Stochastic Local Search and the Lovasz Local Lemma

Fotios Iliopoulos

Electrical Engineering and Computer Sciences
University of California at Berkeley

Technical Report No. UCB/EECS-2019-125
http://www2.eecs.berkeley.edu/Pubs/TechRpts/2019/EECS-2019-125.html

August 16, 2019

Acknowledgement

I thank Alistair Sinclair and Dimitris Achlioptas for various comments and remarks.

Stochastic Local Search and the Lovász Local Lemma

By

Fotios Iliopoulos

A dissertation submitted in partial satisfaction of the

requirements for the degree of

Doctor of Philosophy

in

Computer Science

in the

Graduate Division

of the

University of California, Berkeley

Committee in charge:

Professor Alistair Sinclair, Chair
Associate Professor Prasad Raghavendra
Assistant Professor Nikhil Srivastava

Summer 2019

Stochastic Local Search and the Lovász Local Lemma

Abstract

Stochastic Local Search and the Lovász Local Lemma

by

Fotios Iliopoulos

Doctor of Philosophy in Computer Science

University of California, Berkeley

Professor Alistair Sinclair, Chair

This thesis studies randomized local search algorithms for finding solutions of constraint satisfaction problems inspired by and extending the Lovász Local Lemma (LLL).

The LLL is a powerful probabilistic tool for establishing the *existence* of objects satisfying certain properties (constraints). As a probability statement it asserts that, given a family of "bad" events, if each bad event is individually not very likely and independent of all but a small number of other bad events, then the probability of avoiding all bad events is strictly positive. In a celebrated breakthrough, Moser and Tardos made the LLL constructive for any product probability measure over explicitly presented variables. Specifically, they proved that whenever the LLL condition holds, their *Resample* algorithm, which repeatedly selects any occurring bad event and resamples all its variables according to the measure, quickly converges to an object with desired properties.

In this dissertation we present a framework that extends the work of Moser and Tardos and can be used to analyze arbitrary, possibly complex, *focused* local search algorithms, i.e., search algorithms whose process for addressing violated constraints, while local, is more sophisticated than obliviously resampling their variables independently of the current configuration. We give several applications of this framework, notably a new vertex coloring algorithm for graphs with sparse vertex neighborhoods that uses a number of colors that matches the algorithmic barrier for random graphs, and polynomial time algorithms for the celebrated (non-constructive) results of Kahn for the Goldberg-Seymour and List-Edge-Coloring Conjectures.

Finally, we introduce a generalization of Kolmogorov's notion of commutative algorithms, cast as matrix commutativity, and show that their output distribution approximates the so-called "LLL-distribution", i.e., the distribution obtained by conditioning on avoiding all bad events. This fact allows us to consider questions such as the number of possible distinct final states and the probability that certain portions of the state space are visited by a local search algorithm, extending existing results for the Moser-Tardos algorithm to commutative algorithms.

Dedicated to my parents and my brother

Contents

Acknowledgements

Throughout my graduate studies, I have been fortunate to be surrounded and supported by mentors, friends and family, and I offer my heartfelt thanks:

To my advisor, Alistair Sinclair. Collaborating with Alistair has been a great privilege and an amazing learning experience. His loyalty to high quality research and teaching, together with his uncompromising ethics, have been a constant source of inspiration to me throughout my studies. I thank him for his generous support and advice, both in research and in life.

To Dimitris Achlioptas, for advising me since my years as an undergraduate student and throughout my PhD, and for being there for me as a mentor and as a friend. I am indebted to him for all the things he has taught me, both about research and life, and the countless hours he has spent working and discussing with me. I admire his relentless enthusiasm for science and his great taste for it. Indeed, Dimitris was the one who introduced me to the Probabilistic Method and its connections to local search algorithms, which is the topic of this thesis.

To my thesis and quals committee members, Prasad Raghavendra, Satish Rao, and Nikhil Srivastava. I thank them for serving on my committee, and for their helpful comments at various stages of this work.

To my collaborators and co-authors, Dimitris Achlioptas, Themis Gouleakis, Vladimir Kolmogorov, Alistair Sinclair and Nikos Vlassis. Also to David Harris, Nick Harvey, Mike Molloy and Jan Vondrák for many insightful and inspiring conversations. Without them, this thesis would not be possible.

To Dimitris Fotakis for introducing me to theoretical computer science during my undergraduate studies. He is a wonderful teacher, and I thank him for his support and advice.

To all the theory graduate students in Berkeley for creating an amazing environment. I especially thank Antonio Blanca, Jonah Brown Cohen, Ben Caulfield, Jingcheng Liu, Pasin Manurangsi, Peihan Miao, Alex Psomas, Aviad Rubinstein, Manuel Sabin, Aaron Schild, Tselil Schramm, and Sam Wong for their friendship and the many funny moments we shared together.

To Christos Adamopoulos, Vrettos Moulos, George Patsakis and Panagiotis Zarkos for being my family in Berkeley.

To Paris Syminelakis for the many insightful conversations, and for being a great friend.

To Kyriakos Axiotis and Dimitris Tsipras for their friendship and the amazing time we had together during their visit to Berkeley.

To Eleni Dimitropoulou for the radiant warmth and positive attitude.

To the Simons Institute for giving me the chance to meet some of the greatest researchers in our community.

To Caffe Strada for the many hours I have spent there studying and working on my thesis.

To the Onassis Foundation, for financially supporting my graduate studies.

And to my mother Christine, my father Dimos, and my brother Kostas for their endless love and support during my PhD.

Chapter 1

Introduction

Constraint satisfaction problems over discrete variables, such as satisfiability and graph coloring, arise naturally in a wide variety of areas, among them artificial intelligence [32], bioinformatics [89], combinatorics [11], complexity theory [30], communication theory [41], and statistical image processing [43]. Solving them is an NP-hard task in general and, therefore, we cannot expect of a universal algorithm for it. Instead, researchers throughout the years have developed an arsenal of successful heuristics [19, 72, 75] for tackling them in practice.

A large class of algorithms for solving constraint satisfaction problems employ "stochastic local search"; such algorithms start in a violating configuration and try to reach a satisfying assignment via small randomized changes that in each step focus on satisfying a specific violated constraint (while potentially introducing others). Given their great practical success, it is natural to ask whether there are conditions under which stochastic local search algorithms provably work efficiently, and use these conditions to show that interesting families of instances of hard problems are in fact tractable.

This thesis concerns the design and analysis of such algorithms, using techniques that are inspired by the *Lovász Local Lemma* [36], a powerful and widely applied tool of the Probabilistic Method that has had far-reaching consequences in computer science and combinatorics (see, e.g., [11, 79] for examples).

Besides presenting a compilation of results and applications developed by the author and collaborators [2, 3, 4, 5, 6, 59, 60], the purpose of this document is to serve as an overview of the state of the art in the area. Our hope is that it will be useful to researchers who want to understand, or further contribute to, this exciting line of work. To that end, the material is neither presented in historical order nor is it exhaustive.

1.1 The Probabilistic Method and the Lovász Local Lemma

Numerous problems in computer science and combinatorics can be formulated as searching for objects lacking certain bad properties, or "flaws". For example, constraint satisfaction problems can be seen as searching for objects (satisfying assignments) that are *flawless*, in the sense that they do not violate any constraint.

We can often prove the *existence* of flawless objects via the Probabilistic Method, i.e., by showing that a random object chosen from an appropriate probability distribution has the desired

properties with positive probability. The Probabilistic Method was pioneered by Paul Erdős, who applied it to many problems in combinatorics and number theory.

In classical applications of the Probabilistic Method, the result obtained was not just that the random object had the desired properties with positive probability, but rather that an overwhelming proportion of the random objects did. Indeed, early on the technique applied only to scenarios where the probability that the random object is flawless was fairly large. (Note that in these restrictive settings we can find an object with the desired properties by simply picking one at random.)

In contrast, the Lovász Local Lemma (LLL) [36], which is one of the most important and powerful tools of the Probabilistic Method, allows one to prove the existence of flawless objects even in situations where the probability of avoiding every bad property is exponentially small. Roughly speaking, it asserts that, given a collection of bad events in a probability space, if each of them is individually not too likely, and independent of most other bad events, then the probability that none of them occurs is strictly positive. For example, the LLL implies that every k-CNF formula in which each clause shares variables with fewer than $2^k/e$ other clauses is satisfiable.

