to occur than others. Better algorithmic solutions are described in Section 2.7 and 2.8. We present two algorithms that, given the number of vertices and no other counting information, generate an unlabelled graph in $O(\log n)$ parallel steps and $O(n^2 \log n)$ work on an EREW PRAM, with high probability. The first one mimics in a tighter way the original Dixon and Wilf's algorithm, whereas the second one, based on a solution proposed by Wormald [Wor87], moves further from Dixon and Wilf's framework.

2.1 Introduction

Graphs are very popular partly because they may be used to model many different problems and partly because, in most real-world situations, they are relatively easy to handle computationally.

To get a better understanding of some property of a family $\mathcal{F}$ of graphs in many cases it may seem useful to list all the graphs in $\mathcal{F}$. For graphs with a large number of vertices this may become impractical because the cardinality of $\mathcal{F}$ can be super-polynomial in the order (i.e. number of vertices) of the graph. The listing of few "typical" representatives obtained by a randomised procedure which outputs graphs with a uniform distribution can be a viable alternative.

As mentioned in Section 1.1, assumptions about the input distribution sometimes make the analysis of a particular algorithm easier. Theoretical results on the performance of a given algorithm can then be tested experimentally if a procedure exists for generating input graphs according to the desired distribution.

Uniform generation problems are also related to counting problems [JVV86]. Efficient uniform generation algorithms can be exploited, through sampling techniques, to devise efficient counting algorithms and conversely counting information is often crucial in uniform generation procedures. Since counting problems are often computationally demanding tasks (see for example [Val79]), uniform generation problems are difficult as well. If the requirement of getting exact counting information or an output which is exactly uniform is relaxed then both types of problems sometimes become solvable in polynomial time.

Sequential algorithms exist for the uniform generation of several classes of combinatorial objects. The reader is referred to [NW78] for a large collection of early results and to surveys like [Tin90, Sin93, DVW96, Wil97] for a few more up-to-date results. In this chapter we address the problem of determining the parallel complexity of some uniform generation problems on graphs.

Very little seems to be known about this problem [AS94]. The problems considered are not computationally demanding and the main aim is to show the existence of good PRAM algorithms. Also the material in this thesis can be a starting point for a deeper understanding of the issues involved in the seeking of parallel algorithmic solutions to other uniform generation problems.

The definition of a parallel uniform generator can be stated in the style of [JVV86]:

Definition 3 *A randomised algorithm $\mathcal{A}$ is a* uniform generator *for a relation $\mathcal{SOL}$ if and only if*

1. *there exists a function $\phi : \Sigma^* \to (0, 1]$ such that, for all $x, y \in \Sigma^*$,*

$$\Pr[\mathcal{A}(x) = y] = \begin{cases} 0 & (x, y) \notin \mathcal{SOL} \\ \phi(x) & \text{otherwise} \end{cases}$$

2. *For all $x \in \Sigma^*$ such that $\mathcal{SOL}(x) \neq \emptyset$, $\Pr[\exists y \in \mathcal{SOL}(x) : \mathcal{A}(x) = y] \geq 1/2$.*

If the algorithm is designed for a PRAM then $\mathcal{A}$ is a *parallel* uniform generator. In this chapter we consider PRAM algorithms that run in polylogarithmic time using a polynomial number of processors. Hence Definition 3 is actually a refinement of Definition 1 with the property that the output is uniformly distributed.

In some cases the second property in the definition above is easily met. For example the algorithm below always generates (the adjacency matrix of) a labelled graph on n vertices which is distributed uniformly at random over all such graphs.

```
Input: n (* the number of vertices *)
for all i, j ∈ {1, ..., n} with i < j in parallel do
    if (rand(0, 1) > 1/2)
        G[i, j] ← 1;
        G[j, i] ← 1
    else
        G[i, j] ← 0;
        G[j, i] ← 0
```

The main focus of this chapter will be on the uniform generation of unlabelled graphs. Unlabelled generation in general shows a strong interplay between Group Theory, Combinatorics and Computer Science. In the most general (and rather informal) setting we want to sample a set of structures disregarding possible existing symmetries. Action theory gives a way to describe and account for these symmetries. Section 1.2.6 provides the reader with all relevant definitions.

Unlabelled uniformly generated random graphs are often even more useful than labelled ones for experimental studies, especially when the properties of interest are preserved under vertex relabellings. A quantity of examples come from graph-theoretic applications in Chemistry [Har69, Chapter 1]. Molecules can be represented as multi-graphs whose vertices are labelled by a particular chemical element. Different vertices may receive the same label and in many applications the only characteristic that matters is the way in which these vertices are connected. Thus these multi-graphs can be considered as unlabelled. An important open problem is to count the different possible unlabelled multi-graphs that correspond to existing molecules [GJ97].

In the next section we review the key ingredients of the basic algorithmic solution for the selection of an unlabelled undirected graph on a given number of vertices.

2.2 Sampling Orbits

The parallel algorithms for generating unlabelled graphs uniformly at random presented in the following sections are based on a general procedure $\mathcal{DW}$ for generating an orbit ω of the set Ω under the action of a permutation group $\mathcal{G}$, first described in [DW83].

(1) Select a conjugacy class $C \subseteq \mathcal{G}$ with probability $\Pr[C] = \frac{|C||Fix(g)|}{m|\mathcal{G}|}$ (by Theorem 16.2 g can be any member of C since $|Fix(g)|$ is the same for all $g \in C$; also m is the number of orbits).

(2) Select uniformly at random $\alpha \in Fix(g)$ and return its orbit.

The important observation in the analysis of $\mathcal{DW}$ (stated in Theorem 15.2) is the fact that if we list pairs (g, α) where g is a permutation and $\alpha \in \omega \cap Fig(g)$ then each orbit occurs exactly $|\mathcal{G}|$ times in this listing.

Notice that the number of orbits is an input to the algorithm above. Therefore $\mathcal{DW}$ is indeed a nice and clean example of the relationship between counting and generation mentioned above. If the number of orbits is known, using this number $\mathcal{DW}$ will return an orbit distributed uniformly over the set of all orbits.

Algorithm $\mathcal{DW}$ can be adapted to generate unlabelled graphs of given order assuming that their number is known in advance. In this case $\Omega = \mathcal{G}^n$, the set of all labelled graphs with n vertices, and $\mathcal{G} = S_n$, the symmetric group of order n. The action of S_n on $\mathcal{G}^n$ is a mapping which, given a graph and a permutation, relabels the vertices of the graph according to the permutation. The

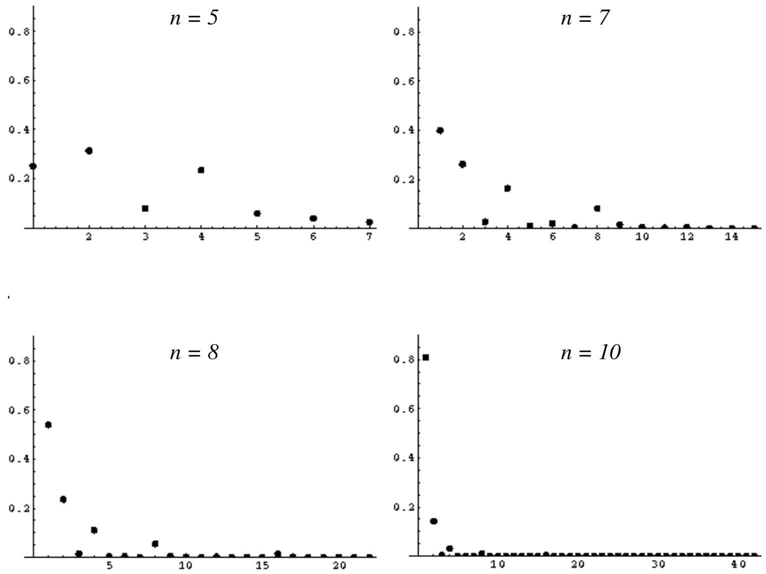

Figure 2.1: Probability distributions on conjugacy classes for $n = 5, 7, 8, 10$.

parameter m is the number of unlabelled graphs on n vertices. A sequential algorithm running in $O(n^2)$ expected time is obtained by noticing that, although the number of conjugacy classes is super-polynomial (see Theorem 19 below), the probability distribution defined in step (1) assigns very high probability to conjugacy classes containing permutations moving very few elements. Figure 2.1 (to be read in row major order) shows this distribution for values of $n = 5, 7, 8, 10$. More formally we give the following definition:

Definition 4 *If C_i and C_j are two conjugacy classes in S_n, then $C_i \prec C_j$ if the number of elements fixed by the permutations in C_i is larger than the number of elements fixed by the permutations in C_j. The partial order $\prec$ is called* weak ascending lexicographic order *(w.a.l. order for short). In particular $C_i \approx C_j$ if the permutations in the two conjugacy classes fix the same number of elements.*

Ideally we would like to prove that $\Pr[C_i]$ is a decreasing function of the number of positions that are not fixed. But this is clearly false in all the examples in Figure 2.1. However it is possible to prove monotonicity on a subset of all conjugacy classes and then, with a little more effort, to establish that the probability of choosing a small (in w.a.l. order) conjugacy class in this subset is much larger than that of choosing any other conjugacy class.

Definition 5 *A conjugacy class is called* dominant *if its cycle type is of the form $[n-2k, k, 0, \ldots, 0]$ for some $k \in \{0, \ldots, \lfloor n/2 \rfloor\}$. D_n is the family of all dominant conjugacy classes in S_n.*

It follows from the definition that if $C \in D_n$ then $\Pr[C]$ as defined in $\mathcal{DW}$ only depends on the number of cycles of length two of the permutations belonging to C.

Lemma 2 *For all n sufficiently large, if $C \in D_n$ then $\Pr[C]$ is a decreasing function of the number of cycles of length two in the permutations in C.*

Proof. Theorem 16 gives an expression for the cardinality of the set $Fix(g)$. If C is dominant then $l(1) = n - k$ for some $k \in \{0, \ldots, \lfloor n/2 \rfloor\}$, $l(2) = k$, all other $l(i)$ are null and

$$q(C) = \frac{1}{2}\{\varphi(1)(n-k)^2 + \varphi(2)k^2 - (n-k) + k\}$$

Function $\varphi(i)$, as defined in Theorem 16.*3*, is the well known Euler function giving the number of positive integers less than i that are prime to i. In particular (see, for example, [Rio58, p. 62]) $\varphi(1) = \varphi(2) = 1$. Hence

$$\begin{aligned} q(C) &= \frac{1}{2}\{n^2 - 2nk + k^2 + k^2 - n + k + k\} \\ &= \binom{n}{2} - k(n-1) + k^2 \end{aligned}$$

Thus it is possible to write

$$m \Pr[C] = \frac{2^{\binom{n}{2} - kn + k^2}}{(n-2k)!\, k!}$$

and the proof will be completed by showing that $f_n(k) : \mathbb{N} \to \mathbb{R}$ defined by

$$f_n(k) =_{df} \frac{2^{\binom{n}{2} - kn + k^2}}{(n-2k)!\, k!}$$

is a decreasing function of k. For any fixed n, the ratio

$$\frac{f_n(k)}{f_n(k+1)} = \frac{(k+1)2^{n-1-2k}}{(n-2k)(n-2k-1)}$$

If $k \leq n/2 - \log[(\sqrt{2} + \epsilon)n]$ for any $\epsilon > 0$, then

$$\frac{f_n(k)}{f_n(k+1)} \geq \frac{2^{2\log[(\sqrt{2}+\epsilon)n]-1}}{n(n-1)} = \frac{[(\sqrt{2}+\epsilon)n]^2}{2n(n-1)}$$

which is larger than one for any $n \geq 2$. If $n/2 - \log[(\sqrt{2} + \epsilon)n] < k \leq \lfloor n/2 \rfloor - 1$ then

$$\begin{aligned} \frac{f_n(k)}{f_n(k+1)} &\geq \frac{n/2 - \log[(\sqrt{2}+\epsilon)n]}{2(2\log[(\sqrt{2}+\epsilon)n] - 2)(2\log[(\sqrt{2}+\epsilon)n] - 3)} \\ &= \frac{n - 2\log[(\sqrt{2}+\epsilon)n]}{8(\log[(\sqrt{2}+\epsilon)n] - 1)(2\log[(\sqrt{2}+\epsilon)n] - 3)} \end{aligned}$$

which again is larger than one for n sufficiently large. □

Lemma 3 *For every integer k such that $k = O(n/\log n)$, if C_1 is a dominant conjugacy class with associated cycle type $[n-2k, k, 0, \ldots, 0]$, then $\Pr[C] \le \Pr[C_1]$ for every conjugacy class C with $C_1 \prec C$.*

Proof. Let C_1 be a dominant conjugacy class with associated cycle type $[n-2k, k, 0, \ldots, 0]$. By Lemma 2 we only need to prove that $\Pr[C_1] \ge \Pr[C]$ for every conjugacy class whose permutations move $2k+1$ or $2k+2$ elements. We state the argument explicitly for permutations moving $2k+1$ elements. Following [HP73] let $g_n^{(i)} = m \sum \Pr[C]$ where the sum is over all conjugacy classes whose permutations move exactly i elements. Harary and Palmer [HP73] prove that

$$g_n^{(i)} \le \frac{2^{\binom{n}{2}+(i-ni+i^2/2)/2}}{(n-i)!}$$

Let C_1 be a conjugacy class whose permutations have cycle type $[n-2k, k, 0, \ldots, 0]$. We have

$$g_n^{(2k+1)} \le \frac{2^{\binom{n}{2}-k(n-2)+k^2-\frac{n}{2}+\frac{3}{4}}}{(n-2k-1)!}$$

By Lemma 2

$$m \Pr[C_1] = \frac{2^{\binom{n}{2}-kn+k^2}}{(n-2k)!\, k!}$$

Hence if C is a conjugacy class whose permutations move $2k+1$ elements

$$\Pr[C] \le \sum_{C \text{ moving } 2k+1} \Pr[C] \le (n-2k)k!\, 2^{2k-\frac{n}{2}+\frac{3}{4}} \Pr[C_1]$$

The result follows for sufficiently large n. The argument for conjugacy classes whose permutations move $2k+2$ objects is the same and the result follows since $g_n^{(i)}$ is a decreasing function of i. $\square$

A simple way to implement the first step of algorithm $\mathcal{DW}$ is to select a random real number ξ between zero and one, to list conjugacy classes in w.a.l. order and pick the first conjugacy class for which the sum of $\frac{|C||Fix(g)|}{m|\mathcal{G}|}$ over all conjugacy classes listed so far is larger than the threshold ξ. Let t_n be the random variable counting the number of conjugacy classes listed by this algorithm. The following result is proved in [DW83].

Theorem 17 $1 \le \mathrm{E}(t_n) \le 3$ *for every $n \in \mathbb{N}$.*

The second step of $\mathcal{DW}$ can be implemented deterministically in $O(n^2)$ sequential time so that the overall algorithm has expected running time $O(n^2)$.

The same algorithm can be simulated in parallel. The following result appears in [Pu97] where, also, other parallel uniform generation algorithms for labelled graphs and subgraphs are presented.

Theorem 18 *There exists an algorithm running in $O(\log n)$ expected time using n^2 processors of a CREW PRAM which generates uniformly at random an unlabelled graph on n vertices, assuming their number is known in advance.*

This result can be improved by using the algorithms described in Section 2.5 instead of some of the procedures embedded in the proof of Theorem 18 to obtain an optimal EREW algorithm. However there are two major problems which are inherent to Dixon and Wilf's algorithmic solution. First of all, the number of unlabelled graphs of order n is assumed to be known. Although an exact formula for this number exists, its computation uses a listing of all conjugacy classes in S_n. The reader is referred to Section 2.4 for further details on the complexity of listing conjugacy classes. Secondly it is not easy to convert $\mathcal{DW}$ into an RNC algorithm which succeeds with high probability.

By the Markov inequality and monotonicity of probability, Theorem 17 implies that, for every $\lambda > 2$ with probability at least $1 - \lambda^{-1}$, t_n is smaller than 3λ. It might then be argued that fixing λ (for example to some polynomial function of the number of vertices), selecting a random $\xi \in [0, 1]$ and listing at most 3λ conjugacy classes during step (1), provides an algorithm which always runs in polynomial time and returns a graph with probability at least $1/2$. In Section 2.6 a parallelisation of $\mathcal{DW}$ based on this idea is described. The resulting algorithm belongs to RNC and achieves optimal work and very low failure probability. Unfortunately it will be shown that the distribution over the output graphs is not completely uniform. In a sense, using $\mathcal{DW}$, exact uniformity seems possible only if the whole process is allowed to run for a super-polynomial number of steps from time to time.

In Section 2.7 and 2.8 a combination of the main ideas in $\mathcal{DW}$ and a different technique presented in Section 2.3 will result in RNC algorithms with exactly uniform output probability. Lower failure probability will be traded-off for higher efficiency.

2.3 Restarting Algorithms

To construct parallel algorithms which run in polylogarithmic time using a polynomial number of processors and produce an output with exactly uniform distribution with high probability, Dixon and Wilf's algorithm can be modified with a "restarting" facility. *Acceptance-rejection sampling* is a well-known method in Statistics to generate a value according to an unknown probability distribution (see [Rub81] or [vN51]). It has been used more recently in Computer Science by [KLM89] to solve approximately some hard counting problems and applied successfully to uniform generation problems by Wormald [Wor87]. In the first part of this section the basic conceptual ingredients of the technique will be spelled out by looking at a simple "toy" problem. In the second part a short description of Wormald's use of a restarting algorithm to solve some uniform generation problems in the context of unlabelled graphs will be given. The algorithms in Sections 2.7 and 2.8 show how this idea can be exploited to define efficient parallel uniform generators as well.

Sampling in a Set Union. Consider the following problem. Let $S_1, S_2, \ldots, S_n$ be n given sets. An element in the *universe* $S = \cup_{i=1}^{n} S_i$ must be sampled uniformly at random. We further assume that the counting and uniform generation problems for S_i for all i can be solved in polynomial time and that the membership in S_i can be decided in polynomial time. Under these conditions the following algorithm $\mathcal{R}$ generates $a \in S$ with probability that does not depend on a and respects also the second condition in Definition 3.

(1) Repeat n times steps (2)-(5):

(2) Select S_j with probability $|S_j|/\sum |S_i|$.

(3) Select $a \in S_j$ uniformly at random.

(4) Compute $N = |\{k \in [n] : a \in S_k\}|$.

(5) With probability $1/N$ output a and stop.

The probability of outputting a specific a during an iteration is

$$\Pr[a] = \Pr[\text{output}|a] \sum_{j:a \in S_j} \Pr[a|j] \Pr[j] = \frac{1}{N} \frac{N}{\sum |S_i|} = \frac{1}{\sum |S_i|}$$

The probability of getting an output in one iteration is at least $1/n$. Hence over n iterations the probability of getting no output is at most $(1 - 1/n)^n < 1/\mathrm{e} < 1/2$.

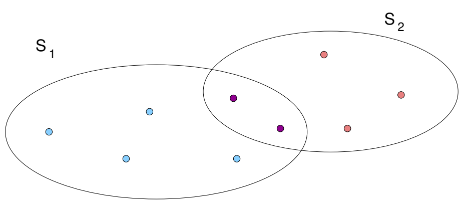

Figure 2.2: Example of set family.

Figure 2.2 gives a graphical description of the situation in a simple case. The universe S is partitioned into three classes $C_1 = S_1 \setminus S_2$, $C_2 = S_2 \setminus S_1$ and $C_3 = S_1 \cap S_2$. Ideally the following algorithm $\mathcal{E}$ should be used.

(1) Select C_j with probability $|C_j|/S$;

(2) Select $a \in C_j$ uniformly at random

Unfortunately in most real world cases the cardinalities of the C_i are unknown and the uniform generation inside C_i might be difficult. Hence we resort to sampling in the set of pairs (a, i). From the point of view of the original problem this is equivalent to sampling each a with a probability proportional to the number of S_i it belongs to. So we need to scale this down by allowing the selection to take place with probability $1/N$. This is exactly what algorithm $\mathcal{R}$ does.

Restarting for Unlabelled Graph Generation. Rejection sampling can be applied to the generation of unlabelled graphs as well. Restarting procedures for generating uniformly at random unlabelled graphs of various types are described in [Wor87]. In the case of unlabelled graphs on a given number n of vertices the symmetric group S_n is acting on the set $\mathcal{G}^n$. Also, the set of all pairs (g, α), where g is a permutation and α is a graph fixed by g, can be partitioned in classes $\Upsilon_1, \ldots, \Upsilon_p$. This time only approximate counting information on the Υ_i is given: $|\Upsilon_i| \leq r_i$ for each $i = 1, \ldots, p$. The following algorithm $\mathcal{W}$ can be used instead of $\mathcal{DW}$ (as long as the r_i satisfy certain conditions, and $\iota(n) = \Omega(\log n)$)

(0) Repeat $\iota(n)$ times steps (1.1), (1.2), (1.3), and (2):

(1.1) select a class Υ_i with probability $r_i / \sum r_i$.

(1.2) Select a permutation g uniformly at random among those such that $(g, \alpha) \in \Upsilon_i$.

(1.3) Restart with a probability $\frac{f(\Upsilon_i)|Fix(g)|}{r_i}$ (where f is some real valued function of Υ_i).

(2) select uniformly at random $\alpha \in Fix(g)$ and return its orbit.

Notice that step (2) is exactly the same as before and the only change with respect to Dixon and Wilf's solution is in the way a permutation is selected. No exact counting information on the number of unlabelled graphs with n vertices is needed. If the algorithm gets to step (2), the probability distribution of the chosen α's orbit is exactly uniform over all orbits (see [Wor87] for details). Moreover with high probability the algorithm will succeed in generating a graph.

In Sections 2.7 and 2.8 two possible parallel implementations of this idea will be described. Before that, in the next section, we will make a detour on integer partitions.

2.4 Integer Partitions

Step (1) in Dixon and Wilf's algorithm selects a conjugacy class at random with a particular probability distribution.

Given a probability space over a finite sample space $\{e_1, \ldots, e_x\}$, the obvious way to select one such event is to compute all the probabilities $\Pr[e_i]$ for $i = 1, \ldots, x$, compute $\sum_{j=1}^{i} \Pr[e_j]$ for $i = 1, \ldots, x$, then select a random number ξ between zero and one and choose the event e_i such that

$$\sum_{j=1}^{i-1} \Pr[e_j] \leq \xi < \sum_{j=1}^{i} \Pr[e_j]$$

(in the above expression $\sum_{j=1}^{0} \Pr[e_j] = 0$). To compute the probabilities $\Pr[e_i]$ a method must be available to list all the e_i's.

In the first part of this section we describe a well-known bijection between the conjugacy classes in the permutation group of order n and the set of so called integer partitions of n. By Theorem 19 below and Theorem 16, for each conjugacy class C, all the information required to compute $\Pr[C]$ can be extracted from the integer partition corresponding to the cycle type of C. Therefore the selection of a conjugacy class in Dixon and Wilf's algorithm can be done by listing the corresponding integer partitions and using them to compute the required probabilities.

As a side result the bijection mentioned above along with some old results on the number of different integer partitions of a number, implies that the number of different conjugacy classes in a permutation group is superpolynomial in the order of the group. Hence an explicit listing of all of them is not a very effective way to compute the required probabilities. However in Section 2.2 we showed that if conjugacy classes are listed in w.a.l. order the first few are much more probable than all the others. In the final part of the section we describe an NC parallel algorithm for listing a polynomial number of integer partitions in the corresponding partial order.

2.4.1 Definitions and Relationship with Conjugacy Classes

For $n \in \mathbb{N}^+$ any sequence $\pi = (a_i)_{i=1,\ldots,h}$ with $1 \leq a_1 \leq a_2 \leq \ldots \leq a_h \leq n$ and $n = \sum_{i=1}^{h} a_i$ is called an *integer partition* of n. Following Theorem 19 below, we will occasionally drop the distinction between π and the cycle type of the conjugacy class associated to it, and represent π by the *cycle type notation*, $[k_1, \ldots, k_n]$ (compare with Section 1.2.6), where $n = \sum_{i=1}^{n} ik_i$ and k_i gives the number of parts of size i. A more compact *multiplicity notation* is obtained by taking the pairs (k_i, i) such that $k_i \neq 0$. Figure 2.3 shows all integer partitions of 8 in the three different representations.

Lemma 4 *The number of non-zero pairs in multiplicity notation of any integer partition of n is $O(\sqrt{n})$.*

Proof. Let $(i_1, k_1)(i_2, k_2)\ldots(i_h, k_h)$ be an integer partition of n into some h non-zero parts. We can write $n = \sum_{j=1}^{h} i_j k_j \geq \sum_{j=1}^{h} i_j$ but also $n \geq \sum_{i=1}^{h} i = h(h-1)/2$, since $i_j \geq j$. This implies $h = O(\sqrt{n})$. In particular $h \leq \sqrt{2n}$ for every positive integer n. □

```
standard           cycle type          multiplicity
1 1 1 1 1 1 1 1    [8,0,0,0,0,0,0,0]    8,1

1 1 1 1 1 1 2      [6,1,0,0,0,0,0,0]    6,1 1,2

1 1 1 1 1 3        [5,0,1,0,0,0,0,0]    5,1 1,3

1 1 1 1 2 2        [4,2,0,0,0,0,0,0]    4,1 2,2
1 1 1 1 4          [4,0,0,1,0,0,0,0]    4,1 1,4

1 1 1 2 3          [3,1,1,0,0,0,0,0]    3,1 1,2 1,3
1 1 1 5            [3,0,0,0,1,0,0,0]    3,1 1,5

1 1 2 2 2          [2,3,0,0,0,0,0,0]    2,1 3,2
1 1 2 4            [2,1,0,1,0,0,0,0]    2,1 1,2 1,4
1 1 3 3            [2,0,2,0,0,0,0,0]    2,1 2,3
1 1 6              [2,0,0,0,0,1,0,0]    2,1 1,6

1 2 2 3            [1,2,1,0,0,0,0,0]    1,1 2,2 1,3
1 2 5              [1,1,0,0,1,0,0,0]    1,1 1,2 1,5
1 3 4              [1,0,1,1,0,0,0,0]    1,1 1,3 1,4
1 7                [1,0,0,0,0,0,1,0]    1,1 1,7

2 2 2 2            [0,4,0,0,0,0,0,0]    4,2
2 2 4              [0,2,0,1,0,0,0,0]    2,2 1,4
2 3 3              [0,1,2,0,0,0,0,0]    1,2 2,3
2 6                [0,1,0,0,0,1,0,0]    1,2 1,6
3 5                [0,0,1,0,1,0,0,0]    1,3 1,5
4 4                [0,0,0,2,0,0,0,0]    2,4
8                  [0,0,0,0,0,0,0,1]    1,8
```

Figure 2.3: Integer partitions of 8 in different representations.

The next result shows a relationship between integer partitions and conjugacy classes in permutation groups.

Theorem 19 *The number of different conjugacy classes in the permutation group S_n is exactly the number $p(n)$ of integer partitions of n.*

Proof. The result is proved by showing that every conjugacy class in S_n defines an integer partition of n and conversely for every integer partition of n there exists a conjugacy class in S_n.

By Theorem 13 all permutations in a given conjugacy class have the same cycle type. Since permutations are bijections every $i \in [n]$ belongs to some cycle and to only one cycle. Hence the set of cycle lengths of a permutation forms an integer partition of n.

Conversely if $\pi = [k_1, \ldots, k_n]$ is an integer partition of n then the permutation g described by

$$\underbrace{(k_1+1\ \ k_1+2)\ldots(k_1+2k_2-1\ \ k_1+2k_2)}_{k_2 \text{cycles of length } 2}\underbrace{(k_1+2k_2+1\ \ k_1+2k_2+2\ \ k_1+2k_2+3)\ldots}_{k_3 \text{cycles of length } 3}\ldots$$

has k_1 fixed elements, k_2 cycles of length 2 and so on. □

Both the counting and the uniform generation problem for integer partitions can be solved fast in parallel. The following result states a well known combinatorial identity on the number $p(n)$ and the existence of an NC algorithm for calculating $p(n)$.

Theorem 20 [SS96] *For every $n \in \mathbb{N}^+$ the numbers $p(i)$ for $i \in \{0, \ldots, n\}$ can be computed in $O(\log^2 n)$ time using n^2 processors on an EREW PRAM.*

Proof. It is well known (see for example [Rio58, p. 111]) that the numbers $p(i)$ for $i \in \{0, \ldots, n\}$ occur as coefficients of x^i in the polynomial

$$\prod_{j=1}^{n} \sum_{i=0}^{\lfloor n/j \rfloor} x^{ij}$$

Let $Q_j(x) = \sum_{i=0}^{\lfloor n/j \rfloor} x^{ij}$ and $Q[j]$ be its internal representation. For example if $n = 10$ then

$$Q[3] = (1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0)$$

corresponding to $1+x^3+x^6+x^9$. The product of two polynomials can be implemented by a function poly_prod in $O(\log n)$ time using n processors on an EREW PRAM using a parallel algorithm to perform a Fast Fourier Transform (see [Wil94] for a nice and gentle description of the algorithmic ideas and [Lei92] for details about the parallel algorithm). Finally n polynomials can be multiplied using the function tree$(Q, n, \mathrm{poly_prod})$ as defined in Section 1.2.2. The result follows. □

Example. For $n = 10$,

$$\begin{aligned} \prod_{j=1}^{1} 0 \sum_{i=0}^{\lfloor 10/j \rfloor} x^{ij} \quad = \quad & (1 + x + x^2 + x^3 + x^4 + x^5 + x^6 + x^7 + x^8 + x^9 + x^{10}) \cdot \\ & (1 + x^2 + x^4 + x^6 + x^8 + x^{10})(1 + x^3 + x^6 + x^9)(1 + x^4 + x^8) \cdot \\ & (1 + x^5 + x^{10})(1 + x^6)(1 + x^7)(1 + x^8)(1 + x^9)(1 + x^{10}) \end{aligned}$$

which eventually gives the polynomial

$$1 + x + 2x^2 + 3x^3 + 5x^4 + 7x^5 + 11x^6 + 15x^7 + 22x^8 + 30x^9 + 42x^{10} + \dots$$

It is worth noticing that there is a simpler way to compute the $p(i)$'s. It is easy to design an algorithm for multiplying two polynomials in $O(\log n)$ parallel steps using n^2 processors on an EREW PRAM. Such an algorithm can then be used in the proof of Theorem 20 instead of the one based on Fast Fourier Transform. This results in a $O(\log^2 n)$ time, $O(n^3)$ processors algorithm for computing all the $p(i)$'s. Since the value of n for which $p(1), \dots, p(n)$ are computed is logarithmic in the overall input order, the complexity values mentioned above do not represent any penalty.

We conclude this section by stating without proof a well-known asymptotic identity on $p(n)$. It will be used in analysing the algorithms presented in the next sub-section.

Theorem 21 [And76, Th. 6.3] $p(n) \sim \frac{e^{\pi\sqrt{2n/3}}}{4n\sqrt{3}}$.

2.4.2 Parallel Algorithm for Listing Integer Partitions

Due to their super-polynomial number there is no hope of finding an NC algorithm for listing all integer partitions of n in any order. The w.a.l. order defined on conjugacy classes in Section 2.2 induces a homonymous partial order on integer partitions in the most obvious way: given two partitions π_1 and π_2, we say that π_1 precedes π_2 (and we write $\pi_1 \prec \pi_2$) in w.a.l. order if the number of parts of size one in π_1 is greater than that of π_2. In this section an NC algorithm for listing a polynomial number of partitions in w.a.l. order in multiplicity notation will be described.

We start by defining two parameters related to integer partitions of n. Let $s_2(n)$ be the number of partitions of $n \in \mathbb{N}^+$ with smallest part of size at least 2; in particular $s_2(0) = s_2(1) = 0$.

Lemma 5 $s_2(j) = p(j) - p(j-1)$ *for every* $j \geq 1$.

Proof. The number of partitions of j with no part of size one is the number of partitions of j less the number of partitions of j starting with a one and ending in any possible way, that is $p(j-1)$. □

Let $u(k)$ denote the number of integer partitions of n with at least $n-k$ ones, for every $k \in \{0, \dots, n\}$. The following lemma states the most important properties of $u(k)$.

Lemma 6 *For every* $n \in \mathbb{N}$*:*

1. *The parameter* $u(k)$ *is independent of* n*. More specifically it is* $u(k) = p(k) = 1 + \sum_{j=0}^{k} s_2(j)$ *for every* $k \in \{0, \dots, n\}$.

2. *For every* $c \in \mathbb{R}^+$ *if* $k = \lfloor c \log^2 n \rfloor$ *then* $u(k) = \Theta\left(\frac{n^{(\pi/\log 2)\sqrt{2c/3}}}{\log^2 n}\right)$.

Proof. The number of partitions of n containing at least $n-k$ parts of size one is exactly the number of integer partitions of k. To understand the second equality, a partition of k is either formed by all ones or contains $k-j$ ones and its remaining parts form a partition of j with smallest part of size at least 2.

For the second part, If $k = \lfloor c \log^2 n \rfloor$, by Theorem 21

$$(1-\varepsilon)\frac{\mathrm{e}^{\pi\sqrt{2k/3}}}{4k\sqrt{3}} \leq u(k) = p(k) \leq (1+\varepsilon)\frac{\mathrm{e}^{\pi\sqrt{2k/3}}}{4k\sqrt{3}}.$$

The result follows. □

The key idea of the algorithm is to consider each partition as formed by two disjoint components: a prefix of $n-i$ ones (for some $i \in \{0\} \cup \{2, \ldots, k\}$) and a tail consisting of an arbitrary integer partition of i with smallest part of size at least two. A more detailed description is as follows (partitions are represented in multiplicity notation, polynomials of degree d by the vector of their $d+1$ coefficients).

```
Input: n, k and polynomials Q[j] for j = 1, ..., k
p ← tree(Q, k, poly_prod);
B ← prefix(p[1 : k], k, +);
List[1, 1] ← (n, 1);
Cycle_Type[1, 1] ← (n, 1);
if k ≥ 2
    for all i ∈ {2, ..., k}, j ∈ {1, ..., p[i]} in parallel do
        List[B[i − 1] + j, 1] ← (n − i, 1);
    for all i ∈ {2, ..., k} in parallel do
        List[B[i − 1] + 1 : B[i − 1] + p[i], 2 : 2⌈√k⌉ + 1] ← FL(i);
    Address[1] ← 1;
    for all i ∈ {2, ..., B[k]} in parallel do
        if List[i, 1] = (·, 1) and List[i, 2] = (·, 1)
            Address[i] ← 0;
        else Address[i] ← 1;
    Address ← prefix(Address, B[k], +);
    for all i ∈ {2, ..., B[k]} and i odd in parallel do
        if Address[i − 1] ≠ Address[i]
```

(18) Cycle_Type$[i-1,\cdot] \leftarrow$ List$[i-1,\cdot]$;

(19) **for** all $i \in \{2,\ldots,B[k]\}$ and i even **in parallel do**

(20) **if** Address$[i-1] \neq$ Address$[i]$

(21) Cycle_Type$[i-1,\cdot] \leftarrow$ List$[i-1,\cdot]$;

A 2-dimensional array Cycle_Type of size

$$\left(\sum_{i=1}^{k} p(i)\right) \times \left(2\lceil\sqrt{k}\rceil + 1\right)$$

is used to store all required integer partitions (we assume the array as well as the partial result array List are initialised so that at the beginning Cycle_Type$[x,y] =$ List$[x,y] = 0$ for all x and y). For every legal x and y, $\text{Cycle\_Type}[x,y]$ will contain the y-th pair (k_{i_y}, i_y) of the x-th partition in the listing.

A preliminary bookkeeping stage (steps (1) and (2)) is needed to allocate the right number of processors and amount of space to complete the listing. The i-th entry of the array p gives the number of partitions of i and is computed by a tree computation using the function poly_prod defined in Theorem 20. The values of another vector B are obtained from p by parallel prefix. For $i = 2,\ldots,k$, using $p(i)$ processors all partitions of i will be computed and stored in positions $B[i-1]+1,\ldots,B[i-1]+p(i)$ of List.

Since partitions are stored in multiplicity notation, prefixes of $n-i$ ones are encoded as pairs $(n-i,1)$. In step (3) List[1, 1] is set to $(n,1)$ and then, (step (6) and (7)) for every $i \in \{2,\ldots,k\}$, List$[B[i-1]+j,1]$ are set to the value $(n-i,1)$ for $p[i]$ different values of j.

The algorithm then uses $k-1$ groups of $p(i)$ processors to complete the listing (steps (8) and (9)). The i-th group (for $i \in \{2,\ldots,k\}$) lists all partitions of i using a natural parallelisation (breadth-first traversal of a binary tree) of the algorithm in [FL83]. Conceptually a tree T_i is built whose nodes correspond to partitions of i. From a partition π two "children" partitions are obtained by applying (in parallel using two different processors) the following rules

A. add one part of size two and remove two parts of size one.

B. if the smallest part greater than one has multiplicity one then increase it by one and remove one part of size one.

So for example if $i = 8$, partition $(8,1)$ is the label for the root of T_i. This has one child $(6,1)(1,2)$ obtained applying rule A and this in turn has two children $(4,1)(2,2)$ and $(5,1)(1,3)$ obtained from rule A and B respectively. We then apply, if possible, the rules to these new leaves until the process is complete. Notice that the concurrent access to π to generate its two children can be avoided, for instance by storing two copies of every node, one to be used in building its left children and the other one for the right one.

A final clean-up stage is needed to remove from List all partitions whose tail is a partition of i having smallest element one and compact the remaining elements. A sufficient condition for List$[x,\cdot]$ to contain a "useless" partition is that

$$\text{List}[x,1] = (n-i,1) \qquad\qquad \text{List}[x,2] = (j,1)$$

For every $x \in \left\{2,\dots\sum_{j=1}^{k} p(j)\right\}$ this condition can be tested in one parallel step. The clean-up stage is implemented by first (steps (11) through (14)) filling an array Address with zeroes and ones depending on the condition above. Then parallel prefix is run on Address. After this Address$[i]$ is the address of List$[i,\cdot]$ in the final array Cycle_Type. The two final **for** loops (two distinct loops are needed to avoid memory conflicts) copy A into Cycle_Type avoiding the copying of "useless" elements.

Lemma 7 *The depth of the tree T_i in step (9) of the algorithm above is $O(i)$ for all i.*

Proof. The partitions of i into j parts is the set of nodes at level $i-j+1$ in T_i. □

Theorem 22 *For any given k, the algorithm above generates $u(k)$ partitions of n in w.a.l. order in $O(k^{3/2})$ time using $O(u(k))$ processors on an EREW PRAM.*

Proof. If k stages are allowed the algorithm uses $O(\sum_{i=0}^{k} p(i))$ processors to compute $u(k)$ partitions and the running time is proportional to the depth of the deepest T_i times the maximum time spent at each node (which is at least the time to write down a partition in multiplicity notation, i.e. $O(\sqrt{k})$).

By Lemma 5, deleting terms equal to zero,

$$\sum_{j=0}^{k} p(j) = 1 + \sum_{j=2}^{k} s_2(j) + \sum_{j=1}^{k} p(j-1).$$

By Lemma 6.2 for every j sufficiently large

$$\begin{aligned} p(j) \quad &> \quad (1-\epsilon)\frac{\mathrm{e}^{\pi\sqrt{2j/3}}}{4j\sqrt{3}} = \frac{1-\epsilon}{1+\epsilon}\left(1+\frac{1}{j-1}\right)\mathrm{e}^{\sqrt{2/3}}(1-\epsilon)\frac{\mathrm{e}^{\pi\sqrt{2(j-1)/3}}}{4(j-1)\sqrt{3}} \\ &\geq \quad \frac{1-\epsilon}{1+\epsilon}\left(1+\frac{1}{j-1}\right)\mathrm{e}^{\sqrt{2/3}}p(j-1) \end{aligned}$$

where ϵ is chosen so that the constant multiplying $p(j-1)$ is larger than one. Hence we can write

$$\sum_{j=0}^{k} p(j) \leq 1 + \sum_{j=2}^{k} s_2(j) + \frac{1}{c}\sum_{j=1}^{k} p(j)$$

for some $c > 1$ and from this conclude

$$\sum_{j=0}^{k} p(j) \leq \frac{c}{c-1}\left(\sum_{j=1}^{k} s_2(j)\right)$$

□

2.5 Reducing Uniform Generation to Sampling in a Permutation Group

In this section an implementation of the second step of algorithm $\mathcal{DW}$ will be given. It is not difficult to prove that if a permutation $g \in S_n$ has been selected (for instance according to the distributions given by algorithms $\mathcal{DW}$ or $\mathcal{W}$ in Section 2.2 and 2.3), then a graph can be constructed efficiently in parallel with optimal work. As a by-product the proof of Theorem 23 provides an optimal NC solution to the counting problem associated with $Fix(g)$.

Theorem 23 *Given a permutation $g \in S_n$ the cardinality of the set $Fix(g)$ can be determined in $O(\log n)$ deterministic parallel time on an EREW PRAM with $O(n^2/\log n)$ processors. Also a graph in $Fix(g)$ can be generated uniformly at random in $O(\log n)$ parallel time on an EREW PRAM with $O(n^2/\log n)$ processors.*

Proof. Given $g \in S_n$, a graph in $Fix(g)$ is obtained by generating $g^* \in S_n^{(2)}$ and picking sets of edges to be part of the graph by selecting cycles of g^* with probability $1/2$.

The PRAM implementation of this algorithm is rather straightforward. Sometimes the code is a bit convoluted because of the need to avoid concurrent read operations. The first six steps of the algorithm, given g, constructs $g^* \in S_n^{(2)}$.

Input: n, g be stored as an array G such that $G[i] = g(i)$

(1) **for** all $i \in \{1, \ldots, n\}$ **in parallel do**

(2) $\quad H[i, \cdot] \leftarrow \text{copy}(G[i], n)$;

(3) $\quad G^*[i, \cdot].\text{val} \leftarrow \text{copy}(G[i], n)$;

(4) **for** all $i, j \in \{1, \ldots, n\}$ and $i < j$ **in parallel do**

(5) $\quad G^*[i, j].\text{val} \leftarrow (\min\{G^*[i, j].\text{val}, H[j, i]\}, \max\{G^*[i, j].\text{val}, H[j, i]\})$;

(6) $\quad G^*[i, j].\text{leader} \leftarrow (i, j)$;

$G^*[i,j]$.val contains the pair $\{g(i), g(j)\}$, whereas $G^*[i,j]$.leader will be needed for the next stage. Its intended meaning is to keep track of the lexicographically smallest pair $\{x, y\}$ belonging to the same cycle (in g^*) that $\{i, j\}$ belong to. The parallel time complexity of the piece of code above is $O(1)$ using $n(n-1)/2$ processors on an EREW PRAM.

The next stage computes the cycle structure of g^*. This is the bottleneck computation. The main technique used here is pointer jumping. The following algorithm associates with every $\{i, j\}$ the value of the lexicographically smallest pair $\{x, y\}$ belonging to the same cycle as $\{i, j\}$ in g^*. The "$>$" used in step (9) is shorthand for the function implementing the lexicographic order.

(7) **for** $k \in \{1, \ldots, \lceil 2 \log n \rceil\}$ **do**
(8)   **for** all $i, j \in \{1, \ldots, n\}$ and $i < j$ **in parallel do**
(9)   **if** $(G^*[i,j].\text{leader} > G^*[G^*[i,j].\text{val.first}, G^*[i,j].\text{val.second}].\text{leader})$
(10) $G^*[i,j].\text{leader} \leftarrow G^*[G^*[i,j].\text{val.first}, G^*[i,j].\text{val.second}].\text{leader};$
(11) $G^*[i,j].\text{val} \leftarrow G^*[G^*[i,j].\text{val.first}, G^*[i,j].\text{val.second}].\text{val};$

For clarity this is a simple $O(\log n)$-time $O(n^2)$-processor EREW PRAM algorithm. An optimal EREW algorithm for computing the cycle structure of g^* can be obtained by adapting the optimal $O(\log n)$ time deterministic list ranking procedure due to Cole and Vishkin (see [CV88]).

Having computed the cycle structure of g^*, a label is associated with each cycle and the array of labels is ranked.

(12) **for** all $i, j \in \{1, \ldots, n\}$ and $i < j$ **in parallel do**
(13)   **if** $(G^*[i,j].\text{val} = G^*[i,j].\text{leader})$
(14)   Ord$[(i-1)n + j] \leftarrow 1;$
(15)   **else** Ord$[(i-1)n + j] \leftarrow 0;$
(16) Ord $\leftarrow$ prefix$(\text{Ord}, n(n-1)/2, +);$

After this Ord$[n(n-1)/2]$ contains the value of $\log |Fix(g)|$. The bottle-neck from the complexity point of view is the cycle structure computation and the first half of the theorem is thus proved.

Finally an array Cycles is created which contains with probability a half the label of the cycle leader for each cycle. All other elements of Cycles are assumed to be set to zero by some constant parallel time initialisation procedure.

(17) **for** all $i, j \in \{1, \ldots, n\}$ and $i < j$ **in parallel do**

if $((G^*[i,j].\mathrm{val} = G^*[i,j].\mathrm{leader})$ and $(\mathrm{rand}(0,1) \leq 1/2))$

Cycles[Ord[$(i-1)n + j$]] $\leftarrow G^*[i,j].\mathrm{val}$;

It is now easy to devise an algorithm that, using Cycles, deletes from G^* all edges which have not been selected. □

The algorithms contained in the proof of Theorem 23 will be used to implement the final part (step (2) in the algorithm in Section 2.2) of the selection in all RNC algorithms described in the following sections.

2.6 RNC Non Uniform Selection

In Section 2.2 (see Theorem 18) it was mentioned that $\mathcal{DW}$ can be simulated on a PRAM using a polynomial number of processors by an algorithm running in $O(\log n)$ expected parallel time.

The same sequential algorithm can be converted into an optimal parallel algorithm having polylogarithmic worst case parallel time with high probability. The resulting algorithm, which will be called $\mathcal{NU}$, is based on a different conjugacy class selection procedure. The number of conjugacy classes that will ever be considered by the process is fixed in advance as a certain function of n. Here is a high level description of $\mathcal{NU}$.

(1.1) For a given k (to be defined later) select $i \in \{1, \ldots, u(k)\}$ with probability $\Pr[C_i] = \frac{|C_i| 2^{q(C_i)}}{mn!}$ and return "ERROR" with probability $1 - \sum_{i=1}^{u(k)} \Pr[C_i]$.

(1.2) Construct a representative g of C_i.

(2) Select uniformly at random $\alpha \in Fix(g)$ and return its orbit.

The major problem with this algorithm is that, although reasonable in practice, the distribution of the output graphs is not uniform. A super-polynomial number of conjugacy classes is not considered at all by the process and this introduces some imbalance in the probability distribution. For the original Dixon and Wilf's procedure, using Theorem 15 it is easy to show that it is equally likely for the output graph α to be in any of the orbits:

$$\begin{aligned}
\Pr[\omega_\alpha] &= \sum_{C \subseteq S_n} \Pr[\omega_\alpha | C] \Pr[C] = \sum_{C \subseteq S_n} \frac{|Fix(g) \cap \omega|}{|Fix(g)|} \frac{|C||Fix(g)|}{m|S_n|} \\
&= \frac{1}{m|S_n|} \sum_{C \subseteq S_n} |C||Fix(g) \cap \omega| = \frac{1}{m|S_n|} \sum_{S_n} |Fix(g) \cap \omega| = \frac{1}{m|S_n|} |S_n|
\end{aligned}$$

If $S_n^{u(k)} = C_1 \cup C_2 \cup \ldots \cup C_{u(k)}$, then

$$\begin{aligned}
\Pr[\omega_\alpha] &= \sum_{C \subseteq S_n^{u(k)}} \frac{|Fix(g) \cap \omega|}{|Fix(g)|} \frac{|C||Fix(g)|}{m|S_n|} = \frac{1}{m|S_n|} \sum_{C \subseteq S_n^{u(k)}} |C||Fix(g) \cap \omega| \\
&= \frac{1}{m|S_n|} \sum_{g \in S_n^{u(k)}} |Fix(g) \cap \omega|
\end{aligned}$$

An upper bound on $\Pr[\omega_\alpha]$ is derived remembering that, by Theorem 15.*1*,

$$\sum_{g \in S_n^u} |Fix(g) \cap \omega| \leq |S_n| = n!.$$

Hence $\Pr[\omega] \leq 1/m$. If $u = 1$ then $S_n^1 = C_1 = \{id\}$. The set of graphs fixed by the identity permutation coincides with the whole set of labelled graphs on n vertices. The distribution of representatives of different orbits in $\mathcal{G}^n$ is very imbalanced. The (orbit of the) complete graph appears only once whereas orbits corresponding to labelled graphs with trivial automorphism group have $n!$ representatives. Hence $\Pr[\omega] \geq 1/(mn!)$. Figure 2.4 shows a small example. Rows are associ-

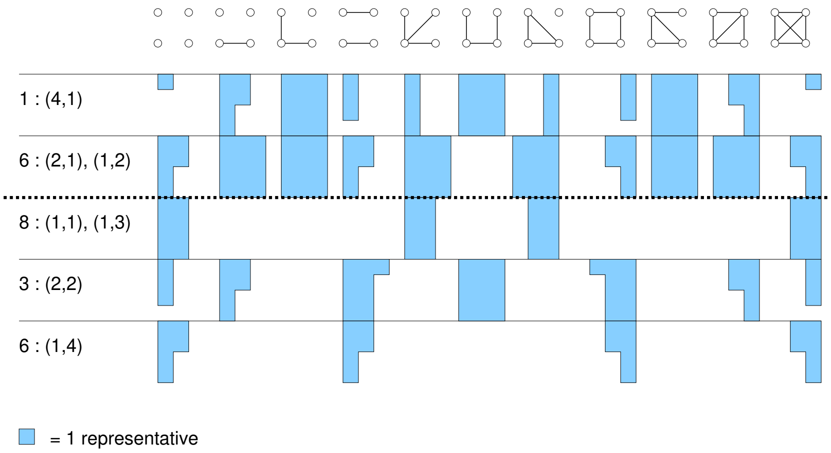

Figure 2.4: Distribution of different orbits for $n = 4$.

ated with conjugacy classes, columns with orbits (the eleven unlabelled graphs on $n = 4$ vertices are drawn on the top to represent them). Notation $x : (i_1, k_1)(i_2, k_2) \ldots (i_h, k_h)$ means "there are x permutations with cycle type $(i_1, k_1)(i_2, k_2) \ldots (i_h, k_h)$. A small square in the row marked by conjugacy class C stands for a labelled graph belonging to the orbit in the particular column and to some set $Fix(g)$ for some $g \in C$, with occurrences of the same graphs in two different sets being counted twice. If only conjugacy classes above the dashed line are listed one of the two unlabelled graphs with four edges occur more than three times as often as the complete graph.

However, for sufficiently large n, graphs "behaving" like K_n are very rare. This implies that the distribution on unlabelled graphs given by the algorithm above, although not exactly uniform, is skewed in very few cases.

Definition 6 *A randomised algorithm $\mathcal{A}$ is a (ψ, ϕ)-*uniform generator *for a relation $\mathcal{SOL}$ if and only if*

1. $\Pr[\mathcal{A}(x) = y] = 0$ *for all $(x, y) \notin \mathcal{SOL}$.*

2. *there exist functions $\psi, \phi : \Sigma^* \to (0, 1]$ with $\psi(x) < \phi(x)$ for all $x \in \Sigma^*$, such that if $x, y \in \mathcal{I}_n$,*

$$\psi(x) \leq \Pr[\mathcal{A}(x) = y] \leq \phi(x)$$

and

$$\lim_{n \to \infty} \frac{|\{y : \Pr[\mathcal{A}(x) = y] < \phi(x)\}|}{|\mathcal{SOL}(x)|} = 0$$

3. *For all $x \in \Sigma^*$ such that $\mathcal{SOL}(x) \neq \emptyset$,* $\Pr[\exists y \in \mathcal{SOL}(x) : \mathcal{A}(x) = y] \geq 1/2$.

The qualifiers "parallel", "RNC", and "with high probability" can be applied to this definition in the usual way. Definition 6 is meant to capture the type of approximation to the uniform distribution given by the algorithm sketched at the beginning of the section. For n sufficiently large the probability that the graph generated by the algorithm belongs to a specific orbit is uniform "most of the time". It is interesting to compare Definition 6 with another notion of approximation to the uniform distribution.

Definition 7 [JVV86] *A randomised algorithm $\mathcal{A}$ is an* almost uniform generator *with* tolerance ϵ *($0 \leq \epsilon < 1$) for a relation $\mathcal{SOL}$ if and only if*

1. *there exists a function $\phi : \Sigma^* \to (0, 1]$ such that, for all $x, y \in \Sigma^*$ and for all $\epsilon \in [0, 1)$,* $\Pr[\mathcal{A}(x, \epsilon) = y] \in I(x, \epsilon)$ *with*

$$I(x, \epsilon) = \begin{cases} \{0\} & (x, y) \notin \mathcal{SOL} \\ \left[\frac{\phi(x)}{1+\epsilon}, (1 + \epsilon)\phi(x)\right] & \text{otherwise} \end{cases}$$

2. *For all $x \in \Sigma^*$ such that $\mathcal{SOL}(x) \neq \emptyset$,* $\Pr[\exists y \in \mathcal{SOL}(x) : \mathcal{A}(x, \epsilon) = y] \geq 1/2$.

An almost uniform generator always produces an output with probability which approximates tightly the uniform but is never guaranteed to produce output with exactly uniform distribution (i.e. with constant probability). If $y \in \mathcal{SOL}(x)$ and a cost function $P_A(x, y) =_{df} \Pr[\mathcal{A}(x) = y]$ is associated with x and y then a generation problem can be recast as an optimisation problem in which the goal is to optimise the "uniformity" of the output distribution. An almost uniform generator is, in some sense, an approximation scheme (see Section 4.1 in Chapter 4) for the exact uniform distribution. On the other hand, a (ψ, ϕ)-uniform generator produces exactly uniform outputs in most cases in the sense that it fails to do so on a small proportion of elements of $\mathcal{SOL}(x)$ as stated in the second condition of Definition 6. However some outputs can be generated with rather small probability.

Theorem 24 *There is an optimal EREW PRAM RNC $(1/n!m, 1/m)$-uniform generator for the class of unlabelled graphs on n vertices, if their number m is known in advance. Moreover an output is produced with high probability.*

Proof. (Sketch) The result follows from the statement in [Bol85, p. 204] that almost all labelled graphs on n vertices has trivial automorphism group. Hence the sequential algorithm at the beginning of the section will produce a uniform output with probability (over all labelled graphs) going to one. The proof is completed by the paragraphs following this theorem which provide details on the computation of the probabilities associated with each conjugacy class and on the construction of the representative permutation. The resulting algorithm runs in $O(\log n)$ parallel steps using $O(n^2/\log n)$ processors. □

Selecting a conjugacy class. The implementation of step (1.1) in algorithm $\mathcal{NU}$ assumes that probabilities $\Pr[C_i]$ for $i = 1, \ldots, u(k)$ have been computed. Lemma 9 and Theorem 25 below describe an algorithm for computing these probabilities that runs in polylogarithmic time using a polynomial number of processors as long as not too many conjugacy classes are listed. The value of k can be either kept to a constant, in which case an optimal algorithm running in $O(\log n)$ parallel steps using $O(n^2/\log n)$ processors is obtained, or chosen to be in $\Omega(\log n)$, in which case a slower but still optimal algorithm is obtained. Notice that by the bounds in Lemma 6.2, k cannot be chosen to be asymptotically larger than $\log^2 n$ otherwise $u(k)$ becomes too large.

First a tight bound on the number of cycles of the permutations in $S_n^{(2)}$ associated with those g belonging to one of the conjugacy classes generated by our parallelisation of $\mathcal{DW}$ is proved.

Lemma 8 *For every $n \in \mathbb{N}^+$ and for every $x \in \{0, 2, \ldots, n-1\}$, let $h(x, n) = \frac{xn}{2} - \frac{x(x+2)}{4}$. If C is a conjugacy class in S_n with cycle type $[n-x, k_2, k_3, \ldots]$ then*

$$\binom{n}{2} - 2h(x, n) \leq q(C) \leq \binom{n}{2} - h(x, n)$$

Proof. Let n be given and C be an arbitrary conjugacy class in S_n. From calculations in [HP73], if $[k_1, k_2, k_3, \ldots]$ is C's cycle type, then

$$q(C) = \sum_{i=1}^{n} \left\lfloor \frac{i}{2} \right\rfloor k_i + \sum_{i=1}^{n} i \binom{k_i}{2} + \sum_{i<j} \gcd(i,j) k_i k_j$$

The upper bound is proved in [HP73, p. 197]. For the lower bound, if $x = 0$ then $q(C) = \binom{n}{2} + \frac{n}{2}$. Otherwise since $\lfloor x \rfloor \geq x - 1/2$ and $\sum_{i<j} \gcd(i,j) k_i k_j > (n-x)x$, we have

$$q(C) \geq \sum_{i=1}^{n} \left(\frac{i-1}{2} \right) k_i + \sum_{i=1}^{n} \frac{i {k_i}^2}{2} - \sum_{i=1}^{n} \frac{i k_i}{2} + (n-x)x$$

and therefore

$$\begin{aligned} q(C) &\geq \sum_{i=1}^{n} \frac{i {k_i}^2}{2} - \sum_{i=1}^{n} \frac{k_i}{2} + (n-x)x \\ &= \frac{(n-x)^2}{2} - \frac{(n-x)}{2} + \sum_{i=2}^{n} \frac{i {k_i}^2}{2} - \sum_{i=1}^{n} \frac{k_i}{2} + (n-x)x \\ &= \binom{n}{2} - nx + \frac{x^2}{2} + \frac{x}{2} + \frac{x}{2} + (n-x)x - \sum_{i=2}^{n} \frac{k_i}{2} \\ &\geq \binom{n}{2} - 2h(x,n) + (n-x)x - \frac{x}{2} \end{aligned}$$

The result follows since $nx - x^2 - x/2 > 0$ if $x < n$. □

Lemma 9 *For every $n \in \mathbb{N}^+$ and for every $k \in \{0, \ldots, n\}$, let*

$$g(n,k) = \max \left\{ \frac{n^2}{\log n}, \frac{u(k)n}{\log n}, \frac{u(k)\sqrt{k}n}{\log^2 n} \right\}.$$

Let $C_1, \ldots, C_{u(k)}$ be the first $u(k)$ conjugacy classes in S_n in w.a.l. order. If the cycle types of the conjugacy classes have been computed then all the numbers $|C_s| 2^{q(C_s)}$ for $s = 1, \ldots, u(k)$ are computable in $O(\sqrt{k} \log n)$ parallel steps using $O(g(n,k))$ processors on an EREW PRAM.

Proof. Let n and k be given. Assume $\pi_1, \ldots, \pi_{u(k)}$ are the partitions of n describing the cycle types of conjugacy classes $C_1, \ldots, C_{u(k)}$. The algorithm starts off performing some computation which only depends on n, not on the conjugacy classes. First all "useful" powers of two are computed. Since only conjugacy classes whose permutations move at most k objects are listed, by Lemma 8 an array Pow of size $= h(k,n)$ will suffice. After the execution of the code below for every $j \in \{1, \ldots, \text{size}\}$, $\text{Pow}[j,i] = 2^{\text{bound}+j}$ where $\text{bound} = \binom{n}{2} - 2h(k,n) - 1$, for all $i \in \{1, \ldots, u(k)\}$ (again multiple copies are needed to avoid concurrent reads in following stages). A first piece of pseudo-code is need to generate the value $2^{\text{bound}+1}$:

(1) rep $\leftarrow \lceil \log n \rceil$;

(2) **for** all $j \in \{1, \ldots, \lceil (\mathrm{bound} + 1)/\mathrm{rep} \rceil\}$ **in parallel do**

(3) $\quad s \leftarrow (j - 1)\,\mathrm{rep} + 1$;

(4) $\quad$ **while** $(s \leq j\ \mathrm{rep})$ **do**

(5) $\qquad$ Small_Pow$[s] \leftarrow 2$;

(6) $\qquad s \leftarrow s + 1$;

(7) Pow$[1, 1] \leftarrow$ tree(Small_Pow, bound $+ 1, \times)$

Each processor initialises $\lceil \log n \rceil$ elements of the array Small_Pow. Then a tree computation is run on this array to compute $2^{\mathrm{bound}+1}$. Without loss of generality step (7) is run in $O(\log n)$ steps using $O(n^2/\log n)$ processors. All the other elements of Pow are computed as follows.

(8) **for** all $j \in \{2, \ldots, \mathrm{size}\}$ **in parallel do**

(9) $\quad$ Pow$[j, 1] \leftarrow 2$;

(10) Pow$[\cdot, 1] \leftarrow$ prefix(Pow$[\cdot, 1]$, size, $\times)$

(11) **for** all $j \in \{1, \ldots, \mathrm{size}\}$ **in parallel do**

(12) $\quad$ Pow$[j, \cdot] \leftarrow$ copy(Pow$[j, 1], u(k))$;

Steps (8) through (10) run in $O(\log n)$ parallel steps using $O(\mathrm{size}/\log n)$ processors. For the final part, if $k = O(\log^2 n)$ and it is such that $u(k) = O(n/\log^2 n)$, there are about $h(k, n) = O(nk)$ objects to be copied $u(k)$ times each. By slowing down the copy function to make it run in $O(\log n)$ parallel steps, the whole algorithm's running time is in the same order of magnitude using $O(n^2/\log n)$ processors.

All factorials of j for $j = 1, \ldots, n$ are computed similarly. The following algorithm runs in $O(\log n)$ parallel steps and has optimal work $O(n\,u(k))$.

(13) Fact$[0, 1] \leftarrow 1$;

(14) **for** all $j \in \{1, \ldots, n\}$ **in parallel do**

(15) $\quad$ Fact$[j, 1] \leftarrow j$;

(16) Fact$[\cdot, 1] \leftarrow$ prefix(Fact$[\cdot, 1], n + 1, \times)$

(17) **for** all $j \in \{0, \ldots, n\}$ **in parallel do**

(18) $\quad$ Fact$[j, \cdot] \leftarrow$ copy(Fact$[j, 1], u(k))$;

Finally we compute $\varphi(j)$ for all $j = 1, \ldots, n$ using the equation (see [GKP89, p. 134])

$$\varphi(j) = j \prod_{i=1}^{x} \left(1 - \frac{1}{p_i}\right) = \frac{j}{\prod_{i=1}^{x} p_i} \prod_{i=1}^{x} (p_i - 1)$$

(where p_i are the primes occurring in the prime factorisation of j). There exists a real constant $c > 0$ such that there are at most $\lceil cj/\log j \rceil$ primes less than j (see for instance the discussion in [GKP89, Sect. 4.3]). In the following code the array primes is a one dimensional array of size $x = \lceil cn/\log n \rceil$.

(19) primes$[\cdot, 1] \leftarrow$ SP(n);

(20) **for** all $i \in \{1, \dots, x\}$ **in parallel do**

(21)   primes$[i, \cdot] \leftarrow \mathrm{copy}(\mathrm{primes}[i, 1], n)$;

(22) **for** all $i \in \{1, \dots, x\}, j \in \{1, \dots, n\}$ **in parallel do**

(23)   **if** $(j \bmod \mathrm{primes}[i, j]) = 0$

(24)   Vect$[i, j] \leftarrow\ \mathrm{primes}[i, j]$

(25)   **else** Vect$[i, j] \leftarrow 1$;

(26) **for** all $j \in \{1, \dots, n\}$ **in parallel do**

(27)   Prod$[j]$.first $\leftarrow\ \mathrm{tree}(\mathrm{Vect}[\cdot, j], x, \times)$

(28)   Prod$[j]$.second $\leftarrow\ \mathrm{tree}(\mathrm{Vect}[\cdot, j] - 1, x, \times)$

(29)   Phi$[j, 1] \leftarrow j\ \mathrm{Prod}[j].\mathrm{second}/\mathrm{Prod}[j].\mathrm{first}$

(30) **for** all $j \in \{1, \dots, n\}$ **in parallel do**

(31)   Phi$[j, \cdot] \leftarrow\ \mathrm{copy}(\mathrm{Phi}[j, 1], u(k))$;

Steps (19) through (21) in the algorithm above lists all primes up to n in $O(\log n)$ parallel steps using $O(n/(\log n \log\log n))$ processors (see [SP94]) copying them to avoid subsequent concurrent reads. Then (steps (22) through (25)) in parallel for every j, x processors are used to single out the primes belonging to the prime decomposition of j. For each i, the value primes$[i, j]$ is assigned to Vect$[i, j]$ if primes$[i, j]$ divides j. After this using standard PRAM functions we can define Phi$[j]$ (this is done in steps (26) through (31)). The best trade-off for the algorithm complexities is obtained if the copying steps are made to run in $O(\log n)$ parallel steps. The resulting complexity of the piece of code above is $O(\log n)$ parallel steps using $O((n/\log n)^2)$ processors.

After the computation above, given partitions $\pi_1, \pi_2, \dots, \pi_{u(k)}$ stored in multiplicity notation in the 2-dimensional array Cycle_Type as discussed in Section 2.4 the data structures Pow, Fact and Phi are used to compute the desired cardinalities. A set of $n/\log n$ processors $\{P_i^s : i = 1, \dots, n/\log n\}$ is associated with each partition Cycle_Type$[s, \cdot]$ for $s = 1, \dots, u(k)$. Each processor will compute a logarithmic number of $l(i) = \sum_{j=1}^{\lceil n/i \rceil} k_{ij}$ (and store them into a portion of an array L). Since all partitions have at least $n - k$ ones, by Lemma 4, there are only $O(\sqrt{k})$ non-zero elements in each of the vectors Cycle_Type$[s, \cdot]$. Hence a logarithmic number of $l(i)$ can be computed in $O(k)$ sequential time (after the required copies of Cycle_Type$[s,\cdot]$ have been made to avoid concurrent reads). After $O(k)$ parallel steps all the $l(i)$ for each partition have been computed using $O\left(\frac{n\ u(k)}{\log n}\right)$ processors. The pseudo code is as follows:

```
rep ← ⌈log n⌉;
for all s ∈ {1, ..., u(k)}, j ∈ {1, ..., 2⌈√k⌉ + 1} in parallel do
    Copies_CT[s, j, ·] ← copy(Cycle_Type[s, j], ⌈n/rep⌉)
for all s ∈ {1, ..., u(k)}, r ∈ {1, ..., ⌈n/rep⌉} in parallel do
    t ← 1;
    while t ≤ rep do
        i ← (r − 1) rep + t;
        L[s, i] ← 0;
        for j = 1 to 2√k + 1 do
            if (Copies_CT[s, j, i].second mod i = 0)
                L[s, i] ← L[s, i] + Copies_CT[s, j, i].first;
        t ← t + 1
```

The complexity of this part of the algorithm is $O(\sqrt{k}\log n)$ parallel steps using $o(n^2/\log n)$ processors.

For every s the value

$$q(C_s) = \frac{1}{2}\left\{\sum_{i=1}^{n} l(i)^2\varphi(i) - l(1) + l(2)\right\}$$

is then computed using Phi$[s,\cdot]$ and L$[s,\cdot]$ (see [DW83]) in $O(\log n)$ parallel steps using $O\left(\frac{n\ u(k)}{\log n}\right)$ processors. The value of $q(C_s)$ will be a pointer in the array Pow.

In parallel for each conjugacy class C_s, $O(n/\log n)$ processors can compute the product Product$[s] = \Pi_{i=1}^{n}\left(i^{k_i}k_i!\right)$ in $O(\sqrt{k}\log n)$ time (each of the i^{k_i} factors require a tree computation) by just scanning the array Cycle_Type$[s,\cdot]$, containing the partition in multiplicity notation (see Section 2.4). Then the cardinality of C_s is stored in Card$[s]$ = $\mathrm{Fact}[n,s]/\mathrm{Product}[s]$ and the computation of the numerator of $\Pr[C_s]$ can be easily completed.

The running time of the whole algorithm is dominated by the time to compute Product, $O(\sqrt{k}\log n)$. Depending on the chosen value of k, the most expensive step in terms of number of processors is either the initial computation of Pow ($O(n^2/\log n)$ processors) or some of the copy operations. $\square$

Theorem 25 *For every $n \in \mathbb{N}^+$ and for every $k \in \{0, \ldots, n\}$, let $g(n,k)$ be the function defined in Lemma 9. Let $C_1, \ldots, C_{u(k)}$ be the first $u(k)$ conjugacy classes in S_n in w.a.l. order. If m, the number of unlabelled graphs on n vertices is given, the probabilities $\Pr[C_s]$ involved in algorithm $\mathcal{NU}$ above, for $s = 1, \ldots, u(k)$, are computable in $O(\sqrt{k} \log n)$ steps using $O(g(n,k))$ processors on an EREW PRAM.*

Proof. Given m, the number of orbits and the quantity computed as described in Lemma 9 $\Pr[C_s] = |C_s| 2^{q(C_s)}/mn!$ for $s = 1, \ldots, u(k)$ can be computed quite easily. $\square$

Theorem 26 *For every $n \in \mathbb{N}^+$, there exists a constant $c \in \mathbb{R}^+$ such that if $k \leq c \log^2 n$ and all probabilities $\Pr[C_s]$ for $s = 1, \ldots, u(k)$ have been computed, then step (1.1) of $\mathcal{NU}$ can be implemented in constant parallel time and sublinear work on an EREW PRAM with high probability.*

Proof. The parallel algorithm implementing step (1.1) of $\mathcal{NU}$ is described below.

Input: n the number of vertices, the values $\Pr[C_s]$ for all $s = 1, \ldots, u(k)$

$P[0] \leftarrow 0$;

$P[1 : u(k)] \leftarrow$ prefix($\Pr[C], u(k), +$);

$\xi \leftarrow$ rand$(0, 1)$;

for all $s \in \{1, \ldots, u(k)\}$ **in parallel do**

    if $((P[s-1] \leq \xi)$ **and** $(P[s] > \xi))$

        output(C_s) and stop;

The value of c can be found from Lemma 6.

The probability that the process does not succeed is

$$1 - \sum_{i=1}^{u(k)} \Pr[C_i] = \sum_{i=u(k)+1}^{p(n)} \Pr[C_i]$$

The following result is from [HP73, p. 198]:

$$\sum_{i=u(k)+1}^{p(n)} \Pr[C_i] \leq \frac{2^{\binom{n}{2}}}{m\, n!} O\left(\frac{n^k}{2^{nk/2}}\right)$$

From which we deduce that the probability of failure is $O\left(\frac{n^k}{2^{nk/2}}\right)$. $\square$

Selecting a permutation. Given a conjugacy class, for the purposes of the algorithm $\mathcal{NU}$, any $g \in C$ "behaves" in the same way: by Theorem 16.2, $|Fix(g)|$ is the same for every $g \in C$.

i =	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
min =	1	1	3	3	3	3	3	3	3	3	11	11	11	11	11	11	11	11	11	20	20	20	20	20	20
length =	1	1	2	2	2	2	2	2	2	2	3	3	3	3	3	3	3	3	3	6	6	6	6	6	6
next =	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-

Figure 2.5: Values of G after step (2).

Definition 8 *If C has cycle type whose non-zero elements are $((k_{j_1}, j_1), \ldots, (k_{j_c}, j_c))$ then the* canonical representative $g \in C$ *is the permutation such that the numbers involved in cycles of length j_x (for $x \in \{1, \ldots, c\}$) are the numbers between $1 + \sum_{y<j_x} yk_y$ and $j_x k_{j_x} + \sum_{y<j_x} yk_y$ in the usual order.*

Example. If $n = 16$ and the conjugacy class has the associated cycle type $((3,1)(2,4)(1,5))$ then its canonical representative is $g = (1)(2)(3)(4\ 5\ 6\ 7)(8\ 9\ 10\ 11)(12\ 13\ 14\ 15\ 16)$.

Lemma 10 *Given a conjugacy class C such that all permutations in C have cycle type $((k_{j_1}, j_1), \ldots, (k_{j_c}, j_c))$, the canonical representative of C can be built deterministically in $O(\log n)$ parallel steps and $O(n)$ work on an EREW PRAM.*

Proof. The permutation g will be again represented by an array of n elements such that the ith element contains the address (i.e. the index) of the element following i in its cycle. The array can be built in $O(\log n)$ steps using $O(n/\log n)$ processors. Indeed the real bottleneck in the running time is $O(\log k)$ in the first stage; all other stages can be run in optimal linear work and even constant parallel time. Given the chosen cycle type in the array Cycle_Type$[s, \cdot]$ the algorithm will define the contents of an array of length n whose elements are four-tuples:

- G$[i]$.min is the smallest number belonging to a cycle of the same length as the one that i belongs to.
- G$[i]$.length is the length of the cycle that i belongs to.
- G$[i]$.next is the value of $g(i)$.

A detailed algorithmic description is as follows:

(1) Initialise a vector Offst such that Offst$[1]$ is equal to one and, for all $i \in \{2, \ldots, n\}$, Offst$[i]$ is equal to

$$\text{Cycle\_Type}[s, i].\text{first} \cdot \text{Cycle\_Type}[s, i].\text{second},$$

and run parallel prefix on it. This takes $O(\log k)$ parallel steps using $O(\sqrt{k}/\log k)$ processors on an EREW PRAM since there are only at most $O(\sqrt{k})$ non-zero elements in Cycle_Type. After this stage Offst$[i]$ will contain the first element involved in a cycle of length i (for every length that is actually present). If $n = 25$ and the given cycle type is $(2,1),\ (4,2),\ (3,3),\ (1,6)$ then Offst $\equiv 1, 3, 11, 20, 26$.

(2) Define the elements in position Offst$[j]$ (for $j < |C|$) in G by the tuple:

$$(\mathrm{Offst}[j], \mathrm{Cycle\_Type}[s,i].\mathrm{second}, 0)$$

then replicate elements min and length so that each (processor associated with a particular) cell in the result array knows what is the length of the cycle it belongs to and what is the first element in a cycle of the same length. In the example above after this stage the fields of G contain the values reported in Figure 2.5. The worst case is when all cycles have the same length. In this case the optimal copying steps run in $O(n)$ work.

(3) It is now possible to compute the field next in constant parallel time and optimal linear work. G$[i]$.next is always $i + 1$ unless
(a) G$[i].\mathrm{length} = 1$, in this case G$[i].\mathrm{next} = i$; or
(b) $(i+1-$ G$[i].\mathrm{min}) \bmod$ G$[i].\mathrm{length} = 0$, in which case G$[i].\mathrm{next} =$ G$[i].\mathrm{min} +$ G$[i].\mathrm{length}((i -$ G$[i].\mathrm{min}) \div$ G$[i].\mathrm{length})$ where "$\div$" returns the integral part of the division between its two operands.

□

2.7 Algorithm Based on Conjugacy Class Listing

The use of the restarting paradigm described in Section 2.3 seems to be crucial to the definition of RNC algorithms for sampling unlabelled graphs of given order with uniform distribution. The RNC algorithm described in this section does not require the number of unlabelled graphs on n vertices as input.

Let $C_1, \ldots, C_{u(k)}$ be conjugacy classes with permutations containing at least $n - k$ fixed elements and let $C_0 = S_n \setminus \bigcup_{i=1}^{u(k)} C_i$. Define

$$B_0 = 2^{\binom{n}{2}} \sum_{i=k+1}^{n} 2^{\frac{i(i+2)}{4} - \frac{in}{2}} \frac{n!}{(n-i)!}$$

and $B = B_0 + \sum_{i=1}^{u(k)} |C_i||Fix(g)|$. Consider the following sequential algorithm $\mathcal{A}$,

(1.1) select $i \in \{0, 1, \ldots, u(k)\}$ with probability

$$\Pr[C_i] = \begin{cases} \frac{B_0}{B} & \text{if } i = 0; \\ \frac{|C_i| 2^{q(C_i)}}{B} & \text{otherwise} \end{cases}$$

(1.2) if $i = 0$ then select a permutation $g \in C_0$ uniformly at random otherwise construct a representative $g \in C_i$ (by Lemma 16.*1* in Section 1.2, if $i > 0$ then g can be any member of C_i).

(1.3) goto (2) with probability

$$\Pr[\text{Out}|(i, g)] = \begin{cases} \frac{|C_0||Fix(g)|}{B_0} & \text{if } i = 0; \\ 1 & \text{otherwise} \end{cases}$$

(otherwise go back to step (1.1)).

(2) select uniformly at random $\alpha \in Fix(g)$ and return its orbit.

A single execution of steps (1.1) through (1.3) of $\mathcal{A}$ is called a *run*. A *successful run* is a run which ends with goto (2).

Theorem 27 *Algorithm $\mathcal{A}$ generates unlabelled graphs on n vertices uniformly at random.*

Proof. The probability that $\mathcal{A}$ generates a particular graph is

$$\begin{aligned} \Pr[\mathcal{A} = \alpha] &= \sum_{i=1}^{u(k)} \frac{1}{2^{q(C_i)}} \frac{|C_i| 2^{q(C_i)}}{B} + \sum_{g \in C_0} \frac{|C_0||Fix(g)|}{B_0} \frac{1}{|Fix(g)|} \frac{1}{|C_0|} \frac{B_0}{B} \\ &= \frac{1}{B} \sum_{i=1}^{u} |C_i| + \sum_{g \in C_0} \frac{1}{B} = \frac{n!}{B} \end{aligned}$$

which is the same for all graphs. □

The natural parallelisation of algorithm $\mathcal{A}$ consists of executing ρ runs independently and in parallel and outputting a graph if at least one run is successful. It will be shown that if $\rho = \Omega(\log n)$ then the probability that there is no successful run is $O(1/n)$. The following paragraphs provide details of the implementation of the single steps. Theorem 32 is the main algorithmic result of this section.

Selecting a conjugacy class. The implementation of step (1.1) in algorithm $\mathcal{A}$ assumes that probabilities $\Pr[C_i]$ for $i = 0, \ldots, u(k)$ have been computed. The pseudo-code for this pre-processing phase is as follows. The value of k will be fixed in Theorem 29.

(1) Form a list of the first $u(k)$ partitions π_j in w.a.l. order.

(2) Compute $\Pr[C_j]$ for all $j \in \{1, \ldots, u(k)\}$.

(3) Run parallel prefix on the probability vector and store the result into an array P; then define $\Pr[C_0]$ as $1 - P[u(k)]$.

(4) Finally update the vector P so that $P[j] \leftarrow P[j] + \Pr[C_0]$ for all $j \in \{0, \ldots, u(k)\}$.

The most expensive steps in this process are (1) and (2)-(3). Theorem 22 deals with the partition generation. Theorem 28 below shows that as long as not too many conjugacy classes are listed, the selection probabilities can be tabulated in NC.

Theorem 28 *For every $n \in \mathbb{N}^+$ and for every $k \in \{0, \ldots, n\}$, let $g(n,k)$ be the function defined in Lemma 9. The probabilities $\Pr[C_s]$ for $s = 0, \ldots, u(k)$ are computable in $O(\sqrt{k} \log n)$ steps using $O(g(n,k))$ processors on an EREW PRAM.*

Proof. Step (3) of the algorithm above shows how to compute $\Pr[C_0]$, given all other probabilities, using parallel prefix. Hence we only need to show how to define $\Pr[C_s] = |C_s| 2^{q(C_s)}/B$ for $s = 1, \ldots, u(k)$. By Lemma 9 the quantities $|C_s| 2^{q(C_s)}$ for all s can be computed in $O(\sqrt{k} \log n)$ parallel steps using $O(g(n,k))$ processors on an EREW PRAM. A final piece of code is needed to compute B_0. This can be described as follows:

(1) Initialise $B_0 = 2^{\binom{n}{2}}$. This can be done in $O(\log n)$ parallel steps using optimal $O(n^2)$ work.

(2) Copy n times Fact$[n]$. Then processor P_i for $i = k+1, \ldots, n$ uses Fact$[n,i]$ to define Ratio$[i] =$ Fact$[n,i]/$Fact$[n-k+i, 1]$ in optimal linear work.

(3) The number $h(k+1, n)^{\frac{1}{k+1}} = 2^{-\frac{n-1}{2} + \frac{k+1}{4}}$ is computed using an optimal linear work algorithm and stored in Pow2$[1,1]$.

(4) For $i = 2, \ldots n-k$, Pow2$[i,1]$ is initialised to $\sqrt[4]{2}$ and then parallel prefix is run on it. After this Pow2$[i,1] = 2^{-\frac{n}{2} + \frac{k+i+2}{4}}$.

(5) Then, each Pow2$[i,1]$ is copied $k+i-1$ times and a tree compu-

tation is run. After this $\text{Pow2}[i,1] = 2^{-\frac{(k+i)n}{2}+\frac{(k+i)(k+i+2)}{4}}$ (for all $i = 1,\ldots,n-k$).

(6) Using a tree computation on the product $\text{Pow2}[i,1]\text{Ratio}[i]$ the summation in the definition of B_0 is easily computed in $O(\log n)$ parallel time and linear work. Then B_0 can be redefined in terms of its initial value and the sum above.

(7) $B - B_0$ can be easily computed by tree$(\text{Card}, u(k), +)$ and B defined consequently. Finally $\Pr[s] \leftarrow \text{Card}[s]/B$ in constant time for all $s = 1,\ldots,u(k)$. All this can be achieved in $O(\log n)$ parallel steps using $O(u(k)/\log n)$ processors.

$\square$

Theorem 29 *For every $n \in \mathbb{N}^+$, there exists a constant $c \in \mathbb{R}^+$ such that if $k \leq c\log^2 n$ and all probabilities $\Pr[C_s]$ for $s = 0,\ldots,u(k)$ have been computed, then step (1.1) of $\mathcal{A}$ can be implemented in constant parallel time and sublinear work on an EREW PRAM.*

Proof. Similar to that of Theorem 26. $\square$

Selecting a permutation. If the class chosen by step (1.1) is C_0, a permutation $g \in C_0$ is generated uniformly at random. There are several algorithms for generating permutations in parallel uniformly at random (see for instance [AS91]). If R_i is the set of permutations which move exactly i elements out of n then $g \in C_0 = \bigcup_{i\geq k+1} R_i$ is built by first selecting one of the R_i with probability proportional to its size and subsequently selecting a random permutation g moving exactly i objects using a method described in [Wor87].

Lemma 11 *For every $n \in \mathbb{N}^+$, $|R_i| = \frac{n!}{(n-i)!}\sum_{j=0}^{i}(-1)^j\frac{1}{j!}$ for all $i \in \{1,\ldots,n\}$. Moreover all the numbers $|R_i|$ can be computed in $O(\log n)$ time and linear work on an EREW PRAM.*

Proof. Given a finite set S, a simple combinatorial principle, known as *inclusion-exclusion* states (see for instance [Rio58, Ch. 3]) that if $A_i = \{x \in S : P_i(x)\}$ for some property P_i with $i = 1,2,\ldots,m$ then the number of objects in S that have none of the properties P_i is

$$|\bigcap_{i=1}^{m}\overline{A_i}| = |S| - \sum|A_i| + \sum|A_i\cap A_j| - \sum|A_i\cap A_j\cap A_k| + \ldots + (-1)^m|A_1\cap A_2\cap\ldots\cap A_m|$$

In the case of interest, let $A_i = \{g \in S_n : g(i) = i\}$. Then $|A_i| = (n-1)!$ and in general $|A_{i_1} \cap \ldots \cap A_{i_j}| = (n-j)!$. This implies that the number of derangements of i objects is

$$D(i) = \sum_{j=0}^{i} (-1)^j \binom{i}{j} (i-j)! = i! \sum_{j=0}^{n} (-1)^j \frac{1}{j!}.$$

The number of ways of selecting an element in R_i is the number of ways of choosing $n-i$ fixed objects out of n, times the number of derangements on the remaining i objects.

(1) In Theorem 28 it was proved that a vector Fact of length n whose j-th position contains $j!$ can be computed in $O(\log n)$ time and linear work. Using this vector another vector Sums such that Sums$[i] = \sum_{j=0}^{i} (-1)^j \frac{1}{j!}$ can then be computed within the same complexity.

(2) A third vector Frac containing the ratios $\frac{n!}{(n-i)!}$ can again be computed within the same complexity.

(3) Finally (in constant time using a linear number of processors) $|R_i| = \mathrm{Frac}[i]\,\mathrm{Sums}[i]$.

The pseudo-code for the algorithm described above is

```
Fact[0, 1] ← 1;
Ft ← Fact[n, 1];
for all i ∈ {0, ..., n} in parallel do
    if odd i
        Sums[i] ← −1/Fact[i, 1];
    else Sums[i] ← 1/Fact[i, 1];
    F[i] ← Ft/Fact[n − i, 1]
Sums ← prefix(Sums, n + 1, +);
for all i ∈ {0, ..., n} in parallel do
    R[i] ← F[i] Sums[i];
```

□

Lemma 12 *Elements of C_0 can be generated uniformly at random in $O(\log n)$ parallel steps using δn processors with high probability if $\delta = \Omega(\log n)$.*

Proof. First a class R_i is selected with probability $\frac{|R_i|}{\sum |R_i|}$. Then δ groups of n processors are allocated. Each of them, independently, selects a random permutation of $1, 2, \ldots, i$ (in $O(\log i)$ time steps and i processors using a result in [AS91]). Let the random variable X denote the number of derangements in δ trials. X is binomially distributed with success probability $p = D(i)/i!$, where $D(i)$ was defined in Lemma 11. Since $\mathrm{e}^{-1} = \sum_{i=0}^{\infty} \frac{(-1)^i}{i!}$, it follows that $p \sim \mathrm{e}^{-1}$ (details are in [GKP89, pp. 195]). Then $\Pr[X = 0]$ can be bounded above by $\mathrm{e}^{-O(\delta)}$ using standard results on the tail of the binomial distribution (for example Theorem 9) and the result follows. □

If step (1.1) returns a conjugacy class then, for the purposes of the algorithm $\mathcal{A}$, any $g \in C$ "behaves" in the same way: by Lemma 16.2, $|Fix(g)|$ is the same for every $g \in C$. By Lemma 10 the canonical permutation can be generated deterministically in $O(\log n)$ parallel steps and linear work on a EREW PRAM.

Theorem 30 *Step (1.2) of $\mathcal{A}$ can be implemented in $O(\log n)$ time and $O(n \log^2 n)$ work on an EREW PRAM with high probability.*

Restarting Probability. If C_0 was chosen in step (1.1) and a random permutation $g \in C_0$ was selected in step (1.2) the algorithm $\mathcal{A}$ proceeds to the selection of a graph in $Fix(g)$ with probability $|C_0||Fix(g)|/B_0$. The implementation of step (1.3) chosen here does not present any particular difficulty. The values of B_0 has been computed already (see Theorem 28). Also, $|C_0| = n! - \sum_{i=1}^{u(k)} |C_i|$ can be easily computed in $O(\log n)$ parallel steps and optimal linear work using the data structures defined in Theorem 28. Finally, the cycle type of g^* can be defined using pointer jumping and algorithmic solutions as in Section 2.5.

Probabilistic Analysis. The parallelisation of algorithm $\mathcal{A}$ is obtained by performing ρ independent trials each simulating a run of $\mathcal{A}$, in parallel. In this final paragraph an upper bound on the probability that the whole process fails to produce a graph over ρ trials is given. Let X_s be a random indicator equal to one if the s-th run is successful.

Lemma 13 *If m is the number of unlabelled graphs on n vertices then* $\Pr[X_s = 1] \sim \frac{n! m}{B}$.

Proof. The algorithm chooses an index i and a permutation $g \in C_i$. Let $\mathrm{Out} = \mathrm{Out}_{(i,g)}$ denote the event "the algorithm terminates given that some i and g were generated".

$$\Pr[X_s = 1] = \sum_{(i,g)} \Pr[\mathrm{Out}|(i,g)] \Pr[(i,g)]$$

Remembering that

$$\Pr[\mathrm{Out}|(i,g)] = \begin{cases} \frac{|C_0||Fix(g)|}{B_0} & \text{if } i = 0; \\ 1 & \text{otherwise} \end{cases}$$

and

$$\Pr[(i,g)] = \begin{cases} \frac{B_0}{B|C_0|} & \text{if } i = 0; \\ \frac{|C_i||Fix(g)|}{B} & \text{otherwise} \end{cases}$$

Simplifying and making the sums explicit we get (here $\sigma \sim 1$ is the probability that Step (2) does generate a permutation in C_i)

$$\Pr[X_s = 1] = \sigma \sum_{i=0}^{u(k)} \sum_{g \in C_i} \frac{|Fix(g)|}{B} = \frac{\sigma}{B} \sum_{i=0}^{u(k)} \sum_{g \in C_i} |Fix(g)| = \frac{\sigma n! m}{B}$$

The last equality follows by Fröbenius' lemma since $\sum_{i=0}^{u(k)} \sum_{g \in C_i} |Fix(g)| = \sum_{g \in S_n} |Fix(g)|$. □

Let $X = \sum_s X_s$ count the number of successful runs over ρ trials. By Theorem 9, $\Pr[X = 0] \leq \mathrm{e}^{-O(\mathrm{E}(X))}$. The following result shows that the expected value of X is sufficiently high.

Lemma 14 $\mathrm{E}(X_s) = \Theta(1)$.

Proof. By Theorem 15.2 $n!m = \sum |C| 2^{q(C)}$ where the sum is over all conjugacy classes. We need to prove that $B_0 \sim \sum_{i > u(k)} |C_i| 2^{q(C_i)}$. Let $R_i = \{g : g \in S_n, g \text{ moves } i \text{ elements}\}$. $C_0 = \bigcup_{i=k+1}^{n} R_i$ and $|R_i| \sim \frac{n!}{(n-i)!}$ (for all $i \geq 1$).

Now we write B_0 in terms of the conjugacy classes and by Lemma 8

$$B_0 \sim \sum_{i=k+1}^{n} 2^{\binom{n}{2} - h(i,n)} \sum_{C \subseteq R_i} |C| \geq \sum_{i > u(k)} |C_i| 2^{q(C_i)}$$

From above, direct calculations show that

$$B_0 = \sum_{i=k+1}^{n} 2^{\binom{n}{2} - h(i,n)} \frac{n!}{(n-i)!} \leq n!\, 2^{\binom{n}{2}} \sum_{i=k+1}^{n} \left(\frac{2^{\frac{i+2}{4}}}{2^{\frac{n}{2}}} \right)^i \leq n!\, 2^{\binom{n}{2}} \sum_{i=k+1}^{n} \left(\frac{1}{2^{\frac{n-2}{4}}} \right)^i$$

from this $B_0 \leq n!\, |C_1| 2^{q(C_1)}$ (where C_1 is the conjugacy class formed by the identity permutation). Hence the contribution of B_0 to B is asymptotically smaller than that of $\sum_{i=1}^{u(k)} |C_i| 2^{q(C_i)}$ and since $\sum_{i=1}^{u(k)} |C_i| 2^{q(C_i)} \sim n!m$ (see [HP73]) the result follows. □

Theorem 31 *If $\rho = \Omega(\log n)$ then with high probability there exists a successful run.*

Proof. Follows from Lemma 14 and the previous discussion. □

Putting together Theorem 28, Theorem 29, Theorem 30, Theorem 23 and the argument on the restarting probabilities we get an RNC algorithm for generating uniformly at random unlabelled graphs of given order.

Theorem 32 *For every n, there exists an RNC algorithm for generating unlabelled graphs on n vertices uniformly at random in $O(\log n)$ time and $O(n^2 \log n)$ work on an EREW PRAM. Moreover an output is generated with high probability.*

Proof. The most expensive step in the proposed parallel implementation of algorithm $\mathcal{A}$ is the preprocessing required to compute the probabilities $\Pr[C]$. However this only needs to be done once. If ρ trials are run in parallel and k is chosen to be a constant, the resulting algorithm runs in $O(\log n)$ time and $O(\rho n^2)$ work on a EREW PRAM. The result follows by choosing $\rho = O(\log n)$. □

2.8 Avoiding Conjugacy Classes

In [Wor87] an alternative sequential algorithm is described which does not require any information about the conjugacy classes in S_n. The collection $(R_i)_{i=1}^n$, as defined in Lemma 14, is a partition of S_n. If $B_1 = 2^{\binom{n}{2}}$ and for $2 \leq i \leq n$, $B_i = 2^{\binom{n}{2}-h(i,n)} \frac{n!}{(n-i)!}$ where $h(i,n)$ is the function defined in Lemma 8 then $|R_i| \leq B_i$ for all $i = 1, \ldots, n$.

The sequential generation algorithm $\mathcal{W}$ is described by the following steps (in what follows $M = \sum_{i=1}^n B_i$):

(1.1) select R_i with probability $\frac{B_i}{M}$;

(1.2) select uniformly at random $g \in R_i$;

(1.3) goto (2) with probability $\frac{|R_i||Fix(g)|}{B_i}$ (otherwise the run is failing and the whole process is started again).

(2) select uniformly at random $\alpha \in Fix(g)$ and return its orbit.

As before on a successful iteration the distribution of the output of $\mathcal{W}$ is uniform over unlabelled graphs on n vertices and each iteration has a fixed (very high) success probability $\frac{n!m}{M}$.

The parallel implementation of $\mathcal{W}$ does not need to be described in great details since most of the work has been done already. After a preprocessing stage in which all the B_i, for $i = 1, \ldots, n$ are computed, ρ trials are run in parallel each consisting in the successive selection of a class R_i, a permutation $g \in R_i$ and the computation of the probabilities to restart the process. If $\rho = \Omega(\log n)$ at least one of the trials ends with high probability (see [Wor87]). Finally one of the terminating processes is chosen and a graph in $Fix(g)$ is generated. If all the B_i have been computed, the numerical calculations involved in (the parallel version of) Step (1.3) above can be performed in

$O(\log n)$ steps using $O(n^2/\log n)$ processors since $q(C) \leq n^2$. Step (1.2) can be performed using the same algorithm described in Section 2.7. A graph can be output using the "usual" final phase (see Section 2.5).

Lemma 15 *For all $n \in \mathbb{N}^+$, the bounds B_i, the selection probabilities B_i/M and all the $|R_i|$ for all $i \in \{0,\ldots,n\}$, in algorithm $\mathcal{W}$ can be computed in $O(\log n)$ steps using $O(n^2/\log n)$ processors on an EREW PRAM.*

Proof. An array B$[i]$ is used to store the bounds defined at the beginning of the section. In particular B$[1]$ can be defined in $O(\log n)$ steps using $O(n^2/\log n)$ processors. Then all other B$[i]$ for $i = 2,\ldots,n$ are computed as follows

$B \leftarrow \text{copy}(B[1], n)$;
for all i, $2 \leq i \leq n$ **in parallel do**
    $V[i] \leftarrow n - i + 1$;
$V \leftarrow \text{prefix}(V, n-1, \times)$
Compute Pow2$[i, 1]$ as in Section 2.7
for all i, $2 \leq i \leq n$ **in parallel do**
    Define $B[i]$ in terms of their initial value, $V[i]$ and Pow2$[i, 1]$

$M \leftarrow \text{tree}(B[\cdot], n, +)$ and for all $1 \leq i \leq n$ $\Pr[i] = B[i]/M$. Finally compute the vector $|R_i|$ for $i = 1,\ldots,n$ as in Section 2.7. □

Theorem 33 *There exists a randomised parallel algorithm for generating unlabelled graphs uniformly at random in time $O(\log n)$ and $O(n^2 \log n)$ work on an EREW PRAM which succeeds with high probability.*

Proof. The algorithm listed after this proof solves the problem. The time to compute every iteration of the inner loop is dominated by the time to generate $g \in R_i$ uniformly at random (which is $O(\log n)$ and $O(\delta n)$ processors by Lemma 12) and the time to compute $|Fix(g)| = 2^{q(C)}$. To compute $|Fix(g)|$ the cycle type of g has to be determined first and then $2^{q(C)}$ can be computed using a simplified version of the algorithm in Theorem 28 in $O(\log n)$ time using $O(n^2/\log n)$ processors (all the required powers of 2 have been computed already). The final selection of α can be performed again as in the algorithm $\mathcal{A}$. So the overall running time is $O(\log n)$ using at most $O\left(\rho\frac{n^2}{\log n} + \frac{n^2}{\log n}\right)$ processors. □

for all $j, 1 \leq j \leq \rho$ **in parallel do**

$P \leftarrow \text{prefix}(\Pr[i], n, +)$;

$\xi \leftarrow \text{rand}(0, 1)$;

for all $l \in \{1, \ldots, n\}$ **in parallel do**

if $((P_{l-1} \leq \xi)$ **and** $(P_l > \xi))$

$i_j \leftarrow l$;

Generate a random permutation $g_j \in R_{i_j}$;

Compute $|Fix(g_{i_j})|$ as in Section 2.5;

$\zeta_j \leftarrow \text{rand}(0, 1)$

if $(g_j$ generated $) \wedge \left(\zeta_j \leq \frac{|R_{i_j}||Fix(g_j)|}{B_{i_j}}\right)$

$A_j \leftarrow 1$;

else $A_j \leftarrow 0$;

$\hat{j} = \min\{j \mid A_j \neq 0\}$;

if defined$(\hat{j})$

Choose $\alpha \in Fix(g_{\hat{j}})$;

Return the orbit of α; Stop;

2.9 Conclusions

In this chapter we have presented some of the issues involved in the efficient parallel generation of unlabelled undirected graphs. After presenting our definition of uniform generator we analysed the main steps involved in sequential algorithms. We showed how some of them can be parallelised quite easily, whereas some others present some problems. Using rejection sampling and algorithmic techniques from [Wor87] we showed the existence of an RNC for generating unlabelled graphs on n vertices uniformly at random in $O(\log n)$ parallel steps using $O(n^2)$ processors.

Chapter 3

Approximating Combinatorial Thresholds

This chapter explores another use of randomness in Computer Science. Probabilistic techniques can be used to gain some structural information about specific decision problems. Empirical studies have indicated that a number of decision problems exhibit the following behaviour: if two integer functions order(x) and size(x) (see Section 1.2.1) are defined for every instance x, and $\mathcal{I}_n$ is the set of instances of order n, then most of those $x \in \mathcal{I}_n$ for which $m = \mathrm{size}(x)$ is "small" have $\mathcal{SOL}(x) \neq \emptyset$ and most instances for which m is "large" have $\mathcal{SOL}(x) = \emptyset$. Furthermore, there is an apparent *threshold function* $\theta = \theta(n)$ such that, if $x \in \mathcal{I}_n$ whenever $m < \theta(n)$ then $\mathcal{SOL}(x) \neq \emptyset$ and whenever $m > \theta(n)$ then $\mathcal{SOL}(x) = \emptyset$. Behaviour of this nature is known as a *phase transition*.

For the satisfiability problem introduced in Section 1.2, the order of an instance ϕ is the number of variables occurring in ϕ and m is the number of clauses in ϕ. Experiments reported in [HW94] suggest that a phase transition occurs at $m = 4.24\ n$. By purely analytical means Goerdt [Goe92] proved that a sharp threshold exists at $m = n$ for the polynomial time solvable 2-SAT (similar results were obtained independently in [CR92] who also considered the general k-SAT case). Results for $k > 2$ are much weaker and, although the existence of a threshold has been recently confirmed [Fri97], its exact "location" has yet to be found. Initial upper bounds on $\theta(n)$ for $k = 3$ were obtained in [CS88] where it was proved that "almost all" m-clause, n-variable instances with $m > 5.19\ n$ are unsatisfiable. The best results to date are

Theorem 34 [KKKS98] *Almost all formulae with more than* $4.601\ n$ *clauses are not satisfiable.*

Theorem 35 [FS96] *Almost all formulae with less than* $3.003\ n$ *clauses are satisfiable.*

Phase transition phenomena are not restricted to decision problems. In fact the last two sections in Chapter 4 will provide examples of such phenomena in the context of optimisation problems.

The importance of studying these problems lies partly in the fact that they shed more light on the combinatorial properties of specific problems, partly in that the presence and location of the threshold function seems to be related to regions in the parameter space where the problem is most difficult to solve [CS88, HW94]. Far from attempting to settle such ambitious claims we concentrate on the combinatorial aspects of this type of problems. The aim of this chapter is twofold. In the first part we will define and analyse the phase transitional behaviour of another graph theoretic property, that of vertex k-colourability (k-COL for short) [GJ79]. A very simple argument given in Section 3.1.1 shows that almost all n vertex graphs with more than $2.71n$ edges cannot be "legally" coloured with only three colours. By a more careful analysis of a necessary condition for the existence of one such colouring, in Section 3.1.2 we are then able to strengthen this bound to $2.61n$.

In the second part of the chapter, we describe a probabilistic technique which can be useful in approximating the threshold function of several combinatorial problems. Roughly speaking sometimes the non-existence of a property in a given input instance is implied by the simultaneous truth of a number of local conditions involving elementary building blocks of the instance. This is well modelled in a random setting by the so-called *coupon collector* problem. In Section 3.2 we give the relevant definitions and then, using some tight asymptotic expressions for the probabilities associated with coupon collector instances, we improve the upper bound on the unsatisfiability threshold given by Theorem 34 to $4.5793n$.

3.1 Improved Upper Bound on the Non 3-Colourability Threshold

In this Section some new results on the non 3-colourability threshold are presented. After giving all relevant definitions and a simple bound (see Theorem 36 below), the main result of the Section is presented in Theorem 37. The Section finishes with a number of remarks and possibilities of further developments.

3.1.1 Definitions and Preliminary Results

The informal notion of phase transition described above can be formalised once the measure with respect to which the sharp change in behaviour is observed is defined. The approach taken in this thesis is probabilistic. Given a decision problem Q, a probability space can be associated with the class of all instances $\mathcal{I} = \bigcup_n \mathcal{I}_n$ (where, as usual, $\mathcal{I}_n$ is the set of instances of order n), by assigning a probability to each $x \in \mathcal{I}$. The expression $\Pr[x \in Q \mid x \in \mathcal{I}_n]$, abbreviated as $\Pr_n[Q]$, is the probability that $x \in \mathcal{I}_n$ is in Q. Usually $\Pr_n[Q]$ will also depend on some other parameters of the specific problem. For graph problems, if n is the number of vertices a natural second parameter is m, the number of edges. Thus $\Pr_{n,m}[Q]$ is the probability that an instance of order n and size m belongs to Q. We are now ready to define the notion of threshold function and phase transition.

Definition 9 *A monotone decreasing decision problem Q has a* phase transition *with* threshold function $\theta(n)$ *if*

(a) if $m/\theta(n) < 1 - o(1)$ *then* $\Pr_{n,m}[Q] \to 1$

(b) if $m/\theta(n) > 1 + o(1)$ *then* $\Pr_{n,m}[Q] \to 0$.

A recent result [FK96] states that many monotone graph properties have a phase transition in the sense of Definition 9. Unfortunately, this result does not imply that the exact location of the threshold function is known for every specific problem. It is thus interesting to prove constructive bounds on the threshold function for specific problems.

The proofs of Theorem 34 and 35 are based on fairly standard techniques for approximating a threshold function. For a monotone decreasing problem Q an upper bound on the threshold function is given by the so called *first moment method*. This is described as follows:

1. We define a r.v. $X = X_{n,m}$ counting the number of solutions for a random instance of order n and size m. The event "$X = 0$" is equivalent to the event "$\mathcal{SOL}(x) = \emptyset$".

2. We find $\mathrm{E}(X)$ as a function of n and m.

3. We find m_0, the smallest value of m such that for all $m > m_0$ we have that $\lim_{n\to\infty} \mathrm{E}(X) = 0$.

4. We use the Markov inequality (see Theorem 6) to prove that $\lim_{n\to\infty} \Pr_n[Q] = 0$.

If $\lim_{n\to\infty} \mathrm{E}(X) = \infty$, sometimes a lower bound on the threshold function can be proved by the more refined *second moment method* which is based on an analysis of $\mathrm{Var}(X)$ and Corollary

1 in Section 1.2.3. Unfortunately in many interesting cases the variance is rather large relative to the mean, so that we cannot hope to prove that $X > 0$ almost always using the second moment method. In some cases, such as the constructive proof of Theorem 35, it is possible to define an algorithm and to prove that if the size of the input is sufficiently small then $X > 0$ almost always.

Given an undirected graph $G = (V, E)$ a *k-colouring* is a mapping $\chi : V \to \{1, \ldots, k\}$. Notice that the mapping is not required to be surjective so any $(k-i)$-colouring (for $1 \leq i < k$) is a k-colouring. The k-colouring is *legal* if

$$\{u, v\} \in E \Rightarrow \chi(u) \neq \chi(v)$$

The *vertex k-colourability* problem (or k-COL for short) is the decision problem that takes as input a graph $G = (V, E)$ and whose solution set is the class of all graphs for which there exists a legal k-colouring. To simplify notations $\mathcal{SOL}(G)$ will be abbreviated as Ξ_G or simply Ξ when the graph G is clear from the context. Also, in the case $k = 3$, we use {red, blue, white} as the set of colours. For any 3-colouring χ let R_χ, B_χ and W_χ be the set of vertices in $V(G)$ coloured red, blue and white, respectively. Lower case letters will be used for their cardinalities, e.g. $r_\chi = |R_\chi|$. The dependence on χ will normally not be shown explicitly.

Since a multigraph and its skeleton have the same number of k-colourings we can study the 3-colourability of random multigraphs and then apply Theorem 11 to deduce corresponding results for simple graphs with a given number of edges. Before stating a very simple upper bound on the 3-colourability threshold we need the following technical result.

Lemma 16 *If $G \in \mathcal{M}(n, m)$ and $e \in E(G)$, the probability that e is legally coloured by a given 3-colouring χ is at most* $2/3 + O(1/n)$.

Proof. If only one colour is used (e.g. for example $r = b = 0$) then trivially the required probability is null. Otherwise there are precisely

$$e(n, b, w) =_{df} \binom{n}{2} - \left[\binom{n-(b+w)}{2} + \binom{b}{2} + \binom{w}{2}\right] = n(b+w) - (b^2 + w^2 + bw)$$

ways of choosing an edge in G that is legally coloured by χ. For every fixed value of n define the function $\tilde{e}_n(x, y) : (\mathbb{R}^+)^2 \to \mathbb{R}$ by $\tilde{e}_n(x, y) = e(n, x, y)$. The point $(x, y) = (n/3, n/3)$ is stationary for $\tilde{e}_n(x, y)$, the Hessian matrix of $\tilde{e}_n(x, y)$ is $\left(\begin{smallmatrix} -2 & -1 \\ -1 & -2 \end{smallmatrix}\right)$. This implies that for every $(x, y) \in [0, n] \times [0, n]$ we have $\tilde{e}_n(x, y) \leq n^2/3$. The probability that an edge is chosen so that its endpoints have different colours is at most

$$\frac{n^2}{3}\frac{2}{n(n-1)} = \frac{2}{3}\frac{n}{n-1}$$

□

Theorem 36 *Almost all graphs on n vertices with more than $2.71\,n$ edges are not 3-colourable.*

Proof. If $X = X(G)$ is the r.v. counting the number of legal 3-colourings of a random G sampled according to the model $\mathcal{M}(n,m)$ then the event "$X > 0$" is equivalent to the event "$G \in$ 3-COL" (given that $G \in \mathcal{G}^n$). We use linearity of expectation to compute an upper bound on $\mathrm{E}(X)$. If χ is a 3-colouring of the n vertices (i.e. a tri-partition of $\{1,\ldots,n\}$ in the colour classes R, B and W) let X_χ be the random indicator equal to one if and only if χ is a legal colouring of G. Since each edge in G is chosen independently, the probability that χ is a legal colouring for G is the product of the probabilities that each selected edge is legally coloured by χ. By Lemma 16 this is at most $2n/3(n-1)$. We have $X = \sum_\chi X_\chi$, and since $\mathrm{E}(X_\chi) = \Pr[\chi \in \Xi] \leq [2n/3(n-1)]^m$, the expectation of X is at most $\mathrm{e}^{n\log 3 - m[\log 3(n-1) - \log 2n]}$ and the upper bound is asymptotically very small if $m/n > \log 3/(\log 3 - \log 2) \simeq 2.7096$. The result about simple graphs follows from Theorem 11. □

It might be worth noticing the asymptotic nature of this result. There is only one simple graph on $n = 4$ vertices and $m = 6$ edges (the complete graph K_4) and it is not 3-colourable, whereas there are $6^6 = 46656$ pairs (G,σ) where $G \in \mathcal{M}^{4,6}$ and $\sigma \in S_6$. and only $6! = 720$ of them are not 3-colourable.

3.1.2 Main Result

If 3-$\mathrm{COL}_{n,m}$ is the set of 3-colourable graphs of order n and size m, then

$$\Pr[X > 0] =_{df} \frac{|3\text{-}\mathrm{COL}_{n,m}|}{|\mathcal{G}^{n,m}|}$$

If X_G is a random indicator equal to one if and only if G is 3-colourable then we can write

$$\Pr[X > 0] = \sum_{G \in 3\text{-}\mathrm{COL}_{n,m}} \frac{1}{|\mathcal{G}^{n,m}|} = \sum_{G \in \mathcal{G}^{n,m}} X_G \Pr[G] \leq \sum_{G \in \mathcal{G}^{n,m}} X(G)\Pr[G] = \mathrm{E}(X)$$

This simple proof of the Markov inequality explains the problem we face when using this probabilistic tool in Theorem 36. Graphs with a large number of colourings make a large contribution to the expectation even though they occur with a fairly small probability. In order to reduce this effect

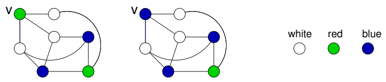

Figure 3.1: A legal 3-colouring (left) and a maximal 3-colouring (right).

and improve Theorem 36 we need to identify a subset of the legal colourings of a graph which is easy to handle combinatorially and compute an upper bound on $\Pr[X > 0]$ based only on these "special" colourings.

Definition 10 *For any graph G a* maximal 3-colouring *is a legal 3-colouring such that by changing the colour of any single red vertex we get an illegal 3-colouring.*

In other words every red vertex in a maximal 3-colouring must be adjacent to a blue and a white vertex. Vertex v in Figure 3.1 is only adjacent to white vertices hence the given colour is not maximal. By recolouring v in blue we obtain a maximal 3-colouring (Figure 3.1 on the right). If G is bipartite it can be vertex coloured using only two colours and without loss of generality we can assume these to be "blue" and "white". Any such 2-colouring is a maximal 3-colouring simply because there is no red coloured vertex. Let $\Xi^{\sharp}_{G}$ (again the dependence on G usually will not be shown explicitly) denote the set of maximal 3-colourings of G. Let $X^{\sharp}$ be the r.v. counting the number of maximal 3-colourings of G.

Lemma 17 *Let $G \in \mathcal{M}(n,m)$. Then*

1. $X_G \leq X^{\sharp} \leq X$.
2. $\Pr[X > 0] \leq \mathrm{E}(X^{\sharp})$.

Proof. If $X_G = 0$ then G is not 3-colourable and $X^{\sharp} = 0$. If $G \in$ 3-COL then there is a very simple procedure which converts every legal 3-colouring of G into a maximal 3-colouring. Let χ be a legal 3-colouring of G and let $R = \{v_1, \ldots, v_r\}$ be the set of vertices that are coloured red. For each $i \in \{1, \ldots, r\}$ recolour v_i either blue or white if this gives a legal colouring (leave $\chi(v_i)$ unchanged, otherwise). After r steps each vertex such that $\chi(v) = R$ has either been recoloured or is such that by changing its colour to blue or white the resulting colouring is not legal anymore. Hence $X_G \leq X^{\sharp}$. $X^{\sharp} \leq X$ is true because all maximal 3-colourings are legal colourings by definition.

The second statement follows from the first one and the chain of inequalities above. $\square$

Using this refined notion of 3-colouring Theorem 36 can be improved as follows.

Theorem 37 *For n sufficiently large, almost all graphs on n vertices with more than $2.61\ n$ edges are not 3-colourable.*

The proof of this result is by the following argument:

1. By Lemma 17 $\Pr[X > 0] \leq \mathrm{E}(X^{\sharp})$.
2. If G has n vertices and $m = c\ n$ edges then there exists a function $h(n,k,c)$ such that $\mathrm{E}(X^{\sharp}) \leq \sum_{k=0}^{n} \mathrm{e}^{nh(n,k,c)}$.
3. For each fixed c it is possible to find two positive real numbers α and ϵ, with $\alpha + \epsilon < 1$, such that for n sufficiently large
   (a) there exists a positive constant δ such that $h(n,k,c) \leq -\delta < 0$ for all $k \in [\alpha n, (\alpha+\epsilon)n]$.
   (b) $h(n,k,c) < h(n,k+1,c)$ for all $k \leq \alpha n$.
   (c) $h(n,k,c) > h(n,k+1,c)$ for all $(\alpha+\epsilon)n \leq k \leq n$.
4. Thus $\mathrm{E}(X^{\sharp}) \leq \sum_{k=0}^{n} \mathrm{e}^{-\delta n}$ and Theorem 37 follows by choosing $c = 2.6028$, $\alpha = 0.696139$ and $\epsilon = 10^{-5}$.

in the remaining part of this Section all these claims are proved. We start by finding an upper bound on $\mathrm{E}(X^{\sharp})$ in terms of the probability that a given 3-colouring is a legal 3-colouring for a random G and the conditional probability that the 3-colouring is also maximal.

Lemma 18 $\mathrm{E}(X^{\sharp}) \leq (2/3 + O(1/n))^{m} \sum_{\chi} \Pr[\chi \in \Xi^{\sharp} \mid \chi \in \Xi]$.

Proof. Let $X^{\sharp}_{\chi}$ be the random indicator equal to one if and only if $\chi \in \Xi^{\sharp}$. Again using linearity of expectation $\mathrm{E}(X^{\sharp}) = \sum_{\chi} \mathrm{E}(X^{\sharp}_{\chi}) = \sum_{\chi} \Pr[\chi \in \Xi^{\sharp}]$. To compute $\Pr[\chi \in \Xi^{\sharp}]$ we use the total probability law (Theorem 2).

$$\begin{aligned}\Pr[\chi \in \Xi^{\sharp}] &= \Pr[\chi \in \Xi^{\sharp} \mid \chi \in \Xi]\ \Pr[\chi \in \Xi] + \Pr[\chi \in \Xi^{\sharp} \mid \chi \notin \Xi]\ \Pr[\chi \notin \Xi] \\ &= \Pr[\chi \in \Xi^{\sharp} \mid \chi \in \Xi]\ \Pr[\chi \in \Xi] + 0 \cdot \Pr[\chi \notin \Xi]\end{aligned}$$

□

Next we need a technical Lemma stating a useful inequality and an important Theorem which will allow us to bound the conditional probabilities involved in the computation of $\mathrm{E}(X^{\sharp})$.

Lemma 19 $[1 - (\alpha n)^{-1}]^{cn} \geq \mathrm{e}^{-\frac{c}{\alpha}} - O(n^{-1})$ *for all positive constants α and c.*

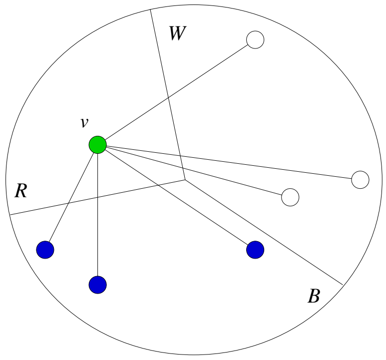

Figure 3.2: Legal edges for a vertex v.

Proof. For $0 < x < 0.69$, $\log(1-x) > -x - x^2$ (see for instance [Bol85, p. 5]), therefore, for sufficiently large n, we have

$$\frac{c}{\alpha} + cn \log\left(1 - \frac{1}{\alpha n}\right) > -\frac{c}{\alpha^2 n}$$

and the result follows by exponentiation. □

The next Theorem is restated from McDiarmid [McD92].

Theorem 38 *Let $\mathcal{V}$ and I be finite non-empty sets. Let $(X_\nu : \nu \in \mathcal{V})$ be a family of independent random variables, each taking values in some set containing I; and for each $i \in I$, let $S_i = \{\nu \in \mathcal{V} : X_\nu = i\}$. Let $(\mathcal{F}_i : i \in I)$ be a family of increasing properties of $\mathcal{V}$. Then*

$$\Pr\left[\bigcap_{i \in I} \{S_i \in \mathcal{F}_i\}\right] \leq \prod_{i \in I} \Pr[S_i \in \mathcal{F}_i].$$

If $e(n, b, w)$ is the function defined in the proof of Lemma 16, for any 3-colouring χ of the set of vertices and any $v \in R_\chi$ there are $e(n, b, w) - b$ (resp. $e(n, b, w) - w$) ways to choose edges connecting v to the rest of the graph so that no blue vertex (resp. white vertex) is adjacent to v (see Figure 3.2 to get some more intuition). Hence if $G \in \mathcal{M}(n, cn)$ with probability

$$\left(\frac{e(n,b,w) - b}{e(n,b,w)}\right)^{cn} = \left(1 - \frac{b}{e(n,b,w)}\right)^{cn}$$

v is not adjacent to any blue vertex. For every $v \in R$ define the events E_v^1 = "v is adjacent to at least one blue vertex" and E_v^2 ="v is adjacent to at least one white vertex". It follows that

$$\Pr[E_v^1] = 1 - \left(1 - \frac{b}{e(n,b,w)}\right)^{cn}.$$

and $\Pr[E_v^2]$ has a similar expression. We have

$$\Pr[\chi \in \Xi^\sharp | \chi \in \Xi] = \Pr\left[\bigcap_{v \in R} (E_v^1 \cap E_v^2)\right]$$

and we aim at using Theorem 38 to upper bound this probability. The following result gives a uniform upper bound on $\Pr[E_v^1]\Pr[E_v^2]$ for every $v \in R$.

Lemma 20 *For every fixed $n \in \mathbb{N}$ and real $c > 0$ let $f_{n,c} : \mathbb{N} \times \mathbb{N} \to [0,1]$ be defined by*

$$f_{n,c}(b,w) =_{df} \left[1 - \left(1 - \frac{b}{e(n,b,w)}\right)^{cn}\right] \left[1 - \left(1 - \frac{w}{e(n,b,w)}\right)^{cn}\right].$$

Then the maximum value of $f_{n,c}(b,w)$ subject to $b + w = k$ is at most $\left\{1 - \left[1 - \frac{1}{2n-(3k/2)}\right]^{cn}\right\}^2$.

Proof. To prove the result we relax integrality assumption and we solve

$$\begin{array}{ll} \max_{(b,w)\in[0,n]\times[0,n]} & f_{n,c}(b,w) \\ \\ \text{s.t.} & b + w = k \\ \\ & k \in [0,n] \end{array}$$

Letting $g_{n,k} = nk - k^2$, we have

$$\begin{aligned} \frac{\partial f_{n,c}(b,k-b)}{\partial b} \quad = \quad & cn\Bigg\{ \left(1 - \frac{b}{g_{n,k} + bk - b^2}\right)^{cn-1} \left[\frac{b^2 + g_{n,k}}{(g_{n,k} + bk - b^2)^2}\right] + \\ & - \left(1 - \frac{k-b}{g_{n,k} + bk - b^2}\right)^{cn-1} \left[\frac{2b^2k^2 - 2kb + g_{n,k}}{(g_{n,k} + bk - b^2)^2}\right] \Bigg\} \end{aligned}$$

The point $b = k/2$ is stationary for $f_{n,c}(b, k-b)$. Moreover the partial derivative is positive if $0 \le b < k/2$ and negative is $k/2 < b \le k$. Hence $b = k/2$ is the only stationary point of $f_{n,c}(b, k-b)$ for $b \in [0,k]$. □

Lemma 21 *If $G \in \mathcal{M}(n, cn)$ then*

$$\sum_{\chi} \Pr[\chi \in \Xi^\sharp \mid \chi \in \Xi] \le \sum_{k=0}^{n} \binom{n}{k} 2^k \left\{1 - \exp\left[-\frac{c}{2-(3k/2n)}\right] + O(1/n)\right\}^{2(n-k)}.$$

Proof. To prove the Lemma an upper bound on $\Pr[\chi \in \Xi^\sharp \mid \chi \in \Xi]$ is defined using Theorem 38. Let G be a graph with b vertices coloured blue by χ, w coloured white and, if $k = b + w$, $r = n - k$ vertices coloured red. The set of red vertices is arbitrarily ordered: let v_j, for $j = 1, \ldots, r$, be the jth vertex in this ordering. Let $I = \{1, \ldots, 2r\}$. Let $\mathcal{V}$ be the set $\{1, \ldots, cn\}$. For each $\nu \in \mathcal{V}$ and $i \in \{1, \ldots, r\}$ (resp. $i \in \{r+1, \ldots, 2r\}$) let $X_\nu = i \in I$ if the νth edge in G is one of the edges that makes the event $E_{v_i}^1$ (resp. $E_{v_{i-r}}^2$) true. If $\mathcal{F}_i$ is, for all $i \in I$, the increasing collection of all non-empty subsets of $\mathcal{V}$ then by Theorem 38

$$\Pr\left[\bigcap_{i\in I}\{S_i \in \mathcal{F}_i\}\right] \le \prod_{i\in I} \Pr\left[S_i \in \mathcal{F}_i\right]$$

Notice that

$$\Pr\left[\bigcap_{v\in R}(E_v^1 \cap E_v^2)\right] = \Pr\left[\bigcap_{i\in I}\{S_i \in (\mathcal{F}_i \setminus \emptyset)\}\right]$$

Therefore

$$\begin{aligned}\Pr\left[\bigcap_{v\in R}(E_v^1 \cap E_v^2)\right] &= \Pr\left[\bigcap_{i\in I}\{S_i \in (\mathcal{F}_i \setminus \emptyset)\}\right] \\ &\le \Pr\left[\bigcap_{i\in I}\{S_i \in \mathcal{F}_i\}\right] \\ &\le \prod_{i\in I}\Pr\left[S_i \in \mathcal{F}_i\right] \\ &\le \prod_{v\in R}\Pr[E_v^1]\Pr[E_v^2]\end{aligned}$$

where the first inequality holds because the event on the right-hand side is bigger (see Theorem 1.2) and the last one is true because the final term contains fewer factors than the previous one.

By Lemma 20 and the argument above, the product of the probability that a particular vertex coloured red is adjacent to a blue and the probability that it is adjacent to a white vertex is at most

$$\left\{1 - \left[1 - \frac{1}{2n-(3k/2)}\right]^{cn}\right\}^2.$$

Thus

$$\Pr[\chi \in \Xi^\sharp \mid \chi \in \Xi] \le \left\{1 - \left[1 - \frac{1}{2n-(3k/2)}\right]^{cn}\right\}^{2(n-k)}$$

By Lemma 19,

$$\left[1 - \frac{1}{2n-(3k/2)}\right]^{cn} \ge \exp\left[-\frac{cn}{2n-(3k/2)}\right] - O(1/n)$$

hence we have

$$\Pr[\chi \in \Xi^\sharp \mid \chi \in \Xi] \le \left\{1 - \exp\left[-\frac{c}{2-(3k/2n)}\right] + O(1/n)\right\}^{2(n-k)}$$

The result follows since there are $\binom{n}{k}2^k$ ways of allocating the sets B and W so that $r = n-k$. $\square$

This last result implies the existence of a function $h(n,k,c)$ such that $\mathrm{E}(X^\sharp)$ is bounded above by $\sum_{k=0}^{n} \mathrm{e}^{nh(n,k,c)}$. More specifically by simple algebraic manipulations of the upper bound on $\mathrm{E}(X^\sharp)$ it is possible to define

$$h(n,k,c) =_{df} c\log\frac{2}{3} + \frac{1}{n}\log\binom{n}{k} + \frac{k}{n}\log 2 + 2\left(1-\frac{k}{n}\right)\log\left[1-\exp\left(-\frac{c}{2-3k/2n}\right)\right]$$

Lemma 22 *For all $c > 0$ there exist $\alpha, \epsilon \in \mathbb{R}^+$ with $0 < \alpha + \epsilon < 1$ and*

$$h(n,k+1,c) - h(n,k,c) > 0 \quad 0 \le k \le \alpha n$$

$$h(n,k+1,c) - h(n,k,c) < 0 \quad (\alpha+\epsilon)n \le k \le n$$

Proof. Let $\gamma_k = 1 - \exp\left(-\frac{c}{2-3k/2n}\right)$. We have

$$h(n,k+1,c) - h(n,k,c) = \frac{1}{n}\left[\log\frac{2(n-k)}{k+1} - 2\log\gamma_k\right] + 2\left(1-\frac{k}{n}\right)\log\frac{\gamma_{k+1}}{\gamma_k}$$

Since $\gamma_k < \gamma_{k+1} < 1$ for all k, if we consider $0 \leq k \leq \alpha n$ and $(\alpha+\epsilon)n \leq k \leq n$ then we have

$$h(n,k+1,c) - h(n,k,c) \geq \frac{1}{n}\left[\log\frac{2(1-\alpha)}{\alpha+1/n} - 2\log\gamma_{\alpha n+1}\right] > 0$$

Similarly for $(\alpha+\epsilon)n \leq k \leq n$

$$h(n,k+1,c)-h(n,k,c) \leq \frac{1}{n}\left[\log\frac{2(1-\alpha-\epsilon)}{\alpha+\epsilon+1/n} - 2\log\gamma_{(\alpha+\epsilon)n}\right]+2\left(1-\alpha-\epsilon\right)\log\frac{\gamma_{(\alpha+\epsilon)n+1}}{\gamma_{(\alpha+\epsilon)n}} < 0$$

□

Lemma 23 *For any $\alpha > n/2$ and $\epsilon > 0$ such that $\alpha+\epsilon < 1$, if $\alpha n \leq k \leq (\alpha+\epsilon)n$, then*

$$h(n,k,c) \leq c\log\frac{2}{3}+(\alpha+\epsilon)\log 2+\log\left(\frac{1}{\alpha(\alpha-1)^{\alpha-1}}\right)+2(1-\alpha-\epsilon)\log(1-\exp\left(-\frac{c}{2-3(\alpha+\epsilon)/2}\right))$$

Proof. The result follows using Stirling approximation for $\binom{n}{\alpha n}$ in the definition of $h(n,k,c)$ and simplifying. □

3.1.3 Concluding Remarks

We end this section by noticing that further improvements seem possible. Definition 10 can be "naturally" strengthened as follows.

Definition 11 *A colouring χ is* better *maximal if it is a maximal colouring in which every blue vertex is adjacent to a white one.*

Indeed Achlioptas and Molloy [AM] recently used this definition and proved that almost all graphs with more than $2.522n$ edges are not 3-colourable. Experiments reported in [HW94] suggest that the threshold function for this problem is located around $2.3n$. Moreover the existence of a sharp threshold has been recently proved [AF99]. We believe that by a more careful analysis of the set of 3-colourings of a graphs it may be possible to move this bound below $2.5n$ and thus get closer to real value of the non 3-colourability threshold. We leave this as an open problem.

3.2 Improved Upper Bound on the Unsatisfiability Threshold

The problem of determining the satisfiability of a Boolean formula is certainly one of the most famous in Computer Science. Its importance lies partly in the fact that the Boolean formula language offers a way to encode many combinatorial problems which preserves the key properties of the combinatorial problem and partly in the nice combinatorial properties of Boolean functions.

The aim of this section is to analyse the phase transitional properties of 3-SAT, the variant of the general satisfiability problem in which the input formula is a conjunction of disjunctions of at most three literals. All relevant definitions were given in Section 1.2.1. Let $\mathcal{C}_n$ denote the set of all clauses with exactly three literals of three distinct variables defined from a set of n variables. A *random formula* ϕ is obtained by selecting m clauses $C \in \mathcal{C}_n$ with replacement. Taking a more static view, similar to the one used to define random graph models, a *random formula model* is a probability space associated with $\Phi_{n,m}$, be the set of all formulae on n variables and m (not necessarily all different) clauses.

Definition 12 *For any Boolean formula ϕ a* maximal *satisfying assignment α is a satisfying assignment such that if the variable x occurs in ϕ and $\alpha(x) = 0$ then the assignment α' defined by*

$$\alpha'(y) = \begin{cases} 1 - \alpha(y) & y = x \\ \alpha(y) & \text{otherwise} \end{cases}$$

does not satisfy ϕ.

Let A_ϕ be the set of satisfying assignments of ϕ and $A^\sharp_\phi$ the set of maximal satisfying assignments of ϕ. Let $X^\sharp$ be the r.v. counting the number of maximal satisfying assignments of ϕ. Again not all satisfying assignments are maximal. Also, if ϕ is satisfiable then there must be at least one maximal satisfying assignment. So by counting only maximal satisfying assignments it is possible to get an improved upper bound on the unsatisfiability threshold.

Lemma 24 *If C is selected at random among the set of all clauses on three literals and α is some truth-assignment then* $\Pr[C\{\alpha\} = 1] = \frac{7}{8}$.

Proof. There are $\binom{n}{3}$ ways of choosing three variables to be part of a clause and 2^3 ways of choosing whether the three variables occur in positive or negated form. Hence $|\mathcal{C}_n| = \binom{n}{3}2^3$. For any fixed truth assignment α and any choice of three variables x_i, x_j, x_k there is a unique clause C containing the three variables such that $C\{\alpha\} = 0$. Hence

$$\Pr[C\{\alpha\} = 1] = 1 - \frac{\binom{n}{3}}{\binom{n}{3}2^3} = 1 - \frac{1}{8}$$

□

From the last lemma and the assumption about the random formula model it follows that $\Pr[\phi\{\alpha\} = 1] = (7/8)^m$. By linearity of expectation,

$$\mathrm{E}(X^\sharp) = \left(\frac{7}{8}\right)^m \sum_\alpha \Pr[\alpha \in A^\sharp_\phi | \alpha \in A_\phi]$$

The final bit of work involves obtaining a good upper bound on the probability that a specific α is a maximal satisfying assignment conditioned on the fact that ϕ is satisfied by α. Let $K_c = 1 - \mathrm{e}^{-3c/7}$ for every $c \in \mathbb{R}^+$.

Theorem 39 [KKKS98] *If ϕ is a random formula on n variables and $m = cn$ clauses, and α sets s variables to zero then* $\Pr[\alpha \in A^\sharp_\phi | \alpha \in A_\phi] \leq (K_c + o(1))^s$.

Theorem 40 [KKKS98] *If ϕ is a random formula on n variables and $m = cn$ clauses, the expected value of $X^\sharp$ is at most $(7/8)^{cn}(1 + K_c + o(1))^n$. It follows that the unique positive solution of the equation*

$$(7/8)^c(2 - \mathrm{e}^{-3c/7}) = 1$$

is an upper bound on κ (this solution is less than 4.667).

Using a stronger notion of maximality Kirousis, Kranakis, Krizanc, and Stamatiou managed to improve this bound to $c_0 = 4.601+$.

The main idea presented in this chapter is a different way of estimating the probability that an assignment is maximal (according to Definition 12). The expectation of $X^\sharp$ is $\sum_\alpha \Pr[\alpha \in A^\sharp_\phi]$. The probabilistic model described in the next Section will allow a tighter estimate on $\Pr[\alpha \in A^\sharp_\phi]$ and this in turn will lead to an improvement on Theorem 40 and on the stronger Theorem 34.

3.2.1 The Young Coupon Collector

A nice way to define the probabilistic model we are interested in is to quote one of the oldest references to it:

> I should like to say straightaway that collecting cigarette-cards is *not* my hobby. But recently the manufacturers of the brand of cigarette I smoke started to issue an attractive series of cards and I said to a friend, "I think I shall save these cards until I obtain the complete set of fifty." He replied, "About how many packets do you think you will have to buy before you get the set?" And this raises an interesting problem in probability

which I do not recollect having seen before. (F. G. Maunsell, *Mathematical Gazette*, 1938)

Maunsell's paper also contains exact expressions for coupon(j, s), the probability of having to buy j packets in order to see all the s different cards and the expected number of trials to see all the cards, assuming that the cards are placed in the packets at random.

Theorem 41 $\mathrm{coupon}(j, s) = \sum_{i=0}^{s} \binom{s}{i}(-1)^i \left(1 - \frac{i}{s}\right)^j$

Unfortunately the formula in the last theorem is only useful when s is a small constant, for otherwise its numerical evaluation becomes very slow. Moreover, for large s, the functional properties of coupon(j, s) are not apparent from the expressions above. For this reasons more recent research has focused on finding useful asymptotic expressions for coupon(j, s) and $E(s)$, the expected number of trials before all coupons have shown up. A simple argument proves that $E(s) \sim s \ln s + O(s)$. Moreover using Chebyshev inequality it is fairly simple to prove that the probability that more than $c\, E(s)$ trials are needed is very small. A deeper analysis of coupon(j, s) is needed to prove high probability results for smaller deviations from the mean. The following is proved for example in Motwani and Raghavan's recent book on randomised algorithms [MR95, Sect. 3.6].

Theorem 42 *Let the random variable X denote the number of trials for collecting each of s types of items. Then, for any constant $c \in \mathbb{R}$ and $j = s \ln s + cs$,*

$$\lim_{s\to\infty} \Pr[X > j] = 1 - \mathrm{e}^{-\mathrm{e}^{-c}}$$

In this section we need to study the probabilities associated with what could be called [Chv91] the *young coupon collector*. The following result, essentially proved in [Arf51], gives an asymptotic expression for coupon(j, s) when $j = \Theta(s)$.

Theorem 43 *If $j = \Theta(s)$.*

$$\mathrm{coupon}(j, s) \sim \sqrt{\frac{j}{\sigma s}}(\mathrm{e}^{r_0} - 1)^s \left(\frac{j}{\mathrm{e} s r_0}\right)^j \tag{3.1}$$

where r_0 is the solution of $f(r) = j/s$ with $f(r) = r\mathrm{e}^r/(\mathrm{e}^r - 1)$ and $\sigma =_{df} r_0 f'(r_0)$.

Theorem 43 has the following useful corollary.

Corollary 2 [Chv91] *Let $x = j/s$ with $j = \Theta(s)$. For all $x > 1$ define $g_1(x) =_{df} (\mathrm{e}^{r_0} - 1)\left(\frac{x}{\mathrm{e} r_0}\right)^x$ where r_0 is the solution of $f(r) = x$. Also let $g_1(1) = \mathrm{e}^{-1}$. Then for all sufficiently large integer s and all $x \geq 1$, coupon$(j, s) \sim g_1(x)^s$.*

3.2.2 Application to the Unsatisfiability Threshold

Let α be a maximal satisfying assignment of $\phi(\vec{x})$. For every variable x set to zero by α (called a *critical variable*), there must exist in ϕ a *critical clause* of the following type: $\{\bar{x}, l_1, l_2\} \in \phi$ with $\alpha(l_1) = \alpha(l_2) = 0$. Therefore the random process by which ϕ is built resembles the coupon collector experiment: if α contains s critical variables then there are s types of coupons, each corresponding to (any) critical clause associated with a different critical variable. In this section this analogy will be described formally and exploited to give an improvement to Theorem 40.

It should be remarked that the connection between maximal satisfying assignments (in the sense of Definition 12) and the coupon collection experiment has been noticed before. In [DB97], Dubois and Boufkhad prove upper bounds on the unsatisfiability threshold for k-SAT for all constant k. Their description is in terms of *Stirling numbers of second kind* $S(j,s) = (1/s!)\sum_{i=0}^{s}\binom{s}{i}(-1)^i(s-i)^j$ (see [Rio58]) and the following identity follows immediately from Theorem 41

$$\text{coupon}(j,s) = (s!/s^j)S(j,s)$$

In Section 3.2.3 it will be shown how the use of a stronger notion of maximality and some results in [KKKS98] in conjunction with the probability associated with the coupon collector will lead to an improvement on Theorem 34 which brings the upper bound on the unsatisfiability threshold for 3-SAT down to 4.5793.

Let $\mathcal{C}_n(x,\alpha)$ be the set of critical clauses for variable x under α. For every critical variable x, $|\mathcal{C}_n(x,\alpha)| = \binom{n-1}{2}$ since there are this many ways of selecting the literals l_1 and l_2. Also, for every pair of critical variables x and y, $\mathcal{C}_n(x,\alpha) \cap \mathcal{C}_n(y,\alpha) = \emptyset$. Assuming α sets s variables to zero, the probability $\Pr[\alpha \in A^\sharp_\phi]$ is the ratio between a function $N(n,m,s)$ and the number of ways to build a formula on m clauses out of n variables. Hence

$$\Pr[\alpha \in A^\sharp_\phi] =_{df} \frac{N(n,m,s)}{\left[8\binom{n}{3}\right]^m}$$

The function $N(n,m,s)$ counts the number of ways to build a formula with m clauses out of n variables containing at least one critical clause for each of the s critical variables. If ϕ contains $j \in \{s, s+1, \ldots, m\}$ critical clauses, then

$$\Pr[\alpha \in A^\sharp_\phi] = \sum_{j=s}^{m} \frac{C(n,m,s,j)\ R(n,m,s,j)}{\left[8\binom{n}{3}\right]^m}$$

where $C(n,m,s,j)$ counts the number of ways of choosing j critical clauses so that at least one member of $\mathcal{C}_n(x, \alpha)$ is chosen for each of the s critical variables and $R(n, m, s, j)$ counts the number of ways of filling up the remainder of ϕ with $m - j$ clauses that are true under α but not critical.

Lemma 25 *For any choice of the parameters* $R(n,m,s,j) = \left(7\binom{n}{3} - s\binom{n-1}{2}\right)^{m-j}$.

Proof. By the argument in the proof of Lemma 24 there are $7\binom{n}{3}$ clauses satisfied by α. If α forces s variables to be critical there are s disjoint groups of $\binom{n-1}{2}$ critical clauses. $\square$

Lemma 26 *For any choice of the parameters* $C(n,m,s,j) = \binom{m}{j}\left[s\binom{n-1}{2}\right]^j \operatorname{coupon}(j,s)$.

Proof. Assume that there are s critical variables associated with a given assignment α. Moreover ϕ contains j critical clauses. There are $\binom{m}{j}$ ways of choosing j positions out of the m available. Also, there are $s\binom{n-1}{2}$ critical clauses. Therefore, if we do not distinguish among the non-critical clauses, there are $\binom{m}{j}\left[s\binom{n-1}{2}\right]^j$ ways of choosing a sequence of m clauses so that exactly j of them are critical. Since $C(n, m, s, j)$ counts the number of these which has at least one occurrence of a critical clause for each of the s critical variables, and since there are equal numbers of possible critical clauses for each variable, the ratio of these terms is the probability $\operatorname{coupon}(j, s)$. $\square$

In terms of the coupon collector problem, there are s items, the s disjoint sets of critical clauses, and j selections to be made. Critical clauses with respect to a specific critical variable are indistinguishable.

Theorem 44 *If* α *sets* s *variables to zero then*

$$\Pr[\alpha \in A_\phi^\sharp] = \sum_{j=s}^{m} \binom{m}{j} \left(\frac{3s}{8n}\right)^j \operatorname{coupon}(j,s) \left(\frac{7}{8} - \frac{3s}{8n}\right)^{m-j}$$

Proof. By the argument at the beginning of the section and Lemma 25 and 26

$$\Pr[\alpha \in A_\phi^\sharp] = \sum_{j=s}^{m} \binom{m}{j} \operatorname{coupon}(j,s) \left(\frac{s\binom{n-1}{2}}{8\binom{n}{3}}\right)^j \left(\frac{7\binom{n}{3} - s\binom{n-1}{2}}{8\binom{n}{3}}\right)^{m-j}$$

and the result follows by straightforward algebraic simplifications. $\square$

It is convenient to split the analysis of $\mathrm{E}(X^\sharp)$ in two main parts.

$$\begin{aligned} \mathrm{E}(X^\sharp) \quad \leq \quad & \left(\frac{7}{8}\right)^m \sum_{s=0}^{\beta n} \binom{n}{s} \Pr[\alpha \in A_\phi^\sharp | \alpha \in A_\phi] + \\ & \sum_{s=\beta n}^{n} \binom{n}{s} \sum_{j=s}^{m} \binom{m}{j} \operatorname{coupon}(j,s) \left(\frac{3s}{8n}\right)^j \left(\frac{7}{8} - \frac{3s}{8n}\right)^{m-j} \end{aligned} \tag{3.2}$$

The first sum is dealt with using Theorem 39. For a sufficiently small $\beta < 1$ the expected number of maximal satisfying assignments with at most βn critical variables is rather small. The second part of $\mathrm{E}(X^\sharp)$ will be bounded using the result on the young coupon collector probability in Theorem 43.

Lemma 27 *$\binom{n}{s} \leq \left(\frac{n\mathrm{e}}{s}\right)^s$ for all $n \in \mathbb{N}$ and $s \in \{0, \ldots, n\}$.*

Proof. For all $n \in \mathbb{N}$ and $s \in \{0, \ldots, n\}$

$$\binom{n}{s} = \frac{n!}{s!\,(n-s)!} \leq \frac{n^s}{s!}$$

The result is then proved by showing that $(s/\mathrm{e})^s$ is a lower bound for $s!$ for all $s \in \mathbb{N}$. By induction on s, clearly $1^1 > 1/\mathrm{e}$. Using the inductive hypothesis

$$(s+1)! \geq (s+1)\left(\frac{s}{\mathrm{e}}\right)^s$$

Multiplying and dividing by $\mathrm{e}(s+1)^{s+1}$,

$$\begin{aligned} (s+1)! &\geq \left(\frac{s+1}{\mathrm{e}}\right)^{s+1} \frac{\mathrm{e}s^s(s+1)}{(s+1)^{s+1}} \\ &= \left(\frac{s+1}{\mathrm{e}}\right)^{s+1} \frac{\mathrm{e}}{(1+\frac{1}{s})^s} > \left(\frac{s+1}{\mathrm{e}}\right)^{s+1} \end{aligned}$$

where the last inequality follows from $1 + x \leq \mathrm{e}^x$. □

Theorem 45 *For n sufficiently large there exists an $\epsilon > 0$ such that for every $c \in (0, c_0)$ there exists a positive real number $\beta_{\epsilon,c} < (K_c + \epsilon)/(1 + K_c + \epsilon)$ such that if ϕ is a random formula on n variables and $m = cn$ clauses then for all $\beta < \beta_{\epsilon,c}$*

$$\left(\frac{7}{8}\right)^{cn} \sum_{s=0}^{\beta n} \binom{n}{s} \Pr[\alpha \in A_\phi^\sharp | \alpha \in A_\phi] = o(1)$$

Proof. Let $\iota : \mathbb{N} \to \mathbb{N}$ be an arbitrary function on natural numbers with $\lim_{n\to\infty} \iota(n) = +\infty$ and $\iota(n) = o(n/\ln n)$. First notice that, by Lemma 27,

$$\sum_{s=0}^{\iota(n)} \binom{n}{s} (K_c + o(1))^s \leq \sum_{s=0}^{\iota(n)} \left[\frac{n\mathrm{e}(K_c + o(1)}{s}\right]^s$$

Also

$$\left[\frac{n\mathrm{e}(K_c + o(1))}{s}\right]^s \leq \left[\frac{n\mathrm{e}(K_c + o(1))}{s+1}\right]^{s+1}$$

Therefore

$$\sum_{s=0}^{\iota(n)} \binom{n}{s} (K_c + o(1))^s \leq (1 + \iota(n)) \left[\frac{n\mathrm{e}(K_c + o(1))}{\iota(n)}\right]^{\iota(n)} = o([(7/8)^{c_0}(1 + K_{c_0} + o(1))]^n)$$

If $\iota(n) \leq s \leq \beta n$ then

$$\begin{aligned}\binom{n}{s} &\sim \sqrt{\frac{n}{2\pi s(n-s)}}\left(\frac{n}{s}\right)^s\left(\frac{n}{n-s}\right)^{n-s}\\ &\leq O\left(1/\sqrt{\iota(n)}\right)\left(\frac{n}{s}\right)^s\left(\frac{n}{n-s}\right)^{n-s}\end{aligned}$$

Claim 1 *The function $[n(K_c + o(1))/s]^s[n/(n-s)]^{n-s}$ is increasing in s for all $s \leq (K_c + o(1))n/(1+K_c+o(1))$.*

Notice that

$$\left[\frac{n(K_c+o(1))}{s}\right]^s\left(\frac{n}{n-s}\right)^{n-s} \leq \left[\frac{n(K_c+o(1))}{s+1}\right]^{s+1}\left(\frac{n}{n-s-1}\right)^{n-s-1}$$

if and only if

$$\frac{s+1}{(K_c+o(1))[n-(s+1)]}\left(1+\frac{1}{s}\right)^s\left(1-\frac{1}{n-s}\right)^{n-s} \leq 1$$

which in turn, since $1+x \leq \mathrm{e}^x$ for all $x \in \mathbb{R}$, is satisfied if

$$\frac{s+1}{(K_c+o(1))[n-(s+1)]} \leq 1$$

The last inequality is true if and only if

$$s \leq n\left[\frac{K_c+o(1)}{1+K_c+o(1)}\right]-1$$

Hence, upper bounding the sum by its largest term,

$$\begin{aligned}\sum_{s=\iota(n)}^{\beta n}\binom{n}{s}(K_c+o(1))^s &\leq O\left(n/\sqrt{\iota(n)}\right)\left[\frac{n(K_c+o(1))}{\beta n}\right]^{\beta n}\left(\frac{n}{n-\beta n}\right)^{n-\beta n}\\ &= O\left(n/\sqrt{\iota(n)}\right)\left\{\left[\frac{K_c+o(1)}{\beta}\right]^{\beta}\left(\frac{1}{1-\beta}\right)^{1-\beta}\right\}^n\end{aligned}$$

For n sufficiently large the term $o(1)$ in the last expression is upper bounded by some $\epsilon > 0$. The remaining part of the proof is aimed at showing that for every positive constants ϵ and c there exists a value $\beta_{\epsilon,c}$ such that

$$\left(\frac{7}{8}\right)^c\left[\frac{K_c+\epsilon}{\beta}\right]^{\beta}\left(\frac{1}{1-\beta}\right)^{1-\beta} < 1 \tag{3.3}$$

for all $\beta < \beta_{\epsilon,c}$. Some simple calculations show that, for every fixed positive ϵ and c, the function $h_{\epsilon,c}(\beta) =_{df} [(K_c+\epsilon)/\beta]^{\beta}(1-\beta)^{\beta-1}$ is continuous and positive for $\beta \in (0,1)$. Also

$$\lim_{\beta\to 0} h_{\epsilon,c}(\beta) = 1 \qquad \lim_{\beta\to 1} h_c(\beta) = K_c+\epsilon$$

Moreover $h_{\epsilon,c}(\beta)$ has a unique maximum at $\beta^* = (K_c+\epsilon)/(1+K_c+\epsilon)$ and $h_{\epsilon,c}(\beta^*) = 1+K_c+\epsilon$. Since $c < c_0$, by Theorem 40 $(7/8)^c(1+K_c+\epsilon) > 1$. Hence $1 < (8/7)^c < 1+K_c+\epsilon$ and therefore there exists a unique $\beta_{\epsilon,c} < (K_c+\epsilon)/(1+K_c+\epsilon)$ solution of the equation $(7/8)^c h_{\epsilon,c}(\beta) = 1$. For all $\beta < \beta_{\epsilon,c}$ inequality (3.3) holds and the theorem follows. □

Using the theory of implicit functions it is possible to prove that the function $c_1(\beta)$ defined implicitly by the equation $(7/8)^c h_{\epsilon,c}(\beta) = 1$ is increasing in β so that smaller values of β result in smaller values for c. This property will be used in Section 3.2.3.

If s is larger that βn then the asymptotic for the coupon collector given by Theorem 43 becomes useful. In order to successfully bound

$$\sum_{s=\beta n}^{n}\sum_{j=s}^{m}\binom{n}{s}\binom{m}{j}\text{coupon}(j,s)\left(\frac{3s}{8n}\right)^j\left(\frac{7}{8}-\frac{3s}{8n}\right)^{m-j}$$

the following approximations are used:

$$\begin{array}{rcll}
\binom{n}{s} & \le & \left(\frac{n}{s}\right)^s\left(\frac{n}{n-s}\right)^{n-s} & \text{if } \beta n \le s \le \gamma n \\
\binom{n}{s} & \le & \left(\frac{ne}{n-s}\right)^{n-s} & \text{if } \gamma n < s \le n \\
\binom{cn}{j} & \le & \left(\frac{cn}{j}\right)^s\left(\frac{cn}{cn-j}\right)^{cn-j} & \text{if } s \le j \le \gamma cn \\
\binom{cn}{j} & \le & \left(\frac{cne}{cn-j}\right)^{cn-j} & \text{if } \gamma cn < j \le cn
\end{array}$$

Then, setting $y = s/n$ and $x = j/s$, the nested summation above is upper bounded by

$$\begin{aligned}
n^{O(1)}\Bigg\{&\int_{\beta}^{\gamma}\int_{1}^{\frac{\gamma c}{y}} f_1(x,y)^n dx\,dy + \int_{\gamma}^{1}\int_{1}^{\frac{\gamma c}{y}} f_2(x,y)^n dx\,dy + \\
+ &\int_{\beta}^{\gamma}\int_{\frac{\gamma c}{y}}^{\frac{c}{y}} f_3(x,y)^n dx\,dy + \int_{\gamma}^{1}\int_{\frac{\gamma c}{y}}^{\frac{c}{y}} f_4(x,y)^n dx\,dy\Bigg\}
\end{aligned}$$

where

$$\begin{array}{rcl}
f_1(x,y) & = & \left(\frac{1}{y}\right)^y\left(\frac{1}{1-y}\right)^{1-y}\left[\frac{c(7-3y)}{8(c-xy)}\right]^{c-xy}\left(\frac{3c}{8x}\right)^{xy} g_1(x)^y \\
f_2(x,y) & = & \left(\frac{e}{1-y}\right)^{1-y}\left[\frac{c(7-3y)}{8(c-xy)}\right]^{c-xy}\left(\frac{3c}{8x}\right)^{xy} \\
f_3(x,y) & = & \left(\frac{1}{y}\right)^y\left(\frac{1}{1-y}\right)^{1-y}\left[\frac{ce(7-3y)}{8(c-xy)}\right]^{c-xy}\left(\frac{3y}{8}\right)^{xy} \\
f_4(x,y) & = & \left(\frac{e}{1-y}\right)^{1-y}\left[\frac{ce(7-3y)}{8(c-xy)}\right]^{c-xy}\left(\frac{3y}{8}\right)^{xy}
\end{array}$$

Figure 3.3 gives a graphical description of the partition induced on the space of values for x and y by the previous definitions. The four integrals above are estimated quite simply by using some uniform upper bounds on each of the f_i's and multiplying them by the area of the region on the plane in which f_i is defined. Also, in order to prove the main result of this section, each of the f_i's is proved to have an upper bound smaller than one for c sufficiently large and some choice of γ. Functions $f_3(x,y)$ and $f_4(x,y)$ only depend on y and xy. They can be upper bounded rather easily.

Theorem 46 *For every $c > 0$ there exists a $\gamma \ge 1/2$ such that $f_3(x,y) < 1$ for all $y \in [\beta,\gamma]$ and x such that $(\gamma c)/y \le x \le c/y$.*

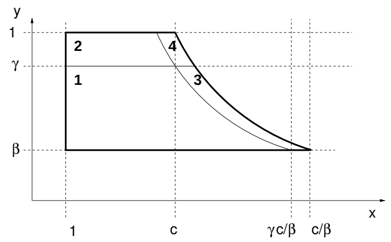

Figure 3.3: Partition of the parameter space used to upper bound $\mathrm{E}(X^\sharp)$.

Proof. The following chain of inequalities holds for all $y \in (0,1]$:

$$\begin{aligned}\left(\frac{1}{y}\right)^y\left(\frac{1}{1-y}\right)^{1-y} &= \frac{1}{y}\left(\frac{1}{y}\right)^{y-1}\left(\frac{1}{1-y}\right)^{1-y}\\ &= \frac{1}{y}\left(\frac{y}{1-y}\right)^{1-y}\\ &= \frac{1}{y}\left(1-\frac{1-2y}{1-y}\right)^{1-y} \leq \frac{\mathrm{e}^{\,2y-1}}{y}\end{aligned}$$

Also, $(\mathrm{e}^{2y-1})/y$ is maximised either at $y=\beta$ or at $y=\gamma$. It follows that there exists a positive real constant β_0 such that if $\beta > \beta_0$ then the maximum of this function is $(\mathrm{e}^{2\gamma-1})/\gamma$. Using this fact and rearranging the expression for f_3

$$f_3(x,y) \leq \frac{\mathrm{e}^{\,2\gamma-1}}{\gamma}\left(\frac{3y}{c\mathrm{e}}\right)^{xy}\left(\frac{c\mathrm{e}}{8}\right)^{c}\left(\frac{7-3y}{c-xy}\right)^{c-xy}$$

$3y/8$ is smaller than one and increasing for all y in the given range; $xy \geq \gamma c$. Hence

$$\left(\frac{3y}{c\mathrm{e}}\right)^{xy} \leq \left(\frac{3\gamma}{c\mathrm{e}}\right)^{\gamma c}$$

In the same way $7-3y$ is always larger than one and thus

$$(7-3y)^{c-xy} \leq 7^{c(1-\gamma)}$$

Finally, setting $1-t = c-xy$,

$$\left(\frac{1}{c-xy}\right)^{c-xy} = \left(\frac{1}{1-t}\right)^{1-t}$$

and the latter is maximised at $t = 1-1/\mathrm{e}$. Hence

$$f_3(x,y) \leq \frac{\mathrm{e}^{\,2\gamma-1+\frac{1}{\mathrm{e}}}}{\gamma}\left[\frac{(3\gamma)^\gamma(7c\mathrm{e})^{1-\gamma}}{8}\right]^c$$

The function $\frac{\mathrm{e}}{\gamma}^{2\gamma-1+\frac{1}{\mathrm{e}}}$ is increasing in $(1/2,1]$ so a weaker upper bound on $f_3(x,y)$ is

$$\mathrm{e}^{1+\frac{1}{\mathrm{e}}}\left[\frac{(3\gamma)^\gamma(7c\mathrm{e})^{1-\gamma}}{8}\right]^c$$

For every given $c > 0$, the expression inside the square brackets is decreasing for every $\gamma \in (0, 7c/3]$. Hence the upper bound on $f_3(x, y)$ is less than one for all $\gamma \in (\gamma_c, 1]$ where γ_c satisfies $[(3\gamma)^{\gamma}(7c\mathrm{e})^{1-\gamma}/8]^c = \mathrm{e}^{-1-1/\mathrm{e}}$. $\square$

Theorem 47 *For every $c > 0$ there exists a $\gamma \geq 1/2$ such that $f_4(x, y) < 1$ for all for all $y \in (\gamma, 1]$ and x such that $(\gamma c)/y \leq x \leq c/y$.*

Proof. By rearrangements similar to those in Theorem 46

$$
\begin{aligned}
f_4(x, y) &\leq \left(\frac{\mathrm{e}}{1-y}\right)^{1-y} \left(\frac{7-3y}{c-xy}\right)^{c-xy} \left(\frac{c\mathrm{e}}{8}\right)^{c} \left(\frac{3}{c\mathrm{e}}\right)^{\gamma c} \\
&\leq \left(\frac{\mathrm{e}}{1-y}\right)^{1-y} \mathrm{e}^{\frac{1}{\mathrm{e}}}(7-3\gamma)^{(1-\gamma)c} \left(\frac{c\mathrm{e}}{8}\right)^{c} \left(\frac{3}{c\mathrm{e}}\right)^{\gamma c} \\
&\leq \mathrm{e}^{\frac{1}{\mathrm{e}}} \left(\frac{\mathrm{e}}{1-\gamma}\right)^{1-\gamma} \left[\frac{3^{\gamma}(7c\mathrm{e})^{1-\gamma}}{8}\right]^{c}
\end{aligned}
$$

and the result follows. $\square$

A little more effort is needed to get uniform upper bounds on $f_2(x, y)$ and $f_1(x, y)$. The following are standard results contained in many introductory analysis texts [Apo57, Giu83]. In what follows if I is an interval in $\mathbb{R}^d$ then for each $i \in \mathbb{N}$ the class $C^i(I)$ contains all functions that are continuous and differentiable i times in I.

Definition 13 *Let $I \subset \mathbb{R}$ be an interval. A function $f : I \to \mathbb{R}$ is* convex *if for all $x_1, x_2 \in I$ and for all $\lambda \in (0, 1)$, $f(\lambda x_1 + (1-\lambda)x_2) \leq \lambda f(x_1) + (1-\lambda) f(x_2)$. A function $f : I \to \mathbb{R}$ is* concave *if $-f$ is convex.*

Theorem 48 [Giu83] *Let $f : I \to \mathbb{R}$ with $f \in C^2(I)$. Then $f(x)$ is concave if and only if $\frac{d^2 f(x)}{dx^2} \leq 0$ for all $x \in I$.*

The following Lemma will be needed in the proof of Theorem 49

Lemma 28 *Let $f : I \times J \to \mathbb{R}$ with $f \in C^2(I \times J)$. Also assume that*

1. *The gradient of f is zero at (a, y) for some $a \in I$ and for all $y \in J$.*
2. *$\frac{\partial^2 f(x,y)}{\partial x^2} \leq 0$ for all $y \in J$.*

Then $\max_{I \times J} f(x, y) = \max_J f(a, y)$.

Proof. By contradiction assume there exists a point $(\bar{x}, \bar{y})$ such that $f(\bar{x}, \bar{y}) > f(a, y)$. Since $\frac{\partial^2 f(x,y)}{\partial x^2} \leq 0$ for all $y \in J$, by Theorem 48 f is concave in y over all J. By definition this implies $f(\bar{x}, \bar{y}) < f(a, \bar{y})$. $\square$

Theorem 49 *For every $c > 0$ there exists a γ such that $f_2(x,y) < 1$ for all $y \in (\gamma, 1]$ and x such that $1 \le x \le (\gamma c)/y$.*

Proof. For all x and y in the given range

$$f_2(x,y) \le \left(\frac{\mathrm{e}}{1-\gamma}\right)^{1-\gamma} \left[\frac{c(7-3y)}{8(c-xy)}\right]^{c-xy} \left(\frac{3c}{8x}\right)^{xy} = \bar{f}_2(x,y)$$

The function $\ln \bar{f}_2(x,y)$ is defined and continuous in the given domain. Also,

$$\frac{\partial}{\partial x} \ln \bar{f}_2(x,y) = y \ln \frac{3(c-xy)}{x(7-3y)}$$

and the partial derivative is zero if and only if $(x,y) \equiv (3c/7, y)$ for all $y \in (\gamma, 1]$. The partial derivative with respect to y is

$$\frac{\partial}{\partial y} \ln \bar{f}_2(x,y) = x \ln \frac{3(c-xy)}{x(7-3y)} - \frac{3c-7x}{7-3y}$$

and it is also null at $(3c/7, y)$. The proof is completed by showing that the points $(3c/7, y)$ are local maxima and that $\frac{\partial}{\partial y} \ln \bar{f}_2(x,y) < 0$ everywhere else in the given domain. Both results are consequences of Lemma 28. □

Finally, the following result describes the main features of f_1. Notice that f_1 is the only function involving the coupon collector's asymptotic.

Theorem 50 *If $c > 4.642$ then $f_1(x,y) < 1$ for all $y \in [\beta, \gamma]$ and x such that $1 \le x \le (\gamma c)/y$.*

Proof. Since f_1 is defined and continuous in the given domain, the critical points can be found by looking at

$$\ln f_1(x,y) = y \ln \frac{1}{y} + (1-y) \ln \frac{1}{1-y} + (c-xy) \ln \frac{c(7-3y)}{8(c-xy)} + xy \ln \frac{3c}{8x} + y \ln g_1(x)$$

By definition of r,

$$\begin{aligned}
\frac{d}{dx} \ln g_1(x) &= \frac{d}{dx} \left\{ \ln(e^r - 1) + x \left(\ln \frac{x}{r} - 1 \right) \right\} \\
&= \frac{\mathrm{e}^r}{\mathrm{e}^r - 1} \frac{dr}{dx} - \frac{x}{r} \frac{dr}{dx} + \ln \frac{x}{r} \\
&= \left(\frac{\mathrm{e}^r}{\mathrm{e}^r - 1} - \frac{x}{r} \right) \frac{dr}{dx} + \ln \frac{x}{r} = \ln \frac{x}{r}
\end{aligned}$$

Hence a straightforward computation shows that

$$G_y(x) = \frac{\partial}{\partial x} \ln f_1(x,y) = y \ln \frac{3(c-xy)}{r(7-3y)}$$

This function, for every fixed y, is defined and continuous. For $x = 1$ since $r < x$, it follows that

$$\frac{3(c - xy)}{r(7 - 3y)} \geq \frac{3(c - xy)}{x(7 - 3y)} = \frac{3c - 3y}{7 - 3y} > 1$$

where the last inequality holds for $c > 7/3$. Hence $G_y(1) > 0$. For $xy = \gamma c$,

$$\frac{3(c - xy)}{r(7 - 3y)} = \frac{3c(1 - \gamma)}{r(7 - 3y)} < \frac{3c(1 - \gamma)}{7 - 3y}$$

The last expression is strictly less than one if

$$c < \frac{7 - 3\gamma}{3(1 - \gamma)}$$

and the upper bound on c is at least 5 for $\gamma \geq 2/3$. Hence $G_y(\gamma c/y) < 0$. So there must be a value $x^* = x^*(y)$ such that

$$\frac{3(c - x^* y)}{x^*(7 - 3y)} = \frac{r}{x^*} \tag{3.4}$$

and hence $G_y(x^*) = 0$. Indeed $G_y(x)$ is actually strictly decreasing for all x in the given domain (and therefore there is a unique x^*). To see this, notice that, by setting $t = xy$ the function $h(t, y) = \frac{3(c-t)}{7-3y}$ is a decreasing function of t. Hence, for each fixed y, $h(xy, y)$ is decreasing in x. Moreover, by the implicit function theorem,

$$\frac{dr}{dx} = \frac{(\mathrm{e}^r - 1)^2}{\mathrm{e}^r(\mathrm{e}^r - 1 - r)} > 0$$

This implies that r is an increasing function of x and in turn this means that $1/r$ is decreasing.

The maximum of $f_1(x, y)$ is to be found among the critical points of

$$F(y) =_{df} \ln f_1(x^*(y), y)$$

Notice that

$$\mathrm{d}F(y) =_{df} G_y(x^*(y))\mathrm{d}x + \frac{\partial}{\partial y} \ln f_1(x, y)\mathrm{d}y$$

and if $x = x^*$ the first term vanishes. Hence $\frac{d}{dy}F(y) = 0$ if and only if

$$\frac{\partial}{\partial y} \ln f_1(x^*, y) = \ln \frac{1 - y}{y} + x^* \ln \frac{3(c - x^* y)}{x^*(7 - 3y)} - \frac{3c - 7x^*}{7 - 3y} + \ln g_1(x^*) = 0 \tag{3.5}$$

Since

$$\frac{3c - 7x}{7 - 3y} = \frac{3c - 3xy + 3xy - 7x}{7 - 3y} = \frac{3(c - xy)}{7 - 3y} - x$$

using (3.4) and the definition of $g_1(x)$ equation (3.5) simplifies to

$$\ln \frac{(1 - y)(\mathrm{e}^r - 1)}{y} - r = 0$$

which is satisfied for

$$y^* = y^*(r) = \frac{e^r - 1}{2e^r - 1}$$

It is also possible to express x^* as a function of r using the definition of $g_1(x)$,

$$x^* = x^*(r) = \frac{re^r}{e^r - 1}$$

Finally, again using (3.4) and the last two definitions,

$$c(r) = \frac{r}{3}(7 - 3y) + xy = \frac{2r(7e^r - 2)}{3(2e^r - 1)}$$

The maximum value of $f_1(x, y)$ in the given range is $f_1(x^*(r), y^*(r))$. By simple calculus it

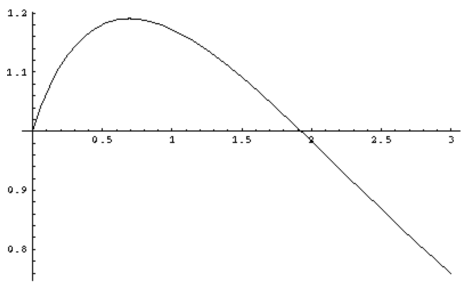

Figure 3.4: Graph of $f_1(x^*(r), y^*(r))$.

is possible to prove that this function is strictly decreasing for $r \geq 1$, $f_1(x^*(1), y^*(1)) > 1$ and $f_1(x^*(3), y^*(3)) < 1$, so that there exists a unique value $r_0 \simeq 1.924$ (numerically found using `Mathematica`'s built-in Newton approximation method) such that $f_1(x^*(r), y^*(r)) < 1$ for all $r > r_0$. $c(r_0) = 4.64248$. □

3.2.3 Refined Analysis

The method described in Section 3.2.2, based on the concept of maximal satisfying assignment in Definition 12 can be further generalised. Smaller and smaller classes of satisfying assignments are captured by the following definition.

Definition 14 *For any boolean formula ϕ an l-*maximal *satisfying assignment α is a satisfying assignment such that any lexicographically larger assignment obtained by changing at most l bits of α does not satisfy ϕ.*

Let A^l_ϕ be the set of l-maximal satisfying assignments of ϕ (so $A^1_\phi \equiv A^\sharp_\phi$ as introduced in Section 3.2) and $X^l =_{df} |A^l_\phi|$. In [KKKS98] a set $A^{2\sharp}_\phi$ (also $X^{2\sharp} =_{df} |A^{2\sharp}_\phi|$) is defined with $A^2_\phi \subseteq A^{2\sharp}_\phi \subseteq A^1_\phi$. The following results are proved in that paper

Theorem 51 *If ϕ is a random formula on n variables and m clauses then*

1. $\mathrm{E}(X^{2\sharp}) = (7/8)^m \sum_\alpha \Pr[\alpha \in A_\phi^{2\sharp} | \alpha \in A_\phi^\sharp] \Pr[\alpha \in A_\phi^\sharp | \alpha \in A_\phi]$.

2. *Let* $u = \mathrm{e}^{-c/7}$. *There exists a function* $d : \{0,1\}^n \to \mathbb{N}$ *such that* $\Pr[\alpha \in A_\phi^{2\sharp} | \alpha \in A_\phi^\sharp] \leq 3m^{1/2} Y^{d(\alpha)}$ *with*

$$Y = 1 - \frac{6u^6 \ln(1/u)}{(1-u^3)n} + \\ + \frac{18u^9 \ln(1/u)}{(1-u^3)^2 n} \psi\left(\frac{6u^6 \ln(1/u)}{1-u^3} + o(1)\right) + o(1/n)$$

where $\psi(x)$ *is the smallest root of* $\psi = \mathrm{e}^{x\psi}$ *for any given* $x \in [0, 1/\mathrm{e}]$.

3. *If* $0 \leq (K_c)^2 \leq Y \leq 1$ *then* $\mathrm{E}(X^{2\sharp}) \leq 3m^{1/2}(7/8)^m \prod_{i=0}^{n-1}(1 + K_c Y^{i/2})$.

4. *Let* $Z = n \ln Y$ *and* $df\_eq(c) =_{df} c(Z/2)\ln(7/8) + \mathrm{dilog}(1+K_c) - \mathrm{dilog}(1 + K_c \mathrm{e}^{Z/2})$ *where* $\mathrm{dilog}(x) = -\int_1^x \frac{\ln t}{1-t} dt$. *If* $df\_eq(c) = 0$ *then* $\lim_{n\to\infty} \mathrm{E}(X^{2\sharp}) = 0$.

The constant obtained by forcing $df\_eq(c) = 0$ is exactly $c_0 = 4.60108\ldots$. Notice that Theorem 51.*3* holds for any number K and Y satisfying the given conditions. One simple way to improve the value of the constant c_0 is to replace K_c by some $F_c < K_c$ and then solve the equation $df\_eq(c) = 0$ using F_c instead of K_c.

As in the analysis in Section 3.2.2, $\mathrm{E}(X^{2\sharp})$ contains a noise component and a main component. The following inequality is implied by the definitions:

$$\mathrm{E}(X^{2\sharp}) \leq \sum_{s=0}^{\beta n} \binom{n}{s} \Pr[\alpha \in A_\phi^\sharp] \\ + \sum_{s=\beta n}^{\gamma n} \binom{n}{s} \Pr[\alpha \in A_\phi^\sharp] Y^{d(\alpha)} + \sum_{s=\gamma n}^{n} \binom{n}{s} \Pr[\alpha \in A_\phi^\sharp]$$

By Theorem 45 the first summation is very small provided a sufficiently small β is used. The bounds given in Theorem 46 and Theorem 47 imply that the last sum in the expression above is very small, provided a sufficiently large value for γ is used. Hence a direct argument is needed only for the central sum. If ϕ has at least some $\gamma c n$ clauses then $\Pr[\alpha \in A_\phi^\sharp]$ is small by Theorem 49. Let

$$f_0(x,y) =_{df} \left(\frac{3c}{7x}\right)^x \left[\frac{c(7-3y)}{7(c-xy)}\right]^{\frac{c}{y}-x} g_1(x)$$

If there is a unique $F_c = \max\{f_0(x,y) : \beta \leq y \leq \gamma, 1 \leq x \leq \gamma c/y\}$ By Lemma 24 and Theorem 44

$$\sum_{s=\beta n}^{\gamma n} \binom{n}{s} \Pr[\alpha \in A_\phi^\sharp] Y^{d(\alpha)} \leq \left(\frac{7}{8}\right)^{cn} \sum_\alpha (F_c)^s Y^{d(\alpha)}$$

Theorem 52 *Function $f_0(x, y)$ has a unique absolute maximum for $\beta \le y \le \gamma$ and $1 \le x \le \gamma c/y$.*

Proof. The partial derivatives of $\ln f_0(x, y)$ are

$$\begin{aligned} \tfrac{\partial}{\partial x} \ln f_0(x, y) &= \ln \tfrac{3(c-xy)}{r(7-3y)} \\ \tfrac{\partial}{\partial y} \ln f_0(x, y) &= \tfrac{c}{y^2} \ln \tfrac{7(c-xy)}{c(7-3y)} - \tfrac{3c-7x}{y(7-3y)} \end{aligned}$$

where r is the solution of $f(r) = x$ as in Theorem 43. Simple analysis as in Theorem 50 implies that, for every fixed y, there is only one point x^* such that $\frac{\partial}{\partial x} \ln f_0(x^*, y) = 0$. For any given y, $x^* = x^*(y, r)$ is the solution to the equation

$$\frac{c - xy}{7 - 3y} = \frac{r}{3} \tag{3.6}$$

As in Theorem 50 let $F(y) = \ln f_0(x^*, y)$. Again the derivative of $F(y)$ is exactly the partial derivative of $f_0(x, y)$ with respect to y evaluated at $x = x^*$. Since

$$\frac{3c - 7x}{y(7 - 3y)} = \frac{3cy - 7c + 7c - 7xy}{y^2(7 - 3y)} = \frac{7(c - xy)}{y^2(7 - 3y)} - \frac{c}{y^2}$$

the condition $\frac{d}{dy} F(y) = 0$ is equivalent to

$$\frac{c}{y^2} \left(\ln \frac{7r}{3c} + 1 - \frac{7r}{3c} \right) = 0$$

Since $\ln z \le z - 1$ for all $z > 0$ and equality holds at $z = 1$, the expression on the left is always negative and only becomes zero when $r = 3c/7$. But substituting this value in the equation $f(r) = x$ and in (3.6) the value for y is not in the given range. This implies that $F(y)$ is strictly decreasing and hence the (unique) maximum of $f_0(x, y)$ for $\beta \le y \le \gamma$ is $f_0(x^*, \beta)$. $\square$

The improved value of c is obtained by using F_c instead of K_c in *df_eq*(c) and solving *df_eq*$(c) = 0$. Notice that the implicit value of c obtained solving *df_eq*$(c) = 0$ is a function of β,

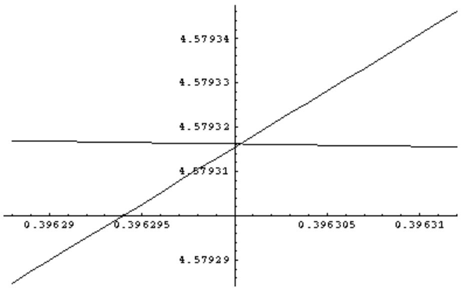

Figure 3.5: Locating the best value of c.

$c_2(\beta)$. It might be possible to prove (experimentally this looks true) that $c = c(\beta)$ is a decreasing function of β. Essentially smaller β make F_c larger (this follows from last Theorem) and the larger

F_c the larger the value of c needed to make the all expression small. The value $c^* = 4.5793$ is obtained by solving numerically $c_1(\beta) = c_2(\beta)$ (for $\beta = 0.3963..$).

3.3 Conclusions

In this chapter we studied some structural properties related to the problems of deciding whether a graph is colourable using only three colours and whether a 3-CNF boolean formula is satisfiable or not.

A simple probabilistic argument implies that "almost all" (in the sense described in Section 1.2.5) graphs on n vertices and more than $2.71n$ edges cannot be legally coloured with only three colours. By a more careful analysis of a necessary condition for the existence of one such colouring we were able to strengthen this bound to $2.61n$. Far from believing that these results are best possible, we concluded the relevant section by pointing out some recent developments and open problems.

The use of a well known probabilistic model allowed us to prove the best result to date on the presence of a satisfying assignment for a random formula. In Section 3.2.3 we proved that almost all 3-CNF boolean formulae with n variables and more than $4.5793n$ clauses are not satisfiable. Again the possibility of further improvements on these bounds is left open by this work.

Chapter 4

Hard Matchings

This chapter will investigate another application of randomness to Computer Science, sometimes called input randomisation. This is based on the simple principle that sometimes the set of instances for which a particular problem is difficult to solve forms a rather sparse or well structured subset of the set of all instances. In this context the assumption that not all inputs occur with the same probability can be used to derive improved performances for specific algorithmic solutions.

The setting will be that of optimisation problems. Many of these problems seem to be quite hard to solve exactly. For example the problem of finding the minimum length tour around a set of towns is the well known NP-complete Travelling Salesman Problem [GJ79]; that of finding the maximum number of non-adjacent vertices in a graph is equivalent to solving the so called Maximum Independent Set problem [GJ79]. The hardness results tell us that it is possible to construct a set of instances on which, under reasonable assumptions, the optimum cannot be found in polynomial time.

One way to cope with these results is to relax the optimality requirement and be happy with an approximate solution. In Section 4.1 the notion of approximation will be made precise: the concepts of approximation algorithm and approximation complexity classes will be defined.

A matching in a graph is a set of disjoint edges. Several optimisation problems are definable in terms of matchings by just changing the cost function or the optimisation criterion. For instance, if for any given graph G and matching M in G, the cost function returns the number of edges in M and the goal is to maximise the value of this function, then the corresponding problem is that of finding a maximum (cardinality) matching in G. The problem of finding a maximum matching in a graph has a glorious history and has an important place among combinatorial problems. The class

NP can be characterised as the set of all decision problems for which finding a solution among all possible candidates can take exponential time but checking whether a candidate is a solution only takes polynomial time (see for example [BDG88, Ch. 8]). Maximum matching is a nice example of a problem for which, despite of the existence of an exponential number of candidates, a solution can be found quickly. This fact, discovered by [Edm65], led to a number of algorithmic applications (see for example [HK73, MV80]).

Few other matching problems share the nice properties of maximum matching. In this chapter two problems will be considered which are not known to be solvable in polynomial time. Section 4.2 provides the reader with the relevant definitions and gives an overview of the known results for these problems. The following four sections present a number of worst case results.

The results in Section 4.3 and 4.6 imply that both problems are NP-hard for graphs with given bounds on the minimum and maximum degree. In Section 4.3 we introduce the concept of almost regular graph and we prove that the first problem is NP-hard for almost regular bipartite graphs of maximum degree $3s$ for every integer $s > 0$. For the second one, we show that even finding a solution whose size is at least a constant fraction ϵ from the optimum is NP-hard for some fixed $\epsilon < 1$, even if the input graph is almost regular of maximum degree $4s$ for every integer $s > 0$.

Section 4.3.2 describes a slightly unrelated hardness result. A relationship is established between one of the matching problems under consideration and the problem of finding a particular subgraph of a given graph called a *2-spanner* (the reader is referred to Section 4.3.2 for further details about this problem). The reduction has been recently used [DWZ] to prove approximation results for an optimisation problem related to 2-spanners in a class of planar graphs. The result was included in this work as an interesting application of the matching problem considered.

A number of simple positive approximation results are given in Section 4.4 and 4.5. In the first of the two a general technique is described for obtaining simple guarantees on the quality of *any* approximation heuristic for the given problems if the input graphs have known bounds on the minimum and maximum degrees. Also, families of regular graphs are constructed that match these guarantees. Secondly, Section 4.5 describes a linear time algorithm which solves optimally the second problem if the input graph is a tree. The problem was known to be solvable exactly on trees but the previously known algorithmic solution involved matrix multiplication and an intricate algorithm for computing a largest independent set in a *chordal* graph. Our solution is greedy in flavour in that it builds the optimal matching by traversing the edges in the tree only once.

The simple positive results presented in Section 4.4 are finally matched by a number of negative results, presented in Section 4.6.

Probabilistic techniques are used in the final part of this chapter. Although difficult to solve exactly in some worst case graphs, the matching problems considered in this chapter can be analysed rather well if assumptions are made on the distribution of the possible input graphs. Section 4.7 and 4.8 contain some discussion about improved approximation results that follow from the assumption that the input graphs belong to different models of random graphs. If the input graphs are very likely to be dense (that is graphs with n vertices and $\Theta(n^2)$ edges) the most likely values of the optima for the problems discussed can be pinned down quite well. Moreover simple greedy strategies provide very good quality approximation algorithms. If the most likely input graphs are sparse then the algorithmic results are weaker, but they still compare favourably with the results of the worst case analysis mentioned above.

4.1 Approximation Algorithms: General Concepts

The definition of optimisation problem was given informally in Section 1.2. This chapter will be entirely concerned with a particular class of optimisation problems. All the definitions in this Section are from [Cre97].

Definition 15 *An* NP optimisation problem *(NPO) P is a tuple $(\mathcal{I}, \mathcal{SOL}, c, opt)$ where:*

(1) $\mathcal{I}$ is the set of the instances *of P and the membership in $\mathcal{I}$ can be decided in polynomial time.*

(2) For each $x \in \mathcal{I}$, $\mathcal{SOL}(x)$ is the set of feasible solutions *of x. Membership in $\mathcal{SOL}(x)$ can be decided in polynomial time and for each $y \in \mathcal{SOL}(x)$, the length of y, $|y|$ is polynomial in the length of x.*

(3) For each $x \in \mathcal{I}$ and each $y \in \mathcal{SOL}(x)$, $c(x, y)$ is an integer, non-negative function, called the objective *or* cost *function.*

(4) opt $\in \{\max, \min\}$ is the optimisation criterion *and tells if the problem P is a maximisation or a minimisation problem.*

For example if $\mathcal{I}$ is the set of undirected graphs, $\mathcal{SOL}(G)$ is, for every $G \in \mathcal{I}$, the collection of all sets of vertices $U \subseteq V(G)$ such that every edge in G has at least one end point in U, $c(G, U) = |U|$ for every $U \in \mathcal{SOL}(G)$, and *opt* $= \min$ then the problem under consideration is that of finding a, so called, *vertex cover* of the edges of minimum cardinality (denoted by MINVC). The number of vertices of this optimal cover is a graph parameter normally denoted by $\tau(G)$ [LP86].

Definition 16 *Let P be an NPO problem. Given an instance x and a feasible solution y of x, the* performance ratio *of y with respect to x is*

$$R(x, y) = \max\left\{\frac{c(x, y)}{opt(x)}, \frac{opt(x)}{c(x, y)}\right\}$$

where $opt(x)$ is the objective function evaluated at an optimal point.

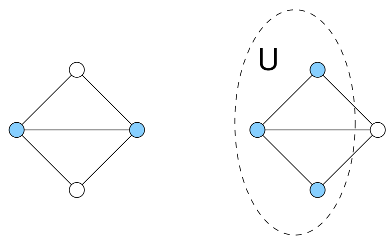

Figure 4.1: Possible Vertex Covers

For example the vertex cover U in Figure 4.1 has cardinality $\frac{3}{2}\tau(G)$ (an optimal cover is shown on the left). Hence $R(G, U) = 3/2$.

Definition 17 *Let P be an NPO problem and let T be an algorithm that, for any given instance x of P, returns a feasible solution $T(x)$ of x. Given an arbitrary function $r : \mathbb{N} \to (1, \infty)$ we say that T is an $r(n)$*-approximation algorithm *for P if, for every instance x of order n,*

$$R(x, T(x)) \leq r(n).$$

Also we say that P can be approximated with ratio *$r > 1$ if there exists an r-approximation algorithm for P.*

For the vertex cover problem a very simple and elegant argument proves that the number of end-points of the edges in a maximal matching in G always approximates $\tau(G)$ within a factor of two (see for instance [CLR90, Sect. 37.1]).

Optimisation problems can be grouped into classes depending on the quality of the approximation algorithms that they have. The class APX contains all NPO problems which admit a polynomial time k-approximation algorithm for *some* constant $k > 1$. The class PTAS contains all NPO problems which admit a polynomial time k-approximation algorithm for *any* constant $k > 1$. The class PTAS takes its name and is characterised in terms of a particular family of approximation algorithms.

Definition 18 *A* polynomial time approximation scheme *(or ptas) for an NPO problem P is an algorithm A which takes as input an $x \in \mathcal{I}$ and an* error bound ϵ *and has a performance ratio*

$$R_\epsilon(x, A(x)) \leq 1 + \epsilon$$

The algorithm A runs in time polynomial in the input order and in ϵ^{-1}.

4.2 Problem Definitions

In this section the relevant optimisation problems will be defined. All graphs in the following discussion will be undirected and labelled. If $G = (V, E)$ is a graph, a set $M \subseteq E$ is a *matching* in G if $e_1 \cap e_2 = \emptyset$ for all $e_1, e_2 \in M$. Let $V(M)$ be the set of vertices belonging to edges in the matching. A matching M is *maximal* if for every $e \in E \setminus M$, there exists $f \in M$ such that $e \cap f \neq \emptyset$ (we say that f *covers* e). A matching M is *induced* if for every edge $e = \{u, v\}$, $e \in M$ if and only if $u, v \in V(M)$ and $e \in E$. A number of parameters can be defined to characterise matchings in graphs:

Definition 19 *If $G = (V, E)$ is a graph then*

1. *$\beta(G)$ denotes the minimum cardinality of a maximal matching in G;*
2. *$\nu(G)$ denotes the maximum cardinality of a matching in G;*
3. *$\nu_I(G)$ denotes the maximum cardinality of an induced matching in G.*

In the following sections some of the combinatorial and computational properties of parameters $\beta(G)$ and $\nu_I(G)$ will be described. The appelations MINMAXLMATCH and MAXINDMATCH will be used to identify the corresponding optimisation problems.

A vast literature is concerned with the parameter $\nu(G)$ (see [LP86] for a beautiful and complete treatment) but much less seems to be known about $\beta(G)$ and $\nu_I(G)$. The following paragraphs give an overview of the known results.

Small Maximal Matchings. The oldest result on $\beta(G)$ is by Forcade [For73] who proved an approximation result on hypercubes. The first negative result is by Yannakakis and Gavril [YG80]. An *edge dominating set* in a graph is a set of edges F such that for all $e \in E(G)$ there exists $f \in F$ with $e \cap f \neq \emptyset$. All maximal matchings are edge dominating sets. Yannakakis and Gavril proved that every edge dominating set can be translated into a possibly smaller maximal matching. Then they proved that finding an edge dominating set with at most k edges is NP-complete even for planar or bipartite graphs with maximum degree three. In the same paper they give a polynomial time algorithm for trees. Subsequently Horton and Kilakos [HK93] extended the NP-completeness to planar bipartite graphs, planar cubic graphs and few other classes of graphs. A graph is *chordal* if for every simple circuit of length at least four there exists an edge not in the circuit connecting two vertices belonging to the circuit. Horton and Kilakos also gave a $O(n^3)$ algorithm for classes of chordal graphs and a few other classes.

Induced Matchings. The first proof of NP-completeness is in [SV82]. The authors present their results in terms of δ-separated matchings. The notion of distance between two vertices, defined in Section 1.2.4, can be extended in the obvious way to pairs of edges. Given a graph $G = (V, E)$, for all $e, f \in E$ the distance $dst_G(e, f)$ is the length of the shortest path among all paths connecting the vertices in e and f. A matching M is δ-separated if the minimum distance between two edges in M is δ. Obviously a 2-separated matching is an induced matching. Stockmeyer and Vazirani prove that finding a δ-separated matching of size at least k is NP-complete. Their reduction is from vertex cover and holds even for bipartite graphs of maximum degree four. An alternative proof of NP-completeness, again even for bipartite graphs, is in [Cam89]. Induced matchings have attracted attention following two questions by Erdős and Nesetril (see [Erd88]):

1. What is the maximum number of edges in a graph of maximum degree Δ and such that $\nu_I(G) \leq k$?
2. What is the minimum t for which the edge set of G can be partitioned into t induced matchings.

A series of improved answers have come in a number of papers [FGST89, HQT93, SY93, LZ97]

Not much is known about the approximability of $\beta(G)$ and $\nu_I(G)$. The only non trivial result is claimed in [Bak94] where a polynomial time approximation scheme for $\beta(G)$ is described for planar graphs.

4.3 NP-hardness Results

In this Section we start describing the new results in this chapter. The class of graphs for which MINMAXLMATCH and MAXINDMATCH are NP-hard to find is extended. The first result deals the hardness of MINMAXLMATCH for graphs which show some regularity whereas the second one relates MAXINDMATCH on bipartite graphs to another interesting combinatorial problem. Other hardness results will follow as by-products of the results in Section 4.6.

A (δ, Δ)-graph is a graph with minimum degree δ and maximum degree Δ. A (d, d)-graph is a regular graph of degree d (or a d-regular graph). If P is an NPO graph problem, then (δ, Δ)-P (resp. d-P) denotes the same problem when the input is restricted to the being a (δ, Δ)-graph (resp. a d-regular graph). Finally, a (δ, Δ)-graph G is *almost regular* if Δ/δ is bounded by a constant.

4.3.1 MINMAXLMATCH in Almost Regular Bipartite Graphs

The NP-completeness proofs in [HK73] and [YG80] show that it is NP-hard to find a maximal matching of minimum cardinality in a planar cubic graph and in planar bipartite $(1, 3)$-graphs. This last result can be extended to bipartite $(ks, 3s)$-graphs for every integer $s > 0$ and $k = 1, 2$. The following result shows how to remove vertices of degree one from a $(1, \Delta)$-graph.

Lemma 29 *There is a polynomial time reduction from* $(1, \Delta)$-MINMAXLMATCH *to* $(2, \Delta)$-MINMAXLMATCH.

Proof. Given a $(1, 3)$-graph G, the graph G' is obtained by replacing each vertex v of degree one in G by the gadget G_v shown in Figure 4.2. The edge $\{v, w\}$ incident to v is attached to the vertex v_0. The resulting graph has minimum degree two and maximum degree three. If M is a maximal matching in G it is easy to build a maximal matching in G' of size $|M|+2|V_1(G)|-|V(M) \cap V_1(G)|$. For every $v \in V_1(G)$ add $\{v_1, v_2\}$ to M' moreover if $\{u, v\} \notin M$ then M' will contain also the edge $\{v_0, v_3\}$. Conversely every matching M' in G' can be transformed into a matching $M'' = M''_1 \cup M''_2$, with $|M''| \leq |M'|$, such that M''_1 is a maximal matching in G and M'' is a set of edges entirely contained in the gadgets G_v. □

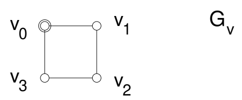

Figure 4.2: Gadget replacing a vertex of degree one in a bipartite graph.

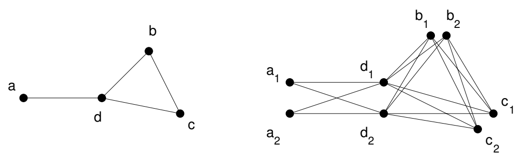

Figure 4.3: A $(1,3)$-graph and its 2-padding.

To prove the main hardness result in this section, the following graph operation will be useful.

Definition 20 *The* s-padding *of a graph G, G_s, is obtained by replacing every vertex v by a distinct set of* twin vertices $v_1, \ldots, v_s$ *with* $\{v_i, u_j\} \in E(G_s)$ *if and only if* $\{u, v\} \in E(G)$.

- *The vertices of G_s are partitioned into s* layers. *Vertices in each layer along with edges connecting pairs of vertices in the same layer form a copy of the original graph G.*
- *For each $e = \{u, v\} \in E(G)$ edges $e_i = \{u_i, v_i\}$ for $i = 1, \ldots, s$ are called* twin edges. *Edges $e_{ij} = \{u_i, v_j\}$ for each $i \neq j$ are called* cross edges.
- *Each copy of $K_{s,s}$ obtained by replacing two vertices and an edge in G is called a* padding copy of $K_{s,s}$ *and sometimes denoted by $K^e_{s,s}$, to show dependency on the edge in the original graph G.*

The following result is a simple consequence of Definition 20.

Lemma 30 *If G is a (δ, Δ)-graph with n vertices and m edges then G_s is a $(s \cdot \delta, s \cdot \Delta)$-graph with sn vertices and $s^2 m$ edges.*

To prove Theorem 53 it is important to relate $\beta(G_s)$ to $\beta(G)$.

Lemma 31 $\beta(G_s) \leq s\,\beta(G)$*, for all graphs G and $s \geq 1$.*

Proof. If M is a maximal matching in G, a maximal matching M_s in G_s is obtained by taking the union of $|M|$ perfect matchings one in each copy of $K^e_{s,s}$ with $e \in M$. □

Lemma 32 $\beta(G_s) \geq s\,\beta(G)$*, for all bipartite graphs G and $s \geq 1$.*

Proof. Let G_s be the padded version of a bipartite graph G, and M_s be a maximal matching in G_s. The s-weighting of the edges of G is a function $w : E(G) \to \{0, \ldots, s\}$. For each $e \in E(G)$ define $w(e) =_{df} |M_s \cap E(K^e_{s,s})|$. The following properties hold:

1. If $w(v) =_{df} \sum_{\{e:v\in e\}} w(e)$ then $w(v) \in \{0, \ldots, s\}$ for all $v \in V(G)$.

   The sum of the weights of the edges incident to v cannot be larger than s otherwise there would be more than s edges in M_s incident to $v_1, \ldots, v_s$; therefore one of these vertices would be incident to more than one edge in M_s.

2. Let $E(i) = \{e \in E(G) : w(e) = i\}$ for $i \in \{0, \ldots, s\}$. Then $\bigcup_{i=1}^{s} E(i)$ is an edge dominating set of G (as defined in Section 4.2).

   This is true because if there was an edge e in G not adjacent to any edge with positive weight then $K_{s,s}^e$ would not be covered by M_s in G_s.

3. Let $G(E(i))$ be the subgraph of G induced by $V(E(i))$. Then $E(s)$ is a maximal matching in $G(E(s))$.

   The edges in $E(s)$ must be independent because each of them corresponds to a perfect matching in a padding copy of $K_{s,s}$.

4. Let $G - G(E(i))$ be the graph obtained by removing from $V(G)$ all vertices in $V(G(E(i)))$ and all edges adjacent to them. Then $\bigcup_{i=1}^{s-1} E(i)$ is an edge dominating set in $G - G(E(s))$.

5. Let $v \in V(G - G(E(s)))$; if $w(v) < s$ then $w(u) = s$ for every $u \in N(v)$.

   Let v_i be one of the twin vertices associated with v that is not in $V(M_s)$. For each $u \in N(v)$ each of the edges $\{v_i, u_j\}$ (for $j = 1, \ldots, s$) must be adjacent to a different edge in M_s otherwise they would not be covered.

Using the s-weighting defined above the edges in M_s can be partitioned into s matchings each corresponding to a maximal matching in G.

First of all, each edge e in $E(s)$ corresponds to s distinct edges $e_1, \ldots, e_s$ in M_s. Define $M(j) = \{e_j : \text{for each } e \in E(s)\}$. We prove, reasoning by induction on s, that the set of remaining edges in M_s, M_s', can be partitioned into s sets M^j such that, $M^j \cup M(j)$ corresponds to a maximal matching in G, for each $j = 1, \ldots, s$.

BASE. If $s = 2$ by property 4 above, the set $E(1)$ is formed by a number of paths and even length cycles, $E_1, \ldots, E_k$. Each cycle of length $2m$ (for some integer $m > 1$) can be decomposed into two matchings M^1 and M^2 of size m by taking alternating edges. If E_j is a path then, by property 5 above, neither of its end-points can be adjacent to a vertex v with $w(v) = 0$. Therefore again two set of edges are added to M^1 and M^2.

STEP. Let H_s be the graph induced by the edges of positive weight less than s. If M^s is a maximal matching in H_s, then $w'(e) = w(e) - 1$ (resp. $w'(e) = w(e)$) if $e \in M^s$ (resp. $e \notin M^s$) is an $(s-1)$-weighting of $E(G - G(E(s)))$ corresponding to the a maximal matching in the $s-1$-padding of $G - G(E(s))$. The inductive hypothesis applies. □

Theorem 53 MINMAXLMATCH *is NP-hard for almost regular bipartite graphs.*

Proof. We will prove hardness for $(ks, 3s)$-graphs, with $k = 1$ or $k = 2$. Yannakakis and Gavril [YG80] proved that $(1, 3)$-MINMAXLMATCH is NP-hard for bipartite graphs. The hardness of $(2, 3)$-MINMAXLMATCH follows from Lemma 29. Then, $k = 1, 2$ the s-padding can be used to obtain an instance of $(ks, 3s)$-MINMAXLMATCH restricted to bipartite graphs. The result then follows from Lemma 31 and Lemma 32. □

4.3.2 MAXINDMATCH and Graph Spanners

Given a graph $G = (V, E)$ a 2-*spanner* is a spanning subgraph G' with the property that $dst_{G'}(u, v) \leq 2 \cdot dst_G(u, v)$ for every $u, v \in V$. Let $s_2(G)$ be the number of edges of a sparsest 2-spanner of G. The problem has many applications in areas like distributed computing, computational geometry and biology [PU89, ADJS93]. Peleg and Ullman [PU89] introduced the concept of graph spanners as a means of constructing synchronisers for Hypercubic networks (their results have been recently improved in [DZ]). The problem of finding a 2-spanner with the minimum number of edges is NP-hard [Pm89]. In this Section we present a reduction from the problem of finding a sparsest 2-spanner in a graph to that of finding a largest induced matching in a bipartite graph without small cycles.

For every G on n vertices and m edges, let $B(G)$ be a bipartite graph with vertex sets $U = \{u_e : e \in E(G)\}$ and $W = \{w_C : C \text{ is a cycle of length 3 in } G\}$. Two vertices $u_e \in U$ and $w_C \in W$ in $B(G)$ are adjacent if the edge e belongs to the cycle C in G.

Lemma 33 $s_2(G) \leq m - \nu_I(B(G))$.

Proof. Let M be an induced matching in $B(G)$. Define $S = \{e \in E(G) : \{u_e, w_C\} \notin M\}$. We claim that S is a spanner in G. This is so because for every $\{u_e, w_C\} \in M$ the edges $f, g \in E(G)$ that form the cycle C along with e are such that $\{u_f, w_C\}, \{u_g, w_C\}$ cannot be in M and therefore are $f, g \in S$. □

Lemma 34 $s_2(G) \geq m - \nu_I(B(G))$.

Proof. Let G' be a 2-spanner in G. We prove that we can construct an induced matching in $B(G)$. If $e \in E(G) \setminus E(G')$ then there exist two edges $f, g \in G'$ such that $C = \{e, f, g\}$ is a triangle in G. We add $\{u_e, w_C\}$ to the matching in $B(G)$ and we say that the triangle C *covers* e. Let M be the set of edges in $E(B(G))$ constructed in this way. Since a triangle can only cover one edge, there are no two edges in M sharing a vertex w_C. Also by our construction every edge in $E(G) \setminus E(G')$ is considered only once so that there are no two edges in M sharing an edge-vertex. We claim that M is an induced matching in $B(G)$. Let $\{u_e, w_C\} \in M$, assume edge e belongs to triangles $C_1, \ldots, C_{t(e)}$, and let $C = \{e, f, g\}$. No edge $\{u_h, w_{C_j}\}$ can be in M because, by the definition of M, $\{u_h, w_{C_j}\} \in M$ would imply that $e \in E(G')$ and therefore $\{u_e, w_C\} \notin M$. Similarly neither u_f, nor u_g can be in $V(M)$. □

The *girth* of a graph is the length of its shortest cycles.

Theorem 54 MAXINDMATCH *is NP-hard on bipartite graphs with girth at least six.*

Proof. For every graph G, the girth of $B(G)$ is at least six, since no two vertices $u_e, u_f \in U$ can share two neighbours in W. The result follows from Lemma 33 and 34. □

4.4 Combinatorial Bounds

The NP-hardness proofs in the previous section (and those in Section 4.6) imply that under reasonable assumptions (see [GJ79]) there can be no polynomial time algorithm which finds a maximal matching of minimum cardinality or one of the largest induced matchings in the given graph. The next best option is to look for algorithms returning approximate solutions. In this section we describe some easy combinatorial results which imply some guarantees on the quality of the solutions for the matching problems under consideration produced by a broad family of algorithms.

The set of all matchings in a graph G, which we denote by $\mathcal{M}(G)$, defines an instance of a type of combinatorial object called an independence system. The following definition is from [KH78].

Definition 21 *An* independence system *is a pair $(E, \mathcal{F})$ where E is a finite set and $\mathcal{F}$ a collection of subsets of E with the property that whenever $F \subset G \in \mathcal{F}$ then $F \in \mathcal{F}$. The elements of $\mathcal{F}$ are called* independent sets*. A* maximal *independent set is an element of $\mathcal{F}$ that is not a subset of any other element of $\mathcal{F}$.*

Korte and Hausmann [KH78] analysed the independence system $(E(G), \mathcal{M}(G))$ and proved an upper bound of 2 on the ratio between the cardinalities of any two maximal matchings. The result they prove is

Theorem 55 $\nu(G) \leq 2\,\beta(G)$ *for any graph* G.

The theorem immediately implies the existence of a simple approximation heuristic for $\beta(G)$: an algorithm returning any maximal matching in G is a 2-approximation algorithm. We therefore have the following theorem.

Theorem 56 MINMAXLMATCH *can be approximated with ratio 2.*

In the next result a similar argument is applied to $(1, \Delta)$-MAXINDMATCH. Let G be a (δ, Δ)-graph. Let $\mathcal{M}_I(G)$ be the set of all induced matchings in G. The pair $(E(G), \mathcal{M}_I(G))$ is an independence system. For every $S \subseteq E$ the *lower rank* of S, $\underline{\rho}(S)$ is the number of edges of the smallest (maximal) induced matching included in S; the *upper rank* $\bar{\rho}$ is defined symmetrically. By a theorem in [KH78], if M is a maximal induced matching, then

$$\frac{\nu_I(G)}{|M|} \leq \max_{S \subseteq E} \frac{\overline{\rho}(S)}{\underline{\rho}(S)}$$

Theorem 57 *Let* G *be a* (δ, Δ)*-graph and* $(E(G), \mathcal{M}_I(G))$ *be given. Then*

$$\max_{S \subseteq E} \frac{\overline{\rho}(S)}{\underline{\rho}(S)} \leq 2(\Delta - 1).$$

Proof. Let M_1 and M_2 be two maximal induced matchings in G and let $e \in M_2 \setminus M_1$. By the maximality condition, the set $M_1 \cup \{e\}$ is not independent (i.e. it is not an induced matching anymore). Hence there exists $\phi(e) \in M_1$ at distance less than two from e and since M_2 is maximal and independent, $\phi(e) \in M_1 \setminus M_2$. Indeed ϕ defines a function from $M_2 \setminus M_1$ to $M_1 \setminus M_2$. Let f be one of the edges in the range of ϕ. A bound on the number of edges $e \in M_2 \setminus M_1$ that can be the pre-image of $f \in M_1 \setminus M_2$ is needed. In the worst case (occurring when e is not adjacent to f) there can be at most $2(\Delta - 1)$ such e. The result follows. □

The last result proves that any algorithm that returns a maximal induced matching in the given graph is a $2(\Delta - 1)$-approximation algorithm.

Theorem 58 $(1, \Delta)$-MAXINDMATCH *can be approximated with ratio* $2(\Delta - 1)$.

The next set of results describes a technique to obtain improved bounds on $\beta(G)$ and $\nu_I(G)$ on (δ, Δ)-graphs. The idea can be readily introduced in the context of vertex cover. Let $G = (V, E)$

be a (δ, Δ)-graph and let U be a vertex cover of cardinality k in G. If U is a vertex cover then $R = V \setminus U$ must be an independent set otherwise there would be an edge not covered by U. This implies that all edges going out from R must end in U. Let $d^1(v)$ be the number of edges adjacent to $v \in U$ and to some $v' \in R$. Then

$$\sum_{v \in R} \deg(v) = \sum_{v \in U} d^1(v) \tag{4.1}$$

If G has minimum degree $\delta > 0$ and maximum degree Δ

$$(n-k)\delta \leq \sum_{v \in R} \deg(v) = \sum_{v \in U} d^1(v) \leq k\Delta$$

from which

$$k \geq \frac{n\delta}{\Delta + \delta}.$$

Similar argument can be applied to bound $\beta(G)$ and $\nu_I(G)$.

Theorem 59 *If G is a (δ, Δ)-graph*

1. $\beta(G) \geq \frac{\delta |V(G)|}{2(\Delta+\delta-1)}$;
2. $\nu_I(G) \leq \frac{\Delta |V(G)|}{2(\Delta+\delta-1)}$.

Proof. If M is a maximal matching then $R = V \setminus V(M)$ is an independent set. Otherwise M could be extended with any edge connecting two different vertices in R. This implies that all vertices adjacent to a vertex in R must belong to $V(M)$. The analogue of (4.1) is

$$\sum_{v \in R} \deg(v) = \sum_{v \in V(M)} d^1(v)$$

If G has minimum degree $\delta > 0$ and maximum degree $\Delta \geq 1$,

$$(n-2k)\delta \leq \sum_{v \in R} \deg(v) = \sum_{v \in V(M)} d^1(v) \leq 2k(\Delta - 1)$$

Thus

$$k \geq \frac{\delta |V(G)|}{2(\Delta + \delta - 1)}$$

If M is a maximal induced matching then for each v such that $\{u, v\} \in M$, $(N(\{v\}) - \{u\}) \cap V(M) = \emptyset$. On the other hand it is not true anymore that R is an independent set (an edge between two vertices in R is at distance one from edges in M so it cannot be chosen, but its presence is legal in a maximal induced matching). In this case we can write (where d^1 now refers to R)

$$\sum_{v \in R} d^1(v) = \sum_{v \in V(M)} (\deg(v) - 1)$$

If G has minimum degree $\delta \geq 1$ and maximum degree $\Delta \geq \delta$

$$(n - 2k)\Delta \geq 2k(\delta - 1) \tag{4.2}$$

from which we get the upper bound

$$k \leq \frac{\Delta|V(G)|}{2(\Delta + \delta - 1)}$$

□

Notice that if G is d-regular the lower bound on $\beta(G)$ and the upper bound on $\nu_I(G)$ coincide. Indeed it is not difficult to construct a family of graphs matching these bounds.

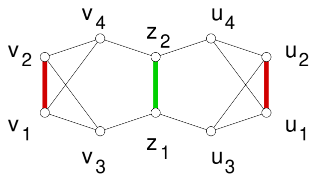

Figure 4.4: A cubic graph with small maximal matching and large induced matching.

Theorem 60 *Let $d \in \mathbb{N}^+$. If G is a d-regular graph on n vertices then*

$$\nu_I(G) \leq \frac{d|V(G)|}{2(2d - 1)} \leq \beta(G)$$

Moreover for every $d = 2i + 1$ with $i \in \mathbb{N}^+$ there exists a graph G_i on $2(2d - 1)$ vertices with $\nu_I(G_i) = \beta(G_i) = d$.

Proof. The first part is an immediate consequence of Theorem 59. The second part can be proved by giving a recursive description of G_i for all $i \in \mathbb{N}^+$. To simplify the description it is convenient to draw G_i so that all its vertices are on five different layers, called *far-left*, *mid-left*, *central*, *mid-right* and *far-right* layer. Figure 4.4 shows G_1. Vertices v_1 and v_2 (respectively u_1 and u_2) are in the far-left (respectively far-right) layer. Vertices v_3 and v_4 (respectively u_3 and u_4) are in the mid-left (respectively mid-right) layer. Vertices z_1 and z_2 are in the central layer. Moreover an horizontal axis separates odd-indexed vertices (which are below it) from even-indexed ones (which are above), with smaller indexes below higher ones.

Let G_{i-1}, for $i \geq 2$, be given. The graph G_i is obtained by adding four central vertices, two mid-left and two mid-right vertices. Since G_1 has two central and two pairs of mid vertices and easy inductions proves that G_{i-1} has $2[2(i-1)-1]$ central vertices and $2(i-1)$ mid-left and mid-right ones. Let $z_{4(i-1)-1}$, $z_{4(i-1)}$, z_{4i-3}, z_{4i-2} be the four "new" central vertices, v_{2i+1} and $v_{2(i+1)}$,

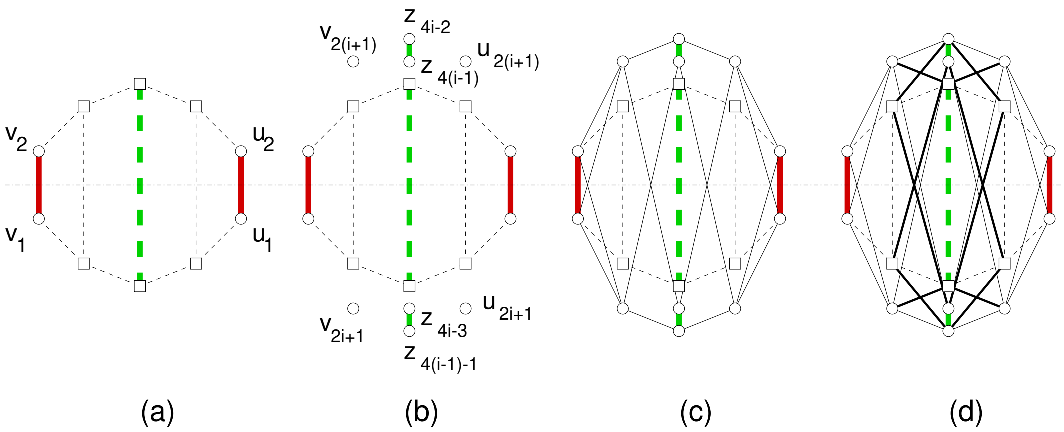

Figure 4.5: A d-regular graph with small maximal matching and large induced matching.

u_{2i+1} and $u_{2(i+1)}$ the mid-left and mid-right ones. G_i has all edges of G_{i-1} plus the following groups:

1. two edges connecting each of v_{2i+1}, $v_{2(i+1)}$, (respectively u_{2i+1} and $u_{2(i+1)}$) to v_1 and v_2 (respectively u_1 and u_2), plus edges

$$\{v_{2i+1}, z_{4(i-1)-1}\}, \{v_{2i+1}, z_{4(i-1)}\},$$

$$\{v_{2(i+1)}, z_{4i-3}\}, \{v_{2(i+1)}, z_{4i-2}\},$$

$$\{u_{2i+1}, z_{4(i-1)-1}\}, \{u_{2i+1}, z_{4(i-1)}\},$$

$$\{u_{2(i+1)}, z_{4i-3}\}, \{u_{2(i+1)}, z_{4i-2}\}$$

All these edges are the continuous black lines in Figure 4.5.(c).

2. A final set of edges connects each of the even index mid vertices with the central vertices of G_{i-1} with indices $4j-2$ and $4j-3$ for $j = 0, 1, \ldots, i-1$. Each of the odd index mid vertices are connected with the central vertices of G_{i-1} with indices $4(j-1)$ and $4(j-1)-1$ for $j = 1, \ldots, i$. The squares in Figure 4.5 represent all mid vertices in G_{i-1}. The bold solid lines in Figure 4.5.(d) represent this kind of edges.

□

Theorem 59.*2* is complemented by the following result, giving a lower bound on the size of a particular family of induced matchings in the given graph.

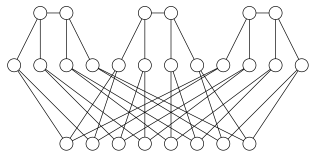

Figure 4.6: A cubic graph with small maximal induced matching.

Theorem 61 *Let* $f(\delta, \Delta) = \frac{4\Delta^2 - 4\Delta + 2}{\delta}$. *If* G *is a* (δ, Δ)*-graph, then* $\nu_I(G) \geq |V(G)|/f(\delta, \Delta)$. *Moreover for every* $d \geq 2$ *there exists a regular graph of degree* d *with* $d \cdot f(d, d)$ *vertices and a maximal induced matching of size* d.

Proof. Let G be a (δ, Δ)-graph on n vertices and M a maximal induced matching in G. An edge $e \in E(G)$ is *covered* by an edge $f \in M$ if there exists a path of length less than two between one of the end-points of e and one of the end-points of f. Each edge in G must be covered by at least one edge in M. Conversely every edge in M can cover at most $2(\Delta - 1)^2 + 2\Delta - 1$ edges. Thus

$$|M| \geq \frac{|E|}{2(\Delta - 1)^2 + 2\Delta - 1} \geq \frac{\delta |V(G)|}{2(2\Delta^2 - 2\Delta + 1)}.$$

A *d-ary depth two tree* T_d is formed by connecting with an edge the roots of two identical copies of a complete d-ary tree on $d^2 - d + 1$ vertices. The graph obtained by taking d copies of T_d all sharing the same set of $(d-1)^2$ leaves is regular of degree d, it has $d \cdot f(d, d)$ vertices and a maximal induced matching of size d. Figure 4.6 shows the appropriate cubic graph. □

4.5 Linear Time Solution for Trees

Although NP-hard for several classes of graphs including planar or bipartite graphs of maximum degree four and almost regular graphs of maximum degree four (see Theorem 66), the problem of finding a largest induced matching admits a polynomial time solution on trees [Cam89]. The algorithmic approach of Cameron reduces the problem to that of finding a largest independent set in a graph H that can be defined starting from the given tree. If $G = (V, E)$ is a tree, the graph $H = (W, F)$ has $|V| - 1$ vertices, one for each edge in G and there is an edge between two members of W if and only if the two original edges in G are either incident or connected by a single edge. Notice that $|F| = O(|V|^2)$. Moreover each induced matching in G is an independent set of the same cardinality in H. Gavril's algorithm [Gav72] finds a largest independent set in a chordal graph with n vertices and m edges in $O(n + m)$ time. Since the graph H is chordal, a largest induced matching in the tree can be found in $O(|V|^2)$ time. In this section we describe a simpler and more efficient way of finding a maximum induced matching in a tree based on dynamic programming. If $G = (V, E)$ is a tree we choose a particular vertex $r \in V$ to be the *root* of the tree (and we say that G is *rooted* at r). If $v \in V \setminus \{r\}$ then parent(v) is the unique neighbour of v in the path from v to r; if parent(v) $\neq r$ then grandparent(v) = parent(parent(v)). In all other cases parent and grandparent are not defined. If u = parent(v) then v is u's *child*. All children of the same node are *siblings* of each other. Let $c(v)$ be the number of children of node v. The *upper neighbourhood* of v (in symbols UN(v)) is empty if $v = r$, it includes r and all v's siblings if v is a child of r and it includes v's siblings, v's parent and v's grandparent otherwise. $E(\mathrm{UN}(v))$ is the set of edges in G connecting the vertices in UN(v).

Claim 2 *If $G = (V, E)$ is a tree and M is an induced matching in G then $|M \cap E(\mathrm{UN}(v))| \leq 1$, for every $v \in V$.*

To believe the claim notice that if M is an induced matching in G, any node v in the tree belongs to one of the following types with respect to the set of edges $E(\mathrm{UN}(v))$:

Type 1. the edge $\{v, \mathrm{parent}(v)\}$ is part of the matching,

Type 2. either $\{\mathrm{parent}(v), \mathrm{grandparent}(v)\}$ or $\{\mathrm{parent}(v), w\}$ (where w is some siblings of v) belongs to the matching,

Type 3. Neither **Type 1.** nor **Type 2.** applies.

The algorithm for finding a largest induced matching in a tree G on n vertices handles an $n \times 3$ matrix Value such that Value$[i, t]$ is the cardinality of the matching in the subtree rooted at i if vertex i is of type t.

Lemma 35 *If G is a tree on n vertices,* Value$[i, t]$ *can be computed in $O(n)$ time for every $i \in \{1, \ldots, n\}$ and $t = 1, 2, 3$.*

Proof. Let G be a tree on n vertices and let r be its root. We assume G is in adjacency list representation and that some linear time preprocessing is performed to order (using standard topological sort [CLR90, Ch. 23]) the vertices in decreasing distance from the root.

The matrix Value can be filled in a bottom-up fashion starting from the deepest vertices of G. If i is a leaf of G then Value$[i, t] = 0$ for $t = 1, 2, 3$. In filling the entry corresponding to node $i \in V$ of type t we only need to consider the entries for all children of i, $j_1, \ldots, j_r$.

1. $\text{Value}[i, 1] = \sum_{k=1}^{c(i)} \text{Value}[j_k, 2]$.

   Since $\{i, \text{parent}(i)\}$ will be part of the matching, we cannot pick any edge from i to one of its children. The matching for the tree rooted at i is just the union of the matchings of the subtrees rooted at each of i's children.

2. $\text{Value}[i, 2] = \sum_{k=1}^{c(i)} \text{Value}[j_k, 3]$.

   We cannot pick any edge from i to one of its children here either.

3. If i has $c(i)$ children then $\text{Value}[i, 3]$ if defined as the maximum between $\sum_{k=1}^{c(i)} \text{Value}[j_k, 3]$ and a number or terms

   $$s_{j_k} = 1 + \text{Value}[j_k, 1] + \sum_{l \neq k} \text{Value}[j_l, 2]$$

   If the upper neighbourhood of i is unmatched we can either combine the matchings in the subtrees rooted at each of i's children (assuming these children are of type 3) or add to the matching an edge from i to one of its children j_k (the one that maximises s_{j_k}) and complete the matching for the subtree rooted at i with the matching for the subtree rooted at j_k (assuming j_k is of type 1) and that of the subtrees rooted at each of i's other children (assuming these children are of type 3).

Option three above is the most expensive involving the maximum over a number of sums equal to the degree of the vertex under consideration. Since the sum of the degrees in a tree is linear in the number of vertices the whole table can be computed in linear time. □

Theorem 62 MAXINDMATCH *can be solved optimally in polynomial time if G is a tree.*

Proof. The largest between $\text{Value}[r, 1]$, $\text{Value}[r, 2]$ and $\text{Value}[r, 3]$ is the cardinality of a largest induced matching in G. By using appropriate date structures it is also possible to store the actual matching. The complexity of the whole process is $O(n)$. □

4.6 Hardness of Approximation

So far two particular graph theoretic parameters both defined on the set $\mathcal{M}(G)$ of the matchings in G have been considered. After proving some NP-hardness results, their algorithmic approximability was studied. In this section we present some negative results.

Polynomial time approximation schemes (as defined in Section 4.1) for NPO problems are considered the next best thing to a polynomial time exact algorithm. For many NPO problems an important question is whether such a scheme exists. The approach taken parallels the development of NP-completeness: some useful notions of reducibility are defined and then problems are grouped together on the basis of their non-approximability properties.

Although several notions of approximation preserving reductions have been proposed (see for example [Cre97]) the L-reduction defined in [PY91] is perhaps the easiest one to use. Let P be an optimisation problem. For every instance x of P, and every solution y of x, let $c_P(x, y)$ be the *cost* of the solution y. Let $opt_P(x)$ be the cost of an optimal solution.

Definition 22 *Let P and Q be two optimisation problems. An* L-reduction *from P to Q is a four-tuple $(t_1, t_2, \alpha, \beta)$ where t_1 and t_2 are polynomial time computable functions and α and β are positive constants with the following properties:*

(1) t_1 maps instances of P to instances of Q and for every instance x of P, $opt_Q(t_1(x)) \leq \alpha \cdot opt_P(x)$.

(2) for every instance x of P, t_2 maps pairs $(t_1(x), y')$ (where y' is a solution of $t_1(x)$) to a solution y of x so that

$$|opt_P(x) - c_P(x, t_2(t_1(x), y'))| \leq \beta |opt_Q(t_1(x)) - c_Q(t_1(x), y')|.$$

We write $P \leq_L Q$ if there exists an L-reduction from P to Q. An important property of L-reductions is given by the following result.

Theorem 63 *Let P and Q be two NPO problems such that $P \leq_L Q$ with parameters α and β, and it is NP-hard to approximate P with ratio c.*

1. *If they are both maximisation problems, then it is NP-hard to approximate Q with ratio $\frac{\alpha\beta c}{(\alpha\beta-1)c+1}$.*

2. *If they are both minimisation problems, then it is NP-hard to approximate Q with ratio $\frac{(\alpha\beta+1)c-1}{\alpha\beta c}$.*

Proof. The result is essentially derived from [Pap94, Proposition 13.2]. To prove the first statement, suppose by contradiction that there is an algorithm which approximates Q with ratio $\frac{\alpha\beta c}{(\alpha\beta-1)c+1}$. For every instance x of P let y' be the result of applying this algorithm to $t_1(x)$. Then, by definition of

L-reduction,

$$\frac{opt_P(x) - c_P(x, t_2(t_1(x), y'))}{opt_P(x)} \le \alpha\beta \frac{opt_Q(t_1(x)) - c_Q(t_1(x), y')}{opt_Q(t_1(x))}$$

By definition of performance ratio, $\frac{opt_Q(t_1(x))}{c_Q(t_1(x),y')} \le \frac{\alpha\beta c}{(\alpha\beta-1)c+1}$, therefore

$$\frac{opt_Q(t_1(x)) - c_Q(t_1(x), y')}{opt_Q(t_1(x))} \le 1 - \frac{(\alpha\beta - 1)c + 1}{\alpha\beta c} = \frac{1}{\alpha\beta}\left(1 - \frac{1}{c}\right)$$

and the result follows.

Similarly if P and Q are minimisation problems and there exists an algorithm which approximates Q with ratio $\frac{(\alpha\beta+1)c-1}{\alpha\beta c}$ then

$$\frac{c_P(x, t_2(t_1(x), y')) - opt_P(x)}{c_P(x, t_2(t_1(x), y'))} \le \alpha\beta \frac{c_Q(t_1(x), y') - opt_Q(t_1(x))}{opt_Q(t_1(x))}$$

By definition of performance ratio $\frac{c_Q(t_1(x),y')}{opt_Q(t_1(x))} \le \frac{(\alpha\beta+1)c-1}{\alpha\beta c}$ and the result follows. □

4.6.1 MinMaxlMatch

Yannakakis and Gavril [YG80] prove the NP-hardness of $(1,3)$-MinMaxlMatch using a reduction from 3-MinVC restricted to planar graphs. Their reduction can be used as a building block for a number of L-reductions.

Theorem 64 3-MinVC $\le_L$ $(k,3)$-MinMaxlMatch *with parameters* $\alpha_k = 1 + 2f(k)$ *and* $\beta_k = 1$ *where* $f(k) = 3\binom{k}{2} + 2$ *for* $k = 1, 2, 3$.

Proof. Let $G = (V, E)$ be a planar cubic graph. Reduce 3-MinVC to $(k,3)$-MinMaxlMatch, by replacing every vertex $v_i \in V$ by the gadget H_k^i shown in Figure 4.7. The three edges incident to v_i are attached to u_i, m_i and p_i (there are 3! possibilities to do this). The resulting graph $t_1(G)_k$ is planar and $\Delta(t_1(G)_k) = 3$ for all k. We claim that $\beta(t_1(G)_k) = f(k)|V(G)| + \tau(G)$. If U is a vertex cover of G define

$$\begin{aligned}
M_1 &= \{\{u_i, w_{i_1}\}, \{m_i, w_{i_3}\}, \{p_i, w_{i_4}\} : v_i \in U\} \cup \{\{w_{i_1}, w_{i_2}\}, \{w_{i_3}, w_{i_4}\} : v_i \notin U\} \\
M_2 &= M_1 \cup \{\{z_{i_1}, z_{i_2}\}, \{z_{i_3}, z_{i_4}\}, \{z_{i_5}, z_{i_6}\} : v_i \in V\} \\
M_3 &= M_1 \cup \{\{t_{i_1}, t_{i_2}\}, \{t_{i_3}, t_{i_4}\}, \{t_{i_5}, t_{i_6}\} : v_i \in V\} \\
&\quad \cup \{\{z_{i_1}, z_{i_7}\}, \{z_{i_2}, z_{i_8}\}, \{z_{i_3}, z_{i_9}\}, \{z_{i_4}, z_{i_1 0}\}, \{z_{i_5}, z_{i_1 1}\}, \{z_{i_6}, z_{i_1 2}\} : v_i \in V\}
\end{aligned}$$

M_k is clearly a matching for all k. It is maximal because all edges in each H_k^i are adjacent to some $e \in M$ and if $e \in E$ was incident to some $v_i \in U$ then $e \in E(t_1(G)_k)$ and it is incident to one of $\{u_i, w_{i_1}\}, \{m_i, w_{i_3}\}, \{p_i, w_{i_4}\} \in M$. We have $|M_k| = (2 + 3\binom{k}{2})n + |U|$.

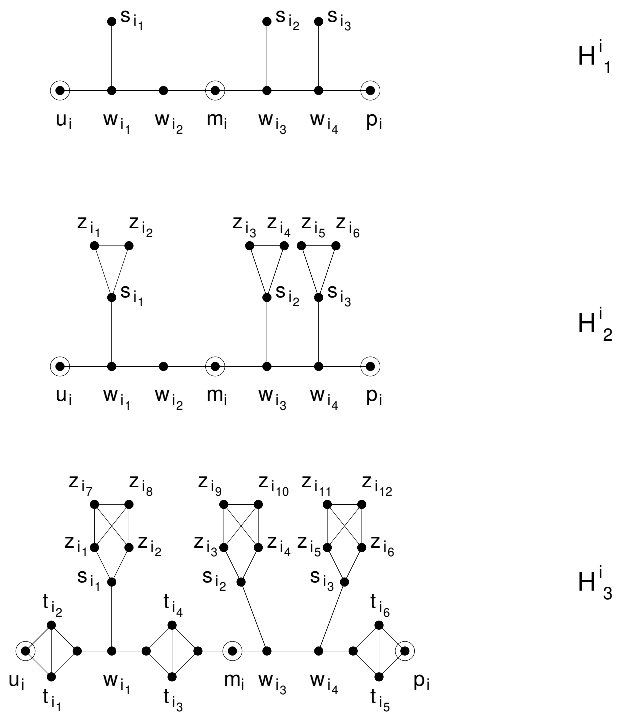

Figure 4.7: Gadgets for a vertex $v_i \in V(G)$.

Conversely to prove $\beta(t_1(G)_1) \geq 2n+\tau(G)$ Yannakakis and Gavril show that any maximal matching M in $t_1(G)_1$ can be transformed (in polynomial time) into another M' containing no edge of the original graph G and such that every H_1^i contains either two or three edges in M. Every edge of the original graph must be adjacent to at least one H_1^i containing three edges in M'; vertices $v_i \in V$ associated with such H_1^i define a vertex cover in G.

For $k > 1$, any maximal matching M in $t_1(G)_k$ can be transformed in another (possibly smaller) matching M' that uses the edges in $M_k \setminus M_1$ to cover the sets $E(H_k^i) \setminus E(H_1^i)$. Then Yannakakis and Gavril's applies to $M' \setminus (M_k \setminus M_1)$ and the edges in $E(t_1(G)_k) \setminus (E(H_k^i) \setminus E(H_1^i))$.
$\square$

3-MINVC is APX-complete [BK99]. Therefore there exists a $c_0 > 1$ such that it is NP-hard to approximate $\tau(G)$ with ratio c even if G is a cubic graph. Using this constant and Theorem 63 we have

Corollary 3 *Let c_0 be a constant such that* 3-MINVC *is NP-hard to approximate with ratio c_0. Then $(k, 3)$-*MINMAXLMATCH *is hard to approximate with ratio $\frac{2(f(k)+1)c_0-1}{(2f(k)+1)c_0}$.*

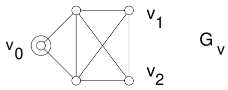

Figure 4.8: Gadget replacing a vertex of degree one.

4.6.2 MAXINDMATCH

Let MIS denote the problem of finding a largest independent set in a graph (problem GT20 in [GJ79]). Appellations (δ, Δ)-MIS and d-MIS are defined in the obvious way. There is [KM] a very simple L-reduction from MIS to MAXINDMATCH with parameters $\alpha = \beta = 1$. Given a graph $G = (V, E)$, define $t_1(G) = (V', E')$ as follows:

$$V' = V \cup \{v' : v \in V\}, \quad E' = E \cup \{\{v, v'\} : v \in V\}.$$

If U is an independent set in G then $F = \{\{v, v'\} : v \in U\}$ is an induced matching in $t_1(G)$. Conversely if F is an induced matching in $t_1(G)$ the set $t_2(t_1(G), F)$ obtained by picking one endpoint from every edge in F is an independent set in G. Therefore the size of a largest independent set in G is $\nu_I(t_1(G))$.

The s-padding of a graph was introduced in Section 4.3.1. The key property of the s-paddings with respect to the induced matchings in the original graph is that they preserve the distances between two vertices. If $G = (V, E)$ is a graph then for every $u, v \in V(G)$, $dst_G(u, v)$ is the *distance* between u and v, defined as the number of edges in a shortest path between u and v.

Lemma 36 *For all graphs G and for every $s \geq 2$, $dst_G(u, v) = dst_{G_s}(u_i, v_j)$ for all $u, v \in V(G)$ with $u \neq v$ and all $i, j \in \{1, 2, \ldots, s\}$.*

Lemma 37 *For all graphs G and for every $s \geq 2$, $\nu_I(G) = \nu_I(G_s)$.*

Proof. Let M be an induced matching in G. Define $M_s = \{\{u_1, v_1\} \in E(G_s) : \{u, v\} \in M\}$. By Lemma 36 all edges in M_s are at distance at least two. Conversely if M_s is an induced matching in G_s define $M = \{\{u, v\} \in E(G) : \{u_i, v_j\} \in M_s \text{ for some } i, j \in \{1, \ldots, s\}\}$. M is an induced matching in G. □

The following Lemmas show how to remove vertices of degree one and two from a $(1, \Delta)$-graph.

Lemma 38 *Any $(1, \Delta)$-graph G can be transformed in polynomial time into a $(2, \Delta)$-graph G' such that $|V_1(G)| = \nu_I(G') - \nu_I(G)$.*

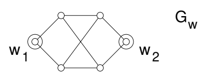

Figure 4.9: Gadget replacing a vertex of degree two.

Proof. Given a $(1, \Delta)$-graph G, the graph G' is obtained by replacing each vertex v of degree one in G by the gadget G_v shown in Figure 4.8. The edge $\{v, w\}$ incident to v is attached to v_0. The resulting graph has minimum degree two and maximum degree Δ. If M is an induced matching in G it is easy to build an induced matching in G' of size $|M| + |V_1(G)|$. Conversely every induced matching M' in G' will contain exactly one edge from every gadget G_i. Replacing (if necessary) each of these edges by the edge $\{v_1, v_2\}$ could only result in a larger matching. The matching obtained by forgetting the gadget-edges is an induced matching in G and its size is (at least) $|M'| - |V_1(G)|$. $\square$

Lemma 39 *Any* $(2, \Delta)$*-graph G can be transformed in polynomial time into a* $(3, \Delta)$*-graph* G' *such that* $|V_2(G)| = \nu_I(G') - \nu_I(G)$.

Proof. Let G be a $(2, \Delta)$-graph. Every vertex w of degree two is replaced by the graph G_w in Figure 4.9. The two edges $\{u, w\}$ and $\{v, w\}$ adjacent to w are replaced by edges $\{u, w_1\}$ and $\{v, w_2\}$. Let G' be the resulting $(3, \Delta)$-graph. If M is a maximal induced matching in G, a matching M' in G' is obtained by taking all edges in M and adding one edge from each of the graphs G_w. Figure 4.10 shows all the relevant cases. If $w \in V(M)$ then without loss of generality we can assume that $w_1 \in V(M')$ and one of the two edges adjacent to w_2 can be added to M'. If $w \notin V(M)$ then any of the four central edges in G_w can be added to M'. After these replacements no vertex in the original graph gets any closer to an edge in the matching. Inequality $\nu_I(G') \geq \nu_I(G) + |V_2(G)|$ follows from the argument above applied to a maximum induced matching in G.

Conversely for any induced matching M' in G' at most one edge from each copy of G_w belongs to M'. The copies of G_w with $M' \cap E(G_w) = \emptyset$ are called *empty*, all others are called *full*. Inequality $\nu_I(G) \geq \nu_I(G') - |V_2(G)|$ is proved by the following claims applied to a maximum

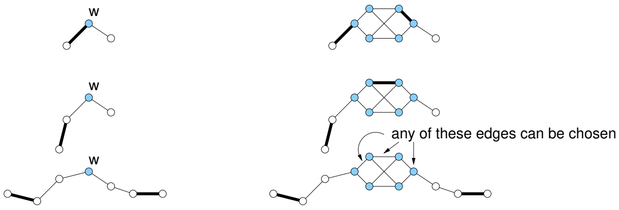

Figure 4.10: Possible ways to define the matching in G' given the one in G.

Figure 4.11: Filling an empty gadget, normal cases.

induced matching in G'.

Claim 3 *Any maximal induced matching M' in G' can be transformed into another induced matching M'' in G' with $|M'| \leq |M''|$ and such that all gadgets in M'' are full.*

Claim 4 $M =_{df} M'' \cap E(G)$ *is an induced matching in G.*

To prove the first claim, an algorithm is described which, given an induced matching $M' \subseteq E(G')$, fills all empty gadgets in M'. The algorithm visits in turn all gadgets in G' that have been created by the reduction and performs the following steps:

(1) If the gadget G_w under consideration is empty some local replacements are performed that fill G_w.

(2) The gadget G_w is then marked as "checked".

(3) A *maximality restoration* phase is performed in which, as a consequence of the local replacements in Step (1), some edges might be added to the induced matching.

Initially all gadgets are "unchecked". Let G_w be an unchecked gadget. If G_w is full the algorithm simply marks it as checked and carries on to the next gadget. Otherwise, since M' is maximal, at least one of the two edges adjacent to vertices w_1 and w_2 must be in M' for otherwise it would be possible to extend M' by picking any of the four central edges in G_w. Without loss of generality let $\{u, w_1\} \in M'$. Figure 4.11 shows (up to reflection symmetries) all possible cases. If vertex v does not belong to another gadget then either of the configurations on the left of Figure 4.11 is replaced by the one shown on the right. If v is part of another gadget few subcases need to be considered. Figure 4.12 shows all possible cases and the replacement rule. In all cases after the replacement the neighbouring gadget is marked as checked. Notice that all replacement rules do not decrease the size of the induced matching. Also as the process goes by, new edges in $E(G)$ can only be added to the current matching during the maximality restoration phase. To prove the second claim, assume by contradiction that two edges $e = \{u, v\}$ and $f = \{w, y\}$ in M are at distance one. Notice that $dst_{G'}(e, f) = dst_G(e, f)$ unless all the shortest paths between them contain a vertex of degree two.

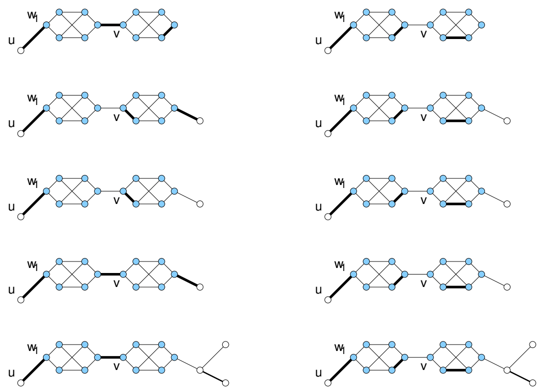

Figure 4.12: Filling an empty gadget, special cases.

The existence of e and f is contradicted by the fact that M' and M'' are induced matchings in G' and all gadgets in G' are filled by M''. □

The non-approximability of $(ks, (\Delta+1)s)$-MAXINDMATCH (for $k = 1, 2, 3$) follows from Theorem 63 applied to known results on independent set [AFWZ95, BK99].

Theorem 65 *Let* $h(\delta, \Delta, c) = \frac{[f(\delta,\Delta)+1+\lfloor \delta/3 \rfloor]c}{f(\delta,\Delta)c+1}$. *Define*

$$\begin{aligned} g(0, \Delta, c) &= c \\ g(i, \Delta, c) &= h(i, \Delta, g(i-1, \Delta, c)) \qquad i \geq 1 \end{aligned}$$

For every $\Delta \geq 3$*, let* c_Δ *be a constant such that it is NP-hard to approximate* $(1, \Delta)$-MIS *with ratio* c_Δ*. Then for* $k = 1, 2, 3$ *and every integer* $s > 0$ *it is NP-hard to approximate* $(ks, (\Delta+1)s)$-MAXINDMATCH *with ratio* $g(k-1, \Delta+1, c_\Delta)$.

Proof. The result for $k = 1$ follows from the L-reduction at the beginning of the section for $s = 1$ and a further L-reduction based on s-paddings for $s \geq 2$. For $k \in \{2, 3\}$ If G has minimum degree $k-1$, Theorem 61 implies $\nu_I(G) \geq |V_{k-1}(G)|/f(k-1, \Delta)$. The result follows using these bounds along with the reductions in Lemma 38 and 39. □

Theorem 66 *Let* c_0 *be a constant such that* 3-MIS *is NP-hard to approximate with ratio* c_0*. Then for every integer* $s > 0$ *it is NP-hard to approximate* $(3s, 4s)$-MAXINDMATCH *with ratio* $\frac{48c_0}{38c_0+5}$.

Proof. The reduction at the beginning of Section 4.6.2 and Theorem 63 imply that it is NP-hard to approximate $(1, 4)$-MAXINDMATCH with ratio c_0. If the original cubic graph G has n vertices, then $t_1(G)$ has $|V_1(t_1(G))| = |V_4(t_1(G))| = n$, no vertex of degree two or three, $5n/2$ edges and the maximum number of edges at distance at most one from a given edge is 19. We call one such graph a *special* $(1, 4)$-graph.

Claim 5 *There is an L-reduction from* $(1,4)$-MAXINDMATCH *restricted to special* $(1,4)$*-graphs to* $(3,4)$-MAXINDMATCH *with parameters* $\alpha = \frac{43}{5}$ *and* $\beta = 1$.

If G is a $(1,4)$-graph with $|V_2(G)| = |V_3(G)| = 0$, then replacing each vertex v of degree one with the gadget in Figure 4.8, gives a $(3,4)$-graph G'. The properties of special $(1,4)$-graphs and the same argument used to prove Theorem 61 imply $\nu_I(G) \geq \frac{5}{38}|V_1(G)|$. Therefore

$$\nu_I(G') = \nu_I(G) + |V_1(G)| \leq \nu_I(G) + \frac{38}{5}\nu_I(G) = \frac{43}{5}\nu_I(G)$$

Also, for every matching M' in G', define $t_2(G', M')$ as described in Lemma 38. It follows that $\nu_I(G) - |t_2(G', M')| \leq \nu_I(G') - |M'|$ and the claim is proved.

Therefore, by Theorem 63, $(3,4)$-MAXINDMATCH is hard to approximate with ratio $c_1 = \frac{43c_0}{38c_0+5}$. The *special* $(3,4)$-graphs H generated by the last reduction have again a lot of structure. In particular, $|V_3(H)| = |V_4(H)|$, $|E(H)| = 7|V_3(H)|/2$ and again the maximum number of edges at distance at most one from a given edge is 23. □

4.7 Small Maximal Matchings in Random Graphs

The previous sections described a few positive and negative results concerning the complexity and algorithmic approximability of the parameters $\beta(G)$ and $\nu_I(G)$. In this Section we look at the combinatorial and algorithmic properties of $\beta(G)$ when G is a random graph or a random bipartite graph. (A similar analysis for $\nu_I(G)$ is in Section 4.8).

More specifically, in Section 4.7.1 the size of the smallest maximal matchings in random graphs generated according to $\mathcal{G}_{n,p}$ are analysed and a number of improved approximation results are proved. Two regions of values for the parameter p are considered. If p is a constant it is proved that, with high probability, $\frac{n}{2} - \frac{\log n}{\log(1/(1-p))} \leq \beta(G) \leq \frac{n}{2} - \frac{\log n}{2\log(1/(1-p))}$. In particular the upper bound is algorithmic in the sense that there is an algorithm that returns a maximal matching of size at most $\frac{n}{2} - \frac{\log n}{2\log(1/(1-p))}$ in a graph $G \in \mathcal{G}_{n,p}$. The results build up on a relationship between maximal matchings and independent sets which maybe useful on its own. If $p = c/n$ with c constant, on average, the graph is much sparser. Again it is possible to exploit the relationship between maximal matchings and independent sets, but the known results for the latter problem are much weaker than in the dense case. Using them we are only able to prove lower bounds on $\beta(G)$. For small values of c the simplest first moment method actually gives better results.

In Section 4.7.2 a similar investigation is performed under the assumption that G is a random bipartite graph belonging to $\mathcal{G}(K_{n,n},p)$. The notion of *split independent set* is introduced, that plays a role similar to that of independent set for general graphs. The analysis is completed for the dense case: it is proved that, with high probability, $n - \frac{2\log n}{\log(1/(1-p))} \leq \beta(G) \leq n - \frac{\log n}{2\log(1/(1-p))}$, if G belongs to $\mathcal{G}(K_{n,n},p)$, with p constant.

It should be remarked that some of the results in these sections can be interpreted as phase transition phenomena. If n and p are fixed and the size (i.e. the cardinality) of the solution is varied, the proportion of graphs containing a solution of that specific size experiences a dramatic change in value. For instance, Theorem 68 and 69 on small maximal matchings imply that, on dense random graphs, as the size of the required matching is increased between two functions in the class $n/2 - \Theta(\log n)$ the proportion of graphs containing a matching of that given size goes from almost all to almost none.

4.7.1 General Graphs

Let $X = X_{p,k}(G)$ be the random variable counting the number of maximal matchings of size k in $G \in \mathcal{G}(n,p)$. Also, let $Z = Z_{p,n-2k}$ be the random variable counting independent sets of size $n-2k$ and let $Y = Y_{p,k}$ be the random variable counting perfect matchings in $H \in \mathcal{G}_{2k,p}$

Theorem 67 *If $G \in \mathcal{G}(n,p)$ then*

1. $\mathrm{E}(X) = \binom{n}{2k}\frac{(2k)!}{k!}\left(\frac{p}{2}\right)^k q^{\binom{n-2k}{2}}$.

2. $\mathrm{E}(X) = \mathrm{E}(Z)\cdot\mathrm{E}(Y)$.

Proof. Let M_i be a set of k independent edges, assume that G is a random graph sampled according to the model $\mathcal{G}(n,p)$ and let $X^i_{p,k}$ be the random indicator equal to one if M_i is a maximal matching in G. $\mathrm{E}(X^i_{p,k}) = \Pr[X^i_{p,k}] = p^k q^{\binom{n-2k}{2}}$. Then by linearity of expectation

$$\mathrm{E}(X) = \sum_{|M_i|=k} \mathrm{E}(X^i_{p,k}) = |\{M_i : |M_i| = k\}| \cdot p^k q^{\binom{n-2k}{2}}$$

The number of matchings of size k is equal to the possible ways of choosing $2k$ vertices out of n times the number of ways of connecting them by k independent edges divided by the number of orderings of these chosen edges:

$$\mathrm{E}(X) = \binom{n}{2k}\frac{(2k)!}{k!}\left(\frac{p}{2}\right)^k q^{\binom{n-2k}{2}}.$$

The second result is true since

$$\mathrm{E}(Z) = \binom{n}{n-2k} q^{\binom{n-2k}{2}} \qquad\qquad \mathrm{E}(Y) = \frac{(2k)!}{k!} \left(\frac{p}{2}\right)^k$$

□

If p is a constant, a natural way to lower bound the size of the smallest maximal matching in a random graph would be to get an upper bound on the expectation of X and then to use the Markov inequality (as we did in Chapter 3). Assuming $2k = n - 2\omega$

$$\mathrm{E}(X) = \frac{n!}{k!\,(n-2k)!} \left(\frac{p}{2}\right)^k q^{\binom{n-2k}{2}} \leq \frac{n^{\frac{n}{2}-\omega}}{(2\omega)!} \left(\frac{p}{2}\right)^{\frac{n}{2}-\omega} q^{2\omega^2-\omega} \leq \left(\frac{pn}{2}\right)^{\frac{n}{2}} \left(\frac{n\mathrm{e}}{pq\omega}\right)^{\omega} q^{2\omega^2}$$

and this goes to zero only if $\omega = \Omega(\sqrt{n})$. However a different argument gives considerably better lower bounds. Following Theorem 67.*2* it is possible to use the natural relationship between maximal matchings and independent sets. If M is a maximal matching in G then $V \setminus V(M)$ is an independent set.

Theorem 68 *$\beta(G) > \frac{n}{2} - \frac{\log n}{\log(1/q)}$ for almost all graphs in $\mathcal{G}(n,p)$ with p constant.*

Proof. Let $Z_{p,2\omega}$ be the random variable counting independent sets of size $2\omega = \frac{2\log n}{\log(1/q)}$ in a random graph G.

$$\begin{aligned}
\Pr[X_{p,\frac{n}{2}-\omega} > 0] &= \Pr[X_{p,\frac{n}{2}-\omega} > 0 \mid Z_{p,2\omega} > 0]\Pr[Z_{p,2\omega} > 0] + \\
&\qquad \Pr[X_{p,\frac{n}{2}-\omega} > 0 \mid Z_{p,2\omega} = 0]\Pr[Z_{p,2\omega} = 0] \\
&\leq \Pr[X_{p,\frac{n}{2}-\omega} > 0 \mid Z_{p,2\omega} > 0]\Pr[Z_{p,2\omega} > 0] + 0 \cdot 1 \\
&\leq \Pr[Z_{p,2\omega} > 0] \to 0
\end{aligned}$$

The last asymptotic result follows from a theorem in [GM75] concerning the independence number of dense random graphs. This implies $\beta(G) > \frac{n}{2} - \frac{\log n}{\log 1/q}$ for almost all graphs $G \in \mathcal{G}(n,p)$. □

The argument before Theorem 68 based on upper bounding $\mathrm{E}(X)$ is weak because even if $\mathrm{E}(Z_{p,2\omega})$ is small $\mathrm{E}(X_{p,\frac{n}{2}-\omega})$ might be very large. The random graph G might have very few independent sets of size 2ω (perhaps, only one) but a large number of maximal matchings of size $\frac{n}{2} - \omega$ (corresponding to the different possible ways of selecting the edges forming the matching).

Results in [GM75] also have algorithmic consequences. Let $\alpha_g(G)$ be the size of the independent set returned by the simplest greedy heuristic which repeatedly places a vertex v in the independent set I (as long as $E(G[I \cup \{v\}]) = \emptyset$) and removes $\{v\} \cup N(v)$ from G. It is easily proved that $\alpha_g(G) \sim \frac{\log n}{\log(1/q)}$.

Theorem 69 $\beta(G) < \frac{n}{2} - \frac{\log n}{2\log(1/q)}$ *for almost all graphs in $\mathcal{G}(n,p)$ with p constant.*

Proof. Let $\mathcal{IS}$ be an algorithm that first finds a maximal independent set I in G using Grimmett and McDiarmid's heuristic and then (attempts to) find a perfect matching in the remaining graph. With probability going to one $|I| \geq (1-\delta)\frac{\log n}{\log(1/q)}$ for all $\delta > 0$. Also, Grimmett and McDiarmid's algorithm does not use any information about $G - I$. Hence $G - I$ is a completely random graph on about $n - |I|$ vertices, each edge in it being chosen with constant probability p. Results in [ER66] imply that almost all such graphs contain a matching leaving at most one vertex unmatched. □

Independent sets are useful also for sparse graphs. If $p = \frac{c}{n}$ a lower bound on $\beta(G)$ can be obtained again by studying $\alpha(G)$.

Theorem 70 $\beta(G) > \frac{n}{2} - \frac{n\log c}{c}$ *for almost all graphs $G \in \mathcal{G}_{n,c/n}$ for $c > 2.27$.*

Proof. $\alpha(G) < \frac{2n\log c}{c}$ for almost all graphs $G \in \mathcal{G}_{n,c/n}$ for $c > 2.27$ (See Theorem XI.22 from Bollobás' book [Bol85]). The result follows by an argument similar to that of Theorem 68 □

In the dense case knowing the exact value of $\mathrm{E}(X)$ does not allow us to improve the lower bound on $\beta(G)$ given by the independent set analysis, since $\mathrm{E}(Y)$ is always large if p is constant and $k \sim \log n$. If $p = \frac{c}{n}$ for c sufficiently small, $\mathrm{E}(Y)$ is small. This implies that the exact expression for $\mathrm{E}(X)$ in Theorem 67 gives some improved lower bounds on $\beta(G)$ for sufficiently sparse graphs. Roughly if c is sufficiently small and U is a large independent set in G then $G[V \setminus U]$ very rarely contains a perfect matching.

Theorem 71 *Let d be a positive constant larger than one. Then $\beta(G) > n/2d$ for almost all graphs $G \in \mathcal{G}(n, c/n)$.*

Proof. By Theorem 67 $\mathrm{E}(X) = \frac{n!}{(n-2k)!\ k!}(p/2)^k(1-p)^{\binom{n-2k}{2}}$. Using Stirling approximation for the factorials involved in this expression we have

$$\mathrm{E}(X) \sim o(1)\left(\frac{n}{n-2k}\right)^n \left[\frac{p(n-2k)^2}{2k\mathrm{e}}\right]^k (1-p)^{\frac{(n-2k)^2}{2}}$$

Since $p = c/n$, it follows that

$$\mathrm{E}(X) \sim o(1)\left(\frac{n}{n-2k}\right)^n \left[\frac{c(n-2k)^2}{2kn\mathrm{e}}\right]^k \mathrm{e}^{-\frac{c(n-2k)^2}{2n}}$$

Now setting $k = n/2d$ we have

$$\mathrm{E}(X) \sim o(1)\left(\frac{d}{d-1}\right)^n \left[\left(\frac{d-1}{d}\right)^2 \frac{cd}{\mathrm{e}}\right]^{\frac{n}{2d}} \mathrm{e}^{-\frac{cn(d-1)^2}{2d^2}}$$

Taking the logarithm of the right hand side, we have that $\mathrm{E}(X) \to 0$ if

$$2d(d-1)\log\frac{d}{d-1} + d\log\frac{dc}{\mathrm{e}} - c(d-1)^2 < 0$$

and this inequality is satisfied for every c larger than a certain $c(d) > 0$. □

4.7.2 Bipartite Graphs

The analysis in the last section can be adapted to the case when G is a bipartite graph. Again $\beta(G)$ is closely related to another graph parameter whose asymptotic behaviour, at least on dense random graphs, can be estimate rather well.

Definition 23 *Given a bipartite graph $G = (V_1, V_2, E)$ with $|V_1| = |V_2| = n$ a* split *independent set in G is a set of vertices S with $|S| = 2\omega$, $|S \cap V_i| = \omega$ and $E(G[S]) = \emptyset$. Let $\sigma(G)$ be the size of a largest split independent set in G.*

One interesting property of split independent sets is that they "control" the existence of small maximal matching in bipartite graphs in the same way as independent sets control the maximal matchings in general graphs. If M is a maximal matching in a bipartite graph G then $V \setminus V(M)$ is a split independent set. Let X be the random variable counting maximal matchings in a random bipartite graph; let $Z = Z_{p,n-k}$ be the random variable counting split independent sets of size $n-k$ and $Y = Y_{p,k}$ the random variable counting perfect matchings in $H \in \mathcal{G}(K_{k,k}, p)$

Theorem 72 *If $G \in \mathcal{G}(K_{n,n}, p)$ then*

1. $\mathrm{E}(X) = \binom{n}{k}^2 k! p^k q^{(n-k)^2}$.

2. $\mathrm{E}(X) = \mathrm{E}(Z) \cdot \mathrm{E}(Y)$.

Proof. Let M_i be a set of k independent edges and $G \in \mathcal{G}(K_{n,n}, p)$ and let $X^i_{p,k}$ be the random indicator equal to one if M_i is a maximal matching in G. $\mathrm{E}(X^i_{p,k}) = \Pr[X^i_{p,k}] = p^k q^{(n-k)^2}$. Then

$$\mathrm{E}(X_{p,k}) = \sum_{|M_i|=k} \mathrm{E}(X^i_{p,k}) = |\{M_i : |M_i| = k\}| \cdot p^k q^{(n-k)^2}$$

The number of matchings of size k is given by the possible ways of choosing k vertices out of n on each side times the number of permutations on k elements. The result follows. □

If p is constant, it is fairly easy to bound the first two moments of Z and get good estimates on the value of $\sigma(G)$.

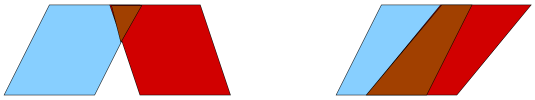

Figure 4.13: Possible relationships between pairs of split independent sets

Theorem 73 *$\sigma(G) \sim \frac{4 \log n}{\log 1/q}$ for almost all graphs in $\mathcal{G}(K_{n,n}, p)$ with p constant.*

Proof. The expected number of split independent sets of size 2ω is $\binom{n}{\omega}^2 q^{\omega^2}$. Hence, if $2\omega = 2\left\lceil \frac{2\log n}{\log 1/q} \right\rceil$

$$\Pr[Z > 0] < \binom{n}{\omega}^2 q^{\omega^2} \leq \left(\frac{n^\omega}{\omega!}\right)^2 q^{\omega^2} = \frac{1}{(\omega!)^2} \exp\left\{2\omega \log n - \omega^2 \log \frac{1}{q}\right\}$$

and the bound on the right goes to zero if $\omega > \frac{2\log n}{\log 1/q}$.

Let $2\omega = 2\left\lfloor \frac{2(1-\epsilon)\log n}{\log 1/q} \right\rfloor$ for any $\epsilon > 0$. The event "$Z = 0$" is equivalent to "$\sigma(G) < 2\omega$" because if there is no split independent set of size 2ω then the largest of such sets can only have less than 2ω elements. By Chebyshev inequality $\Pr[Z = 0] \leq \mathrm{Var}(Z)/\mathrm{E}(Z)^2$. Also $\mathrm{Var}(Z) = \mathrm{E}(Z^2) - \mathrm{E}(Z)^2$. There are $s_\omega = \binom{n}{\omega}^2$ ways of choosing ω vertices from two disjoint sets of n vertices. If Z^i is the random indicator set to one if S^i is a split independent set in G then $Z = \sum Z^i$ and then $\mathrm{E}(Z^2) = \sum_{i,j} \Pr[Z^i \wedge Z^j]$ where the sum is over all $i, j \in \{1, \ldots, s_\omega\}$. From the definition of conditional probability $\Pr[Z^i \wedge Z^j] = \Pr[Z^i \mid Z^j]\Pr[Z^j]$. Hence $\mathrm{E}(Z^2) = \sum_j \Pr[Z^j] \sum_i \Pr[Z^i \mid Z^j]$. Finally by symmetry $\Pr[Z^i \mid Z^j]$ does not actually depend on j but only on the amount of intersection between S^i and S^j. Thus, letting $S^1 = \{1, \ldots, 2\omega\}$, $\mathrm{E}(Z^2) = \left(\sum_j \Pr[Z^j]\right)\left(\sum_i \Pr[Z^i \mid Z^1]\right) = \mathrm{E}(Z) \cdot \mathrm{E}(Z|Z^1)$. Thus to prove that $\Pr[Z = 0]$ converges to zero it is enough to show that the ratio $\mathrm{E}(Z|Z^1)/\mathrm{E}(Z)$ converges to one.

$$\Pr[Z = 0] = \Pr\left[\sigma(G) < \left\lfloor \frac{4(1-\epsilon)\log n}{\log 1/q} \right\rfloor\right] \leq \frac{\mathrm{E}(Z|Z^1)}{\mathrm{E}(Z)} - 1$$

(this is assuming $\mathrm{E}(Z|Z^1) > \mathrm{E}(Z)$). Now

$$\mathrm{E}(Z|Z^1) = \sum_{0 \leq l_1, l_2 \leq \omega} \binom{\omega}{l_1}\binom{\omega}{l_2}\binom{n-\omega}{\omega-l_1}\binom{n-\omega}{\omega-l_2} q^{\omega^2 - l_1 l_2}$$

Define T_{ij} (generic term in $\mathrm{E}(Z|Z^1)/\mathrm{E}(Z)$) by

$$T_{ij}\binom{n}{\omega}^2 = \binom{w}{i}\binom{w}{j}\binom{n-\omega}{\omega-i}\binom{n-\omega}{\omega-j} q^{-ij}$$

Claim 6 $T_{00} \leq 1 - \frac{2\omega^2}{n-\omega+1} + \frac{\omega^3(2\omega-1)}{(n-\omega+1)^2}$.

To see this write

$$\begin{aligned}
T_{00} &= \binom{n-\omega}{\omega}^2 \binom{n}{\omega}^{-2} \\
&= \left[\frac{(n-\omega)\cdot(n-\omega-1)\cdot\ldots\cdot(n-2\omega+1)}{n\cdot(n-1)\cdot\ldots\cdot(n-\omega+1)}\right]^2 \\
&= \left[\left(1-\frac{\omega}{n}\right)\cdot\left(1-\frac{\omega}{n-1}\right)\cdot\ldots\cdot\left(1-\frac{\omega}{n-\omega+1}\right)\right]^2
\end{aligned}$$

From this

$$\left(1-\frac{\omega}{n}\right)^{2\omega} \leq T_{00} \leq \left(1-\frac{\omega}{n-\omega+1}\right)^{2\omega}$$

In particular

$$\begin{aligned}
T_{00} &\leq \left(1-\frac{\omega}{n-\omega+1}\right)^{2\omega} \\
&= 1-\frac{2\omega^2}{n-\omega+1}+\binom{2\omega}{2}\left(\frac{\omega}{n-\omega+1}\right)^2(1-\xi)^{2\omega-3} \\
&< 1-\frac{2\omega^2}{n-\omega+1}+\frac{\omega^3(2\omega-1)}{(n-\omega+1)^2}
\end{aligned}$$

Claim 7 *$T_{ij} \leq \frac{\omega^2}{n-\omega+1}$ for $i+j=1$ and for sufficiently large n.*

The truth of this follows from Claim 6 since

$$T_{ij} = \omega\binom{n-\omega}{\omega-1}\binom{n-\omega}{\omega}\binom{n}{\omega}^{-2} = \frac{\omega^2 T_{00}}{n-\omega+1} \leq \frac{\omega^2}{n-\omega+1}$$

Claim 8 *$T_{ij} \leq T_{10}$ for all $i,j \in \{1,\ldots,\omega\}$ and for sufficiently large n.*

The ratio T_{ij}/T_{10} is bounded above as follows

$$\begin{aligned}
\frac{T_{ij}}{T_{10}} &= \frac{\omega!}{i!\,\omega-i!}\cdot\frac{\omega!}{j!\,\omega-j!}\cdot\frac{n-\omega!}{\omega-i!\,n-2\omega+i!}\cdot\frac{n-\omega!}{\omega-j!\,n-2\omega+j!}\cdot \\
&\quad\cdot\frac{\omega-1!\,n-2\omega+1!}{n-\omega!}\cdot\frac{\omega!\,n-2\omega!}{n-\omega!}\cdot\frac{q^{-ij}}{\omega} \\
&= \frac{(\omega!)^4}{(\omega-i!)^2(\omega-j!)^2}\cdot\frac{n-2\omega!}{n-2\omega+i!}\cdot\frac{n-2\omega+1!}{n-2\omega+j!}\cdot\frac{q^{-ij}}{\omega^2\,i!\,j!} \\
&\leq (\omega)^{2(i+j)-2}\cdot(n-2\omega)^{1-i-j}\cdot q^{-ij} \\
&= \left(\frac{\omega^2 q^{-\frac{ij}{i+j-1}}}{n-2\omega}\right)^{i+j-1}
\end{aligned}$$

The function $\frac{ij}{i+j-1} \leq \frac{\omega^2}{2\omega-1}$ for all $i,j \in \{1,\ldots,\omega\}$. Moreover $\frac{\omega^2}{2\omega-1} \leq \frac{\omega}{2}+1$ for all $\omega \geq 1$. Hence

$$\frac{T_{ij}}{T_{10}} \leq \left(\frac{\omega^2 q^{-\frac{\omega}{2}-1}}{n-2\omega}\right)^{i+j-1}$$

From the claims above

$$\begin{aligned}
\Pr\left[\sigma < 2\left\lfloor\frac{2(1-\epsilon)\log n}{\log 1/q}\right\rfloor\right] &\leq \\
&\leq\ T_{00} + T_{10} + T_{01} + \omega^2 T_{10} - 1 \\
&\leq\ 1 - \frac{2\omega^2}{n-\omega+1} + \frac{\omega^3(2\omega-1)}{(n-\omega+1)^2} + \frac{2\omega^2}{n-\omega+1} + \frac{\omega^4}{n-\omega+1} - 1 \\
&\leq\ \frac{\omega^3(2\omega-1)}{(n-\omega+1)^2} + \frac{\omega^4}{n-\omega+1}
\end{aligned}$$

□

Theorem 74 $\beta(G) > n - \frac{2\log n}{\log 1/q}$ *for almost all graphs in* $\mathcal{G}(K_{n,n}, p)$ *with p constant.*

The similarities between the properties of independent sets in random graphs and those of split independent sets in random bipartite graphs have some algorithmic implications. A rather simple greedy heuristic almost always produces a solution whose cardinality can be predicted quite tightly. Let I be the independent set to be output. Consider the process that visits the vertices of a random bipartite graph $G(V_1, V_2, E)$ in some fixed order. If $V_i = \{v_1^i, \ldots, v_n^i\}$, then the algorithm will look at the pair (v_j^1, v_j^2) during step j. If $\{v_j^1, v_j^2\} \notin E$ and if there is no edge between v_j^i and any of the vertices which are already in I then both v_j^1 and v_j^2 are inserted into I. Let $\sigma_g(G) = |I|$.

Theorem 75 $\sigma_g(G) \sim \frac{\log n}{\log 1/q}$ *for almost all graphs in* $\mathcal{G}(K_{n,n}, p)$ *with p constant.*

Proof. Suppose that $2(k-1)$ vertices are already in I. The algorithm above will add two vertices v_1 and v_2 as the kth pair if $\{v_1, v_2\} \notin E$ and there is no edge between either v_1 or v_2 and any of the vertices which are already in I. The two events are independent in the given model and their joint probability is

$$(1-p) \cdot (1-p)^{2(k-1)} = (1-p)^{2k-1}$$

(since the graph is bipartite, v_1, say, can only possibly be adjacent to $k-1$ of the vertices in I). Let X_k be the random variable equal to the number of pairs considered before the kth pair is added to I. From the previous discussion it is evident that X_k has geometric distribution with parameter $P_k = (1-p)^{2k-1}$. Moreover the variables $X_1, X_2, \ldots$ are all independent. Let $Y_\omega = \sum_{k=1}^{\omega} X_k$. The event "$Y_\omega < n$" is implied by "$\sigma_g(G) > 2\omega$": if the split independent set returned by the greedy algorithm contains more than 2ω vertices that means that the algorithm finds ω independent pairs in strictly less than n trials. Also if $Y_\omega < n$ then certainly each of the X_k cannot be larger than n (the opposite implication is obviously false). Hence

$$\Pr[Y_\omega < n] \leq \Pr[\cap_{k=1}^{\omega}\{X_k \leq n\}] = \prod_{k=1}^{\omega} \Pr[X_k \leq n] = \prod_{k=1}^{\omega}\{1 - [1-(1-p)^{2k-1}]^n\}$$

Let $\omega = \left\lceil \frac{(1+\epsilon)\log n}{2\log 1/q} \right\rceil$ and, given $\epsilon > 0$ and $r \in \mathbb{N}$, choose $m > r/\epsilon$. For sufficiently large n, $\omega - m > 0$. Hence

$$\Pr[Y_\omega < n] \le \prod_{k=\omega-m}^{\omega} \{1 - [1-(1-p)^{2k+1}]^n\} \le \{1-[1-(1-p)^{2(\omega-m)+1}]^n\}^m$$

Now, since $(1-x)^n \ge 1 - nx$,

$$\Pr[Y_\omega < n] \le \{n(1-p)^{2(\omega-m)+1}\}^m = (1-p)^{m-2m^2} n^m (1-p)^{2\omega m} = o(n^{-r})$$

The event "$Y_\omega > n$" is equivalent to "$\sigma_g(G) < 2\omega$". Let $\omega = \left\lfloor \frac{(1-\epsilon)\log n}{2\log 1/q} \right\rfloor$. If $Y_\omega > n$ then there must be at least one k for which $X_k > n/\omega$ (for otherwise $Y = \sum X_k$ would be strictly smaller than n). Hence

$$\Pr[Y_\omega > n] \le \Pr[\cup_{k=1}^{\omega}\{X_k > n/\omega\}] \le \sum_{k=1}^{\omega} \Pr[X_k > n/\omega] \le \omega[1-(1-p)^{2\omega-1}]^{\lfloor n/\omega \rfloor}$$

By the choice of ω,

$$(1-p)^{2\omega-1} > \frac{n^{-(1-\epsilon)}}{1-p}$$

Hence

$$\Pr[Y_\omega > n] \le \omega \left[1 - \frac{n^{-(1-\epsilon)}}{1-p}\right]^{\lfloor n/\omega \rfloor} \le \omega \ \exp\left\{-\frac{n^{-(1-\epsilon)}}{1-p}\left\lfloor \frac{n}{\omega} \right\rfloor\right\}$$

Since $\lfloor n/\omega \rfloor > n/\omega - 1$,

$$\Pr[Y_\omega > n] \le \omega \exp\left\{-\frac{n^{\epsilon}}{(1-p)\omega} - o(1)\right\}$$

and the result follows from the choice of ω. □

The greedy algorithm analysed in Theorem 75 is equivalent to the following algorithm which, at the same time, builds the split independent set in a random graph that is uncovered on the fly. Notice that at the end of the algorithm no choice has been made about the edges connecting pairs of vertices in $(V_1 \cup V_2) \setminus V(M)$.

Input: n, the order of V_1 and V_2.

(1) $I \leftarrow \emptyset$;

(2) $E \leftarrow \emptyset$;

(3) **repeat**

(4) Let v_i be the first vertex in V_i (for $i = 1, 2$);

(5) $N(v_1) \leftarrow \emptyset$;

$N(v_2) \leftarrow \emptyset$;
with probability p **do**
  $E(G) \leftarrow E(G) \cup \{v_1, v_2\}$;
  $N(v_1) \leftarrow \{v_2\}$;
  $N(v_2) \leftarrow \{v_1\}$;
if $(|V_1| = 1) \wedge (E(G) = \emptyset)$
  $I \leftarrow I \cup \{v_1, v_2\}$;
else
  (* let $V_i = \{v_i, u_{i,2}, \ldots, u_{i,|V_i|}\}$ (for $i = 1, 2$) *)
  for $i = 2$ **to** $|V_1| - 1$
    with probability p **do**
      $N(v_1) \leftarrow N(v_1) \cup \{u_{2,i}\}$;
      $E(G) \leftarrow E(G) \cup \{v_1, u_{2,i}\}$;
    with probability p **do**
      $N(v_2) \leftarrow N(v_2) \cup \{u_{1,i}\}$;
      $E(G) \leftarrow E(G) \cup \{v_2, u_{1,i}\}$;
  if $\{v_1, v_2\} \notin E(G)$
    $I \leftarrow I \cup \{v_1, v_2\}$;
$V_1 \leftarrow V_1 \setminus (\{v_1\} \cup N(v_1))$;
$V_2 \leftarrow V_2 \setminus (\{v_2\} \cup N(v_2))$;
until $V_1 = \emptyset$;

The advantage of this algorithm is that, after the split independent set has been found, the graph $G - I$ is still completely undefined. Thus this algorithm can be the first part of a bigger procedure that first finds a (large) split independent set in a random bipartite graph and then a large matching in the remaining random graph. The following result is thus the analogue of Theorem 69 for bipartite random graphs.

Theorem 76 *$\beta(G) < n - \frac{\log n}{2 \log 1/q}$ for almost all graphs in $\mathcal{G}(K_{n,n}, p)$ with p constant.*

4.8 Large Induced Matchings in Random Graphs

We conclude this chapter by looking at the properties of large induced matchings in dense and rather sparse random graphs. The model we use is $\mathcal{G}(n,p)$ and we will consider the case p constant and the case $p = c/\sqrt{n}$, with c constant. In the former case it is possible to prove that, with high probability, $\nu_I(G) \sim \frac{\log n}{\log(1/(1-p))}$. Furthermore there is an algorithm that, with high probability returns an induced matching of size asymptotic to $\frac{\log n}{4\log(1/(1-p))}$. In the case $p = c/\sqrt{n}$, with c constant we were only able to prove that $\frac{\log c}{3c}\sqrt{n} \leq \nu_I(G) \leq \frac{1}{2}\sqrt{n}\log n$.

Let $Y_k = Y_{p,k}(G)$ be the random variable counting the number of induced matchings of size k in $G \in \mathcal{G}(n,p)$.

Expectation. Let M_i be a set of k independent edges in $G \in \mathcal{G}(n,p)$. Define $Y^i_{p,k}$, the random indicator equal to one if M_i is an induced matching in G. The probability that this happens is p^k times the probability $(1-p)^{\binom{2k}{2}-k} = q^{2k(k-1)}$ that the subgraph induced by the vertices in $V(M_i)$ does not contain any other edge apart from those in M_i. Thus $\mathrm{E}(Y^i_{p,k}) = \Pr[Y^i_{p,k} = 1] = p^k q^{2k(k-1)}$. By linearity of expectation

$$\mathrm{E}(Y_k) = \binom{n}{2k}\frac{(2k)!}{k!}\cdot\left(\frac{p}{2}\right)^k q^{2k(k-1)}$$

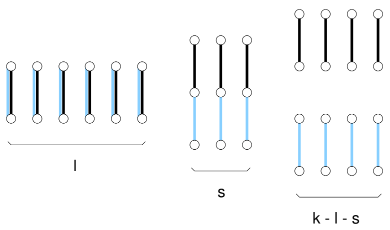

Figure 4.14: Dependence between pairs of induced matchings of size k

Variance. The main ingredient in the variance is $\mathrm{E}(Y_k|Y^i_{p,k})$ for some i: this is the expected number of induced matchings of size k given that M_i is an induced matching of size k in G. Figure 4.14 shows two matchings of the same size and the possible interactions between them. If M_i is given by the black lines, and M_j by the light ones we have

$$\Pr[Y^j_{p,k} \mid Y^i_{p,k}] = p^{k-l} q^{2(k-l)(k-l-1)+4l(k-l)-2ls}$$

There are $k-l$ "new" lines and $s \cdot 2l$ of the possible $\binom{2(k-l)}{2} - (k-l) + 2(k-l)2l$ lines which must not be in M_j have already be accounted for in dealing with the probability $\Pr[Y^i]$. There are $\binom{k}{l}$

ways of choosing l edges in M_i to be in M_j as well. Then there can be s vertices which are adjacent to both M_i and M_j, there are $\binom{2(k-l)}{s}$ ways of choosing them, then $\binom{n-2k}{s}$ ways of choosing the other s endpoints and $s!$ ways to connect them in a matching. Finally $k-l-s$ more independent edges can be chosen in K_{n-2k-s}. We have

$$\begin{aligned}
&\mathrm{E}(Y_k|Y^i_{p,k}) = \\
&= \sum_{l=0}^{k}\binom{k}{l}\sum_{s=0}^{k-l}\binom{2(k-l)}{s}\binom{n-2k}{s}s!\binom{n-2k-s}{2(k-l-s)}\cdot \\
&\qquad \frac{[2(k-l-s)]!}{2^{k-l-s}(k-l-s)!}p^{k-l}q^{2(k-l)(k-l-1)+4l(k-l)-2ls} \\
&= \sum_{l=0}^{k}\binom{k}{l}\left(\frac{p}{2}\right)^{k-l}q^{2(k^2-l^2-k+l)}\sum_{s=0}^{k-l}\binom{2(k-l)}{s}\frac{(n-2k)!}{(k-l-s)!\,(n-4k+2l+s)!}\left(\frac{2}{q^{2l}}\right)^{s}
\end{aligned}$$

Theorem 77 $\nu_I(G) \sim \frac{\log n}{\log 1/q}$ *for almost all graphs in $\mathcal{G}(n,p)$ with p constant.*

Proof. To prove $\nu_I(G) < \frac{\log n}{\log 1/q}$ we use again Markov inequality.

$$\begin{aligned}
\mathrm{E}(Y) &\leq \frac{n^{2k}}{k!}\left(\frac{p}{2}\right)^{k}(1-p)^{2k^2-2k} \\
&\leq \frac{1}{k!}\cdot\exp\left\{2k\left[\log n-(k-1)\log\frac{1}{1-p}\right]\right\}
\end{aligned}$$

and this expression goes to zero as n grows to infinity if $k > \frac{\log n}{\log 1/q}+1$.

Let $k = \left\lfloor\frac{(1-\delta)\log n}{\log 1/q}\right\rfloor$. For $l = 0,\dots,k$ and define $T_l(n)$ by

$$T_l(n)\binom{n}{2k}\frac{2k!}{k!} = \binom{k}{l}\left(\frac{p}{2}\right)^{-l}q^{-2(l^2-l)}\sum_{s=0}^{k-l}\binom{2(k-l)}{s}\frac{(n-2k)!}{(k-l-s)!\,(n-4k+2l+s)!}\left(\frac{2}{q^{2l}}\right)^{s}$$

so that $\frac{\mathrm{E}(Y_k|Y^i_{p,k})}{\mathrm{E}(Y_k)} = \sum_{l=0}^{k}T_l(n)$. The following sequence of claims is aimed at proving that $\sum_{l=0}^{k}T_l(n) \to 1$. Note that $\sum_{l=0}^{k}T_l(n) \geq 1$.

Claim 9 $T_0(n) \leq 1+4k^2/(n-2k)+O((k^2/(n-2k))^2)$.

To see the claim notice that

$$\begin{aligned}
T_0(n) &= \frac{(n-2k)!\,k!}{n!}\sum_{s=0}^{k}\frac{(2k)!}{s!\,(2k-s)!}\,\frac{(n-2k)!}{(k-s)!\,(n-4k+s)!}\,2^s \\
&= \frac{(n-2k)!}{n!}\sum_{s=0}^{k}\binom{k}{s}\frac{(2k)!}{(2k-s)!}\,\frac{(n-2k)!}{(n-4k+s)!}\,2^s \\
&\leq \frac{1}{(n-2k)^{2k}}\sum_{s=0}^{k}\binom{k}{s}(2k)^s(n-2k)^{2k-s}2^s = \left(1+\frac{4k}{n-2k}\right)^{k}
\end{aligned}$$

and the validity of the claim follows.

Claim 10 $T_1(n) = O((k/(n-2k))^2)$.

By simple algebraic manipulations $T_1(n)$ can be expressed in terms of a binomial sum.

$$\begin{aligned}
T_1(n) &= \frac{2k}{p}\frac{(n-2k)!\,k!}{n!}\sum_{s=0}^{k-1}\binom{2(k-1)}{s}\frac{(n-2k)!}{(k-s-1)!\,(n-4k+s+2)!}\left(\frac{2}{q^2}\right)^s \\
&= \frac{2k^2}{p}\frac{(n-2k)!}{n!}\sum_{s=0}^{k-1}\binom{k-1}{s}\frac{(n-2k)!}{(n-4k+s+2)!}\frac{2(k-1)!}{[2(k-1)-s]!}\left(\frac{2}{q^2}\right)^s \\
&\le \frac{2k^2}{p}\frac{1}{(n-2k)^{2k}}\sum_{s=0}^{k-1}\binom{k-1}{s}(n-2k)^{2k-s-2}[2(k-1)]^s\left(\frac{2}{q^2}\right)^s \\
&= \frac{2k^2}{p(n-2k)^2}\sum_{s=0}^{k-1}\binom{k-1}{s}\left(\frac{4(k-1)}{q^2(n-2k)}\right)^s = \frac{2k^2}{p(n-2k)^2}\left(1+\frac{4(k-1)}{q^2(n-2k)}\right)^{k-1}
\end{aligned}$$

Claim 11 $T_l(n) = o(T_1(n))$ *for* $l = 2, \ldots, k$.

It is fairly easy, if tedious, to express $T_{l-1}(n)$ in term of $T_l(n)$.

$$T_{l-1}(n)\binom{n}{2k}\frac{(2k)!}{k!} = \binom{k}{l-1}\left(\frac{2}{p}\right)^{l-1}\left(\frac{1}{q}\right)^{2[(l-1)^2-(l-1)]}\cdot$$
$$\cdot\sum_{s=0}^{k-l+1}\binom{2(k-l+1)}{s}\frac{(n-2k)!}{(k-l+1-s)!\,(n-4k+2l-2+s)!}\left(\frac{2}{q^{2l-2}}\right)^s$$

Since $\binom{k}{l-1} = (l/(k-l+1))\binom{k}{l}$ and $(l-1)^2 - l + 1 = l^2 - l - 2(l-1)$,

$$T_{l-1}(n)\binom{n}{2k}\frac{(2k)!}{k!} = \frac{lpq^{4(l-1)}}{2(k-l+1)}\binom{k}{l}\left(\frac{p}{2}\right)^{-l}q^{-2(l^2-l)}\times$$
$$\sum_{s=0}^{k-l+1}\binom{2(k-l+1)}{s}\frac{(n-2k)!}{(k-l+1-s)!\,(n-4k+2l-2+s)!}\left(\frac{2}{q^{2l-2}}\right)^s$$

Since $\binom{2(k-l+1)}{s} = \binom{2(k-l)}{s}\frac{2(k-l+1)}{2(k-l+1)-s}\frac{2(k-l)+1}{2(k-l)+1-s}$,

$$\begin{aligned}
T_{l-1}(n)\binom{n}{2k}\frac{(2k)!}{k!} &= \frac{lpq^{4(l-1)}}{2(k-l+1)}\binom{k}{l}\left(\frac{p}{2}\right)^{-l}q^{-2(l^2-l)}\cdot \\
&\cdot\sum_{s=0}^{k-l+1}\binom{2(k-l)}{s}\frac{2(k-l+1)}{2(k-l+1)-s}\frac{2(k-l)+1}{2(k-l)+1-s}\cdot \\
&\cdot\frac{(n-2k)!}{(k-l+1-s)!\,(n-4k+2l-2+s)!}\left(\frac{2}{q^{2l-2}}\right)^s \\
&= \frac{lpq^{4(l-1)}}{2(k-l+1)}\binom{k}{l}\left(\frac{p}{2}\right)^{-l}q^{-2(l^2-l)}\sum_{s=0}^{k-l+1}\binom{2(k-l)}{s}\frac{(n-2k)!}{(k-l-s)!\,(n-4k+2l+s)!}\times \\
&\frac{2(k-l+1)}{2(k-l+1)-s}\frac{2(k-l)+1}{2(k-l)+1-s}\frac{(n-4k+2l+s)(n-4k+2l-1+s)}{k-l+1-s}q^{2s}\left(\frac{2}{q^{2l}}\right)^s
\end{aligned}$$

The three fractions involving s are as small as possible when $s=0$. Also q^{2s} is always larger than q^{k-l+1}. Thus, taking everything that does not depend on s out of the sum,

$$T_{l-1}(n)\binom{n}{2k}\frac{(2k)!}{k!} \ge \frac{lpq^{2(k+l-1)}}{2(k-l+1)}\frac{(n-4k+2l)(n-4k+2l-1)}{k-l+1}\times$$

$$\binom{k}{l}\left(\frac{p}{2}\right)^{-l} q^{-2(l^2-l)} \sum_{s=0}^{k-l+1} \binom{2(k-l)}{s} \frac{(n-2k)!}{(k-l-s)!\,(n-4k+2l+s)!} \left(\frac{2}{q^{2l}}\right)^s$$
$$= \frac{lpq^{2(k+l-1)}}{(k-l+1)^2}\binom{n-4k+2l}{2} T_l(n)$$

and, since $1 < l \leq k$,

$$T_{l-1}(n) \geq \frac{lpq^{2k}(n-4k)^2}{2k^2} T_l(n)$$

Finally a simple induction on l shows that, for all $1 \leq l \leq k$,

$$T_l(n) \leq \frac{1}{l!}\left(\frac{2k^2}{q^{2k}p(n-4k)^2}\right)^{l-1} T_1(n)$$

Using the claims above it is now fairly easy to prove that almost always $G \in \mathcal{G}(n,p)$ has an induced matching of size $\left\lfloor \frac{(1-\delta)\log n}{\log 1/q} \right\rfloor$ for any $\delta > 0$. Applying exactly the same argument as in Theorem 73, to prove that $\Pr[Y > 0]$ almost always, we just need to show that the ratio $\mathrm{E}(Y|Y^1)/\mathrm{E}(Y) \to 1$. This follows from the claims above since $\mathrm{E}(Y|Y^1)/\mathrm{E}(Y) = \sum_{l=0}^{k} T_l(n)$. □

It turns out that it is possible to approximate the largest induced matching in a random graph by another variant of the greedy procedure. Pairs of vertices are visited in some (arbitrary but fixed) order and added to the induced matching if they are connected and they are not connected to any other edge in the matching. The following piece of code is one possible implementation.

program Greedy_induced_matching

Input: random $G = (V, E)$ with $V = \{v_1, \ldots, v_n\}$

(1) $M \leftarrow \emptyset$;

(2) **for** $i = 1$ **to** $\lfloor n/2 \rfloor$ **do**

(3) **if** $(\{v_{2i-1}, v_{2i}\} \in E$ and $\mathrm{cut}(\{v_{2i-1}, v_{2i}\}, V(M)) = \emptyset)$

(4) $M \leftarrow M \cup \{v_{2i-1}, v_{2i}\}$;

Let gr_I be the cardinality of the resulting matching.

Theorem 78 $\mathrm{gr}_I(G) \sim \frac{\log n}{4 \log 1/q}$ *for almost all graphs in* $\mathcal{G}_{n,p}$ *with p constant.*

Proof. The proof technique is identical to that of Theorem 75. If $k-1$ edges are in the matching already (let M_{k-1} denote the matching with $k-1$ edges) the pair $\{v_j, v_{j+1}\}$ is added to it if

(1) $\{v_j, v_{j+1}\} \in E$ and,

(2) $\text{cut}(\{v_j, v_{j+1}\}, V(M_{k-1})) = \emptyset$.

The probability that this happens is $P_k = p(1-p)^{4(k-1)}$. For $k = 1, 2, \ldots$, let X_k the geometric random variables equal to the number of pairs considered until the kth pair is added to the matching. Define $Z_l = \sum_{k=1}^{l} X_k$. Let ϵ be a positive real constant. If $l = \left\lceil \frac{(1+\epsilon)\log n}{4\log 1/q} \right\rceil$ and $m > r/\epsilon$ then

$$\begin{aligned}
\Pr[Z_l < n/2] &\leq \prod_{k=1}^{l} \{1 - [1 - p(1-p)^{4(k-1)}]^{n/2}\} \\
&\leq \{1 - [1 - p(1-p)^{4(l-m)}]^{n/2}\}^m \\
&\leq \left(\frac{n}{2}\right)^m p^m (1-p)^{4lm - 4m^2} \\
&= \left(\frac{p}{2(1-p)^{4m}}\right)^m n^m n^{-(1+\epsilon)m} = O(n^{-r})
\end{aligned}$$

If $l = \left\lfloor \frac{(1-\epsilon)\log n}{4\log 1/q} \right\rfloor$

$$\begin{aligned}
\Pr[Z_l > n/2] &\leq \sum_{k=1}^{l} \Pr[X_k > n/2l] \\
&\leq l[1 - p(1-p)^{4(k-1)}]^{\lfloor n/2l \rfloor} \\
&\leq l\left[1 - \frac{pn^{-(1-\epsilon)}}{(1-p)^4}\right]^{\lfloor n/2l \rfloor} \\
&\leq l \cdot \exp\left\{-\frac{pn^{-(1-\epsilon)}}{(1-p)^4}\left\lfloor \frac{n}{2l} \right\rfloor\right\}
\end{aligned}$$

□

Sparse Graphs. The performances of the algorithm "Greedy_induced_matching" described above can be analysed also on sparser graphs. Let $G \in \mathcal{G}_{n,c/\sqrt{n}}$. The following is a technical result needed in Theorem 79.

Lemma 40 *If* $p = \frac{c}{\sqrt{n}}$ *then* $\sum_{j=0}^{\infty} \frac{p^{j+1}}{j+1} \leq \frac{3p}{2}$ *for all* $n > 4c^2$.

Proof.

$$\begin{aligned}
\sum_{j=0}^{\infty} \frac{p^{j+1}}{j+1} &\leq p + \frac{1}{2}\sum_{i=2}^{\infty} p^i \\
&= p + \frac{p^2}{2}\sum_{i=2}^{\infty} p^{i-2} \\
&= p + \frac{p^2}{2}\sum_{t=0}^{\infty} p^t \\
&= p + \frac{p^2}{2}\frac{1}{1-p}
\end{aligned}$$

The result follows using the assumption on p. □

Theorem 79 $\mathrm{gr}_I(G) > \frac{\log c}{3c} \cdot \sqrt{n}$ *for almost all graphs in* $\mathcal{G}(n,p)$ *with* $p = c/\sqrt{n}$ *and c constant.*

Proof. At step i there are $i-1$ edges in M. The probability that algorithm Greedy_induced_matching executes step (4) is $P_i = p(1-p)^{4(i-1)}$. Let X_i, for $i = 1, 2, \ldots$ count the number of edges removed until the i-th edge is added to M. We have $\Pr[X_i = s] = P_i(1-P_i)^{s-1}$. So each of the X_i has geometric distribution and $\mathrm{E}(X_i) = 1/P_i$ and $\mathrm{Var}(X_i) = (1-P_i)/{P_i}^2$. Moreover the random variables X_i are all independent. Let $Z = \sum_{i=1}^{k} X_i$. We have $\Pr[Z > n] = \Pr[\mathrm{Gr}(G) < k]$. Since all the X_i are independent we can derive tight bounds on the expectation and the variance of Z and use Chebyshev inequality to complete the proof of the theorem. The geometric sum $\sum_{i=1}^{k} \alpha^i$ is at most $\frac{\alpha^{k+1}}{\alpha-1}$ for any $\alpha > 1$, therefore

$$\mathrm{E}(Z) = \frac{(1-p)}{p}^4 \sum_{i=1}^{k} [(1-p)^{-4}]^i \leq \frac{(1-p)^{-4k+4}}{p[1-(1-p)^4]}.$$

Since $1-(1-p)^4 = p(4-6p+4p^2-p^3) \geq p$,

$$\mathrm{E}(Z) \leq \frac{(1-p)^{4(1-k)}}{p^2}$$

If $p = c/\sqrt{n}$, using Lemma 40,

$$\mathrm{E}(Z) \leq \frac{n}{c^2}\,\mathrm{e}^{-4k\log(1-c/\sqrt{n})} \leq \frac{n}{c^2}\,\mathrm{e}^{\frac{6ck}{\sqrt{n}}}$$

If $k = \frac{\log c - \delta}{3c}\sqrt{n}$ then

$$\mathrm{E}(Z) \leq \frac{n}{\mathrm{e}^{2\delta}} \leq (1-\eta)n$$

for some fixed η. Similarly

$$\begin{aligned}
\mathrm{Var}(Z) &= \sum_{i=1}^{k} \frac{1-p(1-p)^{4(i-1)}}{p^2(1-p)^{8(i-1)}} \\
&\leq \frac{(1-p)}{p^2}^8 \sum_{i=1}^{k} [(1-p)^{-8}]^i \\
&\leq \frac{(1-p)}{p^3}^{-8k+8}
\end{aligned}$$

Using the assumptions on p and k,

$$\mathrm{Var}(Z) \leq \frac{n^{\frac{3}{2}}}{c^3}\,\mathrm{e}^{\frac{12ck}{\sqrt{n}} - \frac{8c}{\sqrt{n}}} = \frac{cn^{\frac{3}{2}}}{\mathrm{e}^{4\delta + \frac{8c}{\sqrt{n}}}}$$

By Chebyshev inequality, since $\mathrm{E}(Z) \leq n - \eta n$

$$\Pr[Z > n] \leq \Pr[Z \geq \mathrm{E}(Z) + \eta n] \leq \frac{\mathrm{Var}(Z)}{\eta^2 n^2} \leq \frac{c \cdot n^{-1/2}}{(\eta \mathrm{e}^{2\delta + \frac{4c}{\sqrt{n}}})^2}$$

$\square$

The result in Theorem 79 is complemented by the following Theorem. As above let $Y = Y_{p,k}(G)$ be the random variable counting the number of induced matchings of size k in G.

Theorem 80 *$\nu_I(G) < \frac{\sqrt{n} \cdot \log n}{2c}$ for almost all graphs in $\mathcal{G}(n,p)$ with $p = c/\sqrt{n}$ and c constant.*

Proof. Without loss of generality, by Theorem 79 it is possible to assume that $k > \frac{\log c}{3c} \cdot \sqrt{n}$. Using Markov inequality,

$$\begin{aligned}
\Pr[Y > 0] &\leq \frac{n!}{k!\,(n-2k)!} \left(\frac{c}{2\sqrt{n}}\right)^k \mathrm{e}^{-\frac{2ck(k-1)}{\sqrt{n}}} \\
&\leq \frac{n^{2k}}{k!} \left(\frac{c}{2\sqrt{n}}\right)^k \mathrm{e}^{-\frac{2ck(k-1)}{\sqrt{n}}} \\
&\leq \frac{O(1)}{\sqrt{2\pi k}} \left(\frac{\mathrm{e}cn^{\frac{3}{2}}}{2k}\right)^k \mathrm{e}^{-\frac{2ck(k-1)}{\sqrt{n}}} \\
&\leq \frac{O(1)}{\sqrt{2\pi k}} \left(\frac{3\mathrm{e}c^2 n}{2\log c}\right)^k \mathrm{e}^{-\frac{2ck(k-1)}{\sqrt{n}}}
\end{aligned}$$

The last upper bound goes to zero if

$$k > \frac{\sqrt{n}}{2c} \cdot \log\left(\frac{3\mathrm{e}c^2 n}{\log c}\right)$$

By using standard calculus it is not difficult to prove that $f(c) = \frac{3\mathrm{e}c^2}{\log c}$ is minimised for $c = \mathrm{e}^{\frac{1}{2}}$ and $f(c) \geq 6\mathrm{e}^2$. Hence the expected number of induced matchings of size k is very small as long as $k > \frac{\sqrt{n}}{2c} \log 6\mathrm{e}^2 n = \frac{\sqrt{n}(2\log 6 + \log n)}{2c}$. $\square$

4.9 Conclusions

In this chapter we studied two graph-theoretic problems related to the set of matchings in a graph. MINMAXLMATCH denotes the problem of finding a maximal matching of minimum cardinality whereas MAXINDMATCH denotes the problem of finding an induced matching (see the beginning of Section 4.2 for a precise definition of an induced matching) of maximum cardinality in a given graph.

The first part of the chapter described a number of worst-case results. In particular, we proved that MINMAXLMATCH is NP-hard even if the input is an almost regular bipartite graph of maximum degree $3s$, for each integer $s > 0$, and MAXINDMATCH is NP-hard to approximate with approximation ration r (for some $r > 1$) even if the input is an almost regular graph of maximum degree $4s$, for each integer $s > 0$. After presenting simple approximation heuristics for classes of regular and bounded degree graphs based on structural properties of these matchings, and a linear time optimal algorithm for MAXINDMATCH if the input graph is a tree, the investigation of the complexity of these problems in bounded degree graphs was completed by deriving a number of non-approximability results (see Section 4.6) for several classes of bounded degree graphs.

The second part of the chapter described a number of improved results obtained under the assumption that the input graph is generated according to a number of different procedures (see Section 1.2 for all the relevant definitions).

Bibliography

[ADJS93] I. Althofer, G. Das, D. Joseph, and J. Soares. On Sparse Spanners of Weighted Graphs. *Discrete and Computational Geometry*, 9:81–100, 1993.

[AF99] D. Achlioptas and E. Friedgut. A Sharp Threshold for 3-Colourability. *Random Structures and Algorithms*, 14:63–70, 1999.

[AFWZ95] N. Alon, U. Feige, A. Wigderson, and D. Zuckerman. Derandomized Graph Products. *Computational Complexity*, 5:60–75, 1995.

[AHU74] A. V. Aho, J. E. Hopcroft, and J. D. Ullman. *The Design and Analysis of Computer Algorithms*. Addison-Wesley, 1974.

[AM] D. Achlioptas and M. Molloy. Almost All Graphs with 2.522n Edges are not 3-Colourable. Electronic Journal of Combinatorics, to appear.

[And76] G.E. Andrews. *The Theory of Partitions*, volume 2 of *Encyclopedia of Mathematics and its Applications*. Addison-Wesley, 1976.

[Apo57] T. M. Apostol. *Mathematical Analysis*. Addison-Wesley, 1957.

[Arf51] G. Arfwedson. A Probability Distribution Connected With Stirling's Second Class Numbers. *Skandinavisk Aktuarietidskrift*, 36:121–132, 1951.

[AS91] M.D. Atkinson and J.R. Sack. Uniform Generation of Combinatorial Objects in Parallel. Technical Report SCS-TR-185, School of Computer Science, Carleton University, Ottawa, Canada, January 1991.

[AS94] M.D. Atkinson and J.R. Sack. Uniform Generation of Binary Trees in Parallel. *Journal of Parallel and Distributed Computing*, 23:101–103, 1994.

[ASE92] N. Alon, J. H. Spencer, and P. Erdős. *The Probabilistic Method*. Wiley Interscience Series in DMATH and Optimization. John Wiley and Sons, 1992.

[Bak94] B. S. Baker. Approximation Algorithms for NP-complete Problems on Planar Graphs. *Journal of the Association for Computing Machinery*, 41(1):153–180, January 1994.

[BC91] D. P. Bovet and P. Crescenzi. *Teoria della Complessità Computazionale*. Franco Angeli, 1991.

[BDG88] J.L. Balcazar, J. Diaz, and J. Gabarro. *Structural Complexity I*, volume 11 of *EATCS Monographs on Theoretical Computer Science*. Springer-Verlag, 1988.

[BK99] P. Berman and M. Karpinski. On Some Tighter Inapproximability Results. In *Proceedings of the 26th International Colloquium on Automata, Languages, and Programming*, volume ??? of *Lecture Notes in Computer Science*, page ??? Springer - Verlag, 1999.

[Bol79] B. Bollobás. *Graph Theory*, volume 63 of *Graduate Text in Mathematics*. Springer Verlag, 1979.

[Bol85] B. Bollobás. *Random Graphs*. Academic Press, 1985.

[Bre74] R. P. Brent. The Parallel Evaluation of General Arithmetic Expressions. *Journal of the Association for Computing Machinery*, 21:201–206, 1974.

[Cam89] K. Cameron. Induced Matchings. *Discrete and Applied Mathematics*, 24(1-3):97–102, 1989.

[Chv91] V. Chvátal. Almost All Graphs With $1.44n$ Edges Are 3-Colourable. *Random Structures and Algorithms*, 2:11–28, 1991.

[CLR90] T. H. Cormen, C. E. Leiserson, and R. L. Rivest. *Introduction to Algorithms*. M.I.T. Press, 1990.

[CR73] S. A. Cook and R. A. Reckhow. Time Bounded Random Access Machines. *Journal of Computer and System Sciences*, 7:354–375, 1973.

[CR92] V. Chvátal and B. Reed. Mick Gets Some (the Odds Are on His Side. In *Proceedings of the 33rd Annual Symposium on Foundations of Computer Science*, pages 620–627. IEEE, 1992.

[Cre97] P. Crescenzi. A Short Guide to Approximation Preserving Reductions. In *Proceedings of the 12th Annual IEEE Conference on Computational Complexity*, pages 262–273, Ulm, 1997. IEEE.

[CS88] V. Chvátal and E. Szemerédi. Many Hard Examples for Resolution. *Journal of the Association for Computing Machinery*, 35(4):759–768, October 1988.

[CV88] R. Cole and U. Vishkin. Approximate Parallel Schedulting, part I: the Basic Technique With Applications to Optimal Parallel List Ranking in Logarithmic Time. *SIAM Journal on Computing*, 17:128–142, 1988.

[DB97] O. Dubois and Y. Boufkhad. A General Upper Bound for the Satisfiability Threshold of Random r-SAT Formulae. *Journal of Algorithms*, 24(2):395–420, aug 1997.

[DFK91] M. Dyer, A. Frieze, and R. Kannan. A Random Polynomial Time Algorithm for Approximating the Volume of a Convex Body. *Journal of the Association for Computing Machinery*, 38(1):1–17, January 1991.

[Dun88] P. E. Dunne. *The Complexity of Boolean Networks*, volume 29 of *A.P.I.C. Series*. Academic Press, 1988.

[DVW96] A. Denise, M. Vasconcellos, and D. J. A. Welsh. The Random Planar Graph. *Congressus Numerantium*, 113:61–79, 1996.

[DW83] J.D. Dixon and H.S. Wilf. The Random Selection of Unlabelled Graphs. *Journal of Algorithms*, 4:205–213, 1983.

[DWZ] W. Duckworth, N. C. Wormald, and M. Zito. Approximation Algorithms for Finding Sparse 2-Spanners of 4-Connected Planar Triangulations. Submitted to the 10th Australasian Workshop on Combinatorial Algorithms.

[DZ] W. Duckworth and M. Zito. Sparse Hypercube 3-Spanners. To appear in Discrete Applied Mathematics.

[Edm65] J. Edmonds. Paths, Trees and Flowers. *Canadian Journal of Mathematics*, 15:449–467, 1965.

[ER66] P. Erdős and A. Rényi. On the Existence of a Factor of Degree One of a Connected Random Graph. *Acta Mathematica Academiae Scientiarum Hungaricae*, 17(3–4):359–368, 1966.

[Erd88] P. Erdős. Problems and Results in Combinatorial Analysis and Graph Theory. *Discrete Mathematics*, 72:81–92, 1988.

[FGST89] R. J. Faudree, A. Gyárfas, R. H. Schelp, and Z. Tuza. Induced Matchings in Bipartite Graphs. *Discrete Mathematics*, 78(1-2):83–87, 1989.

[FK96] E. Friedgut and G. Kalai. Every Monotone Graph Property Has a Sharp Threshold. *Proceedings of the American Mathematical Society*, 124:2993–3002, 1996.

[FL83] T. I. Fenner and G. Loizou. Tree Traversal Related Algorithms for Generating Integer Partitions. *SIAM Journal on Computing*, 12(3):551–564, August 1983.

[For73] R. Forcade. Smallest Maximal Matchings in the Graph of the d-Dimensional Cube. *Journal of Combinatorial Theory (B)*, 14:153–156, 1973.

[Fri97] E. Friedgut. Necessary and Sufficient Conditions for Sharp Thresholds of Graph Properties, and the k-Sat Problem. with an appendix of J. Bourgain, October 1997.

[FS96] A. M. Frieze and S. Suen. Analysis of two Simple Heuristics on a Random Instance of k-SAT. *Journal of Algorithms*, 20:312–355, 1996.

[Gav72] F. Gavril. Algorithms for Minimum Coloring, Maximum Clique, Minimum Covering by Cliques and Maximum Independent Set of a Chordal Graph. *SIAM Journal on Computing*, 1(2):180–187, June 1972.

[Giu83] E. Giusti. *Analisi Matematica*, volume 1. Boringhieri, Torino, 1983.

[GJ79] M. R. Garey and D. S. Johnson. *Computer and Intractability, a Guide to the Theory of NP-Completeness*. Freeman and Company, 1979.

[GJ97] L. A. Goldberg and M. Jerrum. Randomly Sampling Molecules. In *Proceedings of the Eighth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 183–192, New Orleans, Louisiana, 5–7 January 1997.

[GKP89] R.L. Graham, D.E. Knuth, and O. Patashnik. *Concrete Mathematics*. Addison-Wesley, 1989.

[GM75] G. R. Grimmett and C. J. H. McDiarmid. On Colouring Random Graphs. *Mathematical Proceedings of the Cambridge Philosophical Society*, 77:313–324, 1975.

[Goe92] A. Goerdt. A Threshold for Unsatisfiability. In I. Havel and V. Krbec, editors, *Mathematical Foundations of Computer Science*, volume 629 of *Lecture Notes in Computer Science*, pages 264–274. EACTS, Springer-Verlag, 1992.

[GR88] A.M. Gibbons and W. Rytter. *Efficient Parallel Algorithms*. Cambridge University Press, 1988.

[GS92] G. R. Grimmett and D. R. Stirzaker. *Probability and Random Processes*. Clarendon Press, Oxford, second edition, 1992.

[Har69] F. Harary. *Graph Theory*. Addison-Wesley, 1969.

[HK73] J. Hopcroft and R. Karp. An $n^{5/2}$ Algorithm for Maximal Matching in Bipartite Graphs. *SIAM Journal on Computing*, 2:225–231, 1973.

[HK93] J. D. Horton and K. Kilakos. Minimum Edge Dominating Sets. *SIAM Journal on DMATH*, 6(3):375–387, August 1993.

[Hm90] T. Hagerup and C. Rüb. A Guided Tour of Chernoff Bounds. *Information Processing Letters*, 33(6):305–308, February 1989-90.

[HP73] F. Harary and E.M. Palmer. *Graphical Enumeration*. Academic Press, 1973.

[HQT93] P. Horák, H. Qing, and W. T. Trotter. Induced Matchings in Cubic Graphs. *Journal of Graph Theory*, 17(2):151–160, 1993.

[HW94] T. Hogg and C. P. Williams. The Hardest Constraint Problems: a Double Phase Transition. *Artificial Intelligence*, 69:359–377, 1994.

[JKŁP93] S. Janson, D. E. Knuth, T. Łuczak, and B. Pittel. The Birth of the Giant Component. *Random Structures and Algorithms*, 4(3):233–358, 1993.

[Joh90] D.S. Johnson. *A Catalog of Complexity Classes*, volume A of *Handbook of Theoretical Computer Science*, chapter 2, pages 69–161. Elsevier, 1990.

[JVV86] M.R. Jerrum, L.G. Valiant, and V.V. Vazirani. Random Generation of Combinatorial Structures from a Uniform Distribution. *Theoretical Computer Science*, 43(2–3):169–188, 1986.

[KH78] B. Korte and D. Hausmann. An Analysis of the Greedy Heuristic for Independence Systems. *Annals of Discrete Mathematics*, 2:65–74, 1978.

[KKKS98] L. M. Kirousis, E. Kranakis, D. Krizanc, and Y. C. Stamatiou. Approximating the Unsatisfiability Threshold of Random Formulas. *Random Structures and Algorithms*, 12(3):253–269, 1998.

[KKT95] D. R. Karger, P. N. Klein, and R. E. Tarjan. A Randomized Linear-time Algorithm to Find Minimum Spanning Trees. *Journal of the Association for Computing Machinery*, 42(2), 1995.

[KLM89] R.M. Karp, M. Luby, and N. Madras. Monte-Carlo Approximation Algorithms for Enumeration Problems. *Journal of Algorithms*, 10(3):429–448, September 1989.

[KM] S. Khanna and S. Muthukrishnan. Personal communication.

[Knu73] D. E. Knuth. *The Art of Computer Programming: Searching and Sorting*, volume 3. Addison-Wesley, 1973.

[KR87] R. M. Karp and M. O. Rabin. Efficient Randomized Pattern-matching Algorithms. *IBM Journal of Research and Development*, 31:762–773, March 1987.

[Kri86] V. Krishnamurthy. *Combinatorics: Theory and Applications*. Mathematics and its Applications. John Wiley and Sons, 1986.

[Lei92] F. T. Leighton. *Introduction to Parallel Algorithms and Architectures, Arrays Trees Hypercubes*. Morgan Kaufmann, San Mateo, CA, 1992.

[LP86] L. Lovász and M. D. Plummer. *Matching Theory*, volume 29 of *Annals of Discrete Mathematics*. North Holland, 1986.

[LZ97] J. Liu and H. Zhou. Maximum Induced Matchings in Graphs. *Discrete Mathematics*, 170:277–281, 1997.

[McD92] C. McDiarmid. On a Correlation Inequality of Farr. *Combinatorics, Probability and Computing*, 1:157–160, 1992.

[MR95] R. Motwani and P. Raghavan. *Randomized Algorithms*. Cambridge University Press, 1995.

[MV80] S. Micali and V. V. Vazirani. An $O(v^{1/2}e)$ Algorithm for Finding Maximum Matching in General Graphs. In *Proceedings of the 21st Annual Symposium on Foundations of Computer Science*, pages 17–27, New York, 1980. IEEE Computer Society Press.

[NW78] A. Nijenhuis and H.S. Wilf. *Combinatorial Algorithms*. Academic Press, New York, 1978.

[Pap94] C. Papadimitriou. *Computational Complexity*. Addison-Wesley, 1994.

[Pm89] D. Peleg and A. A. Schäffer. Graph Spanners. *Journal of Graph Theory*, 13(1):99–116, 1989.

[PU89] D. Peleg and J. Ullman. An Optimal Synchronizer for the Hypercube. *SIAM Journal on Computing*, 18:740–747, 1989.

[Pu97] I. Pu. *Algorithms for Economic Storage and Uniform Generation of Graphs*. PhD thesis, University of Warwick, September 1997.

[PY91] C. H. Papadimitriou and M. Yannakakis. Optimization, Approximation and Complexity Classes. *Journal of Computer and System Sciences*, 43:425–440, 1991.

[Rio58] J. Riordan. *An Introduction to Combinatorial Analysis*. Wiley, 1958.

[Rot65] J.J. Rotman. *The Theory of Groups: An Introduction*. Advanced Mathematics. Allyn and Bacon, Boston, 1965.

[Rub81] R. Y. Rubinstein. *Simulation and the Monte Carlo Method*. John Wiley and Sons, 1981.

[Sch97] G. Schaeffer. Bijective Census and Random Generation of Eulerian Planar Maps with Prescribed Vertex Degrees. *Electronic Journal of Combinatorics*, 4(1), 1997. Available from `http://www.combinatorics.org`.

[Sin93] A. Sinclair. *Algorithms for Random Generation and Counting: a Markov Chain Approach*. Birkhäuser, 1993.

[SP94] J. Sorenson and I. Parberry. Two Fast Parallel Prime Number Sieves. *Information and Computation*, 114:115–130, 1994.

[SS96] L. A. Sanchis and M. B. Squire. Parallel Algorithms for Counting and Randomly Generating Integer Partitions. *Journal of Parallel and Distributed Computing*, 34:29–35, 1996.

[SV82] L. J. Stockmeyer and V. V. Vazirani. NP-Completeness of Some Generalizations of the Maximum Matching Problem. *Information Processing Letters*, 15(1):14–19, August 1982.

[SY93] A. Steger and M. Yu. On Induced Matchings. *Discrete Mathematics*, 120:291–295, 1993.

[Tin90] G. Tinhofer. Generating Graphs Uniformly at Random. *Computing*, 7:235–255, 1990.

[Val79] L. G. Valiant. The Complexity of Computing the Permanent. *Theoretical Computer Science*, 8:189–201, 1979.

[vN51] J. von Neumann. Various Techniques Used in Connection with Random Digits. *U.S. National Bureau of Standards Applied Mathematics Series*, 12:36–38, 1951.

[Wil94] H. Wilf. *Algorithms and Complexity*. Prentice-Hall, 1994.

[Wil97] D. B. Wilson. Annotated Bibliography of perfectly Random Sampling with Markov Chains. In D. Aldous and J. Propp, editors, *Microsurveys in Discrete Probability*, volume 41 of *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, page 209. American Mathematical Society, 1997. Also available from `http://dimacs.rutgers.edu/~dbwilson/exact/`.

[Wor87] N.C. Wormald. Generating Random Unlabelled Graphs. *SIAM Journal on Computing*, 16(4):717–727, 1987.

[YG80] M. Yannakakis and F. Gavril. Edge Dominating Sets in Graphs. *SIAM Journal on Applied Mathematics*, 38(3):364–372, June 1980.