Diss. ETH No. 27533

Properties of Randomness in Graph Theory and Algorithms

A thesis submitted to attain the degree of
DOCTOR OF SCIENCES of ETH Zurich
(Dr. sc. ETH Zurich)

presented by
Pascal Su
Master of Science ETH in Mathematics
born on 15.08.1990
citizen of Switzerland

accepted on the recommendation of
Prof. Dr. Angelika Steger, examiner
Prof. Dr. Konstantinos Panagiotou, co-examiner
Prof. Dr. Tibor Szabó, co-examiner

2021

Contents

Abstract

The tools of probability theory have long been introduced to many fields, including many areas of theoretical computer science. The following is a quote, sometimes attributed to G.C. Rota:

"Probability is just combinatorics divided by n"

While there may be some theoretical truth to this statement, the number of insights the probabilistic approach has given into areas of combinatorics, extremal graph theory, and other computer science problems at least suggests that there is some merrit to approaching a problem with probability in mind. We will focus on the field of probabilistic combinatorics, the study of combinatorial structures generated in a probabilistic setting. It is a relatively new but also a highly active field in mathematics.

This dates back to seminal work of Erdős in the 1950s to 60s when he proved that combinatorial structures with nice properties exist, but without explicitly constructing them. Instead, he used a probabilistic argument to prove that one must exist. Interestingly, when trying to deterministically reproduce these constructions this often proves challenging and for many problems, it still remains an open problem to find an explicit solution.

In graph theory, considerable focus is placed on embedding problems, raising the question, can we find substructure in graphs. Random graphs and the probabilistic method have helped substantially here. But not just in graph theory, also in algorithm design, probability and randomness have proven to be valuable for new proofs, new algorithms and a better understanding of problems.

In this thesis, we present four results that use or have been inspired by techniques from probability theory and show that not only do these tools give clean and elegant proofs, but they also help us gain a deeper understanding of the bottlenecks of the problem.

The first result discusses the chromatic number of a graph. Computing the chromatic number is generally $\mathcal{NP}$-hard. However in random graphs, the task is often much easier. In the Erdős-Rényi graph model, $G(n,p)$, the chromatic number is well studied and we extend the asymptotic value of the chromatic number to the stochastic block graph model. This is a natural generalization of $G(n,p)$ in which instead of having a homogeneous probability for all edges, we allow a partition of the vertex set into constantly many parts and may choose different probabilities p for every part and in-between parts where clearly $G(n,p)$ is then a special case. The proof here uses well-known techniques and concentration of random variables but we believe it is elegantly captured in a clean and intuitive result on the exact behavior of independent sets and how they tie to the chromatic number in this random graph model.

The second result we discuss is about embedding a K_r-factor in a graph with some random-inspired property. Randomness gives us considerable power and often in proofs, we do not need most properties of random graphs instead focusing only on for example the expansion properties. Here, we take the famous theorem of Hajnal and Szemerédi. It states that if a graph has minimum degree $\delta(G) \geq \left(1 - \frac{1}{r}\right) n$ and r divides n, it must also have K_r-factor, specifically it is possible to cover all vertices with disjoint cliques of size r. It is easy to find an example that shows that the theorem is tight. However in the random case a much smaller number of edges is required. We restrict ourselves to graphs with sublinear independence number. In this case we are able to find new tight bounds on the minimum degree.

Remaining on the theme of randomness, in the third result we construct a randomized algorithm for finding Hamilton cycles in random graphs with high probability. As mentioned, randomness gives us considerable power, and already in the late 70's Angluin and Valiant observed that

the $\mathcal{NP}$ hard problem of finding a Hamilton cycle becomes close to linear time if we take a random graph. Specifically, we are able to bring the runtime of this algorithm down to $\mathcal{O}(n)$.

In the last result, we present an algorithm to solve the mastermind problem. Mastermind is a game played against an adversary who chooses an arbitrary hidden codeword in $[n]^k$. The Player can ask questions of the form of a string in $[n]^k$ and receives an answer revealing how many of the positions were correct. This problem generalizes the coin weighing problem and first appeared as Mastermind by Chvátal in1983. Here, random queries play a central role in the strategy when k is small compared to n. One of the interesting properties to observe here is the Shannon entropy of a query. A query can receive an answer between 0 and n for the number of correct positions, so the entropy is at most $\log_2(n+1)$, which gives a natural lower bound of $\Omega(n)$ number of queries needed. However a random query without any knowledge of the codeword will recieve an answer distributed with a binomial distribution $Bin(n, 1/k)$, which has constant entropy when k is of order n and it has been shown that non-adaptive strategies need at least $\Omega(n \log n)$ many queries. Therefore, while most existing strategies for small k used random queries, and thus obtained an elegant solution matching the lower bound, when $k = n$, this becomes no longer feasible. We show that by using an adaptive strategy for $k = n$, we can use only $\mathcal{O}(n)$ queries.

Zusammenfassung

Die Methoden der Wahrscheinlichkeitstheorie sind schon längst in verschiedensten andern Gebieten eingeführt worden, dabei auch einige Gebiete der Theoretischen Informatik. Das folgende ist ein Zitat, wird manchmal G.C. Rota zugeschrieben:

"Wahrscheinlichkeit ist einfach Kombinatorik dividiert durch n"

Während es wohl einen gewissen theoretischen kern an Wahrheit an dieser Aussage hat, die Vielzahl an Einsichten die der Ansatz der Wahrscheinlichkeitstheorie eingebracht hat in Gebieten der Kombinatorik, extremalen Graphen Theoie, und anderen Problemen der Informatik deutet darauf hin dass es doch einen Mehrwert bringt, ein Problem von der Perspektive der Wahrscheinlichkeitstheorie zu betrachten. Wir beschäftigen uns mit probabilistischer Kombinatorik, die Lehre der Kombinatorischen Objekten die stochastisch generiert wurden. Es ist ein relativ neues aber höchst aktives Gebiet in der Mathematik.

Dies wurde ins Leben gerufen mit der wegweisenden Arbeit von Erdős in den 1950er bis 69er Jahren. Er bewies mit der Probabilistischen Methode, dass Kombinatorische Objekte mit wünschenswerten Eigenschaften existieren, ohne diese explizit zu konstruieren. Interessanterweise, wenn

man versucht auf deterministische weise diese Konstruktionen zu reproduzieren kann dies of schwierig sein und für viele Probleme bleibt es bis heute eine offen solche konstruktive Lösungen zu finden.

In der Graphentheorie wird ein beträchtlicher Fokus auf einbettungs Probleme gelegt; die Frage ob wir eine Substruktur in einem Graphen finden können. Zufallsgraphen und die Probabilistische Methode haben hier wesentlich geholfen. Aber nicht nur in der Graphentheorie, auch im Algorithmen Design sind Wahrscheinlichkeitstheorie und Zufall wertvoll für neue Beweise, neue Algorithmen und ein besseres Verständnis von Problemen.

In dieser Thesis präsentieren wir vier Resultate die Methoden der Wahrscheinlichkeitstheorie benutzen oder davon inspiriert wurden. Wir möchten zeigen dass diese Methoden nicht nur saubere und elegante Beweise liefern, sondern dass sie auch jeweils besser Einsicht geben wo der Engpass eines Problemes liegt.

Unser erstes Resultat beschäftigt sich mit der chromatischen Zahl eines Graphen. Die chromatische Zahl eines Graphen zu bestimmen ist generell $\mathcal{NP}$-schwer. Jedoch ist es für Zufallsgraphen oft einfacher. Im Erdős-Rényi Graphen Model, $G(n,p)$, ist die chromatische Zahl gut studiert und wir erweitern indem wir den asymptotischen Wert der chromatischen Zahl des Stochastischen Block Graphen Model bestimmen. Dies ist eine natürliche verallgemeinerung des $G(n,p)$ bei dem wir statt einer gleichbleibenden Wahrscheinlichkeit für alle Kanten wir eine Partition erlauben in eine Konstante Anzahl Teile. In jedem Teil und zwischen zwei Teilen dürfen verschiedene Wahrscheinlichkeiten p gewählt werden. Es ist klar dass der $G(n,p)$ der Spezialfall ist wenn die Partition nur aus einem Teil besteht. Der Beweis hier benutzt bekannte Methoden und Konzentration der Zufallsvariabeln aber wir glauben es der Beweis ist sauber und intuitiv und erfasst hervorragend das Verhalten der unabhängigen Mengen und zeigt deren Zusammenhang zur Chromatischen Zahl in diesem Zufallsgraphen Model.

Das zweite Resultat das wir präsentieren beschäftigt sich mit dem einbetten eines K_r-Faktors in einem Graphen mit einer zufalls-inspirierten Eigenschaft. Der Zufall gibt uns beträchtliche Macht und oft in Beweisen benötigen wir viele der Eigenschaften von Zufallsgraphen nicht. Stattdessen fokusieren wir uns zum Beispiel auf die Expansionseigenschaft. Hier betrachten wir das berühmte Theorem von Hajnal und Szemerédi. Es besagt dass falls ein Graph einen Minimalgrad von mindestens $\delta(G) \geq \left(1 - \frac{1}{r}\right) n$ hat und r teilt n, dann hat der Graph auch einen K_r-

faktor, das heisst es ist möglich alle Knoten mit disjunkten Kliquen der grösse r zu bedecken. Es ist einfach ein Beispiel zu finden welches zeigt dass dieses theorem bestmöglich ist. Jedoch benötigen wir bei Zufallsgraphen viel weniger Kanten. Wir betrachten den Fall wo die grösste unabhängige Menge des Graphen nur sublineare grösse haben darf. Dann können wir neue bestmögliche Schranken finden für den minimalgrad.

Wir bleiben bei Zufallsgraphen und zeigen im dritten Resultat welches wir präsentieren, dass es einen randomisierten Algorithmus gibt um einen Hamilton Kreis in einem Zufallsgraphen mit hoher Wahrscheinlichkeit findet. Wie bereits erwähnt gibt uns der Zufall vielseitige Möglichkeiten. Bereits seit den 70ern beobachteten Angluin und Valiant dass das $\mathcal{NP}$-schwere Problem einen Hamilton cycle zu finden fast in linearer Zeit lösbar ist wenn wir einen Zufallsgraphen haben. In unserem Fall zeigen wir dass wir die Laufzeit des Algorithmuses weiter reduzieren können bis $\mathcal{O}(n)$ mit hoher Wahrscheinlichkeit.

In unserem letzten Resultat präsentieren wir einen Algorithmus um das Mastermind Problem zu lösen. Mastermind ist ein Spiel das gegen einen Gegenspieler gespielt wird der ein beliebiges geheimes Codewort in $[n]^k$ auswählt. Der Spieler darf Fagen stellen von der Form von Wörtern in $[n]^k$ und erhält eine Antwort die preisgibt wie viele Positionen korrekt waren. Diese Problem generalisiert das Münz wäge problem und ist das erste Mal von Chvátal 1983 als Mastermind in der Literatur erschienen. Im Werk von Chvátal spiele Zufallsgenerierte Fragen eine entscheidende Rolle und sind asymptotisch optimal solange k klein ist verglichen zu n. Eine der interessanten Eigenschaften die wir beobachten können ist die Shannon Entropie einer Frage. Eine Frage kann eine Antwort zwischen 0 und n erhalten für die Anzahl korrekter Positionen, also ist die Entropie dieser Frage höchstens $\log_2(n+1)$, welches uns eine natürliche untere Schranke von $\Omega(n)$ für die Anzahl benötigte Fragen gibt. Jedoch ist die Antwort auf eine zufällige Frage, ohne jegliches Wissen über das geheime Codewort, binomial verteilt $Bin(n, 1/k)$, welches nur eine konstante Entropie hat wenn k die selbe Grössenordnung hat wie n und zeigt dass Strategien welche nicht adaptiv sind mindestens $\Omega(n \log n)$ viele Fragen stellen müssen. Deshalb, während die meisten Strategien für kleine k zufällig generierte Fragen benutzten und so eine elegante Lösung liefern die asymptotisch optimal ist, ist dies für den Fall $k = n$ nicht mehr möglich. Wir zeigen das mithilfe einer adaptiven Strategie $\mathcal{O}(n)$ viele Fragen genügen.

Acknowledgments

After spending almost a third of my life here, it feels weird to complete the PhD and leave what almost feels like family at ETH. I wouldn't say it was an easy journey but overall it was a very enjoyable and inspiring journey.

First I would like to thank Prof. Angelika Steger, for giving me a chance at my PhD and for being a helpful and motivating advisor. The enthusiasm she has for Mathematics is infectious and I have learned so much from her during my time as a student. Thank you for giving me and all your other students such a smooth introduction to the scientific world.

Of course a big thanks to all the other members of our research group I had the pleasure of interacting with during my PhD, both during work for engaging conversations as well as valuable help and insights, as well as the non-work-related part, for making the entire workplace an enjoyable place to be.

I would like to thank Kosta Panagiotou, for great collaboration, for instructive discussions, and for leading by example and sharing his passion for the problems we solve.

Many thanks go to Tibor Szabó for taking the time to read and coreferee this thesis. You have a big impact in the area of my research

and your opinion of my work accordingly is highly respected.

Of course, I would like to thank all my collaborators for the work featured in this thesis, I enjoyed very much working with all of you: Charlotte Knierim, Konstantinos Panagiotou, Anders Martinsson, Rajko Nenadov, Angelika Steger and Miloš Trujić.

Finally thanks to all the interesting people I have met on my Journey at ETH, which I have experienced as a most welcoming and inspiring place. And of course, all of this is not possible without the love and support from all the people close to me in my personal life. Thanks for all your encouragement which is more valuable to me than anything in the world.

Chapter 1

Introduction

The application of probabilistic tools is rather new but active and thriving in the field of combinatorics and theoretical computer science. Although not the first to apply the probabilistic method, Paul Erdős and Alfred Rényi [ER59] published many cornerstone papers in the 1950s and 60s proving existence of structures in a non-constructive way. The motivation was to attack combinatorial problems with the knowledge provided by probability theory. This allowed to produce combinatorial structures, which were often graphs, that were hard to find with traditional methods. Ideally, among other properties, such graphs have edges which are spread evenly and do not contain large dense subgraphs. These properties are difficult to put into a formal definition, and in many ways, the constructions achieved are tough to replicate with a deterministic approach and the standard tools from combinatorics.

The interest started mainly in the field of Ramsey theory, but the interest in the methods has been an inspiration to many subsequent mathematicians. In general, embedding problems in graphs have become increasingly well understood. Currently the probabilistic method is vital to proofs for a myriad of problems beyond extremal combinatorics.

Since probabilistic tools in graph theory and algorithms are now so broadly applied it is difficult to both go into depth on a topic and simultaneously cover all of the areas. Nevertheless, because the results that we present here are spread out over multiple subareas, we would like to attempt to give some insights into the progress that has been made in a few selected fields of graph theory and some related to algorithms. We will give a more indepth introduction to each problem at the start of each of the following chapters.

We give an overview of the remaining chapters of the thesis at the end of this chapter in Section 1.7. For clarity regarding notation, a table that is useful to readers is included in Chapter 2.

1.1 Ramsey Theory

The probabilistic method was particularly successful in what we now call Ramsey theory. In graph theory, Ramsey's theorem is the equivalent of stating that there can never exist complete chaos. More formally, a graph G is *Ramsey* for a graph H if for any coloring of the edges of G with two colors there exists a copy of H as a monochromatic subgraph of G. As early as in the 1930s, Ramsey proved [Ram30] that there is an integer n for every integer r such that K_n is Ramsey for K_r. Rephrased, for any r, if we take a large enough complete graph, no matter how complex we color it with two colors, we will always find a clique of size r in one of the colors. The smallest such number n we call the diagonal Ramsey number

$$R(r,r) = \min\{n \in \mathbb{N} | K_n \text{ is Ramsey for } K_r\}.$$

While certainly an interesting proof on it's own, Ramsey theory has recieved a lot more attention due to it's application to other problems. A driving force here was Erdős and Szekeres combining it with other areas and showing its usefulness in combinatorial geometry.

The proof that $R(r,r)$ must be a finite number was found, but we are

also interested in obtaining reasonable bounds on the diagonal Ramsey numbers. Currently a part of many lectures on the topic is a bound by Erdős and Szekeres [ES35] using simple induction:

$$R(r,r) \leq \binom{2n-2}{n-1}$$

This bound is surprisingly difficult to improve. The best known current bound was given only recently by Conlon [Con09]. The lower would seem much easier to approach. We would only need to explicitly construct a coloring which avoids cliques. But this is not how the problem was solved. Erdős could prove $R(r,r)$ is at least exponential [Erd47] using the probabilistic method. This proof showed that a random coloring of the graph is actually the most efficient way to avoid a clique and to date we have still not succeeded in proving the lower bound constructively.

Determining the exact diagonal Ramsey numbers is quite out of reach of our current methods, since there remains an exponential gap between the upper and lower bound.

As always in a successful field, a multitude of variations of Ramsey numbers have been considered. We will mention Ramsey-Turán numbers in a later section, and we refer interested readers to excellent literature on Ramsey theory [Bur74, Prö13].

1.2 Random Graphs

Given the new approach of using random graphs in Ramsey theory, naturally there was an urge urge to study them more, as at the time, they were not nearly as well understood as they are now. The theory of random graphs was born with the seminal work of Erdős and Rényi [ER59] and in the following years was also heavily driven by the same two authors.

They first consider $G(n,m)$, the probability space containing all graphs on n vertices and m edges with the uniform distribution. This model was a very natural one to start with, as the condition on the number of edges was a desirable property to avoid degenerate cases. However analysis shows that the *binomial random graph* $G(n,p)$, often called the Erdős-Rényi graph, has a similar behavior for many applications, but is

easier to analyze. In this graph model, we consider an n vertex graph, and every edge is present with probability p independent of all other edges. Exactly this independence is key to what makes it a good graph to analyze. Most results for $G(n,p)$ translate to $G(n,m)$ and vice versa.

More formally, we define the probability space $G(n,p)$ as the probability space with the set of all simple unlabeled n vertex graphs as the sample space, we take a discrete event space, and as the probability function, for any graph G

$$Pr[G] = p^{|E(G)|}(1-p)^{\binom{n}{2}-|E(G)|}.$$

Random graphs for many parameters, have highly predictable behavior. Most problems we ask are of the type: "Given a graph property $\mathcal{P}$, an integer $n \in \mathbb{N}$ and $p \in [0,1]$, what is the probability that a graph drawn from $G(n,p)$ has the property $\mathcal{P}$?". These properties we consider are often monotone increasing or decreasing, meaning that they are preserved under edge inclusion or removal respectively. For these properties, a surprising phenomenon occurs that we call *threshold* behavior. More formally, we say a graph property $\mathcal{P}$ has a threshold if there exists a function $p_0 = p_0(n)$ such that

$$\lim_{n\to\infty} \Pr[G(n,p) \in \mathcal{P}] = \begin{cases} 0, & p \ll p_0 \\ 1, & p \gg p_0. \end{cases}$$

Bollobas and Thomason [BT97] proved that in fact, *all* monotone increasing properties have a threshold. Some interesting graph properties include being connected, having a chromatic number of at least k, or containing H as a subgraph. Particularly for embedding problems, many tools have been discovered for different application purposes, as we discuss in the next section.

1.3 Embedding Problems

In graph theory we often want to know is whether a graph G contains H as a subgraph. H can be a small graph, a triangle, or a spanning one, such as a Hamilton cycle or a perfect matching. The question arises of what is necessary for the graph to contain H and whether there are

properties we can check faster than merely checking all possible subgraph combinations, since that is often an inefficient or even infeasible solution.

A notorious embedding problem is determining whether a graph has a Hamilton cycle. This problem is computationally NP-hard which implies we do not expect to find a polynomial-time algorithm to determine or find a Hamilton cycle in a graph. Nevertheless, if we do not require a statement for all graphs, for some graph families it may be easier.

For Hamilton cycles, we started with Dirac's theorem stating that every graph with a minimum degree of at least $n/2$ is Hamiltonian. Although this is tight, as in there exists a non-Hamiltonian graph with minimum degree $(n-1)/2$, it is weak in the sense that minimum degree alone is maybe the wrong tool of measurement. Even with Ore's theorem, G is Hamiltonian if every pair of non-adjacent vertices together have an added degree of at least n, these are all descriptions of dense graphs; that is, graphs with order of $\Theta(n^2)$ many edges. However, there exist many graphs with many fewer edges that are also Hamiltonian. In fact, one could say an 'average' graph with lower density is Hamiltonian.

What we mean to say is that random graphs $G(n,p)$ prove to be Hamiltonian as soon as $p \geq n(\log n + \log\log n + \omega(1))/2$, which is an average degree of only $\Theta(\log n)$, bringing the edge count from $\Theta(n^2)$ down to $\Theta(n \log n)$.

For random graphs the threshold for Hamiltonicity has been found exactly, but more general questions have been answered, such as counting Hamilton cycles, finding powers of Hamilton cycles, finding them in the resilient setting, and many more. We refer the interested reader to an excellent bibliography by Frieze [Fri19].

However it is not merely the Hamilton cycle here that is interesting; in fact, a general statement was proven by Bollobás [Bol81]:

Theorem 1.1. *For an arbitrary graph H with at least on edge,*

$$\lim_{n\to\infty} \Pr[G(n,p) \supset H] = \begin{cases} 0 & \text{if } p \ll n^{-1/m(H)}, \\ 1 & \text{if } p \gg n^{-1/m(H)}. \end{cases}$$

Where $m(H) = \max_{H' \subseteq H}\{e(H')/v(H')\}$ is the highest possible ratio of edges to vertices in a non-empty subgraph of H.

1.4 Szemerédi's Regularity Lemma

A powerful and now a standard approach for embedding problems is Szemerédi's Regularity Lemma. Szemerédi [Sze75] introduced this tool in the 1970s in a somewhat weaker form than the one in which we present in Chapter 4. It was invented to prove a conjecture of Erdős on sequences of integers [ET36] and since has seen many variations and some strengthenings with additional properties tailored to specific use cases as well as some adaptions for hypergraphs. However, in essence, the lemma states that given any arbitrary dense graph G, we can partition the vertex set into a constant number of parts such that almost all edges between the parts are "ε-regular" meaning random-like. Therefore, this can be seen as imposing structure on a graph.

Moreover, versions for the sparse case exist (see papers by Gerke and Steger [GS05] and Bollobás and Riordan [BR07]). It is particularly useful in embedding graphs of bounded degree. In particular, together with the absorbing method introduced by Rödl, Ruciński, and Szemerédi [RSR08] it is possible to embed spanning graphs. Although the absorbing method has produced many results forward in recent years, we will not expand on it here. We present an application in Chapter 4.

For a more complete survey of Szemerédi's Regularity Lemma see for example [KS96] or [KSSS00].

1.5 Ramsey-Turán

An early version of Szemerédi's Regularity Lemma is used to prove an upper bound for the Ramsey-Turán numbers [Sze72], which is a more recent field combining elements from Ramsey theory and Turán theory.

Turán studied the following question: if an n-vertex graph G may not contain a clique of size r, how many edges can G have? This is known as Turán's theorem: that $e(G) \leq \left(1 - \frac{1}{r-1}\right) \cdot \frac{n^2}{2}$. The extremal graph is the complete $r-1$ partite graph and is a stable extremal example, meaning even 'close' to the bound, the graph must be similar to an $r-1$ partite graph. This was generalized to arbitrary graph families in a result by Erdős, Stone and Simonovits [ES65] showing that the chromatic number χ is essentially the bottleneck for how many edges we can have and the extremal example is again a complete $(\chi - 1)$-partite graph.

Both Turán's theorem and Ramsey's theorem were groundbreaking results in graph theory, pioneering an entire branch of new research. In the late 1960's, Erdős and Sós [ES70] started connecting the two fields. While Turán extremal examples are very structured we rather think of Ramsey extremal examples as random-like, with probabilistic constructions currently being our best lower bound. If we combine these we obtain the following Ramsey-Turán type problem. The independence number of a graph G, $\alpha(G)$, is the size of a largest independent set in G.

Definition 1.2 (Ramsey-Turán number)**.** For a graph G on n vertices not containing a subgraph K_r and having independence number $\alpha(G) = m$ we denote by

$$\mathbf{RT}(n, K_r, m)$$

The maximum number of edges that G can have.

If m is larger than $n/(r-1)$ this is simply Turán's theorem, which gives us the best upper bound. However if m is smaller than that, in particular if $m = o(n)$, then the extremal example, the Turán graph, is no longer allowed.

The study of $\frac{1}{n^2}RT(n, K_r, o(n))$ was of particular interest. Szemerédi provided an upperbound of $1/8$ for $r = 4$, whose optimality was long debated until Bollobás and Erdős provided a fascinating matching constructive lower bound inspired by the geometry of spheres [BE76].

In Chapter 4, we prove a result in which the forbidden graph is not a clique but instead a clique factor, a set of disjoint cliques covering the graph completely. This was studied for triangles by Balogh, Molla and Sharifzadeh [BMS16].

1.6 Algorithms and Entropy

As we increasingly understand the probabilistic tools, it becomes easier to construct algorithms which actually find or calculate combinatorial objects. Randomized algorithms and probabilistic arguments accompany each other in combinatorics. Not only has randomness proven useful in algorithms for making algorithms faster, because we have a good understanding of how random variables work, this has proven to be a method of proving statements which is quite unique. We use a

measure for randomness called entropy. Informally entropy counts the number of random bits needed to generate a random variable. This is defined by specific rules.

Shannon defined the entropy H of a discrete random variable X as the following expression:

$$\mathrm{H}(X) = -\sum_{i=1}^{n} \Pr[x_i] \log_b \Pr[x_i]$$

Any random variable needs entropy to be generated. As an example, if we can generate the equivalent of n random coin flips in an algorithm, the algorithm must take n entropy from a source of randomness.

This can be applied to create lower bounds for runtimes. The entropy lower bound of $n \log n$ for comparison based sorting an example thereof. Entropy can be a tool for inspiration during algorithm design. Alternitively, we can also prove upper bounds using the entropy compression method. The method uses double counting on entropy to show that an algorithm can not run for too long, without generating more entropy than is used during the algorithm. The most famous application is a proof by Moser [Mos09] to show an algorithmic version of the Lovász Local Lemma. This technique is quite new but has already seen several cases of success [EP13, DJKW16, Mol19].

1.7 Overview of the Thesis

In Chapter 3 we provide a classical application of concentration bounds on a variant of the random graph. We study the chromatic number of the random block graph model. Applying tools of concentration, here the Janson inequality, we learn the deeper connection of the chromatic number and the maximal independent sets of this graph, a phenomenon already well known for the Erdős-Rényi Random graph model. Here, however, there are interesting transitions to observe for different relative edge probabilities in and between the blocks of the graph. We give proofs for tight upper and lower bounds for the chromatic number of the stochastic block graph model. The work in this chapter is from a submitted paper by Martinsson, Panagiotou, Trujić and the author ([MPST20]).

In the next Chapter 4 we discuss an embedding problem in a random-like graph but without the randomness. Here we use the tools developed for random graphs, the Szemeredi Regularity Lemma. We take the famous theorem of Hajnal and Szemerédi. It states that if a graph has minimum degree $\delta(G) \geq \left(1 - \frac{1}{r}\right) n$ and r divides n, it must also have K_r-factor. It is easy to find an example that shows that the theorem is tight, however, in the random case a much smaller number of edges is required. We restrict ourselves to graphs with sublinear independence number. We are able to find new tight bounds on the minimum degree conditioned on a sublinear independence number. This is tightly related to Ramsey-Turán theory. This chapter is from a paper by Knierim and the author ([KS21]).

In Chapter 5 we move to an application of randomness in algorithms. Still in the realm of graphs we study the problem of finding a Hamilton cycle in a graph. This is a problem well known to be NP-hard, so it would be surprising to find a general solution. However, in random graphs, the existence of Hamilton cycles has long been proven with help from the probabilistic method. Moreover, finding and outputting the Hamilton cycle was a major breakthrough [Fri88] and is known to be possible in polynomial time. We give an algorithm to improve the runtime to linear in the number of vertices. This result is from a paper by Nenadov, Steger and the author ([NSS21]).

As a last result in Chapter 6 we present an algorithm to solve the mastermind problem. Mastermind is a well known generalization of the coin weighing problem and has some ties to other problems such as resolving sets in graph theory [JP19]. Mastermind was first studied by Chvátal in 1983 [Chv83]. The first strategy was to query randomly and to prove that this is in fact the best you can do asymptotically. This works whenever the number of available colors k is small compared to n, the number of positions. Using the intuition we get from entropy we devise a strategy which finds the hidden codeword in a linear number of queries in the case when $k = n$. Every other case is then reducible to this strategy so we asymptotically solve Mastermind for all k and n. This chapter is from a submitted paper by Martinsson and the author ([MS20]).

Chapter 2

Preliminaries

In this chapter, we introduce the basic notation and tools that will be used throughout this thesis.

2.1 Basic Notation

We use $\log n$ for the base two logarithm and $\ln n$ for the natural logarithm. For an integer $k \in \mathbb{N}$, we use $[k]$ to denote the set $\{1, \ldots, k\}$.

As we are often interested in limits, we use the standard Landau symbols $\mathcal{O}, \Omega, \Theta, o$ and ω to denote the asymptotic behavior of functions. If a hidden constant γ depends on some other constant ε, we write $\gamma(\varepsilon)$ unless it is clear from the context. We write $a \ll b$ and $a \gg b$ to express that $a = o(b)$ and $a = \omega(a)$, respectively. We interchangeably use *with*

high probability (abbreviated by *w.h.p*) and *asymptotically almost surely* (abbreviated by *a.a.s*) to denote that an event holds with probability $1 - o(1)$ as $n \to \infty$.

2.1.1 Graph theory notation

A number of notation standards exist to make it easier to write proofs in graph theory. Most of these conventions have become standard, we follow the notation standards of [Wes01]. All our graphs are simple graphs involving no multiedges and no loops. We will, with few exceptions use the variables G and H for a graph. The letters u, v and w will usually denote vertices and e an edge.

In the following notation we will omit subindices if they are clear from the context.

Symbol	Definition
$V(G)$	Vertex-set of G
$E(G)$	Edge-set of G
$v(G), v_G$	Number of vertices of G, $v(G) = \lvert V(G)\rvert$
$e(G), e_G$	Number of edges of G, $e(G) = \lvert E(G)\rvert$
$N_G(v)$	Neighborhood of a vertex $v \in V(G)$ in G, $$N_G(v) = \{u \in V(G) : \{u, v\} \in E(G)\}$$
$N_G(U)$	Neighborhood of the set of vertices $v \in U \subseteq V(G)$ in G, $$N_G(U) = \bigcup_{v \in U} N_G(v)$$
$\deg_G(v)$	Degree of a vertex v in G, $\deg_G(v) = \lvert N_G(v)\rvert$
$\delta(G)$	Minimum degree of G, $\delta(G) = \min_{v \in V(G)} \deg_G(v)$
$\Delta(G)$	Maximum degree of G, $\Delta(G) = \max_{v \in V(G)} \deg_G(v)$

Symbol	Definition
$E_G(U)$	All the edges of G with both endpoints in U $$E_G(U) = \{\{u,w\} \in E(G) : u,v \in U\}$$
$E_G(U,W)$	All the edges of G with one endpoint in U and W $$E_G(U,W) = \{\{u,w\} \in E(G) : u \in U, w \in W\}$$
$e_G(U,W)$	Size of $E_G(U,W)$, $e_G(U,W) = \lvert E_G(U,W)\rvert$
$N_G(v,W)$	Set of all neighbors of v in $W \subseteq V(G)$ in the graph G
$N_G(U,W)$	Set of all neighbors of U in W, $$N_G(U,W) = \{w \in W : \exists u \in U \text{ s.t. } \{u,w\} \in E(G)\}$$
$\deg_G(u,W)$	Degree of a vertex u in W, $\deg_G(u,W) = \lvert N_G(u,W)\rvert$

2.2 Probabilistic Tools and Estimates

The reason we can obtain useful properties from random variables is that despite the chaotic randomness observed, well-defined bounds exist in the form of concentration inequalities. Therefore, in theory a random variable can deviate far from it's average, however with a high probability it does not.

We start with the first moment inequality.

Lemma 2.1 (Markov's Inequality)**.** *Let X be a non-negative random variable. For all $t > 0$ we have* $\Pr[X \geq t] \leq \frac{\mathbb{E}[X]}{t}$.

Naturally, we have the second moment estimate by Chebyshev and the most commonly used inequality in our work, Chernoffs inequality. This is most useful when trying to show that the sum of many small random variables is not overly large, because they are more or less independent. The Chernoff and Chebyshev inequality are described for example in the book [JLR11].

Theorem 2.2 (Chebyshev Inequality)**.** *For any random variable X for which the variance $Var[X]$ exists,*

$$Pr[|X - \mathbb{E}[X]| \geq t] \leq \frac{Var[X]}{t^2}$$

Theorem 2.3 (Chernoff's inequality)**.** *Let $X = \sum_{i=1}^{n} X_i$ be the sum of n independent Bernoulli distributed variables and $0 < \varepsilon \leq 3/2$. Then*

$$Pr[|X - \mathbb{E}[X]| \geq \varepsilon \mathbb{E}[X]] \leq 2e^{-\frac{\varepsilon^2 \mathbb{E}[X]}{3}}$$

Given the absence of perfect independence, many inequalities attempt to deal with this. There are the well-known Azuma's inequality and Janson's inequality. In this thesis we will be using the second of the two.

The following version of Janson's inequality, tailored for graphs, will suffice for our purposes. This statement follows immediately from Theorems 8.1.1 and 8.1.2 in [AS04]. This inequality will be used and restated in Chapter 3.

Theorem 2.4 (Janson's inequality)**.** *Let $k \in \mathbb{N}$, $\boldsymbol{\alpha} \in (0,1]^k$ with $|\boldsymbol{\alpha}| = 1$, and let $P = (p_{ij})_{i,j \in [k]}$ be a symmetric matrix with all $p_{ij} \in (0,1)$. Consider a family $\{S_i\}_{i \in \mathcal{I}}$ of subsets of the vertex set $[n]$ and let $G \sim G(n, \boldsymbol{\alpha}, P)$. For each $i \in \mathcal{I}$, let X_i denote the indicator random variable for the event $\{S_i$ is an independent set in $G\}$ and, for each ordered pair $(i,j) \in \mathcal{I} \times \mathcal{I}$, write $X_i \sim X_j$ if $E(S_i) \cap E(S_j) \neq \varnothing$. Let*

$$X := \sum_{i \in \mathcal{I}} X_i, \qquad \mu := \mathbb{E}[X], \qquad \text{and} \qquad \overline{\Delta} := \sum_{\substack{(i,j) \in \mathcal{I} \times \mathcal{I} \\ X_i \sim X_j}} \mathbb{E}[X_i X_j].$$

Then

$$\Pr[X = 0] \leq e^{-\mu^2/(2\overline{\Delta})}.$$

For the convenience of some results, we state some known facts about specific distributions. We use the term *negative binomial distribution* in the analysis, and since this term varies in its definition in the literature we provide here the definition we use.

Definition 2.5. Let X_i be independent Bernoulli random variables with the probability of being one is p for any $i \in \mathbb{N}$. For any $r \in \mathbb{N}$, let Y be the index of the r-th X_i that evaluates to 1. Then, Y has a negative binomial distribution $NB(r,p)$.

A negative binomial distribution $Y \sim NB(r, p)$ is equivalent to Y being distributed as the sum of r geometric random variables with success probability p. A simple corollary from the Chebyshev inequality is as follows:

Corollary 2.6. *For a negative binomially distributed variable* $Y \sim NB(r, p)$

$$Pr\left[Y \geq 2\frac{r}{p}\right] \leq \frac{1}{r}$$

Proof. We calculate $\mathbb{E}[Y] = r/p$ and $Var[Y] = r(1-p)/p^2$ and apply Chebyshev.

$$Pr\left[Y \geq 2\frac{r}{p}\right] \leq Pr\left[|Y - \mathbb{E}[Y]| \geq \frac{r}{p}\right] \overset{Chebyshev}{\leq} \frac{1-p}{r} \leq \frac{1}{r}$$

□

Chapter 3

The Chromatic Number of Dense Random Block Graphs

3.1 Introduction

In this chapter, we prove the asymptotic value of the chromatic number in the dense random block graph model. This is a very classic application of concentration bounds to a problem in random graph theory. The content of this chapter is from a submitted paper with Martinsson, Panagiotou, Trujić and the author ([MPST20]).

For a more general overview we give more thorough background information on the chromatic number of the binomial random graph $G(n, p)$

Chromatic Number of Random Graphs

The chromatic number is one of the most central graph parameters in graph theory and has applications in various other areas of computer science. Formally, in a graph G, the *chromatic number* $\chi(G)$ is defined as the smallest number of colors required for coloring the vertices such that no two adjacent vertices of G are assigned the same color. There are a myriad of results for the chromatic number in the *binomial random graph* $G(n,p)$. Nevertheless, the answer to the question is not yet conclusive and is still an active area of research.

The study started with seminal work of Erdős and Rényi [ER59, ER60] when they first introduced random graphs. In a breakthrough paper from 1978, Bollobás [Bol88] obtained the first asymptotically tight result: he established that for $p \in (0,1)$, w.h.p.

$$\chi(G(n,p)) = (1+o(1))\frac{n}{c(p)\ln n}, \quad \text{where } c(p) = -\frac{2}{\ln(1-p)}. \tag{3.1}$$

This result in fact only represents half the picture, because we also know the structure of how the chromatic number is formed. It has long been known that w.h.p. the independence number $\alpha(G(n,p))$, the size of the largest independent set, in $G(n,p)$ equals $(1+o(1))c(p)\ln n$ (see [Mat76]). Therefore, in addition to proving the value of the chromatic number, Bollobás [Bol88] also proved that we can optimally color the graph asymptotically by covering *essentially all* vertices of $G(n,p)$ with almost *maximum* independent sets; that is, independent sets that are of roughly of size $\alpha(G(n,p))$. In other words, w.h.p. any (almost) optimal coloring of $G(n,p)$ consists of color classes with asymptotic size $c(p)\ln n$ that cover $n - o(n/\ln n)$ vertices. This naturally gives a lower bound that matches the upper bound since all color classes must be independent sets and an independent set can be of size at most $\alpha(G(n,p))$.

The paper of Bollobás initiated a long line of research concerned with studying various properties of the distribution of the chromatic number. The currently most accurate result on the asymptotic value of $\chi(G(n,p))$ for $p \in (0,1)$ is due to Heckel [Hec18], who improved previous results by several authors, e.g. [FKM08, McD89, McD90, PS09], and showed upper and lower bounds for $\chi(G(n,p))$ that are within $o(n/\ln^2 n)$.

Apart from the probable asymptotic value of $\chi(G(n,p))$, other parameters of it have been of considerable interest and difficulty. Most notably, the question about the *concentration* of $\chi(G(n,p))$, that is, the smallest

size of an interval in which $\chi(G(n,p))$ is located w.h.p. has been a point of focus since the seminal papers of Erdős and Rényi (see also [Bol04]). In a recent remarkable breakthrough, Heckel [Hec19] showed polynomial non-concentration bounds for $\chi(G(n,p))$, thus answering a long-standing open question.

Stochastic Block Model

Here we study the chromatic number of random graphs in the so-called *stochastic block model* (also known as the *planted partition model*). The model is a generalization of $G(n,p)$ and is defined as follows. Given $k \in \mathbb{N}$, let $P = (p_{ij})_{i,j\in[k]}$ be a symmetric matrix with all entries $p_{ij} \in (0,1)$. Moreover, let $\boldsymbol{\alpha} = (\alpha_1, \dots, \alpha_k)$ be a vector of (1-)norm $|\boldsymbol{\alpha}| = 1$ and with all $\alpha_i \in (0,1]$. For an integer $n \in \mathbb{N}$ we let $G(n, \boldsymbol{\alpha}, P)$ be a random graph (V,E) obtained as follows. The vertex set $V = V_1 \cup \dots \cup V_k$ consists of k disjoint parts such that $|V_i| = \alpha_i n$ for every $i \in [k]$ and $\sum_{i\in[k]} \alpha_i n = n$. Furthermore, for $i,j \in [k]$, two distinct vertices $u \in V_i$ and $v \in V_j$ form an edge $uv \in E$ with probability p_{ij} independently. We think of $k \geq 1$ as a fixed integer; specifically, the number of parts V_i is fixed and independent of n, and P as a fixed matrix, that is, we only consider (dense) graphs that have $\Omega(n^2)$ edges w.h.p.. We call $G \sim G(n, \boldsymbol{\alpha}, P)$ a *random block graph*. That this is a direct generalization of the Erdős-Rényi binomial random graph $G(n,p)$ should be clear when we choose $k = 1$.

The stochastic block model is more flexible and it enables the modeling of communities. The model also is particularly important for other fields; specifically in physics, statistics, machine learning, networks, and other areas of computer science. Its applications range from social networks to image processing and to genetics (see e.g. [NWS02, PSD00, SM00] for some influential papers in this context), and various properties of the model have been studied in physics [DKMZ11, HLL83] and mathematics and computer science [Bop87, BCLS87, CO10, JS98, MNS15]. For further history, reference, and discussion, we refer to the following remarkable survey [Abb17].

Recently several papers have established generalizations of well-known results about properties of the binomial random graph in the more general stochastic block model. For instance, Hamiltonicity [AFG19] and the size of the largest independent set/clique [DHM17]. Our focus here is to determine the asymptotic value of the chromatic number of random

block graphs. As we will see shortly, similar to the consequences of the paper of Bollobás mentioned above, we will not only obtain the value of the chromatic number but also see a clear interdependence with the maximal independent sets of the graph.

3.1.1 Main results

As we previously mentioned, an (almost) optimal coloring of the binomial random graph $G(n,p)$ for $p \in (0,1)$ typically has a rather simple structure and can be constructed *greedily*: almost all n vertices are covered by independent sets that are of nearly maximum size, that is, of size roughly $c(p)\ln n$, where $c(p) = -2/\ln(1-p)$ is defined in (3.1). As we shall see, the structure of optimal colorings is more intricate and diverse when we consider the broader model of random block graphs.

The first result towards the chromatic number that we have is actually a result on independent sets that exist in the graph $G(n,\boldsymbol{\alpha},P)$. This result is heavily inspired by [DHM17] but more generalized for our needs.

Let $k \in \mathbb{N}$ and $G = G(n,\boldsymbol{\alpha},P)$ where $\boldsymbol{\alpha} \in \mathbb{R}^k$ and $P \in \mathbb{R}^{k\times k}$. For a vector $\mathbf{c} \in \mathbb{R}^k$ and $I \subseteq [k]$ define the map

$$g(\mathbf{c},I) := \sum_{i\in I} c_i + \frac{1}{2}\sum_{i,j\in I} c_i c_j \ln(1-p_{ij}). \tag{3.2}$$

Then we define

$$\mathcal{A} := \big\{\mathbf{c} \in \mathbb{R}^k_{\geq 0} : g(\mathbf{c},I) \geq 0 \text{ for all } \varnothing \neq I \subseteq [k]\big\}, \tag{3.3}$$

With $\mathcal{A}$ defined we can state the types of independent sets that exist in the graph $G(n,\boldsymbol{\alpha},P)$ w.h.p..

Let $X_{\mathbf{c}}$ be the number of independent sets in G that intersect each V_i, $i \in [k]$, at roughly $c_i \ln n$ vertices. Then we expect $X_{\mathbf{c}}$ to be zero w.h.p. if and only if $\mathbf{c}$ is outside of $\mathcal{A}$. In other words, with high probability $\mathcal{A}$ categorizes which type of independent sets appear in the the graph $G(n,\boldsymbol{\alpha},P)$. See Figure 3.1 for an illustration when $k = 2$.

Additionally we prove that not only can we find independent sets of the types which are in $\mathcal{A}$ but we can find them in abundance, including in smaller subgraphs of $G(n,\boldsymbol{\alpha},P)$. Therefore, similarly to the approach by Bollobás, we can show for the chromatic number the following theorem which is the main result of this chapter.

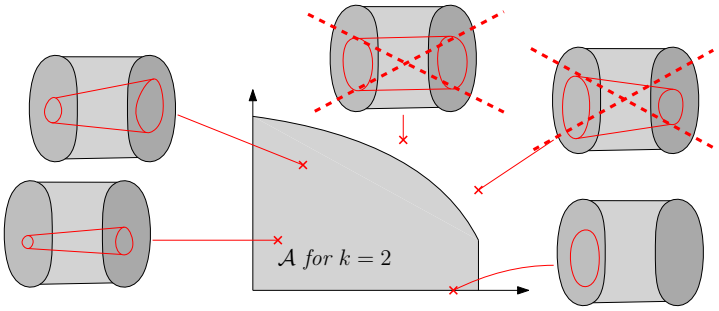

Figure 3.1: Case for $k = 2$. Here $p_{2,2} \geq p_{1,1} \geq p_{1,2}$. $\mathcal{A}$ determines possible independent sets.

Theorem 3.1. *Let $k \geq 1$, $\boldsymbol{\alpha} \in (0,1]^k$ with $|\boldsymbol{\alpha}| = 1$, and let $P = (p_{ij})_{i,j\in[k]}$ be a symmetric matrix with all $p_{ij} \in (0,1)$. Consider a random block graph $G \sim G(n, \boldsymbol{\alpha}, P)$. Then w.h.p.*

$$\chi(G) = (1 + o(1))\frac{n}{c^\star \ln n},$$

where $c^\star$ is given as

$$c^\star = c^\star(\boldsymbol{\alpha}, P) = \max\left\{|\mathbf{c}| : \mathbf{c} \in \operatorname{conv}(\mathcal{A}) \cap \{t \cdot \boldsymbol{\alpha} : t \in \mathbb{R}_{\geq 0}\}\right\}. \tag{3.4}$$

Consider an optimal coloring of a typical instance of G, which is merely a partition of the vertices of G into independent sets $S_1, \ldots, S_{\chi(G)}$. We assume that $\chi(G) = n/(c \ln n)$ and want to determine c. As already mentioned, the S_i's are in $\ln n \cdot \mathcal{A}$. Moreover, each $|V_i|$ can be recovered as $\sum_{1\leq j\leq\chi(G)} |V_i \cap S_j|$. Thus, the average intersection of the color classes with the part V_i is $\overline{s_i} := \alpha_i n/\chi(G) = c\alpha_i \ln n$. In conclusion, in any (in particular, in an *optimal*) coloring, the average intersection of the color classes with each V_i is proportional to $\ln n \cdot \boldsymbol{\alpha}$ and is furthermore a *convex combination* of some types in $\ln n \cdot \mathcal{A}$. Hence, it comes as no surprise that to determine $\chi(G)$, c should be chosen to be maximal under these side constraints, and this is exactly (3.4).

To get the intuition behind the coloring we construct one should analyze the possible types of independent sets that are achievable by evaluating $\mathcal{A}$. In the appropriate ratio, we pick independent sets of the corresponding type, where here we choose greedily, ignoring the rest of the graph, until we cover all but at most $o(n/\ln n)$ vertices. In particular, our findings depend heavily on the shape of $\mathcal{A}$, which in turn depends on the

matrix P as well as the vector α. This implies that the final coloring can follow different schemes. It may be optimal to color $V_1, \dots, V_k$ independently with different colors, or we may choose just a single type $\mathbf{t}$ such that $t_i/t_j \sim \alpha_i/\alpha_j$ for all $i, j \in [k]$ and thus cover (almost) all vertices just with sets of type $\mathbf{t}$ or any interpolation in between. Because of linear dependency, we need at most $k+1$ different types of independent sets to cover almost all vertices.

As a remark, by taking $k = 1$, $\alpha_1 = 1$, and $p_{11} = 1/2$, we recover the classic result of Bollobás [Bol88]. In Section 3.4 we present various applications of the result. Among others, we study the case $k = 2$ in detail and characterize explicitly in all cases the structure of the optimal colorings. Moreover, for general $k \in \mathbb{N}$ we characterize the cases in which

$$\chi(G) \sim \sum_{1 \le i \le k} \chi(G[V_i]),$$

that is, an optimal coloring of G is essentially obtained by coloring each of the k subgraphs individually; as we show, this happens if and only if $p_{ij} \ge 1 - \sqrt{(1-p_i)(1-p_j)}$ for all pairs $i \ne j$. Our last application concerns one further relevant case, namely when there is some homogeneity with respect to the edge probabilities. In particular, we assume that all inter-class probabilities p_{ij}, for $i \ne j$, are equal, and that all intra-class probabilities are also equal. In that case, we explicitly determine the asymptotic value of the chromatic number.

Note that we determine $\chi(G)$ in the case when p_{ij}'s and α_i's are fixed and independent of n. And although some of our results naturally extend for P and $\boldsymbol{\alpha}$ being some functions of n, in general extending this for $p_{ij} = p_{ij}(n)$ and $\alpha_i = \alpha_i(n)$ remains an open problem for further research.

3.2 Independent Sets

In this section we study the distribution of independent sets in the random block graph $G(n, \boldsymbol{\alpha}, P)$. This serves as a main ingredient towards deriving the desired bounds on the chromatic number later on. Throughout, for $\mathbf{t} \in \mathbb{N}_0^k$ we say that a set $S \subseteq V(G)$ is a $\mathbf{t}$-*set* or *of type* $\mathbf{t}$ in $G(n, \boldsymbol{\alpha}, P)$ if $S \cap V_i = t_i$ for every $i \in [k]$. Vectors are denoted by lower-case bold letters. Given vectors $\mathbf{u}, \mathbf{v} \in \mathbb{N}_{\ge 0}^k$, we write $\mathbf{u} \le \mathbf{v}$ if $u_i \le v_i$ for all $i \in [k]$.

Our starting point and main technical tool in this section is a simple consequence of Janson's inequality already mentioned in Chapter 2. We restate it here for convenience.

Theorem 3.2 (Janson's inequality)**.** *Let $k \in \mathbb{N}$, $\boldsymbol{\alpha} \in (0,1]^k$ with $|\boldsymbol{\alpha}| = 1$, and let $P = (p_{ij})_{i,j\in[k]}$ be a symmetric matrix with all $p_{ij} \in (0,1)$. Consider a family $\{S_i\}_{i\in\mathcal{I}}$ of subsets of the vertex set $[n]$ and let $G \sim G(n, \boldsymbol{\alpha}, P)$. For each $i \in \mathcal{I}$, let X_i denote the indicator random variable for the event $\{S_i$ is an independent set in $G\}$ and, for each ordered pair $(i,j) \in \mathcal{I} \times \mathcal{I}$, write $X_i \sim X_j$ if $E(S_i) \cap E(S_j) \neq \varnothing$. Let*

$$X := \sum_{i\in\mathcal{I}} X_i, \qquad \mu := \mathbb{E}[X], \qquad \text{and} \qquad \overline{\Delta} := \sum_{\substack{(i,j)\in\mathcal{I}\times\mathcal{I}\\ X_i\sim X_j}} \mathbb{E}[X_i X_j].$$

Then

$$\Pr[X = 0] \leq e^{-\mu^2/(2\overline{\Delta})}.$$

The next lemma is the central result of this section. In simple terms, it states that w.h.p. whenever we take a sufficiently large subset of vertices of $G \sim G(n, \boldsymbol{\alpha}, P)$, there is an independent $\mathbf{t}$-set, for any $\mathbf{t}$ that "falls" within the set $\mathcal{A}$, that is, $\mathbf{t} \in (1-o(1)) \ln n \cdot \mathcal{A}$. This lemma alone allows us to greedily take out independent sets (color classes) from $G(n, \boldsymbol{\alpha}, P)$ as long as there is some "large" set of vertices remaining in each V_i.

Lemma 3.3. *Let $G \sim G(n, \boldsymbol{\alpha}, P)$. Let $\mathbf{s} \in \mathbb{N}_0^k$ be such that $s_i \geq \alpha_i n / \ln^2 n$ for all $i \in [k]$ and $S \subseteq V(G)$ be an $\mathbf{s}$-set. Then, for every $\mathbf{t} \in (\ln n - 7 \ln\ln n) \cdot \mathcal{A} \cap \mathbb{N}_0^k$ and $X_{\mathbf{t}}$ being the random variable denoting the number of independent $\mathbf{t}$-sets in $G[S]$*

$$\Pr[X_{\mathbf{t}} = 0] = e^{-\Omega(n^2/\ln^8 n)}.$$

Proof. Let us write $X_{\mathbf{t}} = \sum_{I\in\mathcal{S}} X_I$, where $\mathcal{S}$ is the family of all subsets of S that intersect each V_i, $i \in [k]$, in exactly t_i vertices, and X_I is an indicator random variable for the event that I is an independent set in $G[S]$. Set now $\mu := \mathbb{E}[X_{\mathbf{t}}]$ and $\overline{\Delta} := \sum_{(I,J)\in\mathcal{S}\times\mathcal{S}} \mathbb{E}[X_I X_J]$. This puts us directly into the setup of Janson's inequality (Theorem 3.2) with the goal to show

$$\Pr[X_{\mathbf{t}} = 0] \leq e^{-\mu^2/(2\overline{\Delta})} \stackrel{!}{=} e^{-\Omega(n^2/\ln^8 n)}.$$

The whole proof boils down to showing that the $\overline{\Delta}$ term can be bounded by

$$\overline{\Delta} = O\Big(\mu^2 \cdot \frac{\ln^8 n}{n^2}\Big), \tag{3.5}$$

which is what we accomplish in the remainder.

Firstly, it is convenient to determine $\mu = \mathbb{E}[X_{\mathbf{t}}]$ as it helps simplify some calculations. For each $i \in [k]$, there are $\binom{s_i}{t_i}$ choices for the intersection of a $\mathbf{t}$-set with $S \cap V_i$. Additionally, in order for such a $\mathbf{t}$-set to be an independent set in $G[S]$, none of the $\binom{t_i}{2}$ pairs can form an edge, which happens with probability $(1-p_i)^{\binom{t_i}{2}}$. Lastly, no two vertices u, v in the $\mathbf{t}$-set with $u \in V_i$ and $v \in V_j$ can form an edge in $G[S]$, which happens with probability $(1-p_{ij})^{t_i t_j}$. Putting all of this together, we directly get

$$\mu = \prod_{1 \le i \le k} \binom{s_i}{t_i} \cdot \prod_{1 \le i \le k} (1-p_i)^{\binom{t_i}{2}} \cdot \prod_{1 \le i < j \le k} (1-p_{ij})^{t_i t_j}. \tag{3.6}$$

We now turn our attention in bounding the $\overline{\Delta}$ term as promised. Note that the $\overline{\Delta}$ term depends only on those sets which "overlap" in at least two vertices. We denote this "overlap" vector by $\mathbf{o}$ and note that $\mathbf{o} \le \mathbf{t}$ and $|\mathbf{o}| \ge 2$. Each o_i, for $i \in [k]$, measures the "overlap" of the sets inside of the part V_i. For a fixed overlap vector $\mathbf{o}$ and a fixed $i \in [k]$, there are at most

$$\binom{s_i}{t_i}\binom{t_i}{o_i}\binom{s_i - t_i}{t_i - o_i}$$

choices for two $\mathbf{t}$-sets which intersect in exactly o_i vertices within V_i. Similarly as above when deriving the expectation, the probability of both such $\mathbf{t}$-sets being independent is given by a term for intra-class edges and inter-class edges and is exactly

$$(1-p_i)^{2\binom{t_i}{2} - \binom{o_i}{2}} \cdot \prod_{j \ne i} (1-p_{ij})^{2t_i t_j - o_i o_j}.$$

Thus, the contribution to the $\overline{\Delta}$ term of a fixed overlap vector $\mathbf{o}$ is given by

$$\prod_{1 \le i \le k} \binom{s_i}{t_i}\binom{t_i}{o_i}\binom{s_i - t_i}{t_i - o_i} \cdot \prod_{1 \le i \le k} (1-p_i)^{2\binom{t_i}{2} - \binom{o_i}{2}} \cdot \prod_{1 \le i < j \le k} (1-p_{ij})^{2t_i t_j - o_i o_j}.$$

Then, by summing up over all choices of $\mathbf{o}$, we get

$$\overline{\Delta} = \sum_{\substack{\mathbf{o}\le\mathbf{t}\\ |\mathbf{o}|\ge 2}} \Big(\prod_{1\le i\le k} \binom{s_i}{t_i}\binom{t_i}{o_i}\binom{s_i-t_i}{t_i-o_i} \cdot \prod_{1\le i\le k} (1-p_i)^{2\binom{t_i}{2}-\binom{o_i}{2}} \cdot \prod_{1\le i<j\le k} (1-p_{ij})^{2t_it_j-o_io_j}\Big)$$

$$\overset{(3.6)}{=} \mu^2 \cdot \sum_{\substack{\mathbf{o}\le\mathbf{t}\\ |\mathbf{o}|\ge 2}} \Big(\underbrace{\prod_{1\le i\le k} \frac{\binom{t_i}{o_i}\binom{s_i-t_i}{t_i-o_i}}{\binom{s_i}{t_i}} \cdot \prod_{1\le i\le k} (1-p_i)^{-\binom{o_i}{2}} \cdot \prod_{1\le i<j\le k} (1-p_{ij})^{-o_io_j}}_{:=f(\mathbf{o})} \Big)$$

$$= \mu^2 \cdot \sum_{\substack{\mathbf{o}\le\mathbf{t}\\ |\mathbf{o}|\ge 2}} f(\mathbf{o}). \quad (3.7)$$

To complete the proof we aim to give a bound on $\sum_{\mathbf{o}\le\mathbf{t},|\mathbf{o}|\ge 2} f(\mathbf{o})$ of the order $\ln^8 n/n^2$. We first show this the whole sum is essentially dominated by those $f(\mathbf{o})$ with $|\mathbf{o}| = 2$.

Consider an arbitrary $\mathbf{o}$ and let $\mathbf{e}_z \in \{0,1\}^k$ be a unit vector with $(\mathbf{e}_z)_z = 1$ for some $z \in [k]$. We derive

$$\frac{f(\mathbf{o})}{f(\mathbf{o}+\mathbf{e}_z)} \ge \frac{\binom{t_z}{o_z}\binom{s_z-t_z}{t_z-o_z}}{\binom{t_z}{o_z+1}\binom{s_z-t_z}{t_z-(o_z+1)}} \cdot (1-p_z)^{\binom{o_z+1}{2}-\binom{o_z}{2}} \cdot \prod_{j\ne z} (1-p_{zj})^{(o_z+1)o_j-o_zo_j}.$$

Using the fact that $t_i \le c(p_i)\ln n$ (by definition (3.3) of $\mathcal{A}$) and $s_i \ge \alpha_i n/\ln^2 n$, this can be simplified (by standard manipulations of binomial coefficients) to

$$\frac{f(\mathbf{o})}{f(\mathbf{o}+\mathbf{e}_z)} \ge \frac{\delta n}{\ln^4 n} \cdot (1-p_z)^{o_z} \cdot \prod_{j\ne z} (1-p_{zj})^{o_j}, \quad (3.8)$$

for some constant $\delta > 0$ which depends only on α_i's and p_i's. Let $\tilde{\mathbf{o}} \le \mathbf{t}$ be such that $|\tilde{\mathbf{o}}| = 2$ and $\tilde{\mathbf{o}} = \mathbf{e}_x + \mathbf{e}_y$, for some (not necessarily distinct) $x, y \in [k]$. Then from (3.8), for every $\tilde{\mathbf{o}} \le \mathbf{o} \le \mathbf{t}$ with $|\mathbf{o}| \ge 3$, we obtain

$$\frac{f(\tilde{\mathbf{o}})}{f(\mathbf{o})} \ge \Big(\frac{\delta n}{\ln^4 n}\Big)^{|\mathbf{o}|-2} \cdot \prod_{1\le i\le k} (1-p_i)^{\binom{o_i}{2}} \cdot \prod_{1\le i<j\le k} (1-p_{ij})^{o_io_j}.$$

This can be further bounded from below by

$$\exp\Big((|\mathbf{o}|-2)(\ln\delta-4\ln\ln n)-2\ln n \\ +\underbrace{\sum_{1\le i\le k} o_i\ln n+\frac{1}{2}\sum_{1\le i\le k} o_i^2\ln(1-p_i)+\sum_{1\le i<j\le k} o_io_j\ln(1-p_{ij})}_{:=h(\mathbf{o})}\Big). \tag{3.9}$$

Let $\mathbf{d}=\mathbf{o}-\tilde{\mathbf{o}}$ and $\varepsilon=7\ln\ln n/\ln n$. Since $\mathbf{d}\le\mathbf{o}\le\mathbf{t}$ and $\mathbf{t}\in(1-\varepsilon)\ln n\cdot\mathcal{A}$, using the definition of $\mathcal{A}$, we get

$$\sum_{1\le i\le k}\frac{d_i}{(1-\varepsilon)\ln n}+\frac{1}{2}\sum_{1\le i\le k}\frac{d_i^2}{(1-\varepsilon)^2\ln^2 n}\ln(1-p_i)+\sum_{1\le i<j\le k} \\ \frac{d_id_j}{(1-\varepsilon)^2\ln^2 n}\ln(1-p_{ij})\ge 0. \tag{3.10}$$

Multiplying the whole inequality by $(1-\varepsilon)^2\ln^2 n$ gives

$$h(\mathbf{d}):=\sum_{1\le i\le k} d_i\ln n+\frac{1}{2}\sum_{1\le i\le k} d_i^2\ln(1-p_i)+\sum_{1\le i<j\le k} d_id_j\ln(1-p_{ij}) \\ \ge\varepsilon|\mathbf{d}|\ln n. \tag{3.11}$$

On the other hand, since $\mathbf{d}=\mathbf{o}-\tilde{\mathbf{o}}$ and $\tilde{\mathbf{o}}=\mathbf{e}_x+\mathbf{e}_y$, we have

$$h(\mathbf{o})-2\ln n\ge h(\mathbf{d})-2\ln n+(|e_x|+|e_y|)\ln n \\ +\frac{1}{2}\big(2o_x\ln(1-p_x)+2o_y\ln(1-p_y)\big) \\ +\sum_{j\ne x} e_xo_j\ln(1-p_{xj})+\sum_{j\ne y} e_yo_j\ln(1-p_{yj}).$$

Therefore, $h(\mathbf{o})-2\ln n\ge h(\mathbf{d})-O(|\mathbf{o}|)$. By plugging in (3.11) into (3.9), and as $|\mathbf{o}|\ge 3$, we get

$$\frac{f(\tilde{\mathbf{o}})}{f(\mathbf{o})}\ge\exp\Big((|\mathbf{o}|-2)(\ln\delta-4\ln\ln n)+7(|\mathbf{o}|-2)\ln\ln n-O(|\mathbf{o}|)\Big) \\ \ge\ln^{3(|\mathbf{o}|-2)} n+o(\ln^{|\mathbf{o}|-2} n). \tag{3.12}$$

Clearly, by the fact that $\mathbf{t} \in (1-\varepsilon)\ln n \cdot \mathcal{A}$ and as $\mathcal{A}$ is bounded, the norm of $\mathbf{o}$ is at most $C \ln n$ for some (large) constant $C > 0$ depending only on α_i's and p_i's. (We may, e.g., choose C as $C = \sum_{1 \le i \le k} c(p_{ii})$.) This finally implies

$$\sum_{\substack{\mathbf{o} \le \mathbf{t} \\ |\mathbf{o}| \ge 2}} f(\mathbf{o}) = O\Big(\sum_{i=2}^{|\mathbf{t}|} k^{i-2} \ln^{-3(i-2)} n \sum_{|\tilde{\mathbf{o}}|=2} f(\tilde{\mathbf{o}}) \Big) = O\Big(\sum_{|\tilde{\mathbf{o}}|=2} f(\tilde{\mathbf{o}}) \Big).$$

Hence, it remains to show that $f(\tilde{\mathbf{o}}) = O(\ln^8 n / n^2)$. Note that each such $\tilde{\mathbf{o}}$ with $|\tilde{\mathbf{o}}| = 2$ can be represented as $\tilde{\mathbf{o}} = \mathbf{e}_x + \mathbf{e}_y$ for some $x, y \in [k]$, where $\mathbf{e}_i$ stands for a unit vector $\mathbf{e}_i \in \{0,1\}^k$ with $(\mathbf{e}_i)_i = 1$. In case $x \ne y$ with $\tilde{o}_x = 1$ and $\tilde{o}_y = 1$, we have

$$f(\tilde{\mathbf{o}}) \le t_x \frac{\binom{s_x - t_x}{t_x - 1}}{\binom{s_x}{t_x}} \cdot t_y \frac{\binom{s_y - t_y}{t_y - 1}}{\binom{s_y}{t_y}} \cdot (1 - p_{xy})^{-1}.$$

Simple manipulations with binomial coefficients give

$$f(\tilde{\mathbf{o}}) \le \frac{(t_x t_y)^2}{s_x s_y (1 - p_{xy})}.$$

On the other hand, if $x = y$ and $\tilde{o}_x = 2$, then

$$f(\tilde{\mathbf{o}}) = \binom{t_x}{2} \frac{\binom{s_x - t_x}{t_x - 2}}{\binom{s_x}{t_x}} \cdot (1 - p_x)^{-1} \le \frac{t_x^4}{s_x^2 (1 - p_x)}.$$

Recalling that $s_i \ge \alpha_i n / \ln^2 n$ and $t_i = O(\ln n)$ shows the desired bound and completes the proof of the lemma. □

The next lemma establishes the fact that $G(n, \boldsymbol{\alpha}, P)$ w.h.p. contains no independent $\mathbf{t}$-sets which lie "outside" of $\mathcal{A}$. We start by making a useful observation about $\mathcal{A}$.

Claim 3.4. *There exists a constant $C = C(P) > 0$ such that any vector $\mathbf{c} \in \mathcal{A}^k_{\ge 0}$ with $|\mathbf{c}| \le C$ is contained in $\mathcal{A}$.*

Proof. For any $I \subseteq [k]$, we have

$$g(\mathbf{c}, I) \ge \sum_{i \in I} c_i + \frac{1}{2} \sum_{i', j'} c_{i'} c_{j'} \ln(1 - \max_{i,j} p_{ij}) \ge |\mathbf{c}| + \frac{1}{2} |\mathbf{c}|^2 \ln(1 - \max_{i,j} p_{ij}).$$

It follows that $g(\mathbf{c}, I) \ge 0$ for all $I \subseteq [k]$ if $|\mathbf{c}| \le C := -2/\ln(1 - \max_{i,j} p_{ij})$. □

Lemma 3.5. *Let $G \sim G(n, \boldsymbol{\alpha}, P)$. Let X be the random variable denoting the number of independent $\mathbf{t}$-sets in G with $\mathbf{t} \in \mathbb{N}_0^k \setminus (\ln n \cdot \mathcal{A})$. Then*

$$\Pr[X > 0] \leq O\big(n^{-1}\big).$$

Proof. Fix any $\mathbf{t} \notin \mathbb{N}_0^k \setminus (\ln n \cdot \mathcal{A})$ and let $X_{\mathbf{t}}$ count the number of independent $\mathbf{t}$-sets for that fixed $\mathbf{t}$. Observe that, by the definition (3.3) of $\mathcal{A}$, there is a $I \subseteq [k]$ such that $g\big((t_i/\ln n)_{i\in I}, I\big) < 0$. Let $\tilde{\mathbf{t}} = (\tilde{t}_1, \ldots, \tilde{t}_k)$ be defined as

$$\tilde{t}_i = \begin{cases} t_i, & \text{if } i \in I, \\ 0, & \text{otherwise,} \end{cases}$$

and let $X_{\tilde{\mathbf{t}}}$ be the random variable denoting the number of independent $\tilde{\mathbf{t}}$-sets in G. Then

$$\mathbb{E}[X_{\tilde{\mathbf{t}}}] = \prod_{i\in I} \binom{\alpha_i n}{t_i} \cdot \prod_{i\in I} (1-p_i)^{\binom{t_i}{2}} \cdot \prod_{\substack{i,j\in I \\ i\neq j}} (1-p_{ij})^{t_i t_j}.$$

Since $\alpha_i < 1$, using that $\binom{n}{k} \leq (\frac{en}{k})^k$, this can further be bounded by

$$\mathbb{E}[X_{\tilde{\mathbf{t}}}] \leq \exp\Big(\sum_{i\in I} t_i + \sum_{i\in I} t_i \ln n - \sum_{i\in I} t_i \ln t_i + \frac{1}{2} \sum_{i\in I} t_i^2 \ln(1-p_i) + \sum_{\substack{i,j\in I \\ i\neq j}} t_i t_j \ln(1-p_{ij}) \Big).$$

Recall, $g\big((t_i/\ln n)_{i\in I}, I\big) < 0$, and thus

$$\sum_{i\in I} \frac{t_i}{\ln n} + \frac{1}{2} \sum_{i\in I} \frac{t_i^2}{\ln^2 n} \ln(1-p_i) + \sum_{\substack{i,j\in I \\ i\neq j}} \frac{t_i t_j}{\ln^2 n} \ln(1-p_{ij}) < 0.$$

which yields

$$\frac{1}{2} \sum_{i\in I} t_i^2 \ln(1-p_i) + \sum_{\substack{i,j\in I \\ i\neq j}} t_i t_j \ln(1-p_{ij}) < -\sum_{i\in I} t_i \ln n.$$

By Jensen's inequality and the fact that $|\tilde{\mathbf{t}}| = \Omega(\ln n)$ by Claim 3.4, it further follows that

$$\mathbb{E}[X_{\tilde{\mathbf{t}}}] \leq \exp\big(-|\tilde{\mathbf{t}}| \ln(|\tilde{\mathbf{t}}|/k) + |\tilde{\mathbf{t}}| \big) \leq O(n^{-2}).$$

Hence, by Markov's inequality $\Pr[X_{\tilde{\mathbf{t}}} > 0] \le \mathbb{E}[X_{\tilde{\mathbf{t}}}] \le O(n^{-2})$.

Let $\mathcal{M}$ be the set of minimal $\mathbf{t}$'s in $\mathbb{N}_0^k \setminus (\ln n \cdot \mathcal{A})$. Then if there is any independent set in $\mathbb{N}_0^k \setminus (\ln n \cdot \mathcal{A})$ it must also contain an $\mathbf{m}$-set as a subset, for some $\mathbf{m} \in \mathcal{M}$. Similarly as in the proof of Lemma 3.3 (as $\mathcal{A}$ is bounded), we know $|\mathbf{m}| \le C \ln n$ for some $C > 0$ depending only on α_i's and p_i's. Therefore, $|\mathcal{M}| \le (C \ln n)^k$ and by a union bound over all vectors in $\mathcal{M}$, we get

$$\Pr[X > 0] \le \sum_{\mathbf{m} \in \mathcal{M}} \Pr\big[X_{\mathbf{m}} > 0\big] \le O(n^{-1}).$$

In particular, w.h.p. there is also no independent $\mathbf{t}$-set for any $\mathbf{t} \in \mathbb{N}_0^k \setminus (\ln n \cdot \mathcal{A})$, which completes the proof. □

3.3 Chromatic Number

In this section we provide the proof of our main theorem. Recall, the goal is to give a precise (up to lower order terms) bound on the chromatic number of a random block graph $G(n, \boldsymbol{\alpha}, P)$. For the convenience of the reader we restate our main result.

Theorem 3.1. *Let $k \ge 1$, $\boldsymbol{\alpha} \in (0,1]^k$ with $|\boldsymbol{\alpha}| = 1$, and let $P = (p_{ij})_{i,j \in [k]}$ be a symmetric matrix with all $p_{ij} \in (0,1)$. Consider a random block graph $G \sim G(n, \boldsymbol{\alpha}, P)$. Then w.h.p.*

$$\chi(G) = (1 + o(1)) \frac{n}{c^\star \ln n},$$

where $c^\star$ is given as

$$c^\star = c^\star(\boldsymbol{\alpha}, P) = \max\big\{|\mathbf{c}| : \mathbf{c} \in \operatorname{conv}(\mathcal{A}) \cap \{t \cdot \boldsymbol{\alpha} : t \in \mathbb{R}_{\ge 0}\}\big\}. \tag{3.4}$$

3.3.1 Upper bound

Set $\varepsilon = 7 \ln\ln n / \ln n$ and let $\mathbf{c}^\star = \arg\max\big\{|\mathbf{c}| : \mathbf{c} \in \operatorname{conv}(\mathcal{A}) \cap \{\boldsymbol{\alpha} t : t \in calA_{\ge 0}\}\big\}$. While constructing a coloring to give an upper bound on $\chi(G)$ we need to distinguish two cases: $\mathbf{c}^\star \in \mathcal{A}$ and $\mathbf{c}^\star \notin \mathcal{A}$.

Assume the former, that is, $\mathbf{c}^\star \in \mathcal{A}$ and let $\mathbf{t} = (1 - \varepsilon) \ln n \cdot \mathbf{c}^\star$. The probability that there is a set $S \subseteq V(G)$ with $s_i := |S \cap V_i| = \alpha_i n / \ln^2 n$

for all $i \in [k]$ and such that $G[S]$ does not contain an independent $\mathbf{t}$-set is, by Lemma 3.3 and a union bound over all choices for S, at most

$$\prod_{1\le i\le k} \binom{\alpha_i n}{s_i} e^{-\Omega(n^2/\ln^8 n)} \le e^{\sum_{i=1}^k s_i \ln n} \cdot e^{-\Omega(n^2/\ln^8 n)} = o(1).$$

As a direct consequence, w.h.p. as long as there are at least $\alpha_i n/\ln^2 n$ vertices remaining in every V_i, there is an independent $\mathbf{t}$-set where $\mathbf{t} = (1-\varepsilon)\ln n \cdot \mathbf{c}^\star$. We construct the coloring in the usual way: repeatedly take out an independent $\mathbf{t}$-set and assign all the vertices in it a new color. By the argument above, this is possible until there are $n/\ln^2 n$ vertices left in total, at which point we assign to each of those uncolored vertices a color different from all the previously used ones. Therefore, the total number of colors used is at most

$$\frac{n}{|\mathbf{t}|} + \frac{n}{\ln^2 n} = \frac{n}{|\mathbf{c}^\star|(\ln n - 7\ln\ln n)} + \frac{n}{\ln^2 n} = (1+o(1))\frac{n}{c^\star \ln n},$$

as claimed.

On the other hand, if $\mathbf{c}^\star \notin \mathcal{A}$, w.h.p. no independent $(\mathbf{c}^\star \ln n)$-set exists in G by Lemma 3.5. In order to circumvent this, we represent $\mathbf{c}^\star$ as a convex combination

$$\mathbf{c}^\star = \sum_{1\le i\le k+1} \lambda_i \mathbf{t_i}, \tag{3.13}$$

where $\mathbf{t_i} \in \mathcal{A}$, $\lambda_i \in [0,1]$ for all $i \in [k+1]$, and $\sum_{1\le i\le k+1} \lambda_i = 1$.

We then construct a coloring of G as follows. Greedily and sequentially select $\lambda_i n/(c^\star \ln n)$ independent $(\mathbf{t_i}(1-\varepsilon)\ln n)$-sets and assign all the vertices in it a new color. We can indeed do this as even after taking all such sets, the number of vertices remaining in every V_i is

$$\begin{aligned}\alpha_i n - \sum_{1\le j\le k+1} \frac{\lambda_j n}{c^\star \ln n}\cdot(\mathbf{t_j})_i \cdot (1-\varepsilon)\ln n &= \alpha_i n - \frac{(1-\varepsilon)n}{c^\star}\cdot \sum_{1\le j\le k+1} \lambda_j (\mathbf{t_j})_i \\ &\overset{(3.13)}{=} \alpha_i n - \frac{(1-\varepsilon)n}{c^\star}\cdot(\mathbf{c}^\star)_i = \frac{7\alpha_i n \ln\ln n}{\ln n},\end{aligned} \tag{3.14}$$

where the last equality follows from the fact that $\mathbf{c}^\star = c^\star \cdot \boldsymbol{\alpha}$ and our choice of ε.

Let Q_i be the set of uncolored vertices in every V_i. Since $G[V_i]$ is distributed as $G(\alpha_i n, p_i)$ and Q_i is a subset of V_i of size $|Q_i| \ge \varepsilon \alpha_i n$ w.h.p.

by Lemma 3.3 it follows that as long as there are at least $\alpha_i n/\ln^2 n$ vertices remaining, generously rounding for $\boldsymbol{\alpha}$, we find an independent set of size at least $c(p_i)\ln n/2$ in $G[Q_i]$. We greedily take such sets one by one and assign all the vertices in each a new color. Lastly, assign every uncolored vertex a new color which was previously unused. Therefore,

$$\chi(G[Q_i]) \leq \frac{14\alpha_i n \ln\ln n}{c(p_i)\ln^2 n} + \frac{n}{\ln^2 n} = o\Big(\frac{n}{\ln n}\Big).$$

Consequently, the number of different colors used for the whole graph G is at most

$$\sum_{1\leq i\leq k+1} \frac{\lambda_i n}{c^\star \ln n} + \sum_{1\leq i\leq k} \chi(G[Q_i]) = \frac{n}{c^\star \ln n} + o\Big(\frac{n}{\ln n}\Big),$$

as $\sum_{1\leq i\leq k+1}\lambda_i = 1$. This confirms the claimed upper bound.

3.3.2 Lower bound

Set $N = \chi(G)$ and consider any coloring with color classes $S_1,\ldots,S_N$. Trivially, for every $j\in[k]$ we have $\sum_{1\leq i\leq N}|S_i\cap V_j| = \alpha_j n$. So by Lemma 3.5 we may assume every color class is an independent $\mathbf{t}$-set for some $\mathbf{t}\in \ln n\cdot\mathcal{A}$. Let $\bar{\mathbf{t}} = \frac{1}{N}\sum_{i\in[N]}\mathbf{t}_i$ and note that $\bar{\mathbf{t}}\in\{t\cdot\boldsymbol{\alpha}\}$ for some $t\in$
$calA_{\geq 0}$ and $\bar{\mathbf{t}}\in\operatorname{conv}(\mathcal{A})$. Therefore,

$$N = \frac{n}{|\bar{\mathbf{t}}|\ln n} \geq \frac{n}{c^\star \ln n},$$

by maximality of $c^\star$ (see (3.4)). □

3.4 Special Cases

3.4.1 Two-block case

Throughout this subsection we assume that $k=2$ and try to in detail describe the possible structure of the set $\mathcal{A}$ and independent sets of $G(n,\boldsymbol{\alpha},P)$. Recall, the set $\mathcal{A}$ is defined in order to describe "feasible" sizes of independent sets the graph $G(n,\boldsymbol{\alpha},P)$ can have and in the case

$k = 2$ is given as follows:

$$\mathcal{A} = \big\{0 \le c_1 \le c(p_1), 0 \le c_2 \le c(p_2), \\ c_1 + c_2 + \frac{c_1^2}{2}\ln(1-p_1) + \frac{c_2^2}{2}\ln(1-p_2) + c_1c_2\ln(1-p_{12}) \ge 0\big\}.$$

The first two equations determine the size of the largest independent set inside each of the parts V_1 and V_2 on their own by treating them as $G(\alpha_1 n, p_1)$ and $G(\alpha_2 n, p_2)$, respectively. The third inequality is what determines the shape of $\mathcal{A}$.

In particular, having p_1 and p_2 fixed, the shape of $\mathcal{A}$ varies significantly depending on p_{12}. Note that, by using (3.1), the inequality

$$c_1 + c_2 + \frac{c_1^2}{2}\ln(1-p_1) + \frac{c_2^2}{2}\ln(1-p_2) + c_1c_2\ln(1-p_{12}) \ge 0$$

can be rewritten as

$$(c_1+c_2)\Big(1 - \frac{c_1}{c(p_1)} - \frac{c_2}{c(p_2)}\Big) + c_1c_2\Big(\ln(1-p_{12}) + \frac{1}{c(p_1)} + \frac{1}{c(p_2)}\Big) \ge 0. \tag{3.15}$$

If we set c_1 or c_2 to 0 we quickly see that $(c(p_1), 0)$ and $(0, c(p_2))$ are points on the boundary of $\mathcal{A}$. If we set (3.15) to be zero we get the boundary between $(c(p_1), 0)$ and $(0, c(p_2))$, which must be part of a conic section as it satisfies a quadratic equation. So it must be concave or convex. In particular it is enough to check whether the points on the line between $(c(p_1), 0)$ and $(0, c(p_2))$, the line represented by $1 - c_1/c(p_1) - c_2/c(p_2) = 0$, are in $\mathcal{A}$ or not. Because we are only considering points where c_1 and c_2 are positive, this line is contained in $\mathcal{A}$ if and only if the second term of (3.15) is positive, that is, if $\ln(1-p_{12}) + 1/c(p_1) + 1/c(p_2) \ge 0$, or, equivalently, $p_{12} \le 1 - \sqrt{(1-p_1)(1-p_2)}$. Then and only then is $\mathcal{A}$ convex. In this case, the constant $c^\star$ defined as

$$c^\star = \max\big\{|\mathbf{c}| : \mathbf{c} \in \operatorname{conv}(\mathcal{A}) \cap \{\boldsymbol{\alpha} t : t \in \mathbb{R}_{\ge 0}\}\big\}$$

is given by a vector $\mathbf{c}^\star$ which actually belongs to the set $\mathcal{A}$ itself. In other words, independent $\mathbf{t}$-sets with $t_1 = c_1^\star(1-o(1))\ln n$ and $t_2 = c_2^\star(1-o(1))\ln n$ w.h.p. exist in $G(n, \boldsymbol{\alpha}, P)$, and a coloring can be found by greedily picking these sets as long as possible and then coloring all remaining vertices with a new color.

On the other hand, if $p_{12} > 1 - \sqrt{(1-p_1)(1-p_2)}$, the situation is quite different. In this case, the set $\mathcal{A}$ is *concave* and the vector $\mathbf{c}^\star$ which

determines the constant $c^\star$ does not belong to the set $\mathcal{A}$, but lies on its convex hull. In particular, it is given by

$$\mathbf{c}^\star = \Big(\frac{\alpha_1}{\frac{\alpha_1}{c(p_1)} + \frac{\alpha_2}{c(p_2)}}, \frac{\alpha_2}{\frac{\alpha_1}{c(p_1)} + \frac{\alpha_2}{c(p_2)}} \Big).$$

The optimal coloring is then achieved by looking at the "extremal points" $\mathbf{c_1} = (c(p_1), 0)$ and $\mathbf{c_2} = (0, c(p_2))$ and using independent $(\mathbf{c_1}(1-\varepsilon)\ln n)$-sets and $(\mathbf{c_2}(1-\varepsilon)\ln n)$-sets as color classes. Perhaps unsurprisingly, it is shown in Proposition 3.6 below that w.h.p. the chromatic number of $G(n, \boldsymbol{\alpha}, P)$ is then and only then the sum of the chromatic number of the two parts $G[V_1]$ and $G[V_2]$, that is

$$\begin{aligned}\chi(G(n, \boldsymbol{\alpha}, P)) &= \big(1 + o(1)\big)\big(\chi(G(\alpha_1 n, p_1)) + \chi(G(\alpha_2 n, p_2))\big) \\ &= \big(1 + o(1)\big)\frac{\alpha_1 c(p_2) + \alpha_2 c(p_1)}{c(p_1)c(p_2)}\frac{n}{\ln n}. \quad (3.16)\end{aligned}$$

For $p_{12} = 1 - \sqrt{(1-p_1)(1-p_2)}$ we have that $\mathcal{A}$ is both convex and concave since it is limited by a line—so any convex combination of $\mathbf{c}$-sets along this line yields a correct chromatic number asymptotically.

The shape of the set $\mathcal{A}$ for fixed $0 < p_1 \le p_2 < 1$ and depending on $p_{12} \in (0, 1)$ is depicted on Figure 3.2 below.

Clearly, the constant $c^\star$ and the vector $\mathbf{c}^\star$ that defines it do not only depend on the set $\mathcal{A}$ but also on the vector $\boldsymbol{\alpha}$. In Figure 3.3 we show how the vector $\mathbf{c}^\star$ is defined.

Worth noting is that if $p_{12} < 1 - \sqrt{(1-p_1)}$ or $p_{12} < 1 - \sqrt{(1-p_2)}$ the inequalities $c_i < c(p_i)$ become relevant for the shape of $\mathcal{A}$. What this means for the chromatic number is that if

$$p_{12} \le 1 - \sqrt{(1-p_1)} \quad \text{and} \quad \alpha_1 \le c(p_1)\big(\ln(1-p_{12}) - \frac{1}{2}\ln(1-p_2)\big),$$

then the vertex set V_1 has become so sparse and small, that we can color it for free. So the chromatic number of $G(n, \boldsymbol{\alpha}, P)$ is asymptotically bounded by the chromatic number of $G[V_2]$ and is therefore

$$\chi(G(n, \boldsymbol{\alpha}, P)) = \big(1 + o(1)\big)\frac{\alpha_2 n}{c(p_2)\ln n}.$$

The equivalent holds if $p_{12} \le 1 - \sqrt{(1-p_2)}$ and $\alpha_2 \le c(p_2)\big(\ln(1 - p_{12}) - (1/2)\ln(1-p_1)\big)$ for the vertex set V_2.

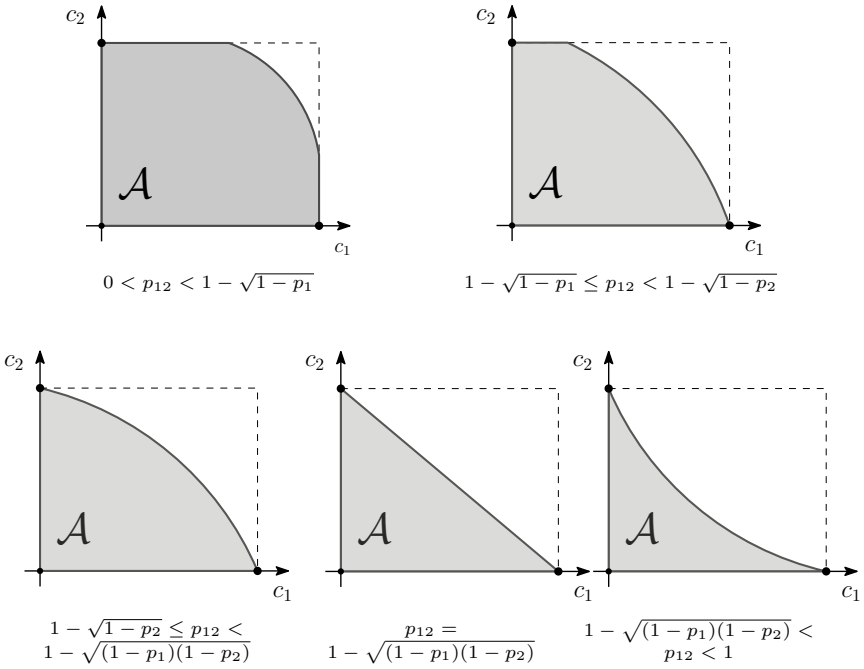

Figure 3.2: Possibilities for $\mathcal{A}$ in case $k = 2$, assuming $p_1 \le p_2$ and then varying p_{12}. The dashed lines correspond to $c(p_1)$ and $c(p_2)$, that is, $c_1 = \frac{2}{-\ln(1-p_1)}$ and $c_2 = \frac{2}{-\ln(1-p_2)}$.

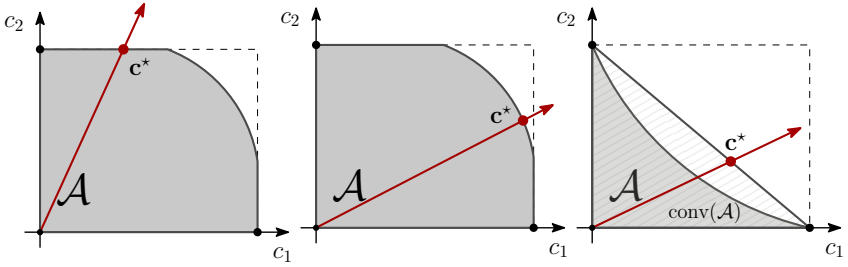

Figure 3.3: The red line represents the vector $\boldsymbol{\alpha}$. The red point represents its intersection with conv($\mathcal{A}$), i.e. the vector $\mathbf{c}^\star$.

3.4.2 Concave set and the union of random graphs

In this subsection we further explore how the random block graph $G(n, \boldsymbol{\alpha}, P)$ behaves in the general case $k \geq 2$ and when

$$p_{ij} > 1 - \sqrt{(1-p_i)(1-p_j)} \qquad \text{for all } \ 1 \leq i < j \leq k.$$

In other words, when all of the bipartite graphs between the parts $G[V_i, V_j]$ are significantly denser than the densest graph $G[V_i]$. In case $k = 2$, this is depicted on the rightmost parts of Figure 3.2 and Figure 3.3.

Proposition 3.6. *With high probability*

$$\chi(G(n, \boldsymbol{\alpha}, P)) = \big(1 + o(1)\big)\Big(\sum_{1 \leq i \leq k} \chi(G(\alpha_i n, p_i))\Big)$$

if and only if $p_{ij} \geq 1 - \sqrt{(1-p_i)(1-p_j)}$ *for all* $1 \leq i < j \leq k$.

Proof. We first show that $p_{ij} \geq 1 - \sqrt{(1-p_i)(1-p_j)}$ for all $1 \leq i < j \leq k$ implies the desired bound on the chromatic number. Recall, for a vector $\mathbf{c} \in$
$calA^k$ and $I \subseteq [k]$, the function $g(\mathbf{c}, I)$ is defined as

$$g(\mathbf{c}, I) = \sum_{i \in I} c_i + \frac{1}{2} \sum_{i,j \in I} c_i c_j \ln(1 - p_{ij}).$$

Note that we can reformulate this as

$$g(\mathbf{c}, I) = \Big(\sum_{i \in I} c_i\Big)\Big(1 - \sum_{i \in I} \frac{c_i}{c(p_i)}\Big) + \sum_{i \neq j \in I} c_i c_j \Big(\ln(1-p_{ij}) + \frac{1}{c(p_i)} + \frac{1}{c(p_j)}\Big).$$

By assumption of $p_{ij} \geq 1 - \sqrt{(1-p_i)(1-p_j)}$ we have that the term $\ln(1-p_{ij}) + 1/c(p_i) + 1/c(p_j)$ is negative or zero for all $i, j \in [k]$. Consequently, $\mathcal{A} \subseteq \mathcal{B} \coloneqq \{\mathbf{c} \in$
$calA^k_{\geq 0} : 1 - \sum_{i \in [k]} \frac{c_i}{c(p_i)} \geq 0\}$ and $\mathcal{B}$ has as boundary a hyperplane and therefore is a convex set.

For every $i \in [k]$, let

$$\mathbf{t_i} = c(p_i) \cdot \mathbf{e}_i, \qquad h = \sum_{1 \leq j \leq k} \frac{\alpha_j}{|\mathbf{t_j}|}, \qquad \text{and} \qquad \lambda_i = \frac{\alpha_i}{h|\mathbf{t_i}|} \tag{3.17}$$

Note that each $\lambda_i \in [0,1]$ and $\sum_{1\le i\le k}\lambda_i = 1$. We claim that we can represent $\mathbf{c}^\star$ as a convex combination of $\mathbf{t_i}$'s like

$$\mathbf{c}^\star = \sum_{1\le i\le k}\lambda_i \mathbf{t_i}.$$

Observe that, by definition

$$c^\star = |\mathbf{c}^\star| = \sum_{1\le i\le k}\lambda_i|\mathbf{t_i}| = \sum_{1\le i\le k}\frac{\alpha_i}{h} = \frac{1}{h}. \tag{3.18}$$

Each $\mathbf{t_i} \in \mathcal{A} \subseteq \mathcal{B}$ is the intersection point of the hyperplane of $\mathcal{B}$ with the corresponding axis. So, in fact, $\mathcal{A} \subseteq \mathrm{conv}(\mathcal{A}) = \mathcal{B}$ and since $\mathbf{c}^\star$ is a linear combination of the $\mathbf{t_i}$'s it must lie on the boundary of $\mathcal{B}$. That gives the upper bound on $\mathbf{c}^\star$ and $\mathbf{c}^\star \in \mathcal{B}$ gives the lower bound. The rest now follows from the same strategy as in Theorem 3.1 and the fact that the number of different colors used is at most

$$\sum_{1\le i\le k}\frac{\alpha_i n}{(1-\varepsilon)|\mathbf{t_i}|\ln n} + o\Big(\frac{n}{\ln n}\Big) \overset{(3.17)}{=} h\cdot\frac{n}{(1-\varepsilon)\ln n} + o\Big(\frac{n}{\ln n}\Big)$$
$$\overset{(3.18)}{=} \frac{n}{c^\star \ln n} + o\Big(\frac{n}{\ln n}\Big). \tag{3.19}$$

As for the other direction, whenever there is a $p_{ij} < 1-\sqrt{(1-p_i)(1-p_j)}$ for a fixed $i \neq j \in [k]$ we can color $G(n,\boldsymbol{\alpha},P)$ in the following way. For every $h \in [k]\backslash\{i,j\}$ color each V_h separately with $\chi(G(\alpha_h n_h, p_h))$ colors. Then look at the graph induced by $V_i \cup V_j$. Clearly, $G[V_i \cup V_j]$ is distributed as the block graph $G(\alpha_i n_i + \alpha_j n_j, \boldsymbol{\alpha}', P')$, where $\boldsymbol{\alpha}' = (\alpha_i, \alpha_j)$ and $P' = \left(\begin{smallmatrix} p_{ii} & p_{ij} \\ p_{ji} & p_{jj}\end{smallmatrix}\right)$. Our observations from analyzing the two-block case in Section 3.4.1 tell us that we can w.h.p. color this graph with asymptotically less colors than the sum of the chromatic numbers of the parts thus proving the proposition. □

3.4.3 Convex set with homogeneous balanced partition

The case where $\mathcal{A}$ is convex can quickly turn out to be quite complicated. Perhaps one of the cases worth mentioning is when the probability matrix P contains only two different values, one for the diagonal and one for the off-diagonal, and additionally $|V_1| = \cdots = |V_k| = n/k$. So, for P

we have

$$p_{ii} = p \quad \forall i \in [k] \qquad \text{and} \qquad p_{ij} = q \quad \forall i \neq j \in [k],$$

with $p \geq q$. Then $\mathcal{A}$ takes a convex shape since all the equations form convex sets and the vector $\mathbf{c}^\star$ must on the boundary of $\mathcal{A}$. In other words, $\mathbf{c}^\star = (\frac{c^\star}{k}, \ldots, \frac{c^\star}{k})$ where $c^\star = |\mathbf{c}^\star|$ and, in particular, it must hold that

$$\sum_{i\in[k]} \frac{c^\star}{k} + \frac{1}{2} \sum_{i\in[k]} \Big(\frac{c^\star}{k}\Big)^2 \ln(1-p) + \sum_{1\leq i<j\leq k} \Big(\frac{c^\star}{k}\Big)^2 \ln(1-q) \leq 0.$$

By rearranging we get that $c^\star \leq -2/\big(\frac{1}{k}\ln(1-p) + \frac{k-1}{k}\ln(1-q)\big)$. Therefore, in case the previous is satisfied with an equality, since $p \geq q$ and due to $\alpha_i = 1/k$, the equations $g(\cdot, I)$ are automatically satisfied for all subsets of the indices $I \subseteq [k]$, and hence $\mathbf{c}^\star$ is maximal and in $\mathcal{A}$. Applying Theorem 3.1, we get

$$\chi(G(n, \boldsymbol{\alpha}, P)) = \big(1 + o(1)\big) \frac{n}{2\ln n} \Big(-\frac{1}{k}\ln(1-p) - \frac{k-1}{k}\ln(1-q)\Big).$$

Chapter 4

K_r-Factors in Graphs with Low Independence Number

In this chapter we prove minimum degree conditions for existence of a K_r-factor in a graph with independence number $\alpha(G) = o(n)$. We apply tools from random graph theory and also implicitly the Szémeredi's Regularity Lemma. This shows how the tools extend to applications in problems which have a priori nothing to do with randomness. The content of this chapter is based on a paper published by Knierim and the author ([KS21]).

4.1 Introduction

The problem we look at in this chapter comes from taking a closer look at the Hajnal and Szemerédi theorem for graph factors. As mentioned in Chapter 1 an H-factor in a graph G is a subgraph of G such that the subgraph covers all vertices of G and is the disjoint union of graphs H. A perfect matching therfore corresponds to a K_2-factor, thus the notion of H-factors is a natural generalization of perfect matchings from edges to arbitrary graphs. For perfect matchings, necessary and sufficient conditions are well-known by Hall's and Tutte's theorems. Ideally we want to formulate such conditions for other subgraph problems.

Often, *global* properties such as factors and Hamilton cycles have *local* necessary conditions. Dirac [Dir52] showed that if an n-vertex graph G has minimum degree at least $n/2$, then it has a Hamilton cycle, in particular if n is even then G has a perfect matching. This was extended to triangle factors by Corrádi and Hajnal [CH63] in 1963 and later generalized to K_r-factors in a classical result by Hajnal and Szemerédi [HS70], who gave the sufficient minimum degree for K_r-factors.

Theorem 4.1 (Hajnal and Szemerédi)**.** *For every graph on n vertices, given an integer $r \geq 2$, if r divides n and the minimum degree of G is at least $\left(1 - \frac{1}{r}\right) n$, then G contains a K_r-factor.*

A short proof was later found by Kierstead and Kostochka [KK08]. The divisibility condition in this theorem is necessary as the vertex set must be divisible by $|H|$ if we want to have an H-factor. The theorem is also tight in a sense that we can not lower the minimum degree condition and still hope to cover any n-vertex graph.

Different results relating to the theorem of Hajnal and Szemerédi have been published. A degree sequence version of the result was published by Treglown [Tre16] proving that, for a $(1/r)$-fraction of the vertices, the degrees can be smaller than prescribed by the Hajnal-Szemerédi theorem. Other results include the minimum degree condition in a 3-partite [MM02], 4-partite [MS08] or multi-partite [KM13] host graph. In each of these results, the known extremal examples all have one or more large independent sets. Naturally the question arises, what happens if we forbid these large independent sets?

To cover any n-vertex graph with independence number at least $n/r+1$ with cliques of arbitrary size we need at least $n/r+1$ cliques, as no clique can contain more than one vertex from the independent set. Taking an

independent set of size exactly $n/r + 1$ and adding edges from each of the remaining vertices to all other vertices gives a graph that does not have a K_r-factor and minimum degree $n - (n/r + 1) = (1 - 1/r)n - 1$.

Problems of this form were first studied in *Ramsey-Turán* theory, elaborated on in Section 1.5, Continuing this line of research, Balogh, Molla and Sharifzadeh [BMS16] proved that the minimum degree requirement for a triangle factor in G decreases if the independence number of G is small showing that $\delta(G) \geq 1/2 + \varepsilon$ suffices in this case. Nenadov and Pehova [NP18] extended their result to larger cliques and a generalization of the independence number. They show that instead of $\delta(G) \geq \left(1 - \frac{1}{r}\right) n$ one only needs roughly $\delta(G) \geq \left(1 - \frac{1}{r-1}\right) n$ for the existence of a K_r-factor if we restrict the independence number of G to be sub-linear.

We further improve the minimum degree condition, doubling the clique size compared to the Hajnal-Szemerédi theorem. We see in the following that this is best possible.

Theorem 4.2. *For every $r \geq 4$ and $\mu > 0$ there are constants γ and $n_0 \in \mathbb{N}$ such that every graph G on $n \geq n_0$ vertices where r divides n, with $\delta(G) \geq \left(1 - \frac{2}{r} + \mu\right) n$ and $\alpha(G) < \gamma n$ has a K_r-factor.*

Note that the bound is not true for $r = 2, 3$. Balogh, Molla and Sharifzadeh [BMS16] observed that a minimum degree of $(1/2+\varepsilon)n$ is needed in the case $r = 3$. This can be seen by considering graphs with a bipartition such that there are no triangles which span over both parts. In particular, for n divisible by 4, the graph $K_{n/2+1} \cup K_{n/2-1}$, the union of two disjoint cliques, has independence number 2 and minimum degree $n/2 - 2$ but does not contain a perfect matching nor a triangle factor because $n/2 - 1$ and $n/2 + 1$ are both odd and cannot both be divisible by three. Balogh, McDowell, Molla and Mycroft [BMMM18] showed that the minimum degree condition can be lowered if an additional divisibility condition on the triangle connected components is added to avoid exactly this case.

The tightness of the Hajnal-Semerédi theorem comes from large independent sets. So what is the bottleneck if we forbid these? By definition, $\alpha(G) = o(1)$ implies that every set of linear size has at least one edge inside, but if we have a large triangle-free set then we can take at most two vertices from this set for every clique. In particular, if we have an n-vertex graph with a triangle-free set of size $2n/r + 1$ then we cannot

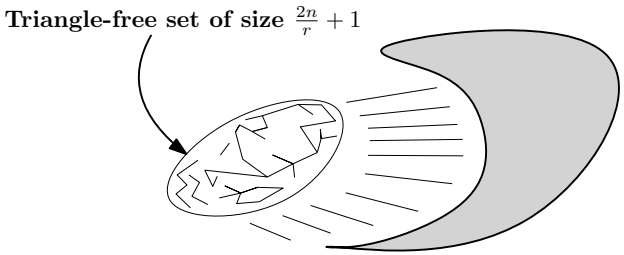

Figure 4.1: An extremal example showing that we cannot improve the degree condition below $\left(1-\frac{2}{r}\right)n$.

hope to find a K_r-factor. The existence of triangle-free graphs with sub-linear independence number is related to the asymmetric Ramsey number $R(3,n)$. This is well studied, results can be found e.g. in [Erd61], [Kim95].

The above construction shows that Theorem 4.2 is asymptotically tight. Look at the following example of an n-vertex graph. Take a triangle-free graph of size $2n/r+1$ and add $\left(1-\frac{2}{r}\right)n-1$ vertices each connected to all other vertices. The triangle-free subgraph of size $2n/r+1$ becomes a bottleneck since we can take at most to two vertices from it to complete to a K_r and we cannot cover the graph with n/r many K_r (see Figure 4.1). So in this graph we have $\delta(G) > \left(1-\frac{2}{r}\right)n$ and $\alpha(G) = o(1)$ but there is no K_r-factor. This construction was first given in [BMS16].

Our proof combines well-known methods like the Regularity Lemma and embedding techniques with new ideas that use the low independence number. The rest of this chapter is structured as follows. In Section 4.2 we introduce some lemmas related to the Regularity Lemma which might be of general interest. The remainder of the chapter contains the proof of Theorem 4.2. The proof consists of two parts.

First, in Section 4.3, we use the absorbing method. This is a technique mainly pushed forward by Rödl, Ruciński and Szemerédi [RRS06a, RRS06b, RSR08]. The method implies that, under the appropriate circumstances, it is enough to find a K_r-tiling covering everything but a small fraction. The method sets aside a small set of vertices at the beginning which we can cover flexibly enough so that we can "absorb" any small fraction of the other vertices which may remain. For the more precise definition see Definition 4.11.

Second, in Section 4.4, we prove that the minimum degree and independence number conditions are enough to cover everything but a small ξ-fraction of all vertices with a K_r-tiling. This is also known as an almost cover of the vertices. To show that there is an almost cover with K_r's in the graph we find a fractional tiling in the reduced graph after applying the Regularity Lemma and convert this back.

We adapt some well-known techniques to make use of the fact that the independence number of G is low. Embedding independent sets into a cluster of the reduced graph of the Regularity Lemma is standard, but we sometimes want to embed edges instead of single vertices. In fact we use that the low independence number implies we can find paths of small length in any small linear sized subset of the vertices. We are required to differentiate between edges in the reduced graph which represent densities above $1/2 + \beta$ and those only above β. We believe this approach might also work for embedding other graphs into a host graph with a low independence number.

4.2 Embedding lemma for low independence number

Since many of our constructions are specifically built for making use of the low independence number we first introduce some definitions, including the statement of a version of Szemerédi's Regularity Lemma. We begin with the notion of ϵ-regular.

Definition 4.3 (ϵ-regular)**.** Given a graph G and disjoint subsets $V_1, V_2 \subseteq V(G)$, we say that the pair (V_1, V_2) is ϵ-regular if for all $X \subseteq V_1, |X| \geq \epsilon|V_1|$ and $Y \subseteq V_2, |Y| \geq \epsilon|V_2|$ we have $|d(X,Y) - d(V_1,V_2)| \leq \epsilon$ where $d(X,Y) = \deg(X,Y)/|X||Y|$

The following fact is an easy consequence from the definition of regularity. It is sometimes known as the Slicing Lemma (cf.[KS96]).

Fact 4.4. *Let $B = (V_1 \cup V_2, E)$ be an ε-regular bipartite graph, let $\alpha > \varepsilon$ and let $V_1' \subset V_1$ and $V_2' \subset V_2$ be subsets with $|V_1'| \geq \alpha|V_1|$ and $|V_2'| \geq \alpha|V_2|$. Then for $\varepsilon' \geq \max\{\varepsilon/\alpha, 2\varepsilon\}$ the graph $B' = B[V_1' \cup V_2']$ induced by V_1' and V_2' is ε'-regular with $|d_B(V_1,V_2) - d_{B'}(V_1',V_2')| < \varepsilon$.*

Our proof builds upon the famous Regularity Lemma by Szemerédi. Originally from [Sze78] there have been many variants making it slightly

stronger or adapted to a particular problem. The following is the degree variant of the Regularity Lemma.

Lemma 4.5 (Regularity Lemma [KS96], Theorem 1.10)**.** *For every $\epsilon > 0$ there is an $M = M(\epsilon)$ such that if G is a graph on $n \geq M$ vertices and $\beta \in [0,1]$ is a real number, then there exists a partition $V(G) = V_0 \cup \ldots \cup V_k$ and a spanning subgraph $G' \subseteq G$ with the following properties:*

1. *$k \leq M$,*
2. *$|V_0| \leq \epsilon n$,*
3. *$|V_i| = m$ for all $1 \leq i \leq k$ with $m \leq \epsilon n$,*
4. *$\deg_{G'}(v) > \deg_G(v) - (\beta + \epsilon)n$ for all $v \in V(G)$,*
5. *V_i is an independent set in G' for all $i \in [k]$,*
6. *all pairs (V_i, V_j) are ϵ-regular with density 0 or at least β.*

What is new in our case is that we must differentiate between dense and very dense pairs of partitions. The following definition replaces the usual reduced graph of the Regularity Lemma. We call it the reduced multigraph throughout Chapter 4.

Definition 4.6 (reduced multigraph)**.** For a graph G and $\beta, \epsilon > 0$ let $V(G) = V_0 \cup \ldots \cup V_k$ be a partition and $G' \subseteq G$ and a subgraph fulfilling the properties of Lemma 4.5. We denote by $R_{\beta,\epsilon}$ the *reduced multigraph* of this partition, which is defined as follows. Let $V(R_{\beta,\epsilon}) = \{1, \ldots, k\}$ and for two distinct vertices i and j we draw two edges between i and j if $d_{G'}(V_i, V_j) \geq 1/2 + \beta$, one edge if $d_{G'}(V_i, V_j) \geq \beta$ and no edge otherwise.

In this reduced multigraph we sometimes refer to the vertices as clusters because of the correspondence to sets of vertices in the original graph. We omit the subscripts β and ϵ whenever it is clear from the context or the parameters are not used.

The following fact connects a minimum degree condition in G to a minimum degree condition in reduced multigraph.

Fact 4.7. *Let G be a graph with $\delta(G) \geq \left(1 - \frac{2}{r} + \mu\right) n$ and $V_1 \cup \ldots \cup V_k$ be the partition given by the Regularity Lemma with the corresponding*

reduced multigraph $R_{\beta,\epsilon}$ for ϵ and β smaller than $\mu/10$. Then for every $i \in V(R_{\beta,\epsilon})$ we have

$$\deg_{R_{\beta,\epsilon}}(i) \geq 2\left(1 - \frac{2}{r} + \mu/2\right)k.$$

Proof. For every $i \in V(R_{\beta,\epsilon})$ we have $\deg_{G'}(V_i, \bigcup_{j\neq i} V_j)$ is at least

$$|V_i|\left(\left(1 - \frac{2}{r} + \mu - (\epsilon+\beta)\right)n - |V_0|\right) \geq \left(1 - \frac{2}{r} + \mu - 2\epsilon - \beta)\right)nm.$$

Every edge in $R_{\beta,\epsilon}$ represents less than $(1/2+\beta)m^2$ edges in $G' \setminus V_0$. So $R_{\beta,\epsilon}$ must have minimum degree at least

$$\deg_{R_{\beta,\epsilon}}(i) \geq \frac{\left(1 - \frac{2}{r} + \mu - 2\epsilon - \beta\right)nm}{(1/2+\beta)m^2} \geq 2\left(1 - \frac{2}{r} + \mu/2\right)k,$$

where in the last step we use the upper bounds on β and ϵ, $m \leq n/k$ and $(1/2+\beta)^{-1} \geq 2(1-2\beta)$. □

Then to formalize the intuition of embedding two vertices into a cluster of the reduced graph we define a multi-embedding.

Definition 4.8 (H-multi-embedding)**.** Let R be a reduced multigraph. We say that a simple graph H is embeddable into the multigraph R if there is a mapping $f : V(H) \to V(R)$ such that the following holds:

1. For any $i \in V(R)$ the induced subgraph on the vertices $f^{-1}(i)$ in H is either an isolated vertex, an edge or a path of length 2.
2. If $\{u, v\} \in E(H)$, then $f(u)$ and $f(v)$ are connected by at least one edge in R (as long as $f(u)$ and $f(v)$ differ).
3. If for $i, j \in V(R)$ we have that $f^{-1}(i)$ and $f^{-1}(j)$ have at least two vertices and are connected in H, then i and j are connected with two edges.
4. The joint neighborhood of the vertices embedded into a single cluster has at most two vertices embedded in any other cluster. That is

   $$|N_H(f^{-1}(i)) \cap f^{-1}(j)| \leq 2 \qquad \forall i, j \in V(R)\ (i \neq j)$$

   where $N_H(f^{-1}(i)) = \bigcup_{w\in f^{-1}(i)} N_H(w)$ is the combined neighborhood of all vertices embedded in cluster i.

We call f a multi-embedding of H.

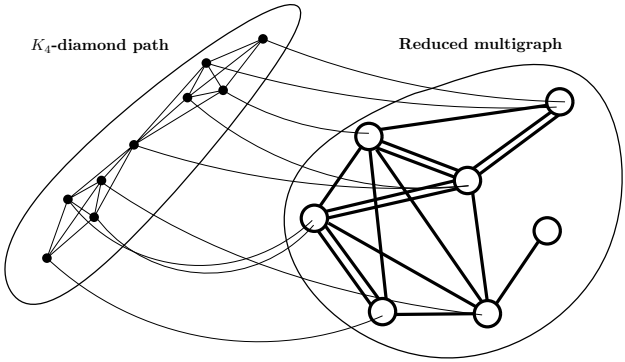

Figure 4.2: Multi-embedding of a K_4-diamond path

Note here that in point 1. we allow for paths of length two, which is needed specifically for our construction later as we may embed paths of length 2 into a single cluster. We could easily adapt the lemma below to also allow constant sized trees instead of paths of length two if it were necessary. We prove that this embedding is useful in the intended way. Whenever we can find a multi-embedding of a graph H in a reduced multigraph of G then we can also find many copies of H as a subgraph of G.

Lemma 4.9 (Embedding Lemma)**.** *For every graph H with $|H| = h$ and $\beta > 0$ there exist $\epsilon, \gamma > 0$ and $n_0 \in \mathbb{N}$ such that the following holds for every graph G on $n > n_0$ vertices and with independence number $\alpha(G) \leq \gamma n$ and the sets $V_1 \cup \ldots \cup V_k$ with $|V_i| = m$ given by the Regularity Lemma with the corresponding reduced multigraph $R_{\beta,\epsilon}$. Let f be a multi-embedding of H into $R_{\beta,\epsilon}$ with $f(V(H)) = \mathcal{I} = \{i_1, \ldots, i_t\}$ for some $1 \leq t \leq |H|$. Then let $V'_{i_1}, \ldots, V'_{i_t}$ be subsets of $V_{i_1}, \ldots, V_{i_t}$ respectively of size at least $(2/\beta)^h \epsilon m$. There exists a copy of H as a subgraph of G such that $v \in V'_{f(v)}$ for each vertex v in $H \subset G$.*

Proof. Ensure $\beta^h \geq (h+1)\varepsilon$ and $\epsilon m \geq 3\gamma n$. The subgraph can be chosen greedily and we show this by induction on the size of $\mathcal{I}$. The base case is clear, if we only have $|\mathcal{I}| = 1$ the multi-embedding can be at most a path of length two. But since V'_{i_1} is of size at least $(2/\beta)^h \epsilon m \geq 3\gamma n \geq 3\alpha(G)$ there is always a path of length two in V'_{i_1}.

For the induction we first check if there is any vertex of R which has a single vertex embedded. If so choose one of these, say i_v and $f^{-1}(i_v) = v \in H$. By the Property (2) of a multi-embedding, there are edges between cluster-vertices in R if their vertices in H are in the neighborhood of v. Consider any of these edges $\{i_w, i_v\}$. This means that V_{i_w} and V_{i_v} must be ε-regular with density at least β. This means that at most ϵm vertices of V'_{i_v} have a neighborhood smaller than $(\beta - \epsilon)|V'_{i_w}|$ in V'_{i_w}. This holds for any neighbor of v in H. As $\epsilon m h < m$, there is at least one vertex in V'_{i_v} which has at least $(\beta/2)|V'_{i_j}|$ neighbors in V'_{i_j} for all $i_j \in f(N_H(v))$, choose one arbitrarily say s_v. Choose V''_{i_j} to be the neighborhood of s_v if $f^{-1}(i_j)$ contains a neighbor of v or set it equal to V'_{i_j} if not. Note that $|V''_j| \geq (2/\beta)^{h-1}\epsilon m$ for all $j \in \mathcal{I}$ and that f restricted to $H \setminus v$ is still a multi-embedding into $\mathcal{I} \setminus i_v$. So we can apply the induction hypothesis and find the subgraph $H \setminus v$ of G such that the vertices are chosen from V''_{i_j}. Since all of the necessary V''_{i_j} are in the neighborhood of s_v, we have that the graph from the induction together with s_v form $H \subset G$ as desired.

The case where each cluster has at least two vertices embedded works analogously. Choose a vertex in $\mathcal{I} \subset R$ arbitrarily, say i_v, and let $f^{-1}(i_v)$ be the vertices v_1, v_2 and possibly v_3 of H. We call $f(N_H(f^{-1}(i_v))) \subset R$, excluding i_v, the set of corresponding neighbors of i_v. Because there are double-edges between i_v and its corresponding neighbors, for any of the corresponding neighbors i_w we have that V_{i_v} and V_{i_w} are regular with density at least $1/2+\beta$. So all but at most ϵm vertices of V'_{i_v} have degree at least $(1/2 + \beta - \epsilon)|V'_{i_w}|$ in V'_{i_w}. Since removing these bad vertices, which are at most $\epsilon m h$ many, from V'_{i_v} still leaves us with at least $3\gamma n$ vertices and there must exist a 2-path (or edge). We choose one of these arbitrarily. By Property (4) of the multi-embedding, at most two of its vertices need to suffice a neighboring condition to any other cluster V'_{i_w} and since the degree of each is at least $(1/2+\beta-\epsilon)|V'_{i_w}|$ also the common neighborhood of these two vertices is larger than $(\beta/2)|V'_{i_w}|$. Take this neighborhood to be V''_{i_w} for all corresponding neighbors of i_v (and $V'_j = V''_j$ where there is no neighborhood condition to be fulfilled). We apply the induction hypothesis on the remaining graph $H \setminus \{v_1, v_2, v_3\}$ with its restricted multi-embedding and the sets $V''_j \ \forall j \in \mathcal{I} \setminus \{i_{v_1}, i_{v_2}, i_{v_3}\}$ to find $H \setminus \{v_1, v_2, v_3\}$ as a subgraph of G which we can extend by the path we chose in V'_{i_v} to get graph $H \subset G$. □

Corollary 4.10. *For every graph H with $|H| = h$ and $\beta > 0$ there exist $\epsilon, \gamma > 0$ and $n_0 \in \mathbb{N}$ such that the following holds for every graph*

G on $n > n_0$ vertices and with independence number $\alpha(G) \leq \gamma n$ and the sets $V_1 \cup \ldots \cup V_k$ with $|V_i| = m$ given by the Regularity Lemma with the corresponding reduced multigraph $R_{\beta,\epsilon}$. Let f be a multi-embedding of H into $R_{\beta,\epsilon}$ with $f(V(H)) = \mathcal{I} = \{i_1, \ldots, i_t\}$ for some $1 \leq t \leq |H|$. Then let $V'_{i_1}, \ldots, V'_{i_t}$ be subsets of $V_{i_1}, \ldots, V_{i_t}$ respectively of size at least $(2/\beta)^h \epsilon m$. Additionally, let $u, v \in V(H)$ and $u_G, v_G \in V(G)$. Then there is an embedding of H in G such that u is mapped to u_G and v is mapped to v_G if the following holds.

(i) There is a multi-embedding f of $H \setminus \{u, v\}$ into $R_{\beta,\epsilon}$.

(ii) For all edges of the form $\{u, x\}$ and $\{v, y\}$ in H also $deg(u_G, V_{f(x)}) \geq \beta|V_{f(x)}|$ and $deg(v_G, V_{f(y)}) \geq \beta|V_{f(y)}|$ in G respectively.

(iii) u and v have distance at least 3 in H.

Proof. The embedding works the same as Lemma 4.9. First fix u and v as u_G and v_G, and then for each neighbor x of u choose $V'_{f(x)} = N(u_G) \cap V_{f(x)}$, same for neighbors of v. For all other vertices in H simply choose $V'_{f(i)} = V_{f(i)}$. So all $|V'_i| \geq \beta|V_i|$ and by Lemma 4.9 we can find an embedding of $H \setminus \{u, v\}$ and the embedding of neighbors u and v will also be neighbors of u_G and v_G respectively. The distance 3 is used to ensure no vertex in H is neighbor to both u and v. □

4.3 Absorbers

Absorbers are a well known tool and they allow us to prove statements about spanning subgraph structures. Often when working with the Regularity Lemma, we only find subgraph structures which cover almost all of the vertices, so all but a small linear fraction. Absorbers allow us to go the last step, they are structures we set aside in advance and which can "absorb" this small fraction of leftover vertices.

Definition 4.11. Let H be a graph with h vertices and let G be a graph with n vertices.

- We say that a subset $A \subseteq V(G)$ is *ξ-absorbing* for some $\xi > 0$ if for every subset $R \subseteq V(G) \setminus A$ such that h divides $|A| + |R|$ and $|R| \leq \xi n$ the induced subgraph $G[A \cup R]$ contains an H-factor.

- Given a subset $S \subseteq V(G)$ of size h and an integer $t \in \mathbb{N}$, we say that a subset $A_S \subseteq V(G) \setminus S$ is (S,t)-*absorbing* if $|A_S| = ht$ and both $G[A_S]$ and $G[A_S \cup S]$ contain an H-factor.

We use the following lemma which gives a sufficient condition for the existence of ξ-absorbers based on abundance of disjoint (S,t)-absorbers. The proof of Lemma 4.12 is based on ideas of Montgomery [Mon14] and relies on the existence of 'robust' sparse bipartite graphs.

Lemma 4.12 (Nenadov, Pehova [NP18])**.** *Let H be a graph with h vertices and let $\varphi > 0$ and $t \in \mathbb{N}$. Then there exists ξ and $n_o \in \mathbb{N}$ such that the following is true. Suppose that G is a graph with $n \geq n_0$ vertices such that for every $S \in \binom{V(G)}{h}$ there is a family of at least φn vertex-disjoint (S,t)-absorbers. Then G contains an ξ-absorbing set of size at most φn.*

We define the following structure as it will be used as the main building block in the remainder of this section.

Definition 4.13 (K_r-diamond path)**.** A K_r-diamond path between vertices u and v is the graph formed by a sequence of disjoint vertices $u = v_1, v_2, .., v_\ell = v$ and disjoint cliques of size $r-1$ in the joint neighborhood of each pair of consecutive vertices. The length of the K_r-diamond path is ℓ, the number of vertices in the sequence.

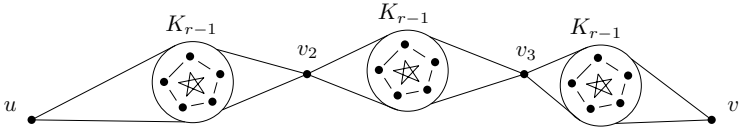

Figure 4.3: K_r-diamond path

4.3.1 Finding K_r-diamond paths

To make use of this lemma we additionally need to find vertex-disjoint (S,t)-absorbers in our graph. Observe that if we can find a K_r with disjoint K_r-diamond paths attached to each of its vertices, then this structure is (S,t)-absorbing for the set S of r free endpoints of the K_r-diamond paths. To find vertex-disjoint (S,t)-absorbers it is sufficient to find many disjoint K_r-diamond paths between any two vertices.

Lemma 4.14. *For every $r \geq 4$ and $\mu > 0$ there exist $\gamma > 0$ and $n_0 \in \mathbb{N}$ such that in every graph G on $n \geq n_0$ vertices with minimum degree $\delta(G) \geq \left(1 - \frac{2}{r} + \mu\right) n$ and $\alpha(G) \leq \gamma n$, after deleting $(\mu/2)n$ many vertices we can still find a K_r-diamond path of length at most 7 between any two remaining vertices.*

Note that, for connectivity issues, the lemma only holds for $r \geq 4$. To prove this lemma we find a multi-embedding of a K_r-diamond path in a reduced multigraph and then extract from that a K_r-diamond path in the original graph. We introduce the notion of a K_r-neighborhood $\Upsilon_r(v)$. These are the neighbors of v such that additionally we can find a multi-embedding of a K_r into the reduced multigraph covering both the vertex and v.

Definition 4.15. Let R be a reduced multigraph. Then for any vertex v the *K_r-neighborhood* $\Upsilon_r(v)$ is defined as follows.

$$\Upsilon_r(v) = \{w \in V(R) | \exists \text{ a multi-embedding } \psi : V(K_r) \to V(R) \text{ s.t. } \psi^{-1}(v) \neq \emptyset \text{ and } \psi^{-1}(w) \neq \emptyset\} \quad (4.1)$$

Further $\Upsilon_r^2(v) = \bigcup_{u \in \Upsilon_r(v)} \Upsilon_r(u)$ is the second-K_r-neighborhood.

Note that by definition any vertex v is in its own K_r-neighborhood assuming there is at least one K_r multi-embedding containing v. Then also $\Upsilon_r(v) \subseteq \Upsilon_r^2(v)$.

In order to find K_r-diamond paths we first show that, for every vertex in the reduced graph, the K_{r+1}-neighborhood is large.

Proposition 4.16. *For $r \geq 4$, let R be a reduced multigraph on k vertices with $\delta(R) > (1 - 2/r)2k$ then we have*

$$\left|\Upsilon_{r+1}^2(v)\right| > \frac{k}{2} \qquad \forall v \in V(R).$$

Before we prove this proposition, we prove a series of lemmas about the size of K_r-neighborhoods. Note that this is easier for large r thus we have to consider some special cases for small values of r. We start with some general lemmas that hold for all r.

In the following, a clique of double-edges denotes a clique where all edges are double-edges and the double-edge-neighborhood of a vertex v is the set of neighbors connected to v with a double-edge.

Lemma 4.17. *For $r \geq 4$, let R be a reduced multigraph on k vertices with $\delta(R) > (1-2/r)2k$. For any vertex v the vertices connected to v by double-edges are contained in the K_{r+1}-neighborhood $\Upsilon_{r+1}(v)$.*

Proof. In order to embed K_{r+1} we need a clique of double-edges of size ℓ and a clique of size $r+1-2\ell$ in the neighborhood of this clique. Note that the double-edge itself is already a clique of size 2. In the following, we show that for every $2 \leq \ell \leq (r+1)/2$ we can find such an embedding given that ℓ is the size of a maximal clique of double-edges.

Fix any double-edge of v and take the largest clique of double-edges containing the double-edge. Let ℓ be the size of the clique, and let S be the set of all vertices which lie in the joint neighborhood of all vertices of the clique. As we assumed the clique of double-edges to be maximal we know that every vertex in S has at most $2\ell-1$ edges into the clique. Every vertex that is not in S has to have at least one non-neighbor in the clique and can thus not have more than $2(\ell-1)$ edges into the clique. Moreover, by our minimum degree condition in R we know that every vertex in the clique has at least $(1-r/2)2k$ edges. Combining this, we get

$$(2\ell-1)|S| + 2(\ell-1)(k-|S|) > \ell\left(1-\frac{2}{r}\right)2k,$$

from which we conclude that

$$|S| > 2k - \frac{4k\ell}{r}. \tag{4.2}$$

For any vertex $v \in R$ it holds that the neighborhood

$$|N(v)| \geq \deg(v)/2 > \left(1-\frac{2}{r}\right)k.$$

In particular the number of vertices not in the neighborhood of a vertex is less than $\frac{2k}{r}$. So by greedily picking vertices one by one we can choose at least

$$\left\lceil \frac{|S|}{\frac{2k}{r}} \right\rceil \geq r - 2\ell + 1$$

many vertices. This gives us a clique of double-edges of size ℓ and in the joint neighborhood a clique of size $r-2\ell+1$ into which we can find a multi-embedding of K_{r+1}. □

Lemma 4.18. *For $r \geq 3$, let R be a reduced multigraph on k vertices with $\delta(R) > (1-2/r)2k$. For any vertex v in the R, if the neighborhood of v is of size at least $(1-1/r)k$, then $N(v) \subseteq \Upsilon_{r+1}(v)$.*

Proof. We apply induction on r by looking at the neighborhood of a vertex finding that the appropriate minimum degree conditions hold. The lemma is true for $r = 3$ since then any neighbor u of v has at least one vertex w in the joint neighborhood with v and we can create a multi-embedding ψ which maps one vertex of a K_4 to v and u and maps the two remaining vertices to w. This is a valid multi-embedding of a K_4 and proves $N(v) \subseteq \Upsilon_4(v)$. This builds our induction base.

For $r > 3$ consider for any vertex $u \in N(v)$ the joint neighborhood with v.

$$\deg(u, N(v)) > (1-2/r)2k - 2(k - |N(v)|) \geq (1-2/(r-1))2|N(v)|,$$

where in the last step we use that $|N(v)| \geq \frac{r-1}{r}k$. To prove that $u \in \Upsilon_{r+1}(v)$ it suffices to show that there is a K_r multi-embedding containing u in $R[N(v)]$, the subgraph induced by $N(v)$. Now $\delta(R[N(v)]) > (1-2/(r-1))2|N(v)|$ so for any vertex $u \in R[N(v)]$, by counting the edges, there must be either a double-edge containing u, in which case Lemma 4.17 gives at least one K_r multi-embedding, or u has a large neighborhood, $(1-2/(r-1))2|N(v)| \geq (1-1/(r-1))|N(v)|$, in which case we apply the induction on the subgraph $R[N(v)]$ so, in fact, in the subgraph $R[N(v)]$ any neighbor of u is in $\Upsilon_r(u)$ and also u is contained in a K_r. □

Lemma 4.19. *For $r = 4$, let R be a reduced multigraph on k vertices with $\delta(R) > (1-2/r)2k$, then $N(v) \subseteq \Upsilon_{r+1}(v)$.*

Proof. For any neighbor of v we want to find a multi-embedding of K_5 mapping to v and that neighbor.

By Lemma 4.17, every double-edge-neighbor of v is in $\Upsilon_{r+1}(v)$. For all other vertices $w \in N(v)$ we claim that either there is an edge between w and a vertex x in the double-edge-neighborhood of v, in which case we can map two vertices to x, two vertices to v and one to w to get a multi-embedding of K_5 or, in the other case, there is a double-edge between w and another vertex x in $N(v)$ and then we can map two vertices to x and w and one to v to get a multi-embedding of K_5.

Let D be the double-edge-neighborhood of v and $S = N(v) \backslash D$. Then for any vertex $w \in N(v)$, if w has no edge to any vertex in D and at most one edge to any vertex in S, then

$$\deg(w) \leq |S| + 2(k - |S| - |D|) \leq 2k - (2|D| + |S|) \overset{(2|D| + |S|) = \deg(v)}{<} k$$

which is a contradiction to the assumption that every vertex in the reduced graph R has degree greater than $(1 - 2/r)2k = k$ for $r = 4$. □

Lemma 4.20. *For $r = 5$, let R be a reduced multigraph on k vertices with $\delta(R) > (1 - 2/r)2k$ then $N(v) \subseteq \Upsilon^2_{r+1}(v)$.*

Proof. Let D be the double-edge-neighborhood of v. If $|D| \leq 2k/5$ we have $N(v) \geq 4k/5$ and by Lemma 4.18 again we have that $N(v) \subseteq \Upsilon_{r+1}(v)$, so we assume $|D| \geq 2k/5$.

Let $u \in N(v)$. We show that $u \in \Upsilon^2_{r+1}(v)$. If u has a double-edge to D, then by Lemma 4.17 it has distance two with regards to the K_r-neighborhood Υ_{r+1} and we are done. So we can assume it has only single edges or no edges to vertices in D. In particular, the double-edge-neighborhood of u does not intersect D so its size is at most $k - |D|$.

So since $|D| \geq 2k/5$, we have that

$$|N(u)| \geq \frac{6k}{5} - (k - |D|) \geq \frac{k}{5} + |D| \geq \frac{7k}{5} - |N(v)|,$$

where the last step follows from $|N(v)| > 6k/5 - |D|$. Since the minimum neighborhood of any other vertex is $3k/5$, the common intersection of u with any other vertex $x \in N(u)$ must be more than $7k/5 - |N(v)| + 3k/5 - k = k - |N(v)|$, so we can choose a vertex y in the joint neighborhood of v, u and x.

Now choose x in $D \cap N(u)$. The multi-embedding of K_6 follows by embedding two vertices each in v and x, and one vertex each in u and y. So then $u \in \Upsilon_{r+1}(v)$. In any case $u \in \Upsilon^2_{r+1}(v)$ and the lemma follows. □

Combining the previous lemmas, we are now ready to prove Proposition 4.16.

Proof of Proposition 4.16. For $r \geq 8$ the double-edge-neighborhood of every vertex is greater than $k/2$ so by Lemma 4.17 this follows immediately. For $r \geq 6$ by looking at the degree, for each vertex either the

double-edge-neighborhood is greater than $k/2$ or the total neighborhood is greater than $(1-1/r)k$, so by Lemma 4.17 or Lemma 4.18 the proposition follows. For $r = 4, 5$ we have Lemmas 4.19 and 4.20 respectively, where in both cases it is easy to see that $|N(v)| \geq k/2$. □

The next lemma is about connecting one fixed vertex v in G to K_{r+1}-embedable structures as follows. Given v, we want to find a multi-embedding of K_{r-1} into the neighborhood of v i.e. clusters that v has many edges to. We then want to extend this K_{r-1} to a K_r by finding a vertex in the joint neighborhood of the clique (not necessarily in $N(v)$). This is a preparation step to apply Corollary 4.10.

Lemma 4.21. *Let G be a graph as above with minimum degree $\delta(G) \geq \left(1-\frac{2}{r}+\mu\right)n$ and $\alpha(G) \leq \gamma n$. Fix a vertex v in G and a reduced multigraph $R_{\beta,\epsilon}$ of G. Let Q_v be the set of vertices $i \in V(R_{\beta,\epsilon})$ such that for their corresponding clusters $V_i \subseteq V(G)$ it holds that $\deg(v, V_i) \geq \beta|V_i|$. Then there exists a multi-embedding of a K_r into $R_{\beta,\epsilon}$ embedding at most one vertex into $V(R_{\beta,\epsilon}) \setminus Q_v$.*

Proof. Note that the number of edges from v to V_0 or any cluster not in Q_v is at most ϵn and $\beta km \leq \beta n$ respectively. The degree of v is at least $(1-2/r+\mu)n$ in G and choosing $\beta, \epsilon < \mu/10$ the number of edges from v to clusters of Q_v is at least $(1-2/r+2\mu/3)n$. In particular since every cluster has size at most n/k this means

$$|Q_v| \geq \left(1-\frac{2}{r}+\mu/2\right)k. \tag{4.3}$$

The proof follows similar arguments as the proof of Lemma 4.17. Let the largest clique with double-edges in Q_v be C of size ℓ. Let $S \subseteq Q_v$ be the joint neighborhood of the vertices from this clique inside Q_v and $T \subseteq V(R) \setminus Q_v$ all vertices which are in the joint neighborhood of the clique but not in Q_v. We want to find a $K_{r-2\ell}$ in $S \cup T$ with at most one vertex in T.

Because C is maximal every vertex in S has at most $2\ell-1$ edges to C and every other vertex in Q_v has at most $2\ell-2$ edges to C. But also every vertex in C has degree greater than $(1-\frac{2}{r})2k$. So we get two bounds for $deg(C, Q_v)$, the sum of degrees between C and Q_v.

$$deg(C, Q_v) > \ell\left((1-2/r)2k - 2(k-|Q_v|)\right) > (1-2/(r-2))\,2\ell|Q_v|,$$

where in the last step we use from (4.3) that $k < \frac{r}{r-2}|Q_v|$.

$$\begin{aligned}\deg(C, Q_v) &\leq \deg(C, S) + \deg(C, Q_v \setminus S)\\ &\leq (2\ell - 1)|S| + (2\ell - 2)(|Q_v| - |S|)\\ &< |S| + (2\ell - 2)|Q_v|.\end{aligned}$$

Together we get a bound on $|S|$. Namely

$$|S| > (1 - 2\ell/(r-2))\, 2|Q_v| > (r - 2\ell - 2)\frac{2k}{r}. \tag{4.4}$$

Next, we bound the size of $S \cup T$ with a similar argument. Counting the edges $\deg(C, V(R))$. Again, vertices in $V(R)$ but not in $S \cup T$ can have at most $2\ell - 2$ edges to C.

$$\begin{aligned}(1 - 2/r)2k\ell &\leq \deg(C, V(R))\\ &= \deg(C, S) + \deg(C, T) + \deg(C, V(R) \setminus (S \cup T))\\ &\leq (2\ell - 1)|S| + 2\ell|T| + (2\ell - 2)(k - |S| - |T|)\\ &< |S| + 2|T| + (2\ell - 2)k.\end{aligned}$$

We get a bound on $|S \cup T|$. Namely $|S| + 2|T| > (1 - 2\ell/r)2k = (r - 2\ell)\frac{2k}{r}$ and in particular since $|T| \leq k - |Q_v| < 2k/r$ because of (4.3) this means

$$|S| + |T| > (1 - 2\ell/r)2k - 2k/r = (r - 2\ell - 1)\frac{2k}{r}. \tag{4.5}$$

Furthermore, observe that for every vertex $w \in V(R)$ we have $N(w) \geq k - 2k/r$, thus every vertex has at most $2k/r$ non-neighbors. This directly implies we can sequentially choose

$$\left\lceil \frac{|S|}{\frac{2k}{r}} \right\rceil \overset{(4.4)}{\geq} r - 2\ell - 1$$

many vertices from S to form a clique and still have at least one vertex from $S \cup T$ because of (4.5) to form the $K_{r-2\ell}$. This together with the K_ℓ of double-edges allows for a multi-embedding of K_r and concludes the proof. □

We now prove Lemma 4.14.

Proof of Lemma 4.14. Choose two arbitrary vertices $s, t \in V(G)$ for which we want to find a K_r-diamond path. For s and t, apply Lemma 4.21 to find two multi-embeddings of K_r's such that at most one of the vertices in R has $\deg(s, V_i) < \beta|V_i|$ and $\deg(t, V_j) < \beta|V_j|$ respectively. Call these vertices s_1 and t_1 respectively. With Proposition 4.16 we find a multi-embedding of at most four K_{r+1}'s connecting s_1 and t_1 since the second Υ_{r+1} neighborhoods overlap.

This almost gives a multi-embedding of a K_r-diamond path connecting s and t. It remains to deal with the multi usage of a cluster in the reduced graph. For the mapping to be a multi-embedding as in Definition 4.8 we need that each vertex/cluster in the reduced graph has only a single vertex, edge or 2-path mapped to it. For this we partition each cluster arbitrarily into enough parts such that we can assign each vertex, edge or 2-path to a unique part. Note that as we have at most six K_r's we only need to split the clusters into constantly many parts.

If we arbitrarily split each cluster of the reduced graph into $6r$ equal parts, then the new partition still satisfies the conditions of the Regularity Lemma because ε-regularity is inherent by Fact 4.4 just with slightly different ε' and β'. So we can have a reduced multigraph R' of this new partition which is just a blowup of R. In particular, we can embed each isolated vertex, edge or 2-path into a separate cluster. In R' the consecutive K_r multi-embedding is in fact a multi-embedding of a K_r-diamond path of length at most seven excluding the endpoints s and t. It follows by Corollary 4.10 that we get a K_r-diamond path in G. □

With Lemma 4.14 we can find K_r-diamond paths from any tuple of r vertices matching them to a different vertex of a K_r somewhere else in the graph. This is now a (S, t)-absorber from Definition 4.11 and together with Lemma 4.12 this is enough to find an absorber of the first kind as in Definition 4.11.

4.4 Almost Spanning Structure

For the second part of the proof we want to show that we can cover most of the vertices with a K_r-tiling. Combining this with the absorber gives a K_r-factor.

Lemma 4.22. *For every $r \in \mathbb{N}$ and $\xi, \mu > 0$, there exist $\gamma > 0$ and $n_0 \in \mathbb{N}$ such that every graph G on $n > n_0$ vertices with $\delta(G) \geq \left(1 - \frac{2}{r} + \mu\right) n$ and $\alpha(G) \leq \gamma n$, we can find a K_r-tiling which covers at least $(1 - \xi)n$ vertices in G.*

We make use of a known result for small subgraphs in the same setting. To find a K_r in a graph with small independence number we only need a certain average degree. The following lemma states this

Lemma 4.23 (Erdős, Sós [ES70])**.** *For every $r \in \mathbb{N}$ and $\mu > 0$ there exist $\gamma > 0$ and $n_0 \in \mathbb{N}$ such that for every graph G on $n > n_0$ vertices with **average degree** $d(G) \geq \left(1 - \frac{2}{r-1} + \mu\right) n$ and $\alpha(G) \leq \gamma n$, then $K_r \subseteq G$.*

First we would like to show, that there exists at least a fractional almost cover of the vertices. A fractional cover is defined as follows:

Definition 4.24. A fractional K_r-tiling $\mathcal{T}$ of a graph G is a weight function from the set $\mathcal{S}$ of all $K_r \subseteq G$ to the interval $[0, 1]$ such that for vertices of G it holds that

$$w_{\mathcal{T}}(v) = \sum_{\substack{\mathcal{K}_i \in \mathcal{S}, \\ v \in \mathcal{K}_i}} w_{\mathcal{T}}(\mathcal{K}_i) \leq 1 \qquad \forall v \in G.$$

We call $\sum_{v \in G} w_{\mathcal{T}}(v)$ the total weight of a tiling and it is a perfect fractional tiling if equality holds for every vertex.

Fractional K_r-tilings are somehow easier to find and we will prove the following lemma later in this section.

Lemma 4.25. *For every $r \in \mathbb{N}$ and η, $\mu > 0$ there exist $\gamma > 0$ and $n_0 \in \mathbb{N}$ such that every graph G on $n \geq n_0$ vertices with $\delta(G) \geq \left(1 - \frac{2}{r} + \mu\right) n$ and $\alpha(G) < \gamma n$ has a fractional K_r-tiling $\mathcal{T}$ such that*

$$|\{v \in G \colon w_{\mathcal{T}}(v) < 1 - \eta\}| \leq \eta n.$$

Observe that the weight of this tiling is at least $(1 - 2\eta)n$. We would like to transform the fractional into an actual tiling. We construct a fractional tiling in the reduced multigraph first, then transfer it to the

original graph greedily. We will slightly abuse notation for the fractional tiling to extend the definition to the reduced multigraph. By a fractional tiling with K_r-embeddable structures we mean we assign the weights to all possible multi-embeddings of K_r onto the reduced multigraph and require that the for every vertex all multi-embeddings mapping to that vertex have a total weight of at most one, counting multiplicity.

Lemma 4.26. *For every $r \in \mathbb{N}$ and $\eta, \beta > 0$ there exist $\epsilon, \gamma > 0$ and $n_0 \in \mathbb{N}$ such that for every graph G on $n \geq n_0$ vertices, if a reduced multigraph $R_{\beta,\epsilon}$ of G has a fractional tiling with K_r-embeddable structures of total weight at least $(1-\eta)k$, then G has an K_r-factor that covers all but $(1-2\eta)n$ vertices.*

Proof. Set ϵ and γ small enough for Lemma 4.9 and such that $(2/\beta)^r\epsilon \leq \eta/2$. The first step is to rescale the tiling. Let $\mathcal{T}$ be the fractional tiling of R as given by the statement. Construct $\mathcal{T}'$ by scaling every K_r-embeddable structure with a factor of $(1-(2/\beta)^r\varepsilon)$ i.e. for any K_r-multi-embedding $\mathcal{K}$ we have $w_{\mathcal{T}'}(\mathcal{K}) = (1-(2/\beta)^r\varepsilon)w_{\mathcal{T}}(\mathcal{K})$. We construct the K_r-tiling in G by greedily taking $w_{\mathcal{T}'}(\mathcal{K})|V_i|$ many K_r given by Lemma 4.9 and remove them from G. Note that, because of the rescaling, the sum of the weights of all K_r-embeddable structures touching one vertex is at most $(1-(2/\beta)^r\varepsilon)$. Thus, in every step of the greedy removal we have at least $(2/\beta)^r\varepsilon|V_i|$ vertices left which ensures that we can always apply Lemma 4.9. Even after rescaling, $\mathcal{T}'$ has total weight at least $(1-(2/\beta)^r\varepsilon-\eta)k$ and $|V_0|$ has at most ϵn many vertices. So the greedy K_r-tiling of G covers at least a $(1-((2/\beta)^r\epsilon+\eta+\epsilon)) \geq (1-2\eta)$ fraction of the vertices which concludes the proof. □

For our proof, we need triangle free graphs with low independence number that we can connect with relatively high density without creating a copy of K_4. A construction by Bollobás and Erdős shows that these graphs exist. We state their results in a slightly different way, but it directly follows from their construction.

Lemma 4.27 ([BE76])**.** *For $\zeta, \gamma > 0$ there is a $n_0 \in \mathbb{N}$ such that for $n \geq n_0$ there is a graph G on $2n$ vertices with a split into V_1, V_2 has the following properties.*

1. *$|V_1| = |V_2| = n$,*
2. *$G[V_1]$ is isomorphic to $G[V_2]$ and they are triangle free,*

3. *G is K_4 free,*
4. *$G[V_1, V_2]$ has density at least $1/2 - \zeta$,*
5. *$\alpha(G) \leq \gamma n$*

The next lemma connects almost tilings and fractional tilings. In order to find an almost tiling in a graph G we apply the Regularity Lemma and need a fractional tiling in the reduced graph. We make use of a second auxiliary graph Γ, which is similar to a blow-up of the reduced graph.

Lemma 4.28. *For every $r \in \mathbb{N}$ and $\mu, \eta > 0$ there exist $\beta, \epsilon, \gamma > 0$ and $n_0 \in \mathbb{N}$ such that for every graph G on $n \geq n_0$ vertices with minimum degree $\delta(G) \geq \left(1 - \frac{2}{r} + \mu\right) n$ and $\alpha(G) \leq \gamma n$ there is a graph Γ with $\delta(\Gamma) \geq \left(1 - \frac{2}{r} + \frac{\mu}{4}\right) |\Gamma|$ and $\alpha(\Gamma) \leq \gamma |\Gamma|$ such that the following holds.*

If Γ has a fractional K_r-tiling with weight at least $(1-\eta)|\Gamma|$, then G has a K_r-tiling covering at least $(1 - 2\eta)n$ vertices.

Proof. Choose β, ε and γ small enough such that Lemma 4.9 and Lemma 4.26 are satisfied and smaller than $\mu/10$. Apply the Regularity Lemma (Lemma 4.5) to G with β and ε. Let $V_0 \cup V_1 \cup \ldots \cup V_k$ be the regular partition resulting from the Regularity Lemma and let $R_{\beta,\epsilon}$ be the reduced multigraph of this partition. Let y_1 be a constant that is larger than n_0 from Lemma 4.27 with $\gamma_{4.27} = \gamma$ and $\zeta_{4.27} = \mu/8$. Construct Γ by taking $y_0 = k \cdot y_1$ vertices and split $V(\Gamma)$ into $W_1, \ldots, W_k$ each of size y_1 where we associate W_i with V_i from the regular partition. On every vertex set W_i we put a triangle-free graph from Lemma 4.27. Then add a complete bipartite graph between two clusters W_i and W_j if i and j are connected by a double-edge in R. Add a K_4-free construction given by Lemma 4.27 between W_i and W_j if i and j are connected by a single edge and the empty graph otherwise. Note that as the graphs inside the clusters are all isomorphic, we are guaranteed that the K_4-free graph construction of Lemma 4.27 is possible between any two clusters.

We consider the minimum degree of Γ. As G has minimum degree $\delta(G) \geq (1-2/r+\mu)n$, using Fact 4.7 we get $\delta(R_{\beta,\epsilon}) \geq 2\left((1 - 2/r + \mu/2)\right) k$ which finally means in Γ every edge from a cluster-vertex i in $R_{\beta,\epsilon}$ contributes to at least $(1/2-\zeta)y_1 = (1/2-\zeta)|\Gamma|/k$ many edges for a vertex in the corresponding set W_i of Γ. Thus Γ has minimum degree

$$\delta(\Gamma) \geq (1 - 2/r + \mu/2 - 2\zeta)|\Gamma| = (1 - 2/r + \mu/4)|\Gamma|$$

where $\zeta = \mu/8$ as we chose for Lemma 4.27.

The important observation now is that every K_r in Γ corresponds to a multi-embedding of K_r in R. This is an easy consequence of the construction of Γ. For every K_r in Γ take the mapping which maps to the vertex i if the vertex of K_r lies in the set W_i in Γ. We never embed three vertices into a vertex of R because all W_i's are triangle free and if there are two clusters W_i and W_j into which we embed two vertices each, these vertices form a K_4 which means in R, i and j must be connected by a double-edge. By construction, the largest independent set of every cluster W_i of Γ is at most $\gamma|W_i|$ so $\alpha(\Gamma) \leq \gamma|\Gamma|$. Then by the assumption of the lemma we have a fractional K_r-tiling $\mathcal{T}$ of Γ. We convert the fractional K_r-tiling of Γ into a fractional K_r-tiling $\mathcal{T}'$ of R by applying the mapping from K_r's to K_r-multi-embeddings of R. So, for every multi-embedding $\mathcal{K}$ of a K_r into R we can define the set $L_{\mathcal{K}}$ to be the set of all K_r in Γ such that the multi-embedding $\mathcal{K}$ maps to the same partitions V_i corresponding to W_i in Γ. Then

$$w_{\mathcal{T}'}(\mathcal{K}) \geq \sum_{K \in L_{\mathcal{K}}} \frac{w_{\mathcal{T}}(K)}{y_1}.$$

So the total weight of $\mathcal{T}'$ must be at least $(1-\eta)k$ in R. Then by Lemma 4.26 we can convert the fractional tiling into an almost cover of G that covers at least $(1-2\eta)n$ vertices. □

For the proof of Lemma 4.25 we need the following lemma which gives us a stepwise improvement of any tiling we have as long as we do not cover a $(1-\eta)$ fraction of the vertices yet. Here a $\{K_r, K_{r+1}\}$-tiling is a disjoint union of K_r's and K_{r+1}'s as subgraph.

Lemma 4.29. *For every $r \in \mathbb{N}$ and $\eta, \mu > 0$ there exist ρ, $\gamma > 0$ and $n_0 \in \mathbb{N}$ such that every graph G on $n \geq n_0$ vertices with $\delta(G) \geq \left(1 - \frac{2}{r} + \mu\right) n$ and $\alpha(G) < \gamma n$ has the following property:*

Let $\mathcal{T}$ be a maximum K_r-tiling in G with $|V(\mathcal{T})| \leq (1-\eta)n$. Then there is a $\{K_r, K_{r+1}\}$-tiling which covers at least $|V(\mathcal{T})| + \rho n$ vertices.

Proof. Let $\mathcal{R} = V(G) \setminus V(\mathcal{T})$ be the set of all uncovered vertices in G. By Lemma 4.23 we know that the average degree inside $\mathcal{R}$ is less than $(1 - 2/(r-1) + \mu)\,|\mathcal{R}|$ as else there would be a K_r inside $\mathcal{R}$ that we could add to $\mathcal{T}$ contradicting the maximality of $\mathcal{T}$. We show that this

implies that we can extend at least ρn of the K_r's in $\mathcal{T}$ to K_{r+1} where ρ is some constant to be chosen later.

Let $T = V(\mathcal{T})$ and we are guaranteed that $|T| \geq \mu n$ as otherwise every vertex in $\mathcal{R}$ would have $\deg(v, \mathcal{R}) \geq (1 - 2/r + \mu)n - \mu n > (1-2/(r-1))|\mathcal{R}|$ contradicting our upper bound on the average degree. Moreover, as every vertex in $\mathcal{R}$ has degree at least $(1 - 2/r + \mu)n$ and inside $\mathcal{R}$ we have an average degree less than $(1-2/(r-1))|\mathcal{R}|$ we know that the edges in between, $\deg(\mathcal{R}, T)$, are at least

$$\left(1 - \frac{2}{r} + \mu\right) n|\mathcal{R}| - \left(1 - \frac{2}{r-1}\right)|\mathcal{R}|^2 \geq \left(1 - \frac{2}{r} + \mu\right)|T||\mathcal{R}| + \frac{2}{r(r-1)}|\mathcal{R}|^2.$$

Let $\mathcal{R}' \subseteq \mathcal{R}$ be the set of all vertices in $\mathcal{R}$ that have $\deg(v, T) > \left(1 - \frac{2}{r} + \mu\right)|T|$ as we know that $\deg(v, T) \leq |T| \leq n$, we conclude that

$$|\mathcal{R}'| \geq \frac{\frac{2}{r(r-1)}|\mathcal{R}|^2}{n} \geq \varphi|\mathcal{R}|$$

for $\varphi = \frac{2}{r(r-1)}\eta$.

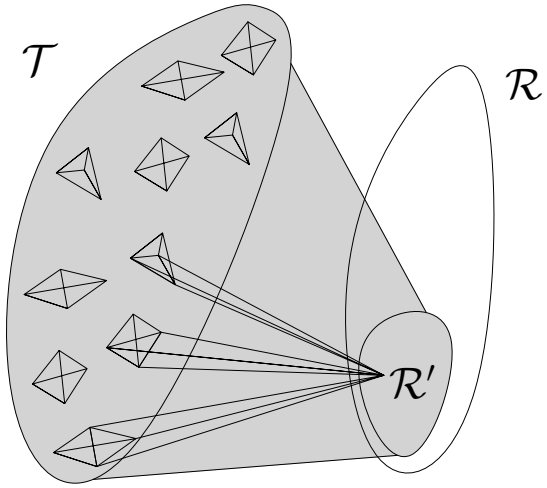

Figure 4.4: Greedy extending to K_{r+1}

We now use vertices or edges from $\mathcal{R}'$ to extend some K_r from $\mathcal{T}$ to a K_{r+1}. Let $\mathcal{T}'$ be the set of all K_r that we did not yet extend in

this process and $\mathcal{R}'' \subseteq \mathcal{R}'$ the set of unused vertices in $\mathcal{R}'$ so far. The following claim asserts that the greedy process works.

Claim 4.30. *If $\mathcal{R}'' \subseteq \mathcal{R}'$ is such that $(\mu/2r)|\mathcal{R}''| \geq \gamma n$ and for every vertex $v \in \mathcal{R}''$ we have $T' = V(\mathcal{T}') \subseteq T$ with*

$$\deg(v, T') \geq \left(1 - \frac{2}{r} + \frac{\mu}{2}\right)|T'|,$$

then we can find a K_r in $\mathcal{T}'$ which can be extended to a K_{r+1}.

Proof. If there is a K_r in $\mathcal{T}'$ such that there is a vertex in $\mathcal{R}''$ which is connected to all vertices from this K_r, then we can extend it to a K_{r+1}. We can thus assume that every vertex in $\mathcal{R}''$ has at most $r-1$ edges to any K_r in $\mathcal{T}'$. Then, the minimum degree condition implies that every vertex has at least $(\mu/2)|T'|$ copies of K_r in $\mathcal{T}'$ such that v is connected to exactly $r-1$ vertices of this K_r.

We can construct an auxiliary bipartite graph where the vertices in one partition are the copies of K_r in $\mathcal{T}'$ and the other partition is formed by the vertices in $\mathcal{R}''$. Then the previous observation implies that this bipartite graph has at least $(\mu/2)|T'||\mathcal{R}''|$ edges and we can thus find a K_r in $\mathcal{T}'$ such that at least $(\mu/2)|\mathcal{R}''|$ vertices from $\mathcal{R}''$ have exactly $r-1$ edges to this particular K_r. Call the set of these vertices $\mathcal{R}'''$ we can then further partition $\mathcal{R}'''$ into $\mathcal{R}'''_1, \dots, \mathcal{R}'''_r$ where we put a vertex $v \in \mathcal{R}'''$ in $\mathcal{R}'''_i$ if and only if v does not have an edge to the ith vertex in the K_r (where the order of the vertices is arbitrary but fixed). Then there is some index j such that $|\mathcal{R}'''_j| \geq (\mu/2r)|\mathcal{R}''|$. As we required that $\alpha(G) < \gamma n \leq (\mu/2r)|\mathcal{R}''| \leq |\mathcal{R}'''_j|$, there is an edge e in $\mathcal{R}'''_j$. We can thus construct a K_{r+1} by removing the jth vertex from the K_r and adding the edge e to the K_r. □

Note that for every K_{r+1} we construct we remove one K_r from $\mathcal{T}'$ and at most two vertices from $\mathcal{R}''$. We choose ρ maximal such that $\rho n \leq (\mu/2r)|T|$ and $2\rho n \leq |\mathcal{R}'| - (2r/\mu)\gamma n$. After the removal of at most ρn greedily formed K_{r+1}'s we are thus left with at least $|\mathcal{R}'| - 2\rho n$ vertices in $\mathcal{R}''$ each of these vertices has $\deg(v, T') \geq (1 - 2/r + \mu/2)|T'|$. Then Claim 4.30 gives that we can chose the K_{r+1}'s in a greedy manner until we extend ρn many K_r's. □

Now we are ready to prove Lemma 4.25. We restate the lemma for convenience of the reader.

Lemma 4.25. *For every $r \in \mathbb{N}$ and η, $\mu > 0$ there exist $\gamma > 0$ and $n_0 \in \mathbb{N}$ such that every graph G on $n \geq n_0$ vertices with $\delta(G) \geq \left(1 - \frac{2}{r} + \mu\right) n$ and $\alpha(G) < \gamma n$ has a fractional K_r-tiling $\mathcal{T}$ such that*

$$|\{v \in G \colon w_{\mathcal{T}}(v) < 1 - \eta\}| \leq \eta n.$$

Proof of Lemma 4.25. We start by taking a maximum K_r-tiling in G. If this covers more than $(1-\eta)n$ vertices, then we are done immediately. Else we repeatedly apply Lemma 4.29 while at every step blowing up each vertex of our graph G with r vertices. This follows the idea which emerged from [Tre16]. After a constant number of blowups we can cover all but a η^2 fraction of the vertices with K_r's. We then convert this tiling of the blown up graph into a fractional tiling of the original graph which misses at most $\eta^2 n$ of total weight, which directly implies that at most ηn vertices can have $w_{\mathcal{T}}(v) < 1 - \eta$.

In each of the steps we blow up the graph by a factor of r, that is we replace every vertex in the previous graph with a set of r vertices and put complete bipartite graphs between all clusters that were connected by an edge in the previous graph. Note that this implies that for a K_{r+1} in the previous graph we can find a perfect K_r tiling in the blown up graph. We will repeat two steps:

- In the first step the *enlargement step* here we start with a K_r tiling which covers a λ fraction of the vertices into a $\{K_r, K_{r+1}\}$ tiling that covers a $\lambda' > \lambda + \rho_{4.29}(\eta^2, \mu, r)$ fraction
- The second step, the *blow up step* blows up the graph and converts the given $\{K_r, K_{r+1}\}$-tiling into a K_r-tiling that covers a λ' fraction.

Note that a K_r-tiling of any graph corresponding to a constant blow up by a factor of s of G which covers a λ fraction of the vertices can be converted into a fractional K_r-tiling in G with weight λn. This can be done as follows. Let $\mathcal{T}'$ be a K_r-tiling in the blown up graph. We construct the fractional K_r-tiling $\mathcal{T}$ in G in the following way. For every $K_r \in \mathcal{T}'$ by construction there is a copy of K_r in G which corresponds to this K_r (in particular we cannot have two vertices which originate from the same vertex in G as these vertices would come from an independent set). We add this K_r to $\mathcal{T}$ with weight $1/s$. When there are multiple instances that correspond to the same K_r in G we just increase the weight by $1/s$ for each copy in the blown up graph. Let G^s be the

blowup of G by factor of s, as the tiling we constructed covers $\lambda|G^s|$ vertices in G^s we get that

$$\sum_{v\in V(G)} w_{\mathcal{T}}(v) = \sum_{K_r\in\mathcal{T}'} r\frac{1}{s} = \lambda|G^s|\frac{1}{s} = \lambda|G|.$$

It thus suffices to show that for some number s, independent of γ and n we can find a K_r-tiling that covers $(1-\eta^2)|G^s|$ vertices in G^s. Let $\gamma = \gamma_{4.29}(\eta^2,\mu,r)$ and $\rho = \rho_{4.29}(\eta^2,\mu,r)$. Every time we apply Lemma 4.29 we newly cover a ρ fraction of the vertices. We thus need to apply this lemma at most $1/\rho$ times. In each blow up step we replace one vertex from the previous graph by r vertices. As we have to do at most $1/\rho$ blow up steps we know that $s \leq r^{1/\rho}$. □

Lemma 4.22 follows directly by applying Lemma 4.25 with $\mu_{4.25} = \mu/4$ to Γ from Lemma 4.28 with μ and $\eta = \xi/4$.

4.5 Finishing the proof

All that is left to do is to combine the results from the previous sections to prove the Theorem 4.2.

Proof of Theorem 4.2. Choose $\varphi \leq \mu/14r^2$ but independent from all other variables. Let $\xi = \xi_{4.12}$ where we apply Lemma 4.12 with φ, $h = r$ and $t = 6r+1$. Choose γ small enough such that it satisfies Lemma 4.14 as well as Lemma 4.22 dependent on the parameters μ, φ and ξ.

In order to apply Lemma 4.14 to get a ξ-absorbing set, we show that for every choice of a r-vertex subset S of $V(G)$ we can find φn vertex disjoint $(S, 6r+1)$-absorbers. We do this as follows. Start with an arbitrary K_r that does not share any vertex with S using Lemma 4.23. Take an arbitrary bijection $g\colon V(K_r) \to S$ of the vertices of this K_r to the vertices in S. Then use Lemma 4.14 to find disjoint K_{r+1}-diamond paths of length at most 7 between each pair $(v, g(v))$ for all $v \in V(K_r)$. Add arbitrary K_r's in case some paths where shorter until there are exactly $6r^2 + r$ vertices in total. We can repeat this φn times without removing more than $(6r^2+r)\varphi n < (\mu/2)n$ vertices from the graph. Having these φn many (S,t)-absorbers implies by Lemma 4.12 that

there is some constant ξ such that there is a ξ-absorbing set of size at most φn. Take such a set A and put it aside.

Note that as $|A| \leq \varphi n$ we know that for $G' = G \setminus A$ we have $\delta(G') \geq (1 - 2/r + \mu')n'$ and $\alpha(G') \leq \gamma' n'$ where $n' = |V(G')|$, $\mu' = \mu/2$ and $\gamma' = 2\gamma$. We then apply Lemma 4.22 with ξ from Lemma 4.12 to G' to get a tiling that covers all but at most $\xi n'$ vertices. Let V_R be the set of vertices that remain uncovered in Lemma 4.22. By construction we have $|V_R| \leq \xi n' \leq \xi n$ and thus we can use the absorber A to cover $A \cup V_R$. □

Chapter 5

An $\mathcal{O}(n)$ Time Algorithm for Finding Hamilton Cycles with High Probability

In this chapter we provide an algorithm for finding Hamilton cycles in time linear in the number of vertices. The content of this chapter is based on a paper published by Nenadov, Steger and the author ([NSS21]).

5.1 Introduction

We have touched briefly on the topic of Hamilton cycles in Chapter 1. As discussed, determining whether a graph has a Hamilton cycle is a notoriously difficult problem that has been tackled in various ways. In general, it is known to be $\mathcal{NP}$-hard, putting it in a bag of complexity

theory together with colorability or SAT, problems for which one has tried to find polynomial time algorithms for a long time without any success so far.

While the Hamilton cycle problem is a difficult problem in general, it turns out that for most graphs it is much easier in the Erdős-Rényi random graph $G(n,p)$. The existence question of the Hamilton cycle problem is very well understood, cf. the comprehensive survey by Frieze [Fri19]. We have the following precise bounds for when Hamilton cycles exist in $G(n,p)$. Let $\mathcal{H}$ be the set of Hamiltonian graphs, then for $G(n,p)$ it holds that (Komlós and Szemerédi [KS83] and Korshunov [Kor76])

$$Pr[G(n,p(n)) \in \mathcal{H}] = \begin{cases} 0, & p(n) = \frac{\log(n)+\log\log(n)-\omega(1)}{n} \\ e^{-e^{-c}}, & p(n) = \frac{\log(n)+\log\log(n)+c+o(1)}{n} \\ 1, & p(n) = \frac{\log(n)+\log\log(n)+\omega(1)}{n} \end{cases}$$

Which is limitwise the same as the threshold for when $G(n,p)$ has minimum degree 2. So really vertices of degree one are the bottleneck for random graphs. In fact, it is known that if we add the edges randomly one by one, the moment we reach minimum degree 2 is the same as the moment the graph becomes Hamiltonian with high probability [AKS85]. And this threshold is also robust (e.g. [NST19, Mon19]). For other random graph models like the random graph with m edges $G(n,m)$, the random regular graph $G(n,r)$ or the k-out which takes k random edges from every vertex the corresponding thresholds for Hamiltonicity are also known [FF84, BFF90, BF09, RW94, SV08]. Similar to the classical random graph case also in these cases the thresholds coincides with a local obstruction such as minimum degree two or any two vertices have a neighborhood of size at least 3. And this is not a coincidence. Randomness gives us such nice expansion properties that only the small structures can be an obstruction to the Hamilton cycle. This phenomenon has been observed also for other properties such as connectivity, containing a perfect matching or colorability.

The proofs of Komlos and Szemeredi and Korshunov are just existential, i.e. they determine the threshold for the existence of Hamilton cycle, but do not provide an efficient algorithm for finding it. In a seminal paper, Angluin and Valiant [AV79] show that with the input given as a random adjacency list one can find Hamilton cycles in $G(n,p)$ for

$p \geq C \log n/n$ in $\mathcal{O}(n \log^2 n)$ time with high probability. There are two ways in which this result is possibly non-optimal: the lower bound on p and the runtime. The first point was considered by Shamir and then Bollobas, Fenner and Frieze, who brought the bound down to the existence threshold of $G(n,p)$. In more recent works the runtime has also been optimized for graphs given in adjacency matrix form, assuming a pair of vertices can be queried in constant time. We summarize these results in the table below. There are various related results that are hard to compare, as their setting is slightly different [FKSV16, ABKP15, FJM+96, Fri88]. Some of the results are assuming the graph is given as an adjacency matrix with black box queries and the runtime $\mathcal{O}(n/p)$ is optimal in that model.

Authors	**Year**	**Time**	**p(n)**
Angluin, Valiant [AV79]	'79	$\mathcal{O}(n \log^2(n))$	$p \geq \frac{C \log(n)}{n}$
Shamir [Sha83]	'83	$\mathcal{O}(n^2)$	$p \geq \frac{\log(n)+o(\log(n))}{n}$
Bollobas et al. [BFF87]	'87	$n^{4+o(1)}$	$p \geq Threshold$
Gurevich, Shelah [GS87]	'87	$\mathcal{O}(n/p)$	p const.
Thomason [Tho89]	'89	$\mathcal{O}(n/p)$	$p \geq Cn^{-1/3}$
Alon, Krivelevich [AK20]	'20	$\mathcal{O}(n/p)$	$p \geq 70n^{-1/2}$

In this chapter we consider the second question that was left open in the Angluin-Valiant paper: can the runtime be improved. Note that a graph with $p \geq C \log n/n$ has $\Theta(n \log n)$ edges. Thus, improving the runtime below this bound requires a *sublinear* algorithm, i.e. sublinear in the input size. These are algorithms that produce an output without reading the input completely (see e.g. [RS11] for an overview of the topic). Such algorithms are less restrictive than those designed for online or a (semi-)streaming model as they allow some control over which part of an input is used. However for graphs with n vertices and $m \gg n$ edges the algorithm is only allowed to read $o(m)$ edges, i.e., a negligible fraction of the input — but nevertheless has to compute the desired output correctly.

5.1.1 Our contribution

In this chapter we show that given a random graph with edge probability $p \geq C \log n/n$, for an appropriately chosen constant C, we can find a

Hamilton cycle in $\mathcal{O}(n)$ time with high probability. This time is clearly optimal, as the algorithm has to return $\Omega(n)$ edges. We assume that the graph is given to us with randomly ordered adjacency lists, such that we can query the next neighbor in those lists for any vertex in constant time.

Theorem 5.1. *There exists a randomized algorithm $\mathcal{R}$ which finds a Hamilton cycle in a random graph $G(n,p)$ in $\mathcal{O}(n)$ time with high probability, provided $p \geq C \log n/n$ for a sufficiently large constant C.*

Note that 'with high probability' is always meant to mean with probability $1 - o(1)$ tending to one as n tends to infinity and takes into account all sources of randomness: i.e., the randomness of the algorithm, the random graph and the randomness of the datastructure used to store the graph (random ordering of the adjacency lists).

The chapter is organized as follows. Section 5.2 contains the algorithm and the proof of Theorem 5.1. It is based on three technical lemmas that we prove in Section 5.3.

5.2 Algorithm

The most commonly used technique for efficient cycle extensions is Posa rotations. This is also the case for the original algorithm of Angluin and Valiant [AV79], which we outline in Section 5.2.1 below, cf. also Figure 5.2. To reduce the runtime to $\mathcal{O}(n)$ we reduce the total *number* of Posa rotations that are required and simultaneously also restrict ourselves to certain types of Posa rotations so that we can realize each of them in $\mathcal{O}(\log n)$ time.

5.2.1 Finding A Hamilton Cycle via Posa Rotations

We sketch here the algorithm of Angluin and Valiant. The main idea of their algorithm is to perform a greedy random walk until all vertices are incorporated in the path/cycle. This means we start from an arbitrary vertex and query a neighbor of that vertex. If the neighbor is already contained in the path we have built so far we consider this a failure and we query a new neighbor. Otherwise we add the neighbor to the path and continue from the new endpoint vertex (see Figure 5.1).

Figure 5.1: Algorithm uses a random walk like strategy, blue edge is the `newneighbor()`.

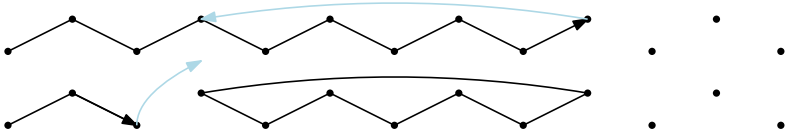

Figure 5.2: Posa rotation, detaching a large cycle.

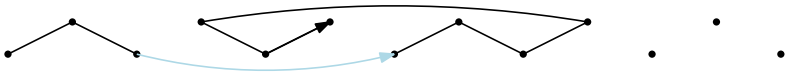

Figure 5.3: Reincorporating the large cycle.

Once the path is long enough (at least $n/2$) we add possible Posa rotations. Assume we start with a path $P = (v_1, \ldots, v_s)$, then if we find two edges such that for some index $i \in [s]$ the edges are of the form $\{v_{i+1}, v_s\}$ and $\{v_i, v_j\}$ for some $j > i + 1$, we can rearrange the path to form a new path $P' = (v_1, \ldots, v_i, v_j, v_{j+1}, \ldots, v_s, v_{i+1}, \ldots, v_{j-1})$ and now the new path has the same vertex set but a different endpoint vertex. This we call a Posa rotation. Additionally we will always want *long* Posa rotations meaning $s - i$ must be at least $n/2$ to ensure that we can find the second edge needed quickly with high probability.

So during our Algorithm if the neighbor (v_{i+1}) of the endpoint of the path (v_s) has distance at least $n/2$ from the endpoint along the path we use that edge to build a cycle and continue from the vertex preceding the neighbor (v_i) on the path (see Figure 5.2). This leaves a cycle of size at least $n/2$ and if we ever find one of the vertices on the cycle to be the neighbor of the current endpoint we reincorporate the large cycle by appending it to the path (again giving a new endpoint).

Many details need to be considered on how random variables interact, etc., but leaving those aside one can easily convince oneself that on average the current vertex changes after a constant number of queries to a new random vertex, and that the number of queries until the path length increases by one is geometrically distributed and has an expec-

tation of $n/(n-i)$ where i is the current length of the path. The total number of Posa rotations is thus bounded by

$$\mathcal{O}\left(\sum_{i=1}^{n} \frac{n}{i}\right) = \mathcal{O}(n \log n).$$

As each Posa rotation takes time $\log n$ to realize this gives a total running time of $\mathcal{O}(n \log^2 n)$.

5.2.2 Our Algorithm

We give a short overview of the new algorithm we propose. The algorithm comes in two phases. In phase 1 we find two random perfect matchings. The union of these two random perfect matchings forms a two regular graph, i.e., a set of disjoint cycles or double edges covering all vertices. It is not difficult to show that the number of cycles is with high probability bounded by $2 \log n$. In phase 2 of the algorithm we stitch these $2 \log n$ cycles together.

For the analysis of the algorithm it is very helpful to assume that a query for a new neighbor of some vertex v returns a vertex w that is *uniformly* distributed over all vertices in $V - v$ and *independent* from all previous queries. Of course such an assumption a priori does not hold if we simply return the next vertex from the adjacency list of v. We realize this by directing the edges and resampling. More formally, we will show the following lemma in Section 5.3; in the remainder of Section 5.2 we will use the corresponding function `newneighbor()` as a black box.

Lemma 5.8 (newneighbor)**.** *It is possible to interact with the graph $G(n,p)$, $p \geq \frac{C \log n}{n}$, with an algorithmic procedure* ***newneighbor(v)*** *which has the following properties with high probability:*
(i) Calling ***newneighbor(v)*** *returns a neighbor of v distributed uniformly among $V - v$ and independent of all calls so far – as long as we make at most $\mathcal{O}(n)$ calls to* ***newneighbor()*** *altogether and every vertex is queried at most $100 \log n$ times. (ii) The total run time of all $\mathcal{O}(n)$ calls is $\mathcal{O}(n)$.*

Note that this algorithm uses both internal randomness as well as the randomness of $G(n,p)$. If `newneighbor(`v`)` ever returns 'there are no

more neighbors' we immediately terminate the entire algorithm and return failure. To avoid this, we will prove that we query `newneighbor(`v`)` from any vertex at most $100 \log n$ times w.h.p. and choose C large enough so that with high probability the minimum degree of the random graph is large enough.

Phase 1: Perfect Matching

In the first phase of the algorithm we show that we can find a perfect matching in $\mathcal{O}(n)$ time. We call the algorithmic procedure described in this section `FastPerfectMatching`, see Algorithm 1. In fact, for an easier understanding of the required ideas, we work in this section with a random *bipartite* random graph. This can easily be done by partitioning the vertex set V into two equal sets A and B arbitrarily (if n is odd we set one vertex aside and include it in phase 2) and only considering the edges between A and B. Formally, the function `newABneighbor(v)` calls `newneighbor(v)` until we receive a neighbor which is in B (resp. A).

Claim 5.2. *If we call `newABneighbor()` for a sequence of $\mathcal{O}(n)$ vertices, in which every vertex $v \in A \cup B$ occurs at most $\log n$ times, then with high probability this results in at most $\mathcal{O}(n)$ calls to `newneighbor()` with at most $6 \log n$ calls per vertex.*

The claim holds because any call of `newneighbor()` has probability at least $1/2$ to be in the correct partition and, by our assumptions on `newneighbor()`, the calls are independent. We can thus apply concentration bounds for binomial distributions and union bound for every vertex. Clearly, `newABneighbor()` still has a uniform and independent distribution over all vertices of the opposite partition.

Let G be the balanced bipartite graph with partitions A and B. During the algorithm we will maintain a matching M which covers some of the vertices and is empty at first. At any point in time, we denote by A_M the vertices in A that are covered by the matching and with A_0 the unmatched vertices. Equivalently for B_M and B_0.

Additionally we need a set of edges that expand well from the vertices of A. And we need to be able to keep track of them efficiently and on the fly. So for any vertex v we define the d-neighborhood of v, $N_d(v) \subseteq V(G)$, to be the set of the first $\lceil d \rceil$ calls to the function `newABneighbor(`v`)`. In particular this implies that for any $d' < d$ the

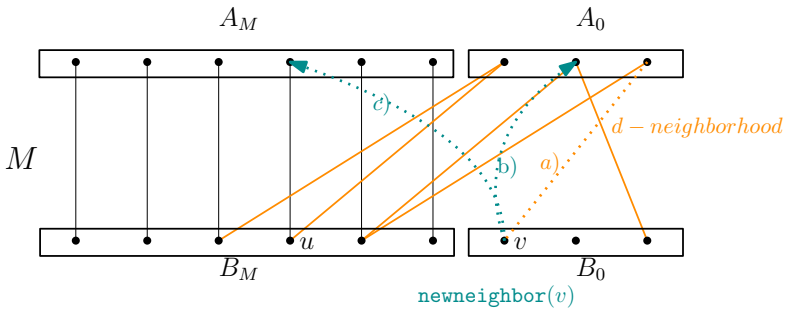

Figure 5.4: For `IncreaseMatching` three things can happen. Either a) the vertex is already in the neighborhood of A_0, in which case we match immediatly, b) the vertex `newABneighbor(`v`)` is in A_0, which also gets matched, or c) `newABneighbor(`v`)` is in A_M. Then we swap the matching and continue from the partner of the `newABneighbor(`v`)`.

d'-neighborhood is contained in the d-neighborhood of v. Similarly, the d-neighborhood of a set of vertices S, denoted by $N_d(S)$, is defined as the union of the d-neighborhoods of all vertices in S. We expose and keep track of the d-neighborhood of the unmatched vertices A_0, $N_{d(|A_0|)}(A_0)$, for the function $d(t) = \min(\sqrt{n/t}, \log n)$. This gives us a neighborhood large enough for the random walks to be effective, but small enough so that we do not need too much time to update/expose.

To increase the matching we call a subroutine `IncreaseMatching`.

`IncreaseMatching` takes as argument the current matching M and an unmatched vertex $v \in B_0$. It proceeds as follows. If v is in $N_d(A_0)$ we add the corresponding neighbor in A_0 and v to the matching. If not we take $w =$ `newABneighbor(v)`. If w is in A_0 we add the edge $\{w, v\}$ to M. If neither of the two is the case, then $w \in A_M$ and there exists a unique u such that $\{w, u\}$ is currently in M. We swap $\{w, u\}$ for $\{w, v\}$, thereby making u a new unmatched vertex, and repeat `IncreaseMatching` with u, cf. Figure 5.4.

Clearly, during the run of the algorithm we also have to dynamically update the d-neighborhood of A_0. In particular this means removing $N_d(w)$ of a newly matched vertex w and, if $d(|A_0|)$ increases, adding vertices from additional calls to `newABneighbor()` for every vertex in A_0.

Algorithm 1 $FastPerfectMatching(G)$

1: $B_0 \leftarrow B$; $B_M \leftarrow \{\}$; $A_0 \leftarrow A$; $A_M \leftarrow \{\}$;
2: d $\leftarrow 0$; $M \leftarrow \{\}$
3: **while** $B_0 \neq \{\}$ **do**
4: $v \leftarrow$ arbitrary vertex from B_0 ▷ and remove from B_0
5: IncreasingMatching(G, M, v); ▷ see Algorithm 2
6: **while** $d < \min\left(\sqrt{\frac{n}{|A_0|}}, \log(n)\right)$ **do**
7: $d \leftarrow d + 1$
8: Add `newABneighbor(`v`)` to the d-neighborhood for every vertex in $v \in A_0$
9: **return** Matching M

To bound the runtime of Algorithm 2, `FastPerfectMatching`, we observe first that we increase the matching exactly n times, which is inline with our desired bound of $\mathcal{O}(n)$. We can thus concentrate on bounding the *recursive* calls to `IncreaseMatching` in line 13 of `IncreaseMatching`.

Lemma 5.3. *Let $\mathcal{L}_i$ denote the number of calls `IncreaseMatching` in line 13, while $|A_0| = i$ for any $i \in [n]$. Then the $\mathcal{L}_i$ are dominated by independent geometric distributions with success probability $p_i = \frac{i \cdot d(i)}{100n}$.*

Proof. Whenever we are at a vertex v in B we expose an edge to a random neighbor in the set A. If that vertex is in A_0 we match v and $|A_0|$ decreases by one so we end the count of $\mathcal{L}_{|A_0|}$. Otherwise we swap with a matched vertex and get a new starting point in B_0. As `newABneighbor()` is independent and uniform, and the matching forms a bijection between A_M and B_M, the fact that the vertex is not in A_0, implies that we get a new *random* vertex u in B_M for the next call. If this vertex is in the exposed d-neighborhood of A_0 we stop and match to a vertex in A_0 also ending the count of $\mathcal{L}_{|A_0|}$.

To assess the probability of stopping, we use the *expansion properties* of the d-neighborhood of A_0 that are inherited from the random graph. This means in particular that the exposed neighborhood of A_0, $N_{d(|A_0|)}(A_0)$, has size at least $\frac{1}{100}|A_0| \cdot d(|A_0|)$, cf. Lemma 5.9 in Section 5.3 for a proof. The probability of hitting a vertex in A_0 or the d-neighborhood of A_0 (while looking at the matched vertex of w in B_M) is thus at least $\frac{|A_0| \cdot d(|A_0|)}{100n}$. Every new call of `newABneighbor()`

Algorithm 2 *IncreaseMatching*(G, M, v)

```
1: if v ∈ N_d(A_0) then
2:     w ← [neighbor of v] ∈ A_0
3:     Add {v, w} to M
4:     Remove w from A_0 and update N_d(A_0)
5:     return
6: w ← newABneighbor(v)
7: if w ∈ A_0 then
8:     Add {v, w} to M
9:     Remove w from A_0 and update N_d(A_0)
10:    return
11: u ← unique vertex with {u, w} ∈ M
12: Remove {u, w} from M and replace with {v, w}
13: IncreaseMatching(G, M, u)
14: return
```

is independent by Lemma 5.8, thus $\mathcal{L}_i$ is dominated by an independent geometric distribution with success probability as claimed. □

We are now ready to proof the desired complexity bound:

Proposition 5.4. *`FastPerfectMatching` finds a perfect matching in a balanced random bipartite graph in time $\mathcal{O}(n)$ with high probability.*

Proof. There are two main contributions to the running time of the Algorithm. First the subroutine `IncreaseMatching`, which we prove to be fast with the help of Lemma 5.3, and secondly the updating and revealing of the d-neighborhood.

Recall that $\mathcal{L}_i$ is the random variable corresponding to the number of calls of `IncreaseMatching` in line 13, while $|A_0| = i$ for any $i \in [n]$. We set $\mathcal{L} = \sum_{i=1}^{n} \mathcal{L}_i$. Note that we can ignore the calls in line 5 of `FastPerfectMatching`, as these add only at total of $\mathcal{O}(n)$ to the run time. From Lemma 5.3 we know that there exists a coupling to a geometrically distributed random variable $\mathcal{L}'$ such that $\mathcal{L}'_i \succeq \mathcal{L}_i$ and $\mathcal{L}'_i$ *is* geometrically distributed with $p_i = \frac{i \cdot d(i)}{100n}$.

From the definition of $\mathcal{L}'_i$ we know that $\mathbb{E}[\mathcal{L}'_i] = \frac{100n}{i \cdot d(i)}$ and $Var[\mathcal{L}'_i] = \frac{1-p_i}{p_i^2} \leq \frac{1}{p_i^2} \leq (\frac{100n}{i \cdot d(i)})^2$. Recall that $d(i) = \sqrt{n/i}$ whenever $i \geq \frac{n}{(\log n)^2}$. The total time used for those sets can thus be bounded in expectation

by

$$\sum_{i=\frac{n}{(\log n)^2}}^{n} \mathbb{E}[\mathcal{L'}_i] = \mathcal{O}\left(\sum_{i=1}^{n} \frac{\sqrt{n}}{\sqrt{i}}\right) = \mathcal{O}(n),$$

as $\sum_{i=1}^{n} i^{-1/2} \leq \int_0^n x^{-1/2}\,\mathrm{d}x = 2\sqrt{n}$. If $i \leq \frac{n}{\log(n)^2}$, then $d(i) = \log n$, and the total expected time used for these sets is thus bounded by

$$\sum_{i=1}^{\frac{n}{(\log n)^2}} \mathbb{E}[\mathcal{L'}_i] = \mathcal{O}\left(\sum_{i=1}^{n} \frac{n}{i \cdot \log(n)}\right) = \mathcal{O}(n).$$

We thus have that $\mathbb{E}[\mathcal{L'}] = \Theta(n)$ as well. To show that the actual run time is concentrated around the expectation we apply Chebyshev's inequality. A similar case distinction as above gives us

$$Var[\mathcal{L'}] \leq \sum_{i=\frac{n}{(\log n)^2}}^{n} \frac{10000n}{i} + \sum_{i=1}^{\frac{n}{(\log n)^2}} \frac{10000n^2}{i^2 \cdot (\log n)^2} = \mathcal{O}\left(\frac{n^2}{(\log n)^2}\right).$$

By Chebyshev's inequality we thus get

$$Pr[\mathcal{L} \geq 2\mathbb{E}[\mathcal{L'}]] \leq Pr[\mathcal{L'} \geq 2\mathbb{E}[\mathcal{L'}]] \leq \frac{Var[\mathcal{L'}]}{\left(\mathbb{E}[\mathcal{L'}]\right)^2} \leq \mathcal{O}\left(\frac{1}{(\log n)^2}\right),$$

which concludes the first part of the proof.

To bound the time needed to expose the d-neighborhoods, we observe first that we can order the vertices in A by the order in which they join the matching. As $N_{d'}(v) \subseteq N_d(v)\ \forall d' \leq d$, we thus have to expose for the i-th vertex in this ordering at most $d(n-i)+1$ edges, where $d(x) = \min\{\sqrt{n/x}, \log n\}$. Thus, the total number of exposed edges is bounded by

$$\begin{aligned}\sum_{i=1}^{n}(d(n-i)+1) &= \sum_{i=1}^{\frac{n}{(\log n)^2}} (d(i)+1) + \sum_{i=\frac{n}{(\log n)^2}}^{n} (d(i)+1) \\ &\leq \sum_{i=1}^{\frac{n}{(\log n)^2}} (\log n + 1) + \sum_{i=\frac{n}{(\log n)^2}}^{n} \sqrt{\frac{n}{i}} \leq 4n.\end{aligned}$$

Additionally we show the number of calls to the `newABneighbor()` function is at most $\log n$ for every vertex w.h.p.. For any $v \in B$ we call `newABneighbor(`v`)` exactly once for each time it appears as the matched partner of `newABneighbor(`v`)`. As the distribution on the neighbors is uniform on A and we only use `IncreaseMatching` $\mathcal{O}(n)$ many times in total, the probability that $v \in B$ occurs at least $\log n$ times is at most

$$\binom{\mathcal{O}(n)}{\log n}\left(\frac{1}{n}\right)^{\log n} = \mathcal{O}(n^{-2}),$$

with room to spare. We can thus apply a union bound over all vertices in B to see that w.h.p. no vertex in B has more than $\log n$ calls to `newABneighbor()`. Clearly the same holds for vertices in A, as we only expose the d-neighborhood and $d(|A_0|) \leq \log n$ always. This concludes the proof of Proposition 5.4. □

Phase 2: Incorporating the Cycle Factor

In the previous section we have seen that we can find a perfect matching in $\mathcal{O}(n)$ time. In this section we show how we can extend this algorithm to find a Hamilton cycle. To do this we first call the perfect matching algorithm *twice*, reseting the d-neighborhoods after the first run. By our assumption on the independence on the calls to the function `newneighbor()`, we thereby get two *independent* random perfect matchings. Their union forms a union of cycles (or double edges) covering all vertices (if the number of vertices was odd we add the single vertex excluded in phase 1 here back as a cycle with one vertex). Our task in this phase is to join these cycles into a single cycle. We start with a lemma that bounds the number of cycles that we need to join.

Lemma 5.5. *The union of two random independent perfect matchings from a complete bipartite graph contains at most* $2\log n$ *cycles with high probability.*

Proof. We claim that the two independent perfect matchings can be seen as a random permutation of $[n/2]$. Indeed, without loss of generality we may assume that M_1 is just the identity (by renumbering the vertices appropriately). M_1 and M_2 are independent which implies M_2 corresponds to a random assignment of B to A. The union of the two matchings thus defines a random permutation of A.

For random permutations the number of cycles has been well studied and is related to the Stirling numbers of the first kind. Using a double counting argument one can easily see that the expected number of cycles of length $2k$ will be $1/k$. The total expected number of cycles is thus equal to the nth harmonic number. It is also well-known that this random variable is concentrated, see e.g. [ABT03] or [AT92, MNZ12]. Thus, with high probability the number of cycles is bounded by $2 \log n$, as claimed. □

Description of Algorithm 3 `JoinCycles`**.** To glue the cycles together we proceed in three phases. First we greedily combine cycles into a path, until this path has length at least $3n/4$. Then we incorporate the remaining cycles one by one using Algorithm 4 `AddSingleCycle`. Finally, we close the Hamilton path into a Hamilton cycle (Lemma 5.7).

The idea behind the first phase is straightforward. We start with an arbitrary cycle and break it apart into a path P. Consider the endvertex p_{end} of that path. We use `newneighbor()` to query a new neighbor of p_{end}. If that neighbor is in a new cycle (which will happen with probability at least $1/4$, as long as the path P contains at most $3n/4$ vertices), we attach that cycle to P, thereby also getting a new endpoint p_{end}. If the latter did not happen, we query a new neighbor. In order to ensure that we do not query to many vertices from a single vertex, we repeat the query for new neighbors at most $40 \log n$ times. If we have not been successful by then, we give up. It is easy to see that the probability for ever giving up at this stage of the algorithm is bounded by $o(1)$. It is also easy to see that the total time spent until the path has length at least $3/4n$ is bounded by $\mathcal{O}(n)$.

Once the path has length at least $3n/4$, the probability that a new neighbor is in one of the remaining cycles gets too small (for our purpose) and we thus change strategy. In particular, we add long Posa rotations, so that we can try various endpoints. This is the purpose of the procedure `AddSingleCycle` (Algorithm 4).

We use a set U to keep track of *used* vertices. Those are vertices for which we already queried neighbors within the algorithm `JoinCycles`. We denote the current path by $P = (p_{start}, .., p_{end})$. We also assume that we have access to a function $pred_P(v)$ that determines the vertex before v on the path (null for p_{start}), and a function $half_p(v)$ which is true iff v is in the first half of P. We denote the cycle C that we want to add as $C = (c_{start}, ..., c_{end})$, where c_{start} is an arbitrary vertex

at which we cut C into a path. We now explore neighborhoods of vertices at once. To do this we denote by `newneighbor`$(v, 40 \log n)$ the set of vertices that we obtain if we apply `newneighbor`(v) $40 \log n$ times. Let $N_{start} = P \cap$ `newneighbor`$(c_{start}, 40 \log n)$ and $N_{end} = P \cap$ `newneighbor`$(c_{end}, 40 \log n)$ denote the intersections of these neighborhood vertices with the path P. Until the cycle C is part of the path P we do the following (Figure 5.5). Let $N(p_{end}) =$ `newneighbor`$(p_{end}, 40 \log n)$ and check for all $v \in N(p_{end})$ if $pred_P(v) \in N_{start}$. If so we also check if $halfp_p(v)$ is true.

If we find a vertex v for which both conditions hold, we join the cycle here. Consider N_{end} and take a vertex $q \in$ `newneighbor`$(c_{end}, 40 \log n)$ such that $halfp_P(q)$ is false and $pred_P(q) \notin U$. Then we add the cycle to the path by constructing the new path $P_{new} = (p_{start}, ..., pred_P(v)) + (c_{start}, ..., c_{end}) + (q, ..., p_{end}) + (v, ..., pred_P(q))$. Then add p_{end}, c_{start} and c_{end} to the used vertices U. (If we cannot find q we abort the algorithm; it will be easy to show that the probability that this happens is negligible.)

If the check fails for all $v \in N(p_{end})$ we perform a Posa rotation. Consider a $v \in N(p_{end}), v \neq p_{start}$, such that $halfp_P(v)$ is true and such that $pred_P(v) \notin U$, and then take a $q \in$ `newneighbor`$(pred_P(v), 40 \log n)$ such that both $halfp_p(q)$ is false and $pred_P(q)$ is unused. We then use v and q to construct a new path with a new endpoint, namely $P_{new} = (p_{start}, ..., pred_P(v)) + (q, ..., p_{end}) + (v, ..., pred_P(q))$. Now we can repeat the above procedure with P_{new} and the new endpoint $p_{newend} = pred_P(q)$. (If we cannot find v or q we abort the algorithm; again it will be easy to show that the probability that this happens is negligible.)

To store the path and cycles we use AVL trees with a linked list. The linked list just stores the vertices in the order as they appear in the path resp. cycle. For the AVL tree we take the ordering in the path/linked list as an ordering of the vertices. With this ordering at hand, the AVL tree is well defined, and it allows for searching resp. answering the query $halfp(v)$ in $\mathcal{O}(\log n)$ time. In addition, splitting the path resp. concatenating two paths correspond to splitting an AVL tree at a given vertex (into a tree containing all smaller vertices and a tree containing the remaining vertices) resp. concatenate two AVL trees in which the largest vertex in one tree is smaller than the smallest vertex in the other tree. It is well known that both of these operations can be done for AVL trees in $\mathcal{O}(\log n)$ time, cf. Lemma 5.10 in Section 5.3 for more details.

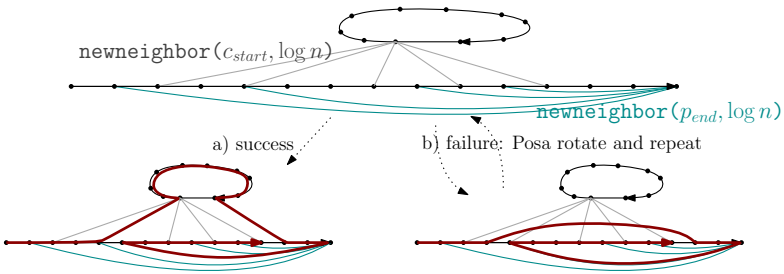

Figure 5.5: Incorporating a single new cycle with `AddSingleCycle`. The dark red path indicating the new Path after an iteration of the while loop.

Proposition 5.6. *Applying the procedure* `AddSingleCycle` *at most* $2\log n$ *times will run in time* $\mathcal{O}(n)$ *with high probability.*

Proof. We want to bound the number of Posa rotations we need to perform while we add at most $2\log n$ cycles. Each Posa rotation occurs at the end of a while loop in the pseudocode.

To incorporate a cycle we want to find a vertex v which, in the order of the path, is right after a vertex in N_{start} and is in the first half of P. P has size at least $3n/4$ so the number of vertices in the first half is at least $n/4$. A random vertex therefore has a chance of at least $1/4$ to be in the first half of P. So every vertex in `newneighbor`$(c_{start}, 40\log n)$ has probability at least $1/4$ independently of being in the first half of P and different from the other vertices. This implies that the number of vertices in N_{start} which are also in the first half of P dominates a binomial distributed random variable $F \sim Bin(40\log n, 1/4)$. For F we know the expectation to be $10\log n$ and by a Chernoff bound (2.3) the probability that F is less than $\log n$ is $\mathcal{O}(n^{-2})$. We observe that where the Posa rotation happens is independent of N_{start}. So we apply a union bound that on fixed $\mathcal{O}(n)$ many rotations of P the probability that there are less than $\log n$ vertices of N_{start} in the first half of P is in $\mathcal{O}(n^{-1})$. This implies that any call to `newneighbor`(p_{end}) has a chance of at least $\log n/n$ to be right after a vertex in N_{start} and also in the first half of P. As each call to `newneighbor()` is independent, the number of tries we must make is geometrically distributed with success probability $\log n/n$ and we must succeed at most $2\log n$ many times. This means the num-

Algorithm 3 *JoinCycles*$(G, \mathcal{C} = M_1 \cup M_2)$

$U \leftarrow \{\}$
$C_0 \leftarrow$ first cycle of $M_1 \cup M_2$, $(c_{0,start}, ..., c_{0,end})$;
$P \leftarrow (c_{0,start}, ..., c_{0,end})$;
$p_{end} \leftarrow$ last vertex of P;
while $|P| \leq \frac{3n}{4}$ **do**
    $N \leftarrow$ `newneighbor`$(p_{end}, 40 \log n)$;
    $U \leftarrow$ add p_{end};
    $v \leftarrow$ Search N for v such that $v \notin P$
    $(v, ..., c_{i,end}) \leftarrow$ cycle of v;
    $P \leftarrow P + (v, ..., c_{i,end})$;
    $p_{end} \leftarrow c_{i,end}$;
while $|P| \neq n$ **do**
    $C_i \leftarrow$ any cycle not in P
    AddSingleCycle(G, P, C_i, U) ▷ See Algorithm 4
return
// If any of the 'Search' parts of the algorithm fail, we abort the algorithm and return failure.

ber of Posa rotations is dominated by a negative binomial distributed random variable $R \sim NB(2 \log n, \log n/n)$. So by the concentration of the negative binomial distribution (Lemma 2.6) the probability that we need to try more than $4n$ times is at most $\mathcal{O}(\log^{-1} n)$. Before every Posa rotation we try `newneighbor`$(p_{end}, 40 \log n)$ so $40 \log n$ tries. This proves an upper bound on the number of Posa rotations of $\mathcal{O}(n/\log n)$ with high probability.

Posa rotation: We summarize the operations we need to do per Posa rotation. This assumes that we already failed to find v which is both after a vertex in N_{start} and also in the first half of P. We expose $40 \log n$ new neighbors of p_{end} and $40 \log n$ of the vertex before v on the path, we need to Posa rotate by splitting the path twice and then joining twice. Checking whether a vertex is in U and adding vertices to U is a constant time operation with a lookup table. All of these operations by choice of proper datastructure (Lemma 5.8 and 5.10) are done in $\mathcal{O}(\log n)$. So over all Posa rotations these sum up to a runtime of at most $\mathcal{O}(n)$. Additionally we need to find the vertex v in the first half of P with $pred_P(v) \notin U$. Since U is much smaller than $n/8$ and $|P| \geq 3n/4$ the number of possible vertices is at least $n/4$. This

Algorithm 4 *AddSingleCycle*(G, P, C_i, U)

// Function $half_P(v)$ returns true if and only if v is in the first half of P;
// For any vertex $v \in P$, $pred_P(v)$ denotes the vertex before v on the path P ;
1: $p_{end} \leftarrow$ last vertex of P;
2: $N_{start} \leftarrow$ `newneighbor(`$c_{start}, 40 \log n$`)` $\cap P$;
3: $N_{end} \leftarrow$ `newneighbor(`$c_{end}, 40 \log n$`)`;
4: $U \leftarrow$ add c_{start} and c_{end};
5: **while** true **do**
6: $N \leftarrow$ `newneighbor(`$p_{end}, 40 \log n$`)`;
7: $U \leftarrow$ add p_{end};
8: **if** $\exists v \in N \;\; s.t. \;\; pred_P(v) \in N_{start}$ and $half_P(v) = true$ **then**
9: $q \leftarrow$ Search N_{end} for q such that $half_p(q) = false$ and $pred_P(q) \notin U$;
10: $P \leftarrow (p_{start}, ..., pred_P(v)) + (c_{start}, ..., c_{end}) + (q, ..., p_{end}) + (v, ..., pred_P(q))$;
11: **return**
12: **else**
13: $v \leftarrow$ Search N for v such that $half_P(v) = true$;
14: $N \leftarrow$ `newneighbor(`$pred_P(v), 40 \log n$`)`;
15: $U \leftarrow$ add $pred_P(v)$;
16: $q \leftarrow$ Search N for q such that $half_P(q) = false$ and $pred_P(q) \notin U$;
17: $P \leftarrow (p_{start}, ..., pred_P(v)) + (q, ..., p_{end}) + (v, ..., pred_P(q))$;
18: $p_{end} \leftarrow pred_P(q)$;
19: // If any of the 'Search' parts of the algorithm fail, we abort the algorithm and return failure.

means that if we test a random vertex, the probability that $half_p()$ returns true and its predecessor is not in U is at least $1/4$. So the number times we need to call $half_P()$ is dominated by a geometric distribution with success probability $1/4$. Similarly to find the vertex q in the second half of P with $pred_P(q) \notin U$, the number of times we need to call $half_P()$ is also dominated by a geometric distribution with success probability $1/4$. So over all rotations, the number of times we need to call $half_P()$ is dominated by a negative binomial distribution $H \sim NB(2 \cdot \mathcal{O}(n/\log n), 1/4)$. So by the concentration of the negative

binomial distribution (Lemma 2.6) the probability that we need to call $halfP$ more than $\mathcal{O}(n/\log n)$ times is $\mathcal{O}(\log n/n)$. And since we can perform $halfP()$ in time $\mathcal{O}(\log n)$ by Lemma 5.10 these have a total runtime of $\mathcal{O}(n)$ with high probability.

Incorporating cycles: Very similarly we bound the time we need to incorporate the cycles. To find the vertex v which in the order of the path is right after a vertex in N_{start} and is in the first half of P we need to call $halfP$ until we succeed. Note that since $|N_{start}| \leq 40 \log n$ and as we proved above at least $\log n$ vertices of them are in the first half of P, every call to $halfP()$ from a random vertex after a vertex in N_{start} has a chance of succeeding of at least $1/40$. This means the number of times we call $halfP$ is again dominated by a negative binomial distribution $NB(2 \log n, 1/40)$ and this runtime is negligible with high probability. As we only incorporate a cycle $2 \log n$ times, also the join and split operations as well as the exposing of N_{end} and searching for q are negligible compared to the $\mathcal{O}(n)$ runtime.

Note also that we only call `newneighbor()` of vertices we then add to U and then not again during the entire algorithm so no vertex has `newneighbor()` called more than $40 \log n$ times. At most $\mathcal{O}(n/\log n)$ many vertices are added to U, and U is small enough so that it is always much smaller than $n/8$.

This concludes the proof of Proposition 5.6. □

Lemma 5.7. *Given a Hamilton path we can transform it to a Hamilton cycle in $\mathcal{O}(n)$ time.*

Proof. Calling the algorithm `AddSingleCycle` with the cycle being p_{start}, but instead looking for v such that a vertex after v is in the neighborhood of p_{start} instead of a predecessor gives us a cycle $C = (p_{start}, ..., v) + (p_{end}, ..., after_P(v))$. The analysis of the runtime is equivalent to the analysis of `AddSingleCycle`. □

Propositions 5.4 and 5.6 as well as Lemma 5.7 show that all components of the algorithm run in time $\mathcal{O}(n)$. It is also easy to check that both phases together require at most $50 \log n$ calls to `newneighbor()` from any fixed vertex, so the assumptions of Lemma 5.8 do hold. So choosing C large enough, say $C = 200$, suffices to guarantee that with high probability the random graph is such that all vertices have more neighbors than we query. This thus concludes the proof of Theorem 5.1.

5.3 Datastructures

In this section we give the details of the data structures that we used within our algorithm for finding the Hamilton cycle.

5.3.1 Querying a new vertex

As explained above, we assumed throughout the analysis of our algorithm that we have access to a function `newneighbor(v)`, that returns for a given vertex v a neighbor w that is *uniformly* distributed in $V - v$ and whose result is *independent* from all previous calls.

Lemma 5.8 (newneighbor)**.** *It is possible to interact with the graph $G(n,p)$, $p \geq \frac{C \log n}{n}$, with an algorithmic procedure `newneighbor(v)` which has the following properties with high probability:*
(i) Calling `newneighbor(v)` returns a neighbor of v distributed uniformly among $V - v$ and independent of all calls so far – as long as we make at most $\mathcal{O}(n)$ calls to `newneighbor()` altogether and every vertex is queried at most $100 \log n$ times. (ii) The total run time of all $\mathcal{O}(n)$ calls is $\mathcal{O}(n)$.

Proof. To realize such a function `newneighbor()`, it is important to make the adjacency lists independent of each other. To realize this we transform the graph G (which is distributed as a random graph $G(n,p)$) into a directed graph G' distributed as $D(n, p/2)$ (in which each directed edge is present independently with probability $p/2$). It is well know how this can be done. In particular, we can sample $D(n, p/2)$ from $G(n,p)$ as a subgraph such that every edge in the directed graph is also an undirected edge in the $G(n,p)$. More precisely, we do the following for every edge $\{i,j\}$ of G: with probability

$$\begin{aligned}
\frac{1}{2} - \frac{p}{4} &\text{ set } (i,j) \in G' \text{ and } (j,i) \notin G' \\
\frac{1}{2} - \frac{p}{4} &\text{ set } (i,j) \notin G' \text{ and } (j,i) \in G' \\
\frac{p}{4} &\text{ set } (i,j) \in G' \text{ and } (j,i) \in G' \\
\frac{p}{4} &\text{ set } (i,j) \notin G' \text{ and } (j,i) \notin G'
\end{aligned}$$

In order to be consistent with the transformation from G to G' and to not lose too much time we only direct the edges once we see it for the first time. To recall the made decision, we store the random choices of the edges that we have we encountered so far into a hashtable. Thus, we can check for each edge that we obtain from querying the adjacency list of a vertex in G, whether we have seen this edge already and if so, which orientation we have chosen. The hash table has size n and we use a hashfunction which is 4-wise independent. This way the variance of the number of collisions is equal to a random function, and therefore the time we need for hashing is $\mathcal{O}(n) + \mathcal{O}(number\ of\ collisions) = \mathcal{O}(n)$, which can be seen by applying Chebyshev's inequality. A more detailed argument of why linear probing with hash functions works in this context can be found in [PPR07, TZ04].

Finally, we want the distribution of the next edge to be uniform among the vertices. For this we need to resample from the already seen edges. Assuming we have revealed d many edges from v we flip a biased coin. With probability $d/(n-1)$ we retake an old neighbor and output it, one chosen uniformly at random, and with probability $1 - d/(n-1)$ we take the next vertex in the adjacency list (which is also in $D(n,p)$). Otherwise, we return one of the previously seen neighbors uniformly at random. In this way any vertex has probability exactly $1/(n-1)$ to be returned by `newneighbor(v)`.

□

5.3.2 Expansion

What we need from the random graph are properties of good expansion. Given the adjacency list of a vertex v we define the d-neighborhood $N_d(v) \subseteq V(G)$ to be the set of the first $\lceil d \rceil$ calls to the function `newABneighbor(v)`. In the analysis of the algorithm we make use of the following lemma.

Lemma 5.9 (Neighborhood Lemma)**.** *Let $G(n/2, n/2, p)$ be a random bipartite graph with $p \geq \frac{C\ \log(n)}{n}$ and partitions A and B. Then with high probability we have for all subsets $A' \subseteq A$ that the d-neighborhood of A' is of size at least*

$$|N_{d(A')}(A')| \geq \frac{1}{100}|A'| \cdot d(A'), \tag{5.1}$$

where $d(A') = min(\sqrt{\frac{n}{|A'|}}, \log(n))$.

Proof. The proof follows from a straight forward calculation of probabilities. Let us assume by contradiction there exists a set $A' \subseteq A$ with $|N_{d(A')}(A')| < \frac{1}{100}|A'| \cdot d(A')$. Then there is a set $B' \subseteq B$ of size $|B'| = \frac{1}{100}|A'| \cdot d(A')$ containing this d-neighborhood, $N_{d(A')}(A') \subseteq B'$. So this is a probability we want to bound from above. The probability for a single vertex in A' to have its d-neighborhood contained in a fixed set B' is $\left(\frac{|B'|}{n}\right)^{d(A')}$ since the d-neighborhood is $d(A')$ vertices chosen from B uniformly and independently at random. The probability for two specific sets $A' \subseteq A$ and $B' \subseteq B$ to have this property is $(|B'|/n)^{|A'| \cdot d(A')}$. Now take the union bound over all possible sets A' and B' (with $|B'| = \frac{1}{100}|A'| \cdot d(A')$):

$$Pr[(5.1) \text{ false}] \leq \sum_{A', B'} Pr[B' \text{ contains } N_{d(A')}(A')]$$

$$= \sum_{i=1}^{n} \binom{n}{i} \binom{n}{\frac{1}{100} i \; d(i)} \left(\frac{\frac{1}{100} i \cdot d(i)}{n}\right)^{i \cdot d(i)} \tag{5.2}$$

Then we apply an approximation for the binomial coefficients: $\binom{n}{k} \leq \left(\frac{en}{k}\right)^k$. We see that $\frac{1}{4} \log \frac{100n}{i \cdot d(i)} \geq \log(e \cdot n/i)/d(i)$ so

$$Pr[(5.1) \text{ false}] \leq \sum_{i=1}^{n} exp\left(i \cdot d(i) \cdot \left(-\frac{1}{2} \log\left(\frac{100n}{i \cdot d(i)}\right)\right)\right)$$

Now $d(i)$ is a known function of i. So we distinguish between two cases. When $i \geq n/\log^2(n)$, then $d(i) = \sqrt{n/i}$. And we can calculate $(i \leq n)$

$$\sum_{i=1}^{n} \left(\frac{i^{1/4} \cdot n^{1/4}}{10 \cdot n^{1/2}}\right)^{\sqrt{n\ i}} \leq \sum_{i=1}^{n} \left(\frac{1}{10}\right)^{\sqrt{n\ i}} \leq \mathcal{O}(n^{-2})$$

On the other hand if $i \leq n/\log^2(n)$, then $d(i) = \log(n)$. And we can calculate

$$\sum_{i=1}^{n/\log^2(n)} \left(\frac{i^{1/2} \cdot \log(n)^{1/2}}{10 \cdot n^{1/2}}\right)^{i \log(n)} \leq \sum_{i=1}^{n/\log^2(n)} \left(\frac{1}{10 \cdot \log(n)^{1/2}}\right)^{i \log(n)}$$

$$\leq \mathcal{O}(n^{-2}) \tag{5.3}$$

Together this implies that the lemma holds for random graphs with probability $\geq 1 - \mathcal{O}(n^{-2})$.

□

5.3.3 AVL Trees

Lemma 5.10 (AVL Trees)**.** *We can store a path (or cycle) in an AVL tree joint with a linked list datastructure and can perform the following operations (where we view the cycle as a path split at an arbitrary point):*

- *For any vertex v find the vertex preceding or succeeding it in the path in constant time $\mathcal{O}(1)$*
- *For any vertex v searching the path it is in and determining whether it is in the first or second half of it in time $\mathcal{O}(\log n)$*
- *Split the path into two paths in time $\mathcal{O}(\log n)$*
- *Concatenate two paths into one by adding the endpoint of one to the start of the other in time $\mathcal{O}(\log n)$*

Proof. We combine an AVL tree, which is a balanced binary search tree, with a linked list. The AVL tree is built on the order sequence of the path as if numbering the vertices along the path from 1 to $|P|$. The linked list ensures that going forward and backward on the path is done in constant time, where the AVL tree can perform search (for the half function) in $\mathcal{O}(\log n)$. A split of the path is nothing other than splitting the AVL tree at a leaf node into two trees such that all the nodes smaller go into one tree and all the nodes larger go into the other. The concatenate is the inverse of the split and only requires attaching the smaller tree to the larger one at the appropriate node and rebalancing up to the root. Both operations run in $\mathcal{O}(\log n)$ time.AVL trees are by now a part of basic datastructure lectures and in particular the split and concatenate operations can be found e.g. in the book by Knuth [Knu98] see page 473, which also cites from [Cra72] or more generally on AVL trees see [Pfa98]. □

Chapter 6

Mastermind with a Linear Number of Queries

In this chapter, we provide an algorithm (sequence of queries) for solving mastermind with linearly many queries. The content of this chapter is from a submitted paper by Martinsson and the author ([MS20]).

6.1 Introduction

Mastermind is a famous code-breaking board game for two players. One player, code-maker, creates a hidden codeword consisting of a sequence of four colors. The goal of the second player, the codebreaker, is to determine this codeword in as few guesses as possible. After each guess, codemaker provides a certain number of black and white pegs indicating

how close the guess is to the real codeword. The game is over when codebreaker has made a guess identical to the hidden string. The board game version was first released in 1971, though older pen-and-paper versions (e.g. bulls and cows) were known before it.

Guessing games such as Mastermind have gained much attention in the scientific community. This is in part out due to their popularity as recreational games, but importantly also as natural problems in the intersection of information theory and algorithms. In particular, it is not too difficult to see that two-color Mastermind is equivalent to coin weighing with a spring scale. This problem was first introduced in 1960 by Shapiro and Fine [SF60]. In subsequent years, a number of different approaches have been devised which solves this problem up to a constant factor of the information-theoretic lower bound.

The general k color n slot Mastermind first appeared in the scientific literature in 1983 in a paper by Chvátal [Chv83]. By extending ideas of Erdős and Rényi [ER63] from coin-weighing he showed that the information-theoretic lower bound is sharp up to constant factor for $k \leq n^{1-\varepsilon}$.

Surprisingly, for a larger number of colors, the number of guesses needed to reconstruct the codeword has remained unknown. In particular, for $k = n$, the best-known algorithm by Doerr, Doerr, Spöhel, and Thomas [DDST16] reconstructs the codeword in $\mathcal{O}(n \log \log n)$ guesses, whereas no significant improvement on the information-theoretic lower bound of $\Omega(n)$ is known.

In this chapter, we resolve this problem after almost 40 years by showing how the n color n slot Mastermind can be solved in $\mathcal{O}(n)$ guesses, matching the information-theoretic lower bound up to constant factor. By combining this with a result by Doerr et al., we determine asymptotically the number of guesses for any combination of k and n.

6.1.1 Related work

The study of two-color Mastermind dates to an American Mathematical Monthly post in 1960 by Shapiro and Fine [SF60]: "*Counterfeit coins weigh 9 grams and genuine coins all weigh 10 grams. One is given n coins of unknown composition, and an accurate scale (not a balance). How many weighings are needed to isolate the counterfeit coins?*"

It can be observed, already for $n = 4$, that fewer than n weighings are

required. The authors consequently conjecture that $o(n)$ weighings suffice for large n. Indeed, the entropy lower bound states that at least $n/\log_2(n+1)$ weighings are necessary. In subsequent years, many techniques were independently discovered that attain this bound within a constant factor [ER63, CM66, Lin64, Lin65], see also [ER63] for further early works. Erdős and Rényi [ER63] showed that a sequence of $(2+o(1))n/\log_2 n$ random weighings would uniquely identify the counterfeit coins with high probability, and by the probabilistic method there is a deterministic sequence of $\mathcal{O}(n/\log n)$ weighings that identifies any set of counterfeit coins.

Cantor and Mills [CM66] proposed a recursive solution to this problem. Here it is natural to consider *signed* coin weighings, where, for each coin on the scale, we may choose whether it contributes with its weight or minus its weight. We call a $\{-1,0,1\}$ valued matrix A an *identification matrix* if any binary vector x of compatible length can be uniquely determined by Ax. It is simple to show that if A is an identification matrix, then so is

$$\begin{pmatrix} A & A & I \\ A & -A & 0 \end{pmatrix}. \tag{6.1}$$

By putting $A_0 = (1)$ and recursing this formula, we obtain an identification matrix A_k with 2^k rows and $(k+2)2^{k-1}$ columns. Thus, we can identify which out of $n = (k+2)2^{k-1}$ coins are counterfeit by using $2^k \sim 2n/\log_2 n$ *signed* weighings, or, by weighing the $+1$s and -1s separately, using $4n\log_2 n$ (unsigned) weighings. Using a more careful analysis, the authors show that $\sim 2n/\log_2 n$ weighings suffice.

It was shown in [ER63] that $2n/\log_2 n$ is best possible, up to lower order terms, for *non-adaptive* strategies. It is a central open problem to determine the optimal constant for general strategies, but it is currently not known whether adaptiveness can be used to obtain a leading term improvement.

In 1983 Chvátal [Chv83] proposed the generalized version of Mastermind with k colors and n slots. Here the entropy lower bound states that $\Omega(n\log k/\log n)$ guesses are necessary. For $k \le n^{1-\varepsilon}$, he showed that a simple random guessing strategy uniquely determines the codeword within a constant factor of the entropy bound.

For larger k, less is known. For k between n and n^2, Chvátal showed that $2n\log_2 k + 4n$ guesses suffice. For any $k \ge n$, this was improved to $2n\log_2 n + 2n + \lceil k/n \rceil + 2$ by Chen, Cunha, and Homer [CCH96],

further to $n\lceil\log_2 k\rceil+\lceil(2-1/k)n\rceil+k$ by Goodrich [Goo09b], and again to $n\lceil\log_2 n\rceil-n+k+1$ by Jäger and Peczarski [JP11]. As a comparison, we note that if $k = k(n)$ is polynomial in n, then the entropy lower bound is simply $\Omega(n)$. This gap is a very natural one as Doerr, Doerr, Spöhel, and Thomas [DDST16] showed in a relatively recent paper that if one uses a non-adaptive strategy, there is in fact a lower bound of $\Omega(n \log n)$ when $k = n$. In the same paper they also use an adaptive strategy to significantly narrow this gap, showing that $\mathcal{O}(n \log\log n)$ guesses suffice for $k = n$. Moreover, by proving a general relation between $n = k$ Mastermind with only black pegs and $k \geq n$ Mastermind with both black and white pegs, they show that $\mathcal{O}(n \log\log n+k/n)$ guesses suffice for any $k \geq n$.

The special case of $n = 4$ and $k = 6$, being the most common commercial version of Mastermind, has received some special attention. Knuth [Knu77] showed that the optimal deterministic strategy needs 5 guesses in the worst case. In the randomized setting, it was shown by Koyama [Koy93] that optimal strategy needs in expectation $5625/1296 = 4.34\ldots$ guesses for the worst case distribution of codewords.

Stuckman and Zhang [SZ06] showed that it is NP-hard to determine whether a sequence of guesses with black and white peg answers is consistent with any codeword. The analogous result was shown by Goodrich [Goo09b] assuming only black peg answers are given. It was shown by Viglietta [Vig12] that both of these results hold even for $k = 2$.

Mastermind has furthermore been proposed to model attacks on genomic data [Goo09a] and cracking bank pins [FL10].

6.1.2 Results

Our contribution lies in resolving the black-peg Mastermind game for $k = n$ where we have as many possible colors as positions.

Theorem 6.1. *For n colors and n positions we can solve black-peg Mastermind with $\mathcal{O}(n)$ many queries and in polynomial time with high probability.*

The above result is best possible up to a constant factor as the natural entropy lower bound shows. Moreover, if we additionally assume that the codeword can only be a permutation of the colors, that is each color must appear exactly once, then we can solve it deterministically with

$\mathcal{O}(n)$ queries.

Combining our results with earlier results by Doerr, Doerr, Spöhel, and Thomas we are able to resolve the randomized query complexity of Mastermind in the full parameter range, thus finally resolving this problem after almost 40 years. Recall that the randomized query complexity is the minimum (over all strategies) maximum (over all codewords) expected number of queries needed to win the game.

Theorem 6.2. *For k colors and n positions, the randomized query complexity of Mastermind is*

$$\Theta(n \log k / \log n + k/n),$$

if codebreaker receive both black-peg and white-peg information for each query, and

$$\Theta(n \log k / \log n + k),$$

if codebreaker only receives black-peg information.

We believe that the same result holds true the deterministic query complexity, but we will not attempt to prove it here.

6.2 Game of Mastermind

We define a game of Mastermind as a two player game, where one player, the codemaker, chooses a hidden codeword $c = (c_1, ..., c_n)$ in $[k]^n$, where c is an n-tuple or string of colors in $[k]$. The other player, the codebreaker, may submit queries of the form $q = (q_1, ..., q_n) \in [k]^n$ which are also stings of n colors. We call k the number of colors and n the number of positions.

For each query q, we associate two integers, which we call the number of black and white pegs respectively. The number of black pegs is the number of correct positions in which the codeword matches the query string. The number of white pegs is often referred to as the number of correctly guessed colors which do not have the correct position. More formally, the number of white pegs is the number of additional correct positions one can maximally obtain by permuting the entries of q. We denote the number of black and white pegs by $b(q) = b_c(q)$ and $w(q) = w_c(q)$ respectively.

We differentiate between two versions of Mastermind, we call black-peg Mastermind the game when codemaker gives only the number of black pegs as an answer to every query and we call black-white-peg Mastermind as the Mastermind game when codemaker gives both the number of black and the number of white pegs as answer to every query.

The game is over as soon as codebreaker makes a query such that $b_c(q) = n$, that is, $q = c$. The goal of the codebreaker is to make the game ends after as few queries as possible.

6.3 Preliminaries

Doerr, Doerr, Spöhel, and Thomas [DDST16] showed that the case $k = n$ is the last remaining case to be able to get tight asymptotic bounds for all k. For this reason we focus on this case only except for Section 6.5. We first run a randomized algorithm so that we can assume afterwards that the codeword is a permutation which makes the analysis cleaner. We believe it is possible to adapt the algorithm to deal with non-permutation codewords deterministically but we will not prove it here.

6.3.1 Finding a zero vector

For constructing our Mastermind queries below, it turns out to be useful to guess "blank" in certain positions of the string. To achieve this, it suffices to find a query z such that $b(z) = 0$, which can be done as follows.

Lemma 6.3. *Let $k \geq 2$. With $n+1$ many queries we can find a string $z \in [k]^n$ which results in $b_c(z) = 0$.*

Proof. Query the string of all 1s, $t^{(0)}$, as well as the strings $t^{(i)}$ for all $i \in \{1, .., n\}$ where $t^{(i)}$ is the string of all ones except at the ith position it has a 2. Now if $b_c(t^{(i)}) = b_c(t^{(0)}) - 1$, then $c_i \neq 2$, otherwise $c_i \neq 1$ and we have at every position a color that is incorrect, so we have a string z which satisfies $b_c(z) = 0$. □

Note that if $k = n$ and we are content with a randomized search then we can find an all zero string by choosing queries $z \in [n]^n$ uniformly

Algorithm 5 *Permutation*

Output: n pairwise disjoint strings $f^{(1)}, \ldots, f^{(n)}$ such that each string contains exactly one correct position

```
for i ∈ [n] do
    S_i^(1) = [n];
k = 1;
while k ≤ n do
    rand ← Choose a random color for each position i from S_i^(k) for each i ∈ [n];
    if b(rand) == 1 then
        f^(k) ← rand;
        for i ∈ [n] do
            S_i^(k+1) = S_i^(k) \ {rand_i};
        k = k + 1;
return f^(1), ..., f^(n)
```

at random until a query is obtained with $b_c(z) = 0$. As the success probability of one iteration $(1 - 1/n)^n \geq 1/4$, this takes on average at most 4 guesses.

6.3.2 Permutation

The following lemma is due to Angelika Steger (from personal communication).

Lemma 6.4. *The Algorithm* 5 *will, with high probability, need at most* $\mathcal{O}(n)$ *many queries to find* n *different strings such that every correct position is contained in exactly one of the strings.*

Proof. The finding of these strings is done by random queries. Let $S^{(1)} = [n]^n$ start out to be the entire space of possible queries. Sample uniform random queries *rand* from $S^{(1)}$ until one of them gives $b(rand) = 1$. Set this query to be $f^{(1)}$. Now set aside all colors at the corresponding positions and keep querying. That is $S_1^{(2)} = [n]\backslash f_1^{(1)} \times ... \times [n]\backslash f_n^{(1)}$ and sample again random queries *rand* from $S^{(2)}$ until one of them gives $b(rand) = 1$. In this way set aside $f^{(i)}$ which was received by querying randomly from $S^{(i)} = [n]\backslash\{f_1^{(1)}, .., f_1^{(i-1)}\} \times ... \times$

$[n]\backslash\{f_n^{(1)}, .., f_n^{(i-1)}\}$. Let us analyze how many queries this takes. The sets $S^{(i)}$ have $n-i+1$ many possible colors at every position and also $n-i+1$ many positions at which there is still a correct color available. So every position with a still available correct color will have chance of $1/(n-i+1)$ chance of being in *rand* and this independently of every other position. So the probability that $b(q)=1$ for a random query is

$$\Pr[b(rand)=1] = (n-i+1)\cdot\frac{1}{n-i+1}\left(1-\frac{1}{n-i+1}\right)^{n-i} \geq e^{-1} \tag{6.2}$$

Let X_i be the number of queries it takes to find the ith string to set aside. Then X_i is geometrically distributed and the total time is the sum of all X_i, $i \in [n]$, which is an independent sum of geometrically distributed random variables with success probabilities as in (6.2), so expected at most e^{-1}. Applying the Chebychev inequality gives us that, w.h.p. we can find $f^{(1)}$ to $f^{(n)}$ with $\mathcal{O}(n)$ many queries. □

6.3.3 Signed Permutation Mastermind

By treating the strings in Lemma 6.4 as colors, we may assume that the codeword is a permutation of $[n]$, i.e. each color appears exactly once. To simplify the presentation of our main algorithm below, we will in addition modify Mastermind to allow for signed queries, as follows.

Definition 6.5. For a given n, we define *signed permutation Mastermind* as the variation of black-peg Mastermind on n slots and colors where the codeword is a permutation of $[n]$ and where codebreaker is allowed to make *signed queries*. Each signed query consists of a string $q \in \{-n, \ldots, n\}^n$, and we define $b(q) = b_c(q) := |\{i \in [n] | c_i = q_i\}| - |\{i \in [n] | c_i = -q_i\}|$.

We denote by $\mathrm{spmm}(n)$ the minimum number of guesses needed by any deterministic strategy to win signed permutation Mastermind.

Lemma 6.6. *Black-peg Mastermind with n colors and slots can be solved in $\mathcal{O}(n + \mathrm{spmm}(n))$ guesses with high probability.*

Proof. We apply Lemma 6.4 to get n disjoint strings each with one correct position and we can remap the colors to these strings. Then instead of colors in $[n]$ we have colors in $\{f^{(1)}, \ldots, f^{(n)}\}$. And then there is a correct position in every one of the new colors.

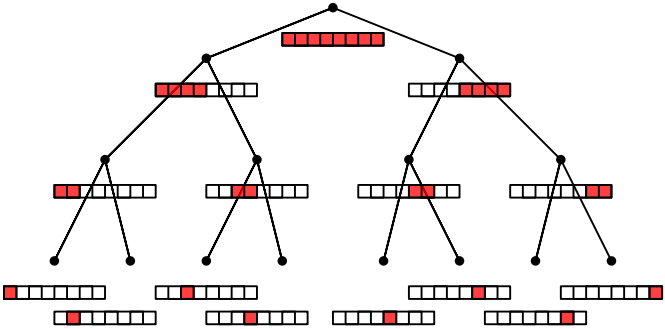

Figure 6.1: Information Tree for $n = 8$. Every node corresponds to an interval, here marked in red.

Then we apply Lemma 6.3 to get an all zero string. Every query in the signed permutation Mastermind we can simulate in the black-peg Mastermind with two queries. For every query q we make in the signed permutation Mastermind we can make a q^+ and a q^- query in the black-peg Mastermind where q^+ is all the positive positions of q and replacing positions which are negative or zero by the corresponding entry at the same position of the zero query we get from Lemma 6.3, and q^+ is in the same way all negative positions. Then $b(q^+) - b(q^-)$ in the black-peg Mastermind is equal to the query answer of $b(q)$ in the signed permutation Mastermind. This transformation can be done in polynomial time and proves the reduction to signed permutation Mastermind. □

6.4 Proof of Theorem 6.1

With the reduction from the previous section at hand, we may assume that each color appears in the hidden codeword exactly once. It remains to show that we can determine the position of every color by using $\mathcal{O}(n)$ signed queries. Before presenting our algorithm, we will first present a token sliding game which will be used to housekeep, at any point while playing signed permutation Mastermind, the information we currently have for each color.

Definition 6.7. Given an instance of signed permutation Mastermind, we define the corresponding information tree T as a rooted complete balanced binary tree of depth $\lceil \log_2 n \rceil$. We denote by n_T the number of

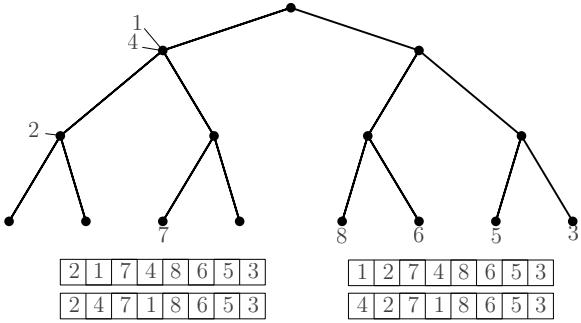

Figure 6.2: Example configuration for $n = 8$. Tokens of colors 1 through 8 are at different positions in the information tree and there are 4 remaining possibilities for the codeword.

leaves of the tree. In other words, n_T is the smallest power of two bigger than or equal to n. For each vertex in T, we associate a (sub-)interval of $[n_T]$ as follows. We order the vertices at depth d in the canonical way from left to right, and associate the jth such vertex with the interval $[(n_T/2^d)(j-1)+1, (n_T/2^d)j]$.

Note that if n is not a power of two, some vertices will be associated to intervals that go outside $[n]$. An example of an information tree for $n = 8$ is illustrated in Figure 6.1.

We introduce handy notation for the complete binary tree T. The root of T is denoted by r. For any vertex we denote by v_L and v_R its left and right child respectively if they exist and if we descend multiple vertices we write v_{LR} for $(v_L)_R$. Further T_L denotes the induced subtree rooted at r_L, similarly T_* is the the induced subtree rooted at $r_\star$ for $\star$ being any combination of R and L such that $r_\star$ exists.

For any instance of signed permutation Mastermind, we perform the following *token game* on the information tree as follows. We initially place n colored tokens at the root r, one for each possible color in our Mastermind instance. At any point, we may take a token at position v and slide it to a either v_L or v_R, if we can prove that the position of its color in the hidden codeword lies in the corresponding sub-interval.

Observation 6.8. *When all tokens are positioned on leaves of T, we know the complete codeword.*

Algorithm 6 `Preprocess(`T`)`

Input: Tree T
Output: Preprocessed tree T
if $n_T \leq 2$ **then**
    Use 1 query to move all tokens to leafs of T;
    return T;
for each token t at r **do**
    Query t and slide token to either r_L or r_R;
for each token t at r_L **do**
    Query t and slide token to either r_{LL} or r_{LR};
Run `Preprocess(`T_{LL}`)`;
Run `Preprocess(`T_{LR}`)`;
return T;

The simplest way to move a token is by performing a query that equals that color on, say, the left half of its current position, and zero everywhere else. We call this step querying a token.

Definition 6.9. For a color f with token at non-leaf node v, we say we query the color f if we make a query of the color f only in the left half of the interval corresponding to v (zero everywhere else).

We note that any query of this form can only give 0 or 1 as output. We will refer to any such queries as *zero-one queries*.

6.4.1 Solving Signed Permutation Mastermind

We are now ready to present our main strategy to solve signed permutation Mastermind by constructing a sequence of entropy dense queries that allows us to slide all tokens on T from the root to their respective leafs. This will be done in two steps, which we call `Preprocess(`T`)` and `Solve(`T`)`.

As intervals corresponding to vertices close to the root of T are so large, there is initially not much room to include many colors in the same query. Thus the idea of the preprocessing step is to perform $\mathcal{O}(n)$ simple queries, in order to move some of the tokens down the tree and thus allow the solve step to query many colors in parallel.

`Preprocess(`T`)`, see Algorithm 6, takes the tree, weighs (Definition 6.9)

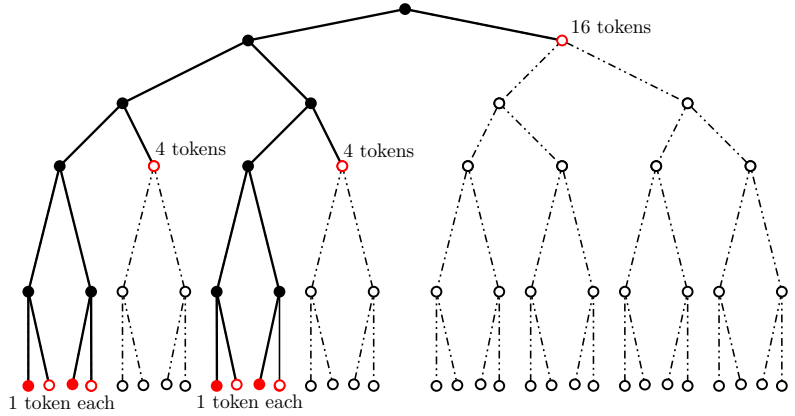

Figure 6.3: Tree preprocessed for n=32, all black vertices have been emptied, tokens are at red vertices.

all colors whose tokens are at the root r and slides the tokens accordingly. Then repeats for the left child of the root r_L . Then we recusivly apply the algorithm to the subtrees of the left two grandchildren of the root T_{LL} and T_{LR}. If the tree has depth 2 or less, we skip all the steps on vertices which do not exist.

Proposition 6.10. *The Algorithm* 6 `Preprocess(T)` *requires at most* $3n_T$ *zero-one queries and runs in polynomial time.*

Proof. Clearly if the depth of the tree is ≤ 2 this holds, as we make only a single zero-one query. Then the rest follows by induction, if we analyze the number of queries needed we see, at the root r we need to query at most n_T tokens, and at the left child r_L we query at most $n_T/2$. In the left grandchildren we recurs. So if $a(n_T)$ is the maximum number of queries we need for `Preprocess(T)` for a tree with n_T leafs, then it holds that

$$a(n_T) \leq n_T + n_T/2 + a(n_{T_{LL}}) + a(n_{T_{LR}}) = n_T + n_T/2 + a(n_T/4) + a(n_T/4)$$

From which follows that $a(n_T) \leq 3n_T$. Since we only query tokens all queries we do are zero-one queries and this can be done in polynomial time concluding the proof. □

The result of running `Preprocess(T)` is illustrated in Figure 6.3. All tokens have either been moved to leafs, or to a vertex of the form

$r_R, r_{*R}, r_{**R}, \ldots$ where each '$*$' denotes either LL or LR. This we call a preprocessed tree.

Once T is preprocessed we can run the second algorithm `Solve(`T`)`, see Algorithm 7. Due to the preproccessing, we know which colors are contained in the leftmost two quarters as well as the rightmost half of the codeword. Thus we can now solve the subproblems formed by T_{LL}, T_{LR}, and T_R independently of each other. Moreover, T_{LL} and T_{LR} are already preprocessed. We now use this observation to construct `Solve(`T`)` recursively. If the depth of the tree is at most 2, we already know the codeword from the preprocessing. For larger trees, we recursively simulate `Solve()` on the subtrees T_{LL} and T_{LR} and also simulate the algorithm `Preprocess(`T_R`)`. Here, simulating a function mean that they are still allowed to move tokens in the information tree, but whenever they try to make a query, the string will instead be pushed to the parent process.

In order to get a speedup from this parallelism, we employ an idea similar to the Cantor-Mills construction (6.1) for coin-weighing.

Lemma 6.11. *Define the* support *of a query q as the set of indices $i \in [n]$ such that $q_i \neq 0$. For any three queries $q^{(1)}, q^{(2)}$, and s with disjoint supports and such that s is a zero-one query, we can determine $b(q^{(1)}), b(q^{(2)})$, and $b(s)$ by making only two queries.*

Proof. We query $w^{(1)} = q^{(1)} + q^{(2)} + s$ and $w^{(2)} = q^{(1)} - q^{(2)}$, where $+$ and $-$ denote element-wise addition subtraction respectively. Then we can retrieve the answers from just the information of $b(w^{(1)})$ and $b(w^{(2)})$. If we combine the queries, $b(w^{(2)}) + b(w^{(2)}) = 2b(q^{(1)}) + b(s)$. So we can retrieve $b(s) = b(w^{(2)}) + b(w^{(2)}) \mod 2$. And then also the queries, $b(q^{(2)}) = (b(w^{(1)}) + b(w^{(2)}) - b(s))/2$ and $b(q^{(2)}) = (b(w^{(1)}) - b(w^{(2)}) - b(s))/2$ are recoverable. □

Proposition 6.12. *Calling Algorithm 7, `Solve()`, for a preprocessed tree will move all the tokens to leaves. Moreover, the algorithm will use at most $6n_T$ queries and runs in polynomial time.*

Proof. The first statement follows by induction. If $n_T \leq 2$ a preprocessed tree already has all its tokens at leaves, so nothing needs to be done. The induction step follows by correctness of Lemma 6.11.

Algorithm 7 `Solve(`T`)`

Input: Preprocessed Tree T
Output: Tree T where all tokens have been queried down to the leaves

1: **if** $n_T \leq 2$ **then**
2: **return** T;
3: Simulate: `Solve(`T_{LL}`)`, `Solve(`T_{LR}`)` and `Preprocess(`T_R`)`;
4: **while** at least one simulated process has not finished **do**
5: Get the next queries $q^{(1)}, q^{(2)}$, and s requested by the processes;
6: Compute the answers using two queries, as in Lemma 6.11, and return to the respective process;
7: `Solve(`T_R`)`;
8: **return** T;

For the runtime analysis, also use induction. If the depth $n_T \leq 2$ then clearly the statement holds. Then if $n_T > 2$ we must simulate `Solve(`T_{LL}`)`, `Solve(`T_{LR}`)` and `Preprocess(`T_R`)` where we only need two queries for every three and we must run `Solve(`T_R`)`. In total we get the following recursion where $c(n_T)$ is the maximum number of queries we must make during `Solve(`T`)` and $a(n_T)$ the maximum number of queries that `Preprocess(`T`)` must make in a tree with n_T leafs.

$$\begin{aligned} c(n_T) &\leq 2 \cdot \max(c(n_{T_{LL}}), c(n_{T_{LR}}), a(n_{T_R})) + c(n_{T_R}) \\ &= 2 \cdot \max(c(n_T/4), a(n_T/2)) + c(n_T/2) \end{aligned}$$

By Proposition 6.10 we know $a(n_T/2) \leq 3n_T/2$ so by induction we get that $c(n_T) \leq 6n_T$. All the operations are clearly in polynomial time so this concludes the proof. □

Now we have all the tools at hand to prove the main Theorem 6.1.

Theorem 6.1. *For n colors and n positions we can solve black-peg Mastermind with $\mathcal{O}(n)$ many queries and in polynomial time with high probability.*

Proof. We have Lemma 6.6 which reduces the problem of solving black-peg Mastermind to solving signed permutation Mastermind. For the new instance of signed permutation Mastermind that results from this we consider the information tree. We move the tokens of this tree to

the leaves with first the algorithm `Preprocess(`T`)` and then `Solve(`T`)` on the preprocessed tree. By Propositions 6.10 and 6.12 this takes at most $9n_T$ signed queries and is done in polynomial time. By Observation 6.8 we have found the hidden codeword of the signed permutation Mastermind, and therefore also the hidden codeword of the black-peg Mastermind game. The transformation can be done in polynomial time and by Lemma 6.6 we need at most $\mathcal{O}(n+9n_T) = \mathcal{O}(n)$ queries to solve black-peg Mastermind. □

6.5 Playing Mastermind with arbitrarily many colors

We briefly make some remarks on other ranges of k and n for Mastermind. By combining Theorem 6.1 with results of Chvátal [Chv83] and Doerr, Doerr, Spöhel, and Thomas [DDST16], we determine up to constant factors the smallest expected number of queries needed to solve mastermind for any n and k.

Theorem 6.2. *For k colors and n positions, the randomized query complexity of Mastermind is*

$$\Theta(n \log k / \log n + k/n),$$

if codebreaker receive both black-peg and white-peg information for each query, and

$$\Theta(n \log k / \log n + k),$$

if codebreaker only receives black-peg information.

For any n and k, let $\mathrm{bmm}(n,k)$ denote the minimum (over all strategies) worst-case (over all codewords) expected number of guesses to solve Mastermind if only black peg information can be used. Similarly, denote $\mathrm{bwmm}(n,k)$ the smallest expected number of queries needed if both black and white peg information can be used. The following relation between was shown by Doerr, Doerr, Spöhel, and Thomas.

Theorem 6.13 (Theorem 4, [DDST16])**.** *For all $k \geq n \geq 1$,*

$$\mathrm{bwmm}(n,k) = \Theta\left(\mathrm{bmm}(n,n) + k/n\right).$$

Proof of Theorem 6.2. For small k, say $k \leq \sqrt{n}$, the result follows by the results of Chvátal [Chv83]. Moreover, for $k \geq n$ the white peg statement follows directly by combining Theorems 6.1 and 6.13. Thus it remains to consider the case of $\sqrt{n} \leq k \leq n$, and the case of $k \geq n$ for black-peg Mastermind.

For $\sqrt{n} \leq k \leq n$, the leading terms in both bounds in Theorem 6.2 is of order n, which matches the entropy lower bound. On the other hand, using Lemma 6.3 we can find a query such that $b(z) = 0$ in $\mathcal{O}(n)$ queries. Having found this, we simply follow the same strategy as for n color black-peg Mastermind by replacing any color $> k$ in a query by the corresponding entry of z. Thus finishing in $\mathcal{O}(n)$ queries.

Finally, for black-peg Mastermind with $k \geq n$, the leading term in Theorem 6.2 is of order k. This can be attained by using k guesses to determine which colors appear in the codeword, and then reduce to the case of n colors. On the other hand, $\Omega(k)$ is clearly necessary as this is the expected number of queries needed to guess the correct color in a single position, provided the codeword is chosen uniformly at random. □

Bibliography

[Abb17] Emmanuel Abbe, *Community detection and stochastic block models: recent developments*, The Journal of Machine Learning Research **18** (2017), no. 1, 6446–6531. 19

[ABKP15] Peter Allen, Julia Böttcher, Yoshiharu Kohayakawa, and Yury Person, *Tight hamilton cycles in random hypergraphs*, Random Structures & Algorithms **46** (2015), no. 3, 446–465. 69

[ABT03] Richard Arratia, Andrew D Barbour, and Simon Tavaré, *Logarithmic combinatorial structures: a probabilistic approach*, vol. 1, European Mathematical Society, 2003. 79

[AFG19] Michael Anastos, Alan Frieze, and Pu Gao, *Hamiltonicity of random graphs in the stochastic block model*, arXiv preprint arXiv:1910.12594 (2019). 19

[AK20] Yahav Alon and Michael Krivelevich, *Finding a Hamilton cycle fast on average using rotations and extensions*, Random Structures & Algorithms **57** (2020), no. 1, 32–46. 69

[AKS85] Miklós Ajtai, János Komlós, and Endre Szemerédi, *First occurrence of Hamilton cycles in random graphs*, North-

Holland Mathematics Studies **115** (1985), no. C, 173–178. 68

[AS04] Noga Alon and Joel H Spencer, *The probabilistic method*, John Wiley & Sons, 2004. 14

[AT92] Richard Arratia and Simon Tavaré, *The cycle structure of random permutations*, The Annals of Probability (1992), 1567–1591. 79

[AV79] Dana Angluin and Leslie G Valiant, *Fast probabilistic algorithms for Hamiltonian circuits and matchings*, Journal of Computer and System Sciences **18** (1979), no. 2, 155–193. 68, 69, 70

[BCLS87] Thang Nguyen Bui, Soma Chaudhuri, Frank Thomson Leighton, and Michael Sipser, *Graph bisection algorithms with good average case behavior*, Combinatorica **7** (1987), no. 2, 171–191. 19

[BE76] Béla Bollobás and Paul Erdős, *On a Ramsey–Turán type problem*, Journal of Combinatorial Theory, Series B **21** (1976), no. 2, 166–168. 7, 58

[BF09] Tom Bohman and Alan Frieze, *Hamilton cycles in 3-out*, Random Structures & Algorithms **35** (2009), no. 4, 393–417. 68

[BFF87] Bela Bollobás, Trevor I. Fenner, and Alan M. Frieze, *An algorithm for finding Hamilton paths and cycles in random graphs*, Combinatorica **7** (1987), no. 4, 327–341. 69

[BFF90] ———, *Hamilton cycles in random graphs with minimal degree at least k*, A tribute to Paul Erdos, edited by A. Baker, B. Bollobás and A. Hajnal (1990), 59–96. 68

[BMMM18] József Balogh, Andrew McDowell, Theodore Molla, and Richard Mycroft, *Triangle-tilings in graphs without large independent sets*, Combinatorics, Probability and Computing **27** (2018), no. 4, 449–474. 41

[BMS16] József Balogh, Theodore Molla, and Maryam Sharifzadeh, *Triangle factors of graphs without large independent sets and of weighted graphs*, Random Structures & Algorithms **49** (2016), no. 4, 669–693. 7, 41, 42

[Bol81] Béla Bollobás, *Random graphs*, Combinatorics (Swansea, 1981), London Math. Soc. Lecture Note Ser., vol. 52, Cambridge Univ. Press, Cambridge-New York, 1981, pp. 80–102. MR 633650 (83e:05039) 5

[Bol88] ———, *The chromatic number of random graphs*, Combinatorica **8** (1988), no. 1, 49–55. 18, 22

[Bol04] ———, *How sharp is the concentration of the chromatic number?*, Combinatorics, Probability & Computing **13** (2004), no. 1, 115. 19

[Bop87] Ravi B Boppana, *Eigenvalues and graph bisection: An average-case analysis*, 28th Annual Symposium on Foundations of Computer Science (sfcs 1987), IEEE, 1987, pp. 280–285. 19

[BR07] Béla Bollobás and Oliver Riordan, *Metrics for sparse graphs*, arXiv preprint arXiv:0708.1919 (2007). 6

[BT97] Béla Bollobás and Andrew Thomason, *Hereditary and monotone properties of graphs*, The Mathematics of Paul Erdös II, Springer, 1997, pp. 70–78. 4

[Bur74] Stefan A Burr, *Generalized ramsey theory for graphs-a survey*, Graphs and combinatorics, Springer, 1974, pp. 52–75. 3

[CCH96] Zhixiang Chen, Carlos Cunha, and Steven Homer, *Finding a hidden code by asking questions*, Computing and Combinatorics (Berlin, Heidelberg) (Jin-Yi Cai and Chak Kuen Wong, eds.), Springer Berlin Heidelberg, 1996, pp. 50–55. 91

[CH63] Keresztély Corradi and András Hajnal, *On the maximal number of independent circuits in a graph*, Acta Mathematica Hungarica **14** (1963), no. 3-4, 423–439. 40

[Chv83] Vasek Chvátal, *Mastermind*, Combinatorica **3** (1983), no. 3-4, 325–329. 9, 90, 91, 103, 104

[CM66] David G Cantor and WH Mills, *Determination of a subset from certain combinatorial properties*, Canadian Journal of Mathematics **18** (1966), 42–48. 91

[CO10] Amin Coja-Oghlan, *Graph partitioning via adaptive spectral techniques*, Combinatorics, Probability & Computing **19** (2010), no. 2, 227–284. 19

[Con09] David Conlon, *A new upper bound for diagonal Ramsey numbers*, Annals of Mathematics (2009), 941–960. 3

[Cra72] C.A. Crane, *Linear lists and priority queues as balanced binary trees*, Computer Science Department, Department of Computer Science, Stanford University., 1972. 88

[DDST16] Benjamin Doerr, Carola Doerr, Reto Spöhel, and Henning Thomas, *Playing mastermind with many colors*, J. ACM **63** (2016), no. 5. 90, 92, 94, 103

[DHM17] Martin Doležal, Jan Hladkỳ, and András Máthé, *Cliques in dense inhomogeneous random graphs*, Random Structures & Algorithms **51** (2017), no. 2, 275–314. 19, 20

[Dir52] Gabriel Andrew Dirac, *Some theorems on abstract graphs*, Proceedings of the London Mathematical Society **3** (1952), no. 1, 69–81. 40

[DJKW16] Vida Dujmović, Gwenaël Joret, Jakub Kozik, and David R Wood, *Nonrepetitive colouring via entropy compression*, Combinatorica **36** (2016), no. 6, 661–686. 8

[DKMZ11] Aurelien Decelle, Florent Krzakala, Cristopher Moore, and Lenka Zdeborova, *Asymptotic analysis of the stochastic block model for modular networks and its algorithmic applications*, Physical review. E, Statistical, nonlinear, and soft matter physics **84** (2011), 066106. 19

[EP13] Louis Esperet and Aline Parreau, *Acyclic edge-coloring using entropy compression*, European Journal of Combinatorics **34** (2013), no. 6, 1019–1027. 8

[ER59] P Erdős and A Rényi, *On random graphs I*, Publicationes Mathematicae Debrecen **6** (1959), 290–297. 1, 3, 18

[ER60] Paul Erdős and Alfréd Rényi, *On the evolution of random graphs*, Publ. Math. Inst. Hung. Acad. Sci **5** (1960), no. 1, 17–60. 18

[ER63] _____, *On two problems of information theory*, Magyar Tud. Akad. Mat. Kutató Int. Közl **8** (1963), 229–243. 90, 91

[Erd47] P. Erdős, *Some remarks on the theory of graphs*, Bull. Amer. Math. Soc. **53** (1947), 292–294. 3

[Erd61] Paul Erdős, *Graph theory and probability II*, Canadian Journal of Mathematics **13** (1961), 346–352. 42

[ES35] Paul Erdős and George Szekeres, *A combinatorial problem in geometry*, Compositio Mathematica **2** (1935), 463–470. 3

[ES65] Paul Erdős and Miklós Simonovits, *A limit theorem in graph theory*, Studia Sci. Math. Hung, Citeseer, 1965. 6

[ES70] Paul Erdős and Vera T Sós, *Some remarks on Ramsey's and Turán's theorem*, Combinatorial theory and its applications, II (Proc. Colloq., Balatonfüred, 1969), 1970, pp. 395–404. 7, 57

[ET36] Paul Erdös and Paul Turán, *On some sequences of integers*, Journal of the London Mathematical Society **1** (1936), no. 4, 261–264. 6

[FF84] Trevor I Fenner and Alan M Frieze, *Hamiltonian cycles in random regular graphs*, Journal of Combinatorial Theory, Series B **37** (1984), no. 2, 103–112. 68

[FJM+96] Alan Frieze, Mark Jerrum, Michael Molloy, Robert Robinson, and Nicholas Wormald, *Generating and counting hamilton cycles in random regular graphs*, Journal of Algorithms **21** (1996), no. 1, 176–198. 69

[FKM08] N Fountoulakis, RJ Kang, and C McDiarmid, *The t-stability number of a random graph*, Electron. J. Combin. 17 (2010),# R59 **17** (2008), no. 1. 18

[FKSV16] Asaf Ferber, Michael Krivelevich, Benny Sudakov, and Pedro Vieira, *Finding hamilton cycles in random graphs with few queries*, Random Structures & Algorithms **49** (2016), no. 4, 635–668. 69

[FL10] Riccardo Focardi and Flaminia L. Luccio, *Cracking bank pins by playing mastermind*, Fun with Algorithms (Berlin, Heidelberg) (Paolo Boldi and Luisa Gargano, eds.), Springer Berlin Heidelberg, 2010, pp. 202–213. 92

[Fri88] Alan M Frieze, *Finding hamilton cycles in sparse random graphs*, Journal of Combinatorial Theory, Series B **44** (1988), no. 2, 230–250. 9, 69

[Fri19] Alan Frieze, *Hamilton cycles in random graphs: a bibliography*, arXiv preprint arXiv:1901.07139 (2019). 5, 68

[Goo09a] M. Goodrich, *The mastermind attack on genomic data*, 2009 30th IEEE Symposium on Security and Privacy, 2009, pp. 204–218. 92

[Goo09b] Michael Goodrich, *On the algorithmic complexity of the mastermind game with black-peg results*, Information Processing Letters **109** (2009), 675–678. 92

[GS87] Yuri Gurevich and Saharon Shelah, *Expected computation time for Hamiltonian path problem*, SIAM Journal on Computing **16** (1987), no. 3, 486–502. 69

[GS05] Stefanie Gerke and Angelika Steger, *The sparse Regularity Lemma and its applications*, Surveys in combinatorics 2005, London Math. Soc. Lecture Note Ser., vol. 327, Cambridge Univ. Press, Cambridge, 2005, pp. 227–258. 6

[Hec18] Annika Heckel, *The chromatic number of dense random graphs*, Random Structures & Algorithms **53** (2018), no. 1, 140–182. 18

[Hec19] ———, *Non-concentration of the chromatic number of a random graph*, arxiv:1906.11808 (2019), Accepted for publication in the Journal of the AMS. 19

[HLL83] Paul W Holland, Kathryn Blackmond Laskey, and Samuel Leinhardt, *Stochastic blockmodels: First steps*, Social networks **5** (1983), no. 2, 109–137. 19

[HS70] András Hajnal and Endre Szemerédi, *Proof of a conjecture of P. Erdős*, Combinatorial theory and its applications **2** (1970), 601–623. 40

[JLR11] Svante Janson, Tomasz Luczak, and Andrzej Rucinski, *Random graphs*, vol. 45, John Wiley & Sons, 2011. 13

[JP11] Gerold Jäger and Marcin Peczarski, *The number of pessimistic guesses in Generalized Black-peg Mastermind*, Inform. Process. Lett. **111** (2011), no. 19, 933–940. MR 2848673 92

[JP19] Zilin Jiang and Nikita Polyanskii, *On the metric dimension of cartesian powers of a graph*, Journal of Combinatorial Theory, Series A **165** (2019), 1–14. 9

[JS98] M. Jerrum and G. B. Sorkin, *The metropolis algorithm for graph bisection*, Discrete Applied Mathematics **82** (1998), no. 1-3, 155–175. 19

[Kim95] Jeong Han Kim, *The Ramsey number R (3, t) has order of magnitude $t^2/\log t$*, Random Structures & Algorithms **7** (1995), no. 3, 173–207. 42

[KK08] Hal A Kierstead and Alexandr V Kostochka, *A short proof of the Hajnal–Szemerédi theorem on equitable colouring*, Combinatorics, Probability and Computing **17** (2008), no. 2, 265–270. 40

[KM13] Peter Keevash and Richard Mycroft, *A multipartite Hajnal-Szemerédi theorem*, The Seventh European Conference on Combinatorics, Graph Theory and Applications, Springer, 2013, pp. 141–146. 40

[Knu98] Donald E Knuth, *The art of computer programming: Volume 3: Sorting and searching*, Addison-Wesley Professional, 1998. 88

[Knu77] Donald E. Knuth, *The computer as master mind*, J. Recreational Math. **9** (1976/77), no. 1, 1–6. MR 434680 92

[Kor76] Aleksei Dmitrievich Korshunov, *Solution of a problem of erdős and renyi on Hamiltonian cycles in nonoriented graphs*, Doklady Akademii Nauk **228** (1976), no. 3, 529–532. 68

[Koy93] K. Koyama, *An optimal mastermind strategy*, Journal of Recreational Mathematics **25** (1993), 251–256. 92

[KS83] János Komlós and Endre Szemerédi, *Limit distribution for the existence of Hamiltonian cycles in a random graph*, Discrete mathematics **43** (1983), no. 1, 55–63. 68

[KS96] János Komlós and Miklós Simonovits, *Szemerédi's Regularity Lemma and its applications in graph theory*, Combinatorics, Paul Erdős is eighty, Vol. 2 (Keszthely, 1993) (1996). 6, 43, 44

[KS21] Charlotte Knierim and Pascal Su, *Kr-factors in graphs with low independence number*, Journal of Combinatorial Theory, Series B **148** (2021), 60–83. 9, 39

[KSSS00] János Komlós, Ali Shokoufandeh, Miklós Simonovits, and Endre Szemerédi, *The regularity lemma and its applications in graph theory*, Summer School on Theoretical Aspects of Computer Science (2000), 84–112. 6

[Lin64] Bernt Lindström, *On a combinatory detection problem I*, I. Magyar Tud. Akad. Mat. Kutató Int. Közl **9** (1964), 195–207. 91

[Lin65] _____, *On a combinatorial problem in number theory*, Canadian Mathematical Bulletin **8** (1965), no. 4, 477–490. 91

[Mat76] David W Matula, *The largest clique size in a random graph*, Department of Computer Science, Southern Methodist University, 1976. 18

[McD89] Colin McDiarmid, *On the method of bounded differences*, Surveys in combinatorics **141** (1989), no. 1, 148–188. 18

[McD90] _____, *On the chromatic number of random graphs*, Random Structures & Algorithms **1** (1990), no. 4, 435–442. 18

[MM02] Csaba Magyar and Ryan R Martin, *Tripartite version of the Corrádi–Hajnal theorem*, Discrete mathematics **254** (2002), no. 1-3, 289–308. 40

[MNS15] E. Mossel, J. Neeman, and A. Sly, *Reconstruction and estimation in the planted partition model*, Probability Theory and Related Fields volume **162** (2015), 431—-461. 19

[MNZ12] Kenneth Maples, Ashkan Nikeghbali, and Dirk Zeindler, *On the number of cycles in a random permutation*, Electron. Commun. Probab. **17** (2012), 13 pp. 79

[Mol19] Michael Molloy, *The list chromatic number of graphs with small clique number*, Journal of Combinatorial Theory, Series B **134** (2019), 264–284. 8

[Mon14] Richard Montgomery, *Embedding bounded degree spanning trees in random graphs*, arXiv preprint arXiv:1405.6559 (2014). 49

[Mon19] ———, *Hamiltonicity in random graphs is born resilient*, Journal of Combinatorial Theory, Series B **139** (2019), 316–341. 68

[Mos09] Robin A. Moser, *A constructive proof of the lovász local lemma*, Proceedings of the Forty-First Annual ACM Symposium on Theory of Computing (New York, NY, USA), STOC '09, Association for Computing Machinery, 2009, p. 343–350. 8

[MPST20] Anders Martinsson, Konstantinos Panagiotou, Pascal Su, and Miloš Trujić, *The chromatic number of dense random block graphs*, arXiv preprint arXiv:2007.07700 (2020). 8, 17

[MS08] Ryan Martin and Endre Szemerédi, *Quadripartite version of the Hajnal–Szemerédi theorem*, Discrete Mathematics **308** (2008), no. 19, 4337–4360. 40

[MS20] Anders Martinsson and Pascal Su, *Mastermind with a linear number of queries*, arXiv preprint arXiv:2011.05921 (2020). 9, 89

[NP18] Rajko Nenadov and Yanitsa Pehova, *On a Ramsey-Turán variant of the Hajnal–Szemerédi theorem*, arXiv preprint arXiv:1806.03530 (2018). 41, 49

[NSS21] Rajko Nenadov, Angelika Steger, and Pascal Su, *An O(N) Time Algorithm for Finding Hamilton Cycles with High Probability*, 12th Innovations in Theoretical Computer Science Conference (ITCS 2021) (Dagstuhl, Germany) (James R. Lee, ed.), Leibniz International Proceed-

ings in Informatics (LIPIcs), vol. 185, Schloss Dagstuhl–Leibniz-Zentrum für Informatik, 2021, pp. 60:1–60:17. 9, 67

[NST19] Rajko Nenadov, Angelika Steger, and Miloš Trujić, *Resilience of perfect matchings and hamiltonicity in random graph processes*, Random Structures & Algorithms **54** (2019), no. 4, 797–819. 68

[NWS02] M. E. J. Newman, D. J. Watts, and S. H. Strogatz, *Random graph models of social networks*, Proceedings of the National Academy of Sciences **99** (2002), no. suppl 1, 2566–2572. 19

[Pfa98] Ben Pfaff, *An introduction to binary search trees and balanced trees*, Libavl Binary Search Tree Library **1** (1998), 19–20. 88

[PPR07] Anna Pagh, Rasmus Pagh, and Milan Ruzic, *Linear probing with constant independence*, Proceedings of the thirty-ninth annual ACM symposium on Theory of computing, 2007, pp. 318–327. 86

[Prö13] Hans Jürgen Prömel, *Ramsey theory for discrete structures*, Springer, 2013. 3

[PS09] Konstantinos Panagiotou and Angelika Steger, *A note on the chromatic number of a dense random graph*, Discrete Mathematics **309** (2009), no. 10, 3420–3423. 18

[PSD00] Jonathan K. Pritchard, Matthew Stephens, and Peter Donnelly, *Inference of population structure using multilocus genotype data*, Genetics **155** (2000), no. 2, 945–959. 19

[Ram30] Frank P. Ramsey, *On a problem of formal logic*, Proceedings of the London Mathematical Society **2** (1930), no. 1, 264–286. 2

[RRS06a] Vojtěch Rödl, Andrzej Ruciński, and Endre Szemerédi, *A Dirac-type theorem for 3-uniform hypergraphs*, Combinatorics, Probability and Computing **15** (2006), no. 1-2, 229–251. 42

[RRS06b] Vojtech Rödl, Andrzej Ruciński, and Endre Szemerédi, *Perfect matchings in uniform hypergraphs with large minimum degree*, European Journal of Combinatorics **27** (2006), no. 8, 1333–1349. 42

[RS11] Ronitt Rubinfeld and Asaf Shapira, *Sublinear time algorithms*, SIAM Journal on Discrete Mathematics **25** (2011), no. 4, 1562–1588. 69

[RSR08] Vojtěch Rödl, Endre Szemerédi, and Andrzej Ruciński, *An approximate Dirac-type theorem for k-uniform hypergraphs*, Combinatorica **28** (2008), no. 2, 229–260. 6, 42

[RW94] Robert W. Robinson and Nicholas C. Wormald, *Almost all regular graphs are hamiltonian*, Random Structures & Algorithms **5** (1994), no. 2, 363–374. 68

[SF60] H. S. Shapiro and N. J. Fine, *E1399*, The American Mathematical Monthly **67** (1960), no. 7, 697–698. 90

[Sha83] Eli Shamir, *How many random edges make a graph Hamiltonian?*, Combinatorica **3** (1983), no. 1, 123–131. 69

[SM00] J. Shi and J. Malik, *Normalized cuts and image segmentation*, IEEE Transactions on Pattern Analysis and Machine Intelligence **22** (2000), no. 8, 888–905. 19

[SV08] Benny Sudakov and Van H Vu, *Local resilience of graphs*, Random Structures & Algorithms **33** (2008), no. 4, 409–433. 68

[SZ06] Jeff Stuckman and Guo-Qiang Zhang, *Mastermind is np-complete*, INFOCOMP Journal of Computer Science **5** (2006). 92

[Sze72] Endre Szemerédi, *On graphs containing no complete subgraph with 4 vertices*, Mat. Lapok **23** (1972), 113–116. 6

[Sze75] ———, *On sets of integers containing k elements in arithmetic progression*, Acta Arithmetica **1** (1975), no. 27, 199–245. 6

[Sze78] Endre Szemerédi, *Regular partitions of graphs*, Problèmes combinatoires et théorie des graphes (Colloq. Internat.

CNRS, Univ. Orsay, Orsay, 1976), Colloq. Internat. CNRS, vol. 260, CNRS, Paris, 1978, pp. 399–401. MR 540024 43

[Tho89] Andrew Thomason, *A simple linear expected time algorithm for finding a hamilton path*, Discrete Mathematics **75** (1989), no. 1-3, 373–379. 69

[Tre16] Andrew Treglown, *A degree sequence Hajnal–Szemerédi theorem*, Journal of Combinatorial Theory, Series B **118** (2016), 13–43. 40, 63

[TZ04] Mikkel Thorup and Yin Zhang, *Tabulation based 4-universal hashing with applications to second moment estimation.*, SODA, vol. 4, 2004, pp. 615–624. 86

[Vig12] Giovanni Viglietta, *Hardness of mastermind*, Proceedings of the 6th International Conference on Fun with Algorithms (Berlin, Heidelberg), FUN'12, Springer-Verlag, 2012, p. 368–378. 92

[Wes01] Douglas B. West, *Introduction to graph theory*, vol. 2, Prentice hall Upper Saddle River, 2001. 12

Curriculum Vitae

Pascal Su

date of birth, 15 August 1990
in Mountain View, USA

2006- 2010	High School at Alte Kantonsschule Aarau, Aarau
2010 - 2016	Bachelor and Master of Sciences (Mathematics) ETH Zürich
2016 - 2021	Ph.D. in Theoretical Computer Science, ETH Zurich