The LLL has a short proof and is easy to apply. It has found applications in various field of combinatorics and computer science, from Ramsey theory [11], to designing error-correcting codes [85] and graph sparsification [14, 40]. The LLL has also had a large impact on graph theory [79], as it is the basic ingredient of the so-called "semi-random method", which is the technique behind some of the strongest results in graph coloring. These include results by Kim, Johansson, Molloy and others on coloring locally sparse graphs [10, 62, 63, 68, 76, 103], Kahn's results on list-edge coloring linear hypergraphs [64], Kahn's proof that asymptotically the Goldberg-Seymour and List-Edge-Coloring conjectures hold [65, 66], and the result of Molloy and Reed [77] on almost optimal total colorings of graphs.

In this thesis we will demonstrate several new applications of the LLL and also discuss its various versions and generalizations.

1.2 Connection to Local Search and Contributions

As one can imagine, after proving via the LLL that objects with desired properties exist, it is natural to ask whether they can actually be found efficiently. However, since the number of such objects is typically exponentially small, *a priori* it seems difficult to construct efficient algorithms to find them. Indeed, making the LLL constructive has been a long quest, starting with the work of Beck [15], with subsequent important developments by Alon [9], Molloy and Reed [78], Czumaj and Scheideler [31], Srinivasan [101] and others. Each such work established a method for finding flawless objects efficiently, but in all cases under significant additional conditions beyond the LLL. The breakthrough was made only in 2009 by Moser [80], who showed that a shockingly simple local search algorithm almost matches the LLL condition for k-CNF formulas. Very shortly afterwards, Moser and Tardos in a landmark paper [81] made the general LLL constructive for all product measures over explicitly presented variables. Specifically, they proved that whenever the LLL condition holds, their *Resample* algorithm, which repeatedly selects any bad event in the current assignment and resamples all its variables according to the measure, quickly converges to a desired object.

Naturally, a plethora of papers has employed and further developed the techniques of Moser and Tardos to design and analyze local search algorithms in a wide variety of settings. For example,

the *entropy compression* method introduced by Moser for the analysis of his satisfiability algorithm has drawn a lot of attention and has been applied to a variety of problems, e.g., [3, 38, 47], without any explicit reference to the LLL. Furthermore, the *backward-looking* analysis of [81] has been used [70, 88] to prove that the Moser-Tardos algorithm in fact converges in polynomial time under the most powerful LLL conditions known, while the so-called *forward-looking* analysis has been employed [4, 57] to make constructive applications of the LLL in non-product probability spaces. One of the goals of this thesis is to provide an introduction to these LLL-inspired algorithmic techniques, so that researchers can study and use them.

The main new contribution of this thesis is a framework that extends the work of Moser and Tardos and can be used to analyze arbitrary, possibly quite complex, *focused* local search algorithms, i.e., search algorithms whose process for addressing violated constraints, while local, is more sophisticated than obliviously resampling their variables independently of the current configuration. Moreover, it can be used to capture applications of the LLL in non-product probability spaces, e.g., random permutations, random matchings of a clique, hard-core distributions etc.

This framework is based on work of the author in collaboration with other researchers, and we note that some of the results presented here have been published elsewhere [3, 4, 5, 59, 60]. Among other things, we will employ our framework to establish concrete connections between local search algorithms and the LLL, to study several new algorithms, and to give several new applications. For instance, we use it to design polynomial time algorithms for the celebrated results of Kahn [65, 66], which use probabilistic arguments to show that the chromatic and list-chromatic index of multigraphs are asymptotically equal to their fractional chromatic index, verifying the famous Goldberg-Seymour and List-Edge-Coloring conjectures asymptotically.

From a technical standpoint, the new insight of this thesis is that LLL-inspired convergence arguments can be seen as method for bounding the spectral radius of a matrix specifying the algorithm to be analyzed. Armed with this viewpoint, we establish a new condition for the convergence of local search algorithms, which captures and unifies the most general results regarding the algorithmic LLL [3, 4, 57], the classical potential function argument and a plethora of other papers based on the entropy compression method.

To get a feeling for the main idea behind our analysis, imagine we wish to analyze the performance of a Markovian algorithm $\mathcal{M}$ for solving a CSP instance. Let Ω be the set of all possible variable assignments, e.g., $\{0,1\}^n$, and let $\Omega^* \subseteq \Omega$ be the set of violating assignments. Let A be the $|\Omega^*| \times |\Omega^*|$ submatrix of $\mathcal{M}$ corresponding to transitions that stay within Ω^*. Our starting observation is that the algorithm will escape Ω^* if and only if the spectral radius of A is strictly less than 1. Of course, since A is huge and defined implicitly by the algorithm, its spectral radius, $\rho(A)$, is not readily available.

To sidestep the inaccessibility of the spectral radius, we bound an operator norm $\|\cdot\|$ of A, possibly after performing a change of basis. That is, we exploit the facts that $\rho(A) = \rho(MAM^{-1})$ for any invertible matrix M and that every operator norm upper bounds the spectral radius. Our next observation is that different norms capture different convergence arguments. For example:

- Let ϕ be a non-negative function on Ω that takes values 0 on satisfying assignments. The *potential function* argument asserts that eventually $\phi = 0$, i.e., the particle escapes Ω^*, if ϕ is always reduced (in expectation) under the action of the algorithm. It is not hard to verify that this is equivalent to requiring that $\|MAM^{-1}\|_\infty < 1$, where M is the diagonal matrix $\mathrm{diag}(1/\phi(\sigma))$.

- What is arguably far less obvious is that LLL-inspired convergence arguments bound $\|MAM^{-1}\|_1$, where $M = \mathrm{diag}(\mu(\sigma))$ and μ is the underlying probability measure on Ω. Thus, the LLL can be viewed as the "dual" of the potential function argument (in the sense that $\|\cdot\|_1$ is the dual norm of $\|\cdot\|_\infty$).

As a demonstration of its power, we use this new condition to analyze a sophisticated local search algorithm for the classical problem of coloring graphs with sparse neighborhoods. Specifically, we give a constructive bound on the chromatic number of graphs as a function of the maximum degree and the number of triangles a vertex can be a part of. In addition, we prove that any improvement over our algorithm would require a major breakthrough in random graph theory, since it coincides with a phase transition in the geometry of the set of colorings known as the shattering threshold [1].

Finally, it is often useful to obtain more detailed information about the execution of a stochastic local search algorithm, such as the number of possible distinct outputs, or the probability that certain portions of the state space are visited. This information is available for the Moser-Tardos algorithm due to Haeupler, Saha and Srinivasan [49] and Harris and Srinivasan [56]. Here we introduce a generalization of Kolmogorov's [71] notion of commutative algorithms, cast as matrix commutativity, and show that their output distribution approximates the so-called "LLL-distribution", i.e., the distribution obtained by conditioning on avoiding all bad events. This fact allows us to extend already existing results regarding distributional properties of the Moser-Tardos algorithms to commutative algorithms, and to give several applications.

1.3 Organization of the Thesis

In Chapter 2 we formally present the non-constructive statement of the Lovász Local Lemma and its various versions and generalizations. In Chapter 3 we introduce our framework for the design and analysis of stochastic local search algorithms, and we use it to present the three main techniques that have been used in the algorithmic LLL literature to date, explaining the advantages and limitations of each of them. Among other things, we state and prove the general algorithmic LLL condition of [4], and use it to draw new connections between arbitrary focused local search algorithms and the LLL. In Chapter 4 we introduce a linear algebraic perspective into the analysis of LLL-inspired algorithms, and a new algorithmic LLL condition inspired by it. Chapter 5 concerns the class of commutative algorithms, for which one can guarantee distributional properties besides fast convergence. In Chapter 6 we apply our framework to design and analyze algorithms for the problem of coloring locally sparse graphs. Finally, in Chapter 7 we give polynomial time algorithms for the results of Kahn [65, 66] regarding the Goldberg-Seymour and List-Edge-Coloring conjectures.

Chapter 2

The Lovász Local Lemma

In this chapter we formally present the non-constructive statement of the Lovász Local Lemma and its various versions and generalizations.

2.1 The Lovász Local Lemma

Let Ω be a finite set and let $F = \{f_1, f_2, \ldots, f_m\}$ be a collection of subsets of Ω, each of which will be referred to as a "flaw." Let $\bigcup_{i\in[m]} f_i = \Omega^*$. For example, for a given CNF formula on n variables with clauses $c_1, \ldots, c_m$, we take $\Omega = \{0,1\}^n$ to be the set of all possible variable assignments, and f_i the set of assignments that fail to satisfy clause c_i. Our goal is to find an assignment in $\Omega \setminus \Omega^*$, i.e., a satisfying ("flawless") assignment.

Given a set of objects Ω and a family of flaws we can often prove the *existence* of flawless objects using the Probabilistic Method. Indeed, in many interesting cases this is the only way we know how to do so. To employ the Probabilistic Method we introduce a probability measure on the set of objects and consider the collection of ("bad") events $\mathcal{B}$ corresponding to the flaws (one event per flaw). The existence of flawless objects is then equivalent to the intersection of the complements of the bad events having strictly positive probability. Clearly, such positivity always holds if the events in $\mathcal{B}$ are independent and none of them has measure 1. One of the most powerful tools of the Probabilistic Method is the Lovász Local Lemma, which asserts that such positivity also holds under a condition of limited dependence among the events in $\mathcal{B}$. The idea of the Local Lemma was first circulated by Lovász in the early 1970s in an unpublished note. It was published by Erdős and Lovász in [36]. Below we state it in its simplest form.

Lovász Local Lemma. *Let (Ω, μ) be a probability space and let $\mathcal{B} = \{B_1, \ldots, B_m\}$ be a set of (bad) events such that for each i:*

(a) $\mu(B_i) \leq p < 1$; and

(b) B_i is mutually independent of all but d events.

If $\mathrm{e}p(d+1) \leq 1$ then with positive probability, none of the events in $\mathcal{B}$ occur.

As a first example, we apply the the LLL to a satisfiability problem.

Theorem 2.1. *Every k-CNF formula in which each clause shares variables with at most $2^k/e - 1$ other clauses is satisfiable.*

Proof. Let $\Omega = \{0,1\}^n$, where n is the number of variables of the formula. Let μ be the probability distribution over Ω induced by setting each variable 0 or 1, independently, and uniformly at random. For each clause c we define B_c to be the (bad) event that c is violated. Let also D_c denote the set of clauses with which c shares variables.

Clearly, each event B_c is independent of the set of events $\{B_{c'} : c' \notin D_c\}$. Moreover, $\mu(B_c) \leq 2^{-k}$ for every clause c and, by assumption, $d := \max_c |D_c| \leq 2^k/e - 1$. Thus, applying the Lovász Local Lemma to this family of bad events concludes the proof.

□

The Local Lemma allows us to reach global conclusions using a local, "scale-free" analysis. From this point of view, a drawback of the so-called "symmetric" version presented above is that it requires *global* bounds for the probability of bad events and the dependencies between them. In cases where the probabilities of bad events vary widely, this can be problematic. The general form below circumvents this issue. It is also due in unpublished form to Lovász and was given by Spencer in [100]. Note that, throughout, we use the convention that a product of devoid factors equals 1, i.e., $\prod_{x\in\emptyset} f(x) = 1$.

General LLL. *Let (Ω, μ) be a probability space and let $\mathcal{B} = \{B_1, B_2, \ldots, B_m\}$ be a set of (bad) events. For each $i \in [m]$ let $D(i) \subseteq ([m] \setminus \{i\})$ be such that $\mu(B_i \mid \cap_{j\in S}\overline{B_j}) = \mu(B_i)$ for every $S \subseteq [m] \setminus (D(i) \cup \{i\})$. If there exist positive real numbers $\{\psi_i\}_{i=1}^m$ such that for all $i \in [m]$,*

$$\mu(B_i) \sum_{S\subseteq D(i)\cup\{i\}} \prod_{j\in S} \psi_j \leq 1 \ , \tag{2.1}$$

then the probability that none of the events in $\mathcal{B}$ occurs is at least $\prod_{i=1}^m 1/(1+\psi_i) > 0$.

The directed graph on $[m]$ where each vertex i points to the vertices of $D(i)$ is known as the *dependency* graph.

Remark 2.2. *Condition* (2.1) *above is equivalent to $\mu(B_i) \leq x_i \prod_{j\in D(i)}(1-x_j)$, where $x_i = \psi_i/(1+\psi_i)$, which often appears in the literature. As we will see the formulation* (2.1) *facilitates comparisons. To see the equivalence, notice that since $x_i = 0$ is uninteresting, we may assume $x_i \in (0,1)$. Taking $\psi_i > 0$, setting $x_i = \psi_i/(1+\psi_i) \in (0,1)$, and simplifying, the condition becomes $\mu(B_i)\prod_{j\in\{i\}\cup D(i)}(1+\psi_j) \leq \psi_i$. Opening up the product yields* (2.1).

While, at first sight, condition (2.1) may seem unwieldy, we note that in most applications setting $\psi_i = \mu(B_i)\psi$ and optimizing over $\psi > 0$ (which is typically a straightforward task) suffices to give the best result. Moreover, there exist several weaker, but more user-friendly conditions, in which the parameters ψ_i do not appear. For example, the symmetric version we presented originally is obtained from (2.1) by setting $\psi_i = \frac{1}{d}$ for every i. The reader is referred to [79] for more examples and details.

2.2 The Lopsided Local Lemma

In [37], Erdős and Spencer noted that one can replace the LLL's requirement that each bad event is independent of all but a few other bad events with the weaker requirement that each bad event is *negatively correlated* with few other bad events, yielding the original version of what is known as the *lospided* LLL. That is, for each bad event B_i there should only be few other bad events whose non-occurrence may boost B_i's probability of occurring; the non-occurrence of any subset of the remaining events should leave B_i either unaffected, or make it less likely. A natural setting for the lopsided LLL arises when one seeks a collection of permutations satisfying a set of constraints and considers the uniform measure on them. While the bad events (constraint violations) are now typically densely dependent (as fixing the image of even just one element affects all others), one can often establish sufficient negative correlation among the bad events to apply the lopsided LLL.

More sophisticated versions of the lopsided LLL have also been established in [8, 33]. Below we state the strongest form of the lopsided LLL that holds for arbitrary probability spaces and families of bad events (see e.g., [79, p.228]).

Lopsided LLL. *Let (Ω, μ) be a probability space and let $\mathcal{B} = \{B_1, B_2, \ldots, B_m\}$ be a set of (bad) events. For each $i \in [m]$, let $L(i) \subseteq [m] \setminus \{i\}$ be such that $\mu(B_i \mid \bigcap_{j \in S} \overline{B_j}) \leq b_i$ for every $S \subseteq [m] \setminus (L(i) \cup \{i\})$. If there exist positive real numbers $\{\psi_i\}_{i=1}^m$ such that for all $i \in [m]$,*

$$\frac{b_i}{\psi_i} \sum_{S \subseteq L(i) \cup \{i\}} \prod_{j \in S} \psi_j \leq 1 \ , \tag{2.2}$$

then the probability that none of the events in $\mathcal{B}$ occurs is at least $\prod_{i=1}^m 1/(1+\psi_i) > 0$.

The directed graph on $[m]$ where each vertex i points to the vertices of $L(i)$ is known as the *lopsidependency* graph.

The above form of the LLL is motivated by the fact that, in complex applications, small but non-vanishing correlations tend to travel arbitrarily far in the space Ω. To isolate these dependencies so that they can be treated locally, it can be crucial to allow mild negative correlations between each bad event B_i and the events outside its "special" set $L(i)$, achieved by allowing $b_i \geq \mu(B_i)$.

2.3 Extensions

While the lopsided LLL condition (2.2) is the weakest known condition in the general setting, it can be improved under additional assumptions. We briefly discuss some of these improvements in this section.

Perhaps the two most popular improvements come from considering the *undirected* graph, G, on $[m]$ such that $L(i)$ is a subset of the neighbors of i, for every $i \in [m]$. (One can trivially get such a G by ignoring the direction of eges in the lopsidependency graph, but at the cost of potentially expanding the "neighborhood" of each vertex.)

The first improvement, which is called the *cluster expansion* condition, was introduced by Bissacot et al. [21] who showed that the conclusion of the lopsided LLL remains valid if the summation in (2.2) is restricted to those sets $S \subseteq \{i\} \cup L(i)$ which are *independent* in G. In particular, letting $\mathrm{Ind}(S) = \mathrm{Ind}_G(S)$ denote the set of independent subsets of S with respect to G, we have the following condition.

Cluster Expansion Condition. *Let (Ω, μ) be a probability space and let $\mathcal{B} = \{B_1, B_2, \ldots, B_m\}$ be a set of (bad) events. For each $i \in [m]$, let $L(i) \subseteq [m] \setminus \{i\}$ be such that $\mu(B_i \mid \bigcap_{j \in S} \overline{B_j}) \leq b_i$ for every $S \subseteq [m] \setminus (L(i) \cup \{i\})$. If there exist positive real numbers $\{\psi_i\}_{i=1}^m$ such that for all $i \in [m]$,*

$$\frac{b_i}{\psi_i} \sum_{S \in \mathrm{Ind}(L(i) \cup \{i\})} \prod_{j \in S} \psi_j \leq 1 \ , \tag{2.3}$$

then the probability that none of the events in $\mathcal{B}$ occurs is at least $\prod_{i=1}^m 1/(1+\psi_i) > 0$.

The second improvement, which was introduced by Shearer [98], is based on certain forms of the multivariate independence polynomial and exploits *global* information about G. In fact, it gives a *tight* criterion under which all events can be avoided for a given graph G and, consequently, it subsumes the cluster expansion condition. Unlike the cluster expansion condition, though, Shearer's condition involves a separate condition for every independent set in G and, therefore, verifying it is typically an intractable task.

Shearer's Condition. *Let (Ω, μ) be a probability space and let $\mathcal{B} = \{B_1, B_2, \ldots, B_m\}$ be a set of m (bad) events. For each $i \in [m]$, let $L(i) \subseteq [m] \setminus \{i\}$ be such that $\mu(B_i \mid \bigcap_{j \in S} \overline{B_j}) \leq b_i$ for every $S \subseteq [m] \setminus (L(i) \cup \{i\})$. Let $b \in \mathbb{R}^m$ be the real vector such that $b(i) = b_i$. Furthermore, for $S \subseteq [m]$ define $b^S = \prod_{j \in S} b_j$ and the polynomial q^S by:*

$$q^S(b) := \sum_{\substack{I \in \mathrm{Ind}([m]) \\ S \subseteq I}} (-1)^{|I \setminus S|} b^I \ . \tag{2.4}$$

If $q^S(b) \geq 0$ for all $S \subseteq [m]$, then the probability that none of the events in $\mathcal{B}$ occurs is at least $q^{\emptyset}(b)$.

Kolipaka, Szegedy and Xu [69] have developed other intermediate LLL versions that, again given G, interpolate between (2.2) and Shearer's condition. Moreover, it is known that when μ is a product measure and the dependencies between events can be expressed in terms of variable sharing (that is, under the assumptions of the variable setting [81]), several works [70, 52, 50, 58] have shown that Shearer's condition can be improved, i.e., that more permissive conditions exist.

Pegden [87] proved the so-called *left-handed*-version of the LLL, which is able to reduce the number of edges in a (directed) lopsidependency graph when there is an ordering underlying the significant dependencies of events.

Finally, Scott and Sokal [95], among many other contributions, introduced a so-called *soft-core* LLL condition, in an effort to quantify interaction strengths between bad events (whereas in all other works two bad events either interact or they do not). Since it relies on relatively involved assumptions, we will not describe it here in the interest of brevity. We note though that finding combinatorial applications for that condition was left as an open question in [95]. To the best of our knowledge, it remains open.

Chapter 3

Algorithmic Techniques

In this chapter we develop tools for the design and analysis of stochastic local search algorithms for finding flawless objects in a state space Ω. The general idea in stochastic local search is that Ω is equipped with a neighborhood structure and that the search starts at some state of Ω and moves from state to state within the neighborhood structure. *Focused* local search corresponds to the case where each state change can be attributed to an effort to rid the state of some specific present flaw.

We start by introducing a general algorithmic framework for focused local search, and then we present the three main techniques that have been used for the analysis of LLL-inspired algorithms: the *entropy-compression* method, *backward-looking* analysis, and *forward-looking* analysis. We explain the advantages and limitations of each of them, as well as their connection to the LLL.

It is worth noting that, using the tools presented in this chapter, one can give an efficient algorithm for effectively all the existential applications of the LLL we are aware of.

3.1 Algorithmic Framework

Recall that Ω is a finite set, $F = \{f_1, f_2, \ldots, f_m\}$ is a collection of subsets of Ω which we call "flaws", and that $\bigcup_{i\in[m]} f_i = \Omega^*$. Since we will be interested in algorithms which traverse the set Ω we will also refer to its elements as "states".

For a state σ, we denote by $U(\sigma) = \{j \in [m] : f_j \ni \sigma\}$ the set of (indices of) flaws present in σ. (Here and elsewhere, we shall blur the distinction between flaws and their indices.) We consider algorithms which start in a state sampled from a probability distribution θ and, in each flawed state $\sigma \in \Omega^*$, choose a flaw $f_i \ni \sigma$, and attempt to leave ("fix") f_i by randomly moving to a neighboring state. We will assume that, for every flaw f_i and every state $\sigma \in f_i$, there is a non-empty set of *actions* $\alpha(i, \sigma) \subseteq \Omega$ such that *addressing* flaw f_i in state σ amounts to selecting the next state τ from $\alpha(i, \sigma)$ according to some probability distribution $\rho_i(\sigma, \tau)$. Note that potentially $\alpha(i, \sigma) \cap f_i \neq \emptyset$, i.e., addressing a flaw does not necessarily imply removing it. We sometimes write $\sigma \xrightarrow{i} \tau$ to denote the fact that the algorithm addresses flaw f_i at σ and moves to τ. Finally, we say that a transition $\sigma \xrightarrow{i} \tau$ *introduces* flaw $f_j \ni \tau$ if $\sigma \notin f_j$ or if $j = i$. (Thus, a flaw (re)introduces itself when a transition fails to address it.)

Definition 3.1 (Causality)**.** *We say that* f_i causes f_j *if there exists a transition* $\sigma \xrightarrow{i} \tau$ *that introduces* f_j*.*

Definition 3.2 (Causality Digraph)**.** *Any digraph $C = C(\Omega, F)$ on $[m]$ that includes every edge $i \to j$ such that f_i causes f_j is called a* causality digraph*. We write $\Gamma(i)$ for the set of out-neighbors of i in this graph.*

The framework described above was introduced in [3]. Throughout this thesis, we will use it to prove convergence results for local search algorithms, to establish connections between them and the LLL, and to give several concrete applications.

As a final remark, we note that all our sufficient conditions for convergence will bound the number of steps of the input algorithm, and will not consider the running time to perform each step. (In most applications though, bounding the latter is a trivial task.)

3.2 The Entropy Compression Method

The entropy compression method is an elegant technique introduced by Moser [80] for the analysis of his original algorithm for generating solutions to k-SAT instances that satisfy the LLL conditions. As was later shown by Achlioptas and Iliopoulos [3], it can also be used to make constructive applications of the lopsided LLL in arbitrary spaces endowed with the uniform measure, but it does not seem to generalize to non-uniform probability spaces.

Besides applications of the LLL this method has also been used to analyze *backtracking* algorithms, e.g., [34, 47, 34, 38, 45, 46, 84]. These natural and potentially powerful local search algorithms operate on *partial non-violating* assignments, starting with the empty assignment, and try to extend it to a complete, satisfying assignment. To do this, at each step they assign a (random) value to a currently unassigned variable; if this leads to the violation of one or more constraints, they backtrack to a partial non-violating assignment by unassigning some set of variables (typically including the last assigned variable). While the spirit of the analysis in all these works is close to that of Moser, they are not derived from a known form of the LLL and indeed often improve on earlier results obtained from the LLL. Instead, the fact that the algorithm under consideration reaches a flawless object is established in each case by a problem-specific counting argument. In Chapter 4 we will see several examples of backtracking algorithms and we will study them systematically via a different approach.

Here, we first introduce the entropy compression method by analyzing Moser's satisfiability algorithm. Then, we extend the analysis to prove a simple sufficient algorithmic LLL condition that applies to general local search algorithms which address flaws by picking actions uniformly at random at each step. (This is a corollary of the main result of [3].) As an example, we apply this condition to give an algorithm for an application of the lopsided LLL to a problem in the space of permutations of $[n]$.

3.2.1 An Algorithm for Satisfiability

The basic idea behind the entropy compression method is that a string of random bits cannot be represented by a shorter string. Moser's ingenious observation was that, quite surprisingly, we can make use of this fact to establish algorithmic convergence.

At a high level, his argument proceeds as follows. Suppose that we wish to prove that a certain algorithm for finding flawless objects eventually reaches one. To do that, we start by giving a

lower bound, $r(t)$, on the number of random bits the algorithm requires in order to perform t steps. Then, towards a contradiction, we show that the assumption that the algorithm has not encountered a flawless object within the first t steps allows us to construct a string, L_t, of length at most $\ell(t)$, such that given access to it we can perfectly reconstruct the random bits used by the algorithm. Finally, we show that there exists $t_0 > 0$ such that for every $t \geq t_0$ we have $r(t) > l(t)$, i.e., L_t *losslessly compresses* the random string used by the algorithm. Since this is impossible, we have established that the algorithm has to terminate with high probability within a bounded number of steps.

To see this idea in action, consider the following algorithm for satisfiability that takes as input a k-CNF formula $\phi = \{c_1, \ldots, c_m\}$ on n variables, and an arbitrary initial assignment $\sigma_1 \in \{0,1\}^n$. Its main operation is the stochastic procedure RESAMPLE which takes as input a clause c and a state $\sigma \in \{0,1\}^n$ and outputs a new state σ' by picking a random assignment for the variables of c. For a clause c, let $N(c)$ denote the set of clauses that share variables with c.

Algorithm 1 Recursive Algorithm for k-SAT

```
1:  procedure SOLVE(φ, σ_1)
2:      σ ← σ_1
3:      while there exists a violated clause in σ do
4:          Let c be the lowest indexed violated clause
5:          σ ← FIX (c, σ)
6:  procedure FIX(c, σ)
7:      σ' ← RESAMPLE(c, σ)
8:      while there exists a violated clause of N(c) ∪ {c} in σ' do
9:          Let c' be the lowest indexed violated clause in N(c) ∪ {c}
10:         σ' ←FIX(c', σ')
```

It is a straightforward exercise to establish that if the above algorithm terminates, then it outputs a satisfying assignment for ϕ. The main idea is that to prove inductively that if on input (c, σ) procedure FIX returns assignment σ', then every clause that was satisfied in σ is also satisfied in σ' and, further, c is also satisfied in σ'. (See also Proposition 8.8 in Appendix 8.) With this observation in mind, we will now establish Theorem 2.1 constructively via the entropy compression method.

Theorem 3.3. *Suppose that ϕ is a k-CNF formula with m clauses on n variables and such that each clause shares variables with at most $(1-\epsilon)2^k/\mathrm{e} - 1$ other clauses, where $\epsilon \in (0,1)$. Then, the probability that Algorithm 1 on input ϕ has not terminated after $(n+m+s)/\log_2(1/(1-\epsilon))$ resampling steps is 2^{-s}.*

Remark 3.4. *Recall Theorem 2.1 and note that Theorem 3.3 requires an "ϵ-slack" in the LLL conditions. This is an assumption that is necessary in general applications of the entropy compression method. However, when the dependency digraph of the LLL application is symmetric, as it is here, this requirement for slack can be avoided [57].*

Proof of Theorem 3.3. Fix an integer $T \geq 0$ to serve as an upper bound on the number of resampling steps allowed. That is, the algorithm terminates as soon as it reaches a satisfying state, or has performed T resampling steps, whichever comes first. If we resample clauses uniformly, the

algorithm "consumes" precisely k random bits per resampling. Consider the set of all 2^{kT} binary strings of length kT. We will prove that, for sufficiently large values of T, the vast majority of these strings are such that the algorithm reaches a satisfying state in strictly fewer than T steps. Specifically, let $\mathcal{B}$ be the set of binary strings of length kT that cause the algorithm to perform precisely T resampling steps (thus, $\mathcal{B}$ also contains those binary strings for which a satisfying state is reached after the very last resampling). We will prove that $|\mathcal{B}| \leq 2^{\lambda T + B}$, where $\lambda < k$ and B is independent of T. Since each random string has the same probability, we see that the probability that the algorithm does not succeed in strictly fewer than T steps is bounded by $2^{B-(k-\lambda)T}$. In particular, for any $t \geq 0$, the probability that it does not succeed within $t + B/(k-\lambda)$ steps is exponentially small in $(k-\lambda)t$. Thus, not only does the algorithm finish quickly on average, but it exhibits a cutoff, i.e., there is a critical number of resampling steps $T_0 = B/(k-\lambda)$, beyond which the probability of non-termination drops exponentially.

To bound $|\mathcal{B}|$ we map each binary string $b \in \mathcal{B}$ to the following object: the sequence of T clauses resampled by the algorithm in the course of consuming b, followed by the state, σ_T , reached after these T resampling steps. The first crucial point is that this map is 1-to-1. To see this, observe that if the resampling step that took us from σ_{T-1} to σ_T involved clause c, then σ_{T-1} is the state that results by changing in σ_T the variables in c to the unique value assignment falsifying c (the uniqueness stemming from the nature of satisfiability constraints). And so on. Therefore, $|\mathcal{B}| \leq 2n$ times the number of possible clause resampling sequences.

To bound the number of possible clause resampling sequences we consider the forest of recursive calls of FIX induced by the execution of the algorithm. This is a labeled forest with T nodes, one for each call of FIX, where each node is labeled with the input clause of FIX in the corresponding call. Note that it consists of at most m trees, one for each call of FIX from SOLVE, and that each node has at most $\Delta := \max_{c \in \phi} |N(c) \cup \{c\}|$ distinct children. (Both of these statements are a direct corollary of the guarantee for the output of FIX we mentioned earlier.) Ordering the roots of the forest as well as the children of each node according to the indices of their labels allows us to recover the clause resampling sequence that occurred in the execution of the algorithm via a preorder transversal of the forest.

The latter observation implies that the number of possible clause resampling sequences is bounded by the number of possible recursive forests. The latter is bounded by the number of Δ-ary forests with at most m roots and T nodes, which is known to be at most $2^m \left(\mathrm{e}\Delta\right)^T \leq 2^{m+\lambda T}$, where $\lambda = k + \log_2(1-\epsilon)$. Note that for the inequality we used the fact that $\Delta \leq (1-\epsilon)2^k/\mathrm{e}$ according to our hypothesis.

Therefore, the probability that the algorithm performs more than $t_* = \frac{(n+m+s)}{\log_2(1/(1-\epsilon))}$ steps is at most $2^{n+m+(k-\lambda)t_*} = 2^{n+m+\log_2(1-\epsilon)t_*} = 2^{-s}$, concluding the proof.

□

3.2.2 Uniform Random Walks in Abstract Spaces

In this section we extend the analysis of Moser's satisfiability algorithm to general local search algorithms which address flaws by picking actions uniformly at random at each step.

Suppose we are given a family of flaws $F = \{f_1, \ldots, f_m\}$, a set of actions $\alpha(i, \sigma)$ for each flaw f_i and $\sigma \in f_i$, and a causality graph C. Fix any permutation π on $[m]$. For any $S \subseteq [m]$ let $\pi(S) = \min_{j \in S} \pi(j)$, i.e., the lowest index in S according to π. Recall that $U(\sigma)$ denotes the set

of indices of flaws present in state σ, and note that we sometimes abbreviate $\pi(U(\sigma))$ as $\pi(\sigma)$. We consider the following recursive algorithm.

Algorithm 2 Recursive Uniform Random Walk

```
1: procedure ELIMINATE
2:     σ ← σ_1
3:     while U(σ) ≠ ∅ do
4:         σ ← ADDRESS (π(σ), σ)
       return σ
5: procedure ADDRESS(i, σ)
6:     σ' ← A uniformly random element of α(i, σ)
7:     while S = U(σ') ∩ Γ(i) ≠ ∅ do
8:         σ' ← ADDRESS(π(S), σ')
```

Algorithm 2 is inspired by Moser's satisfiability algorithm; indeed, notice that procedure ADDRESS has the same property as procedure FIX of Algorithm 1: If on input (i, σ) procedure ADDRESS returns assignment σ', then every flaw that was absent from σ is also absent from σ' and, further, $\sigma' \notin f_i$. Thus, if Algorithm 2 ever terminates, it has reached a flawless object.

We will use the entropy compression method to show a simple algorithmic LLL condition under which Algorithm 2 is guaranteed to terminate fast. For an arbitrary state $\tau \in \Omega$ and flaw f_i, let

$$\mathrm{In}_i(\tau) := \{\sigma \in f_i : \rho_i(\sigma, \tau) > 0\} \ .$$

Theorem 3.5. *Let* $\alpha = \min_{i\in[m],\,\sigma\in f_i} |\alpha(i,\sigma)|$, $\beta = \max_{i\in[m]} |\mathrm{In}_i(\tau)|$ *and* $d = \max_{i\in[m]} |\Gamma(i)|$. *If there exists* $\epsilon \in [0,1)$ *such that* $d \cdot (\beta/\alpha) \cdot \mathrm{e} \leq 1 - \epsilon$ *, then the probability that Algorithm 2 does not reach a flawless object within* $(\log_2 |\Omega| + m + s) \,/\, \log_2(1/(1-\epsilon))$ *steps is at most* 2^{-s}.

Proof Sketch. We will call a walk $\Sigma = \sigma_1 \xrightarrow{w_1} \sigma_2 \xrightarrow{w_2} \ldots \sigma_t \xrightarrow{w_t} \sigma_{t+1}$ a *t-trajectory*. A t-trajectory is *bad* if it only goes through flawed states. Let $\mathrm{Bad}(t)$ be the set of bad t-trajectories starting at σ_1, and define the *witness* of a bad t-trajectory Σ to be the sequence $W(\Sigma) = (w_1, w_2, \ldots, w_t)$.

Note that the probability that the algorithm follows any trajectory Σ is $\prod_{i=1}^{t} \rho_{w_i}(\sigma_i, \sigma_{i+1})) \leq \alpha^{-t}$, and, therefore, the probability that it fails to reach a flawless object within t steps is at most $|\mathrm{Bad}(t)|\alpha^{-t}$. To bound the size of $\mathrm{Bad}(t)$ we will make use of the following key claim: The map from bad t-trajectories $\Sigma \to \langle W(\Sigma), \sigma_{t+1}\rangle$ is β-to-1. To see this, notice that $\sigma_t \in \mathrm{In}_{w_t}(\sigma_{t+1})$ and, therefore, there are at most β possible choices for it.

Thus, $|\mathrm{Bad}(t)|$ is bounded by the number of possible witness t-sequences multiplied by $|\Omega|\beta^t$. The proof is concluded by bounding the number of witness t-sequences by the number of d-ary forests with at most m roots, in an identical way to the proof of Theorem 3.3. In particular, we obtain that the probability that the algorithm fails to reach a flawless object within t steps is at most $|\Omega|2^m(d(1-\epsilon)\beta/\alpha)^t$. Therefore, if $t \geq t_* := (\log_2 |\Omega| + m + s)/\log_2(1/(1-\epsilon))$ the probability of failure is at most 2^{-s}, concluding the proof. □

Theorem 3.5 is a corollary of the main results of [3], where several more refined convergence conditions were developed and applied to various settings. In [6] the reader can find a more involved entropy compression argument that also applies to non-uniform random walks in the context of stochastic control.

Observe that Theorem 3.5 implies Theorem 3.3 by applying it to Moser's satisfiability algorithm. In this case $\Omega = \{0,1\}^n$, we have for each i one flaw f_i comprising the subset of Ω that violates clause c_i, and a symmetric causality graph in which an edge between two flaws exists if and only if the corresponding clauses share variables. Thus, $\alpha = 2, d = (1-\epsilon)2^k/\mathrm{e}$ and $\beta = 1$. (To see that $\beta = 1$, recall that the nature of satisfiability constraints implies that if a resampling step that takes the algorithm from state σ to state τ involves clause c, then σ has to be the unique state that is induced by mutating τ by changing the value of the variables of c to the unique value assignment falsifying c. In other words, given τ and c, σ is uniquely determined.)

We will next see how Theorem 3.5 can be used to make constructive an application of the lopsided Local Lemma to the problem of finding Latin transversals. This application was first given (non-constructively) in the paper of Erdős and Spencer [37] which introduced the original version of the lopsided LLL. We present it here as an example of how the entropy compression method can be used to design and analyze algorithms for applications of the LLL in non-product probability spaces. We note that Bissacot et.al. [21] gave a slightly better bound for this problem, which can also be made constructive via the entropy compression method and, in particular, using the main result of [3]. For the sake of simplicity, we will not show this improved bound.

Application to Latin transversals. Let M be an $n \times n$ matrix whose entries come from a set of colors C. A *Latin transversal* of M is a permutation $\pi \in S_n$ such that the entries $\{M(i,\pi(i))\}_{i=1}^n$ have distinct colors, i.e., a selection of n entries in distinct rows and columns such that no two elements have the same color.

Theorem 3.6. *If each color $c \in C$ appears at most $\Delta \leq \frac{n}{4\mathrm{e}}$ times in M, then there exists an algorithm that finds a Latin transversal of M in $O(n^5 \log n)$ steps with high probability.*

Proof. Let M be any matrix in which each color appears at most Δ times and let $\Omega = S_n$ be the set of all permutations of $[n]$. Let $P = P(M)$ be the set of all quadruples (i,j,i',j') such that $M(i,j) = M(i',j')$. For each quadruple $(i,j,i',j') \in P$ let

$$f_{i,j,i',j'} = \{\pi \in \Omega : \pi(i) = j \text{ and } \pi(i') = j'\} \ .$$

Thus, an element of Ω is flawless iff it is a Latin transversal of M.

To address the flaw induced by a pair of entries (i,j), (i',j') of M in an element $\pi \in \Omega$, we select two other entries (k,ℓ), (k',ℓ'), also selected by π, and replace the four entries (i,j), (i',j'), (k,ℓ), (k',ℓ') with the four entries (i,ℓ), (i',ℓ'), (k,j), and (k',j'). More precisely, for $\pi \in f = f_{i,j,i',j'}$ the set $\alpha(f,\pi)$ consists of all possible outputs of SWITCH(π,i,j,i',j').

Algorithm 3 SWITCH(π,i,j,i',j')

1: Let k be any element of $[n]$. Let $\ell = \pi(k)$.
2: Let $k' \neq k$ be any element of $[n]$. Let $\ell' = \pi(\ell')$.
3: Modify π to ω by the following "switch": $\omega(i) = \ell$, $\omega(i') = \ell'$, $\omega(k) = j$, $\omega(k') = j'$, and $\omega(z) = \pi(z)$ for all $z \notin \{i,i',k,k'\}$.

Enumerating the choices in Steps 1 and 2 we see that $|\alpha(f,\pi)| = n(n-1)$. Let us now consider the following symmetric causality graph G whose validity can easily be verified from the

description of SWITCH. Two flaws $f_{i,j,i',j'}$ and $f_{p,q,p',q'}$ are adjacent in G if and only if $\{i,i'\} \cap \{p,p'\} \neq \emptyset$ or $\{j,j'\} \cap \{q,q'\} \neq \emptyset$. Thus, each flaw $f_{i,j,i',j'}$ is adjacent to four types of flaws, corresponding to the four new entries (i,y), (i',y'), (x,i), and (x',j'). The maximum degree of G is at most $4n(\Delta-1)$, since for a fixed (i,j,i',j') we can choose (s,t) with $s \in \{i,i'\}$ or $t \in \{j,j'\}$ in $4n$ different ways and, as there are at most Δ entries of M with any given color, once (s,t) has been chosen there are at most $\Delta - 1$ choices for (s',t') such that $M(s,t) = M(s',t')$. Thus, the set of vertices in $\Gamma(f_{i,j,i',j'})$ is the union of four subsets, each of cardinality at most $n(\Delta-1)$.

We will next show that, for every flaw f and state ω, it holds that $|\mathrm{In}_f(\omega)| = 1$. To see this, consider any action $\pi \xrightarrow{f_{i,j,i',j'}} \omega$. Suppose that $\omega(i) = y$, $\omega(i') = y'$, $\omega^{-1}(j) = x$, and $\omega^{-1}(j') = x'$. Given ω and (i,j,i',j'), we see that the image of every element under π other than i, i', x, x' is the same as under ω, while $\pi(i) = j$, $\pi(i') = j'$, $\pi(x) = y$ and $\pi(x') = y'$.

The above observations imply that $\alpha = n(n-1)$, $d = 4n(\Delta-1)$ and $\beta = 1$ and, therefore,

$$d \cdot (\beta/\alpha) \cdot \mathrm{e} = \frac{n - 4\mathrm{e}}{n-1} =: 1 - \epsilon \ .$$

To bound the running time notice that there exist at most n^4 flaws, that $\log_2 |\Omega| = \log_2 n! = \Theta(n \log n)$, and that $\log_2(1/(1-\epsilon)) = \Omega(\frac{1}{n})$.

□

3.3 Backward-Looking Analysis

In this section we present the backward-looking analysis that was introduced by Moser and Tardos [81] in their breakthrough paper that made the LLL constructive for any product probability measure over explicitly presented variables. Besides allowing us to handle application of the LLL in non-uniform probability spaces, this kind of analysis has the benefit of being much more informative compared to others, as it often enables us to establish that algorithms have useful properties besides fast convergence. Among other things, it allows for the design of parallel and distributed algorithms, it can be used to give bounds on the entropy of the output distribution of the algorithm and the expected "weight" of a solution, it avoids the need for slack in the LLL conditions, and it does not depend on the strategy for choosing which flaw to address at each step.

Its main drawback is that it does not apply to arbitrary stochastic local search algorithms, a fact that has been established rigorously in [57]. Fortunately though, and as we will see in Chapter 5, it does apply to a large class of so-called *commutative* algorithms, a notion introduced by Kolmogorov [71]. Other minor drawbacks include the requirement of a symmetric input causality graph a and sufficiently random initial state for the algorithm.

Backward-looking analysis gets its name from the fact that it proceeds by bounding the probability of certain "witness structures" being consistent with the execution of the algorithm; these are acyclic graphs or digraphs labelled by flaws which grow backwards in time. The intention is to interpret each witness structure as an explanation of why certain (typically undesirable) events occurred during the execution of the algorithm.

Here we will use the notion of *witness trees* from [81] (slightly reformulated to fit our framework) in order to analyze the Moser-Tardos algorithm. Witness trees are used in the vast majority of applications of backward-looking analysis, many of which we will study in the context of commutative algorithms in Chapter 5.

3.3.1 Witness Trees

Throughout this section we consider algorithms with the property that f_i causes f_j if and only if f_j causes f_i. We will thus view the causality graph as an *undirected* graph G that may contain self-loops. We also write $i \sim j$ to denote that $j \in \Gamma(i)$ (or equivalently, $i \in \Gamma(j)$).

Recall that, given a trajectory $\Sigma = \sigma_1 \xrightarrow{w_1} \dots \sigma_t \xrightarrow{w_t} \sigma_{t+1}$, we denote by $W(\Sigma) = (w_1, \dots, w_t)$ the *witness sequence* of Σ. A *witness tree* $\tau = (T, \ell_T)$ is a finite, rooted, unordered tree T along with a labelling $\ell_T : V(T) \to [m]$ of its vertices with indices of flaws such that the children of a vertex $v \in V(T)$ receives labels from $\Gamma(\ell_T(v))$. To lighten the notation, we will sometimes write (v) to denote $\ell_T(v)$ and $V(\tau)$ instead of $V(T)$. Given a witness sequence $W = (w_1, w_2, \dots, w_t)$ we associate with each $i \in [t]$ a witness tree $\tau_W(i)$ that is constructed as follows: Let $\tau_W^i(i)$ be an isolated vertex labelled by w_i. Then, going backwards for each $j = i-1, i-2, \dots, 1$: if there is a vertex $v \in \tau_W^{j+1}(i)$ such that $(v) \sim w_j$ then we choose among those vertices the one having the maximum distance (breaking ties arbitrarily) from the root and attach a new child vertex u to v that we label w_j to get $\tau_W^j(i)$. If there is no such vertex v then $\tau_W^{j+1}(i) = \tau_W^j(i)$. Finally, let $\tau_W(i) = \tau_W^1(i)$.

We will say that a witness tree τ *occurs* in a trajectory Σ if $W(\Sigma) = (w_1, w_2, \dots, w_t)$ and there is $k \in [t]$ such that $\tau_W(k) = \tau$.

Below we prove some properties of witness trees that have positive probability to occur in the execution of the algorithm and which will be useful to us later.

Recall that for a set $S \subseteq [m]$, $\mathrm{Ind}(S) = \mathrm{Ind}_G(S)$ denotes the set of independent subsets of S with respect to G.

Proposition 3.7. *For a witness tree $\tau = (T, \ell_T)$ let $L_i = L_i(\tau)$ denote the set of labels of the nodes at distance i from the root. For each $i \geq 0$, $L_i \in \mathrm{Ind}([m])$.*

Proof. Fix a witness tree τ and let $W = (w_1, w_2, \dots, w_t)$ be a witness sequence that occurs with positive probability during the execution of the algorithm and such that $\tau_W(t) = \tau$. To prove the proposition, we will show that for each $i \geq 0$, and each pair of labels $\alpha, \beta \in L_i$, $\alpha \neq \beta$, we have that $\alpha \nsim \beta$.

Fix a pair of labels $\alpha, \beta \in L_i$. By the definition of τ, labels α, β correspond to two indices w_{j_1}, w_{j_2} of W. Assume without loss of generality that $j_1 < j_2$. Then, according to the algorithm for constructing τ, the node corresponding to w_{j_2} is attached to the i-th level of τ before the node corresponding to w_{j_1}. The proof is concluded by noticing that if $w_{j_1} = \alpha \sim \beta = w_{j_2}$, then the node corresponding to w_{j_1} is eligible to be a child of the node corresponding to w_{j_2}, and, thus, $\alpha \notin L_i$, which is a contradiction.

□

Proposition 3.8. *For a witness sequence W of length t and any two distinct $i, j \in [t]$ we have $\tau_W(i) \neq \tau_W(j)$.*

Proof. Let $W = (w_1, w_2, \dots, w_t)$ be a witness sequence that can occur with positive probability in an execution of the algorithm, and assume w.l.o.g. that $i < j$. If $w_i \neq w_j$ then the claim is straightforward because the root of $\tau_W(i)$ is w_i while the root of $\tau_W(j)$ is w_j. If $w_i = w_j = w$, then there are two cases. In the first case, $w \in \Gamma(w)$, and so tree $\tau_W(j)$ has at least one more vertex than $\tau_W(i)$. In the second case $w \notin \Gamma(w)$. This implies that at the i-th step of any trajectory Σ such that $W(\Sigma) = W$, flaw f_w was addressed and removed. However, the fact that $w_j = w$ implies that

there has to be $k \in \{i+1, \ldots, j-1\}$ such that addressing w_k introduced w, and thus $w_k \sim w$. Again, this means that $\tau_W(j)$ has at least one more vertex than $\tau_W(i)$.

□

3.3.2 The Moser-Tardos Algorithm

In this section we present and analyze the celebrated Moser-Tardos algorithm [81], which can be used to make constructive applications of the LLL in the so-called *variable setting*. There, the input is a set of variables $\mathcal{X} = \{x_1, \ldots, x_n\}$ with domains $\mathcal{D}_1, \ldots, \mathcal{D}_n$, along with a set of *constraints* $\mathcal{C} = \{c_1, \ldots, c_m\}$. Each constraint c_i is associated with a set of variables $\mathrm{var}(i) \subseteq \mathcal{X}$ and corresponds to a set of forbidden value assignments for these variables, i.e., values that *violate* the constraint. Moreover, we are also given a product measure μ over $\Omega := \prod_{i=1}^{n} \mathcal{D}_i$ (i.e., a collection of n independent random variables, one for each variable in $\mathcal{X}$), and a graph $G = G(V, E)$ with vertex set $V = [m]$ and edge set $E = \{(i,j) : \mathrm{var}(i) \cap \mathrm{var}(j) \neq \emptyset\}$.

The Moser-Tardos algorithm is defined as follows.

Algorithm 4 The Moser-Tardos Algorithm

```
1: procedure RESAMPLE(𝒳, 𝒞, μ)
2:     Sample all variables in 𝒳 according to μ.
3:     while there is some violated constraint do
4:         Choose an arbitrary violated constraint c_i
5:         (Re)sample every variable in var(i) according to μ.
```

For each $i \in [m]$, let f_i be the flaw (bad event) that corresponds to the subset of Ω that comprises the assignments which violate constraint c_i. Let also $D(i) = \{j : (i,j) \in E\}$ denote the set of neighbors of i in G, and observe that for every $S \subseteq [m] \setminus (D(i) \cup \{i\})$ it holds that $\mu(f_i \mid \cap_{j \in S} \overline{f_j}) = \mu(f_i)$, i.e., G is a dependency graph for the family of bad events $F = \{f_1, \ldots, f_m\}$ and the probability distribution μ. Finally, notice that the definition of the Moser-Tardos algorithm implies that G (slightly modified so that each vertex has a self-loop) is also a valid causality graph since a necessary condition for flaw f_i to cause f_j is that constraints c_i and c_j share variables.

Recall the cluster expansion condition (2.3). We will prove the following convergence guarantee for Algorithm 4, which corresponds to a constructive cluster expansion condition for the variable setting.

Theorem 3.9. *Let $(\mathcal{X}, \mathcal{C}, \mu)$ be the input of Algorithm 4 and $F = \{f_1, \ldots, f_m\}$ be the corresponding set of flaws. If there exist positive real numbers $\{\psi_i\}_{i=1}^{m}$ such that, for all $i \in [m]$,*

$$\frac{\mu(f_i)}{\psi_i} \sum_{S \in \mathrm{Ind}(D(i) \cup \{i\})} \prod_{j \in S} \psi_j \leq 1 \ , \tag{3.1}$$

then $\mathbb{E}[N_i] \leq \psi_i$, where N_i is the number of times constraint c_i is addressed during the execution of Algorithm 4.

Corollary 3.10. *Under the assumptions of Theorem 3.9, Algorithm 4 terminates after $\sum_{i=1}^{m} \psi_i$ steps in expectation.*

Remark 3.11. *If Shearer's condition* (2.4) *is satisfied, then one can replace ψ_i in the conclusion of Theorem 3.9 with $\frac{q^{\{i\}}(b)}{q^{\emptyset}(b)}$, where $b = (\mu(f_1), \ldots, \mu(f_m))$ is the vector whose entries correspond to the probabilities of the flaws in F under μ.*

3.3.3 Proof of Theorem 3.9

For a witness tree τ let $\Pr[\tau]$ denote the probability that it occurs during the execution of the algorithm. The key ingredient of the original analysis of Moser and Tardos is the following technical lemma, which is known as the *witness tree lemma*.

Lemma 3.12 (Witness Tree Lemma)**.** *For every witness tree τ, $\Pr[\tau] \leq \prod_{v \in V(\tau)} \mu(f_{(v)})$.*

The witness tree lemma (and its variations) is at the heart of every application of backward-looking analysis, and is also the main tool for establishing properties of stochastic local search algorithms besides fast convergence. In the interest of brevity, we will not present its proof here, as in Chapter 5 we will reformulate it and prove it in the commutative setting, which subsumes the variable setting. The interested reader is referred to the original proof of Moser and Tardos [81], which is based on the elegant "resampling table technique". We note though that this technique does not seem to generalize beyond the variable setting.

Note now that if W is the witness sequence corresponding to the trajectory of the algorithm, then N_i is the number of occurrences of c_i in W and, according to Proposition 3.8, also the number of distinct witness trees occurring in W that have their root labeled i. Therefore, letting $\mathcal{W}_i$ denote the set of witness trees whose root is labeled i, one can bound the expectation of N_i by summing the bounds in Lemma 3.12. In particular, the following lemma concludes the proof of Theorem 3.9.

Lemma 3.13. *Under the assumptions of Theorem 3.9,*

$$\sum_{\tau \in \mathcal{W}_i} \prod_{v \in V(\tau)} \mu\left(f_{(v)}\right) \leq \psi_i \ .$$

Proof of Lemma 3.13. To proceed, we use ideas from [81, 88]. Specifically, we introduce a branching process that is able to produce every tree in $\mathcal{W}_i$ and bound $\sum_{\tau \in \mathcal{W}_i} \prod_{v \in V(\tau)} \mu(f_{(v)})$ by analyzing it.

We start with a single node labelled by i. In each subsequent round each leaf u "gives birth" to a set of nodes whose set of (distinct) labels is a set $S \in \mathrm{List}((u)) := \mathrm{Ind}(D((u))) \cup \{(u)\}$ with probability proportional to $\prod_{j \in S} \psi_j$. Proposition 3.7 guarantees that this process creates every tree in $\mathcal{W}_i$ with positive probability. To express the exact probability received by each $S \subseteq [m]$ we define

$$Q(S) := \prod_{j \in S} \psi_j \tag{3.2}$$

and let $Z_{(u)} = \sum_{S \in \mathrm{List}((u))} Q(S)$. Clearly, each $S \in \mathrm{List}((u))$ receives probability equal to $\frac{Q(S)}{Z_{(u)}}$.

Proposition 3.14. *The branching process described above produces every tree $\tau \in \mathcal{W}_i$ with probability*

$$p_\tau = \frac{1}{\psi_i} \prod_{v \in V(\tau)} \frac{\psi_{(v)}}{\sum_{S \in \mathrm{List}((v))} \prod_{j \in S} \psi_j} \ .$$

Proof. For each tree $\tau \in \mathcal{W}_i$ and each node v of τ, let $N(v)$ denote the set of labels of its children. Then:

$$\begin{aligned} p_\tau &= \prod_{v \in V(\tau)} \frac{Q(N(v))}{\sum_{S \in \mathrm{List}((v))} Q(S)} \\ &= \frac{1}{\psi_i} \prod_{v \in V(\tau)} \frac{\psi_{(v)}}{\sum_{S \in \mathrm{List}((v))} Q(S)} \ . \end{aligned}$$

□

Returning now to the proof of Lemma 3.13, notice that:

$$\begin{aligned} \sum_{\tau \in \mathcal{W}_i} \prod_{v \in V(\tau)} \mu(f_{(v)}) &\leq \sum_{\tau \in \mathcal{W}_i} \prod_{v \in V(\tau)} \frac{\psi_{(v)}}{\sum_{S \in \mathrm{List}((v))} \prod_{j \in S} \psi_j} && (3.3) \\ &= \psi_i \sum_{\tau \in \mathcal{W}_i} p_\tau && (3.4) \\ &\leq \psi_i \ , \end{aligned}$$

where (3.3) follows by the hypothesis of Theorem 3.9 while (3.4) by Proposition 3.14.

□

As a final note, Remark 3.11 follows from the fact that, under the assumptions of Shearer's condition, it can be established that $\sum_{\tau \in \mathcal{W}_i} \prod_{v \in V(\tau)} \mu\left(f_{(v)}\right) \leq \frac{q^{\{i\}}(b)}{q^{\emptyset}(b)}$. The reader is referred to [57, 59, 70, 71] for more details.

3.4 Forward-Looking Analysis

In this section we present the forward-looking analysis which was introduced by Giotis et al. [44] and further developed in [4, 57]. In this kind of analysis, and in contrast to the backward-looking argument of Moser and Tardos, the structures of interest are forests labeled by flaws that grow forwards in time. Its main advantage is that it can be used to make constructive every[1] application of the LLL we are aware of, i.e., even applications outside the variable setting. (Indeed, the primary goal of this section is to state and prove the constructive analogue of condition (2.2) that was developed by Achlioptas, Iliopoulos and Kolmogorov [4].) On the other hand, its main drawbacks are that it cannot be used to guarantee distributional properties besides fast convergence, and that it imposes restrictions on the strategy the algorithm employs in order to choose a flaw to address in each step.

3.4.1 A General Algorithmic LLL Condition

We start by describing the general algorithmic LLL condition of [4] which can be used to analyze arbitrary stochastic local search algorithms under minimal assumptions about how the algorithm

[1] As shown in [57], there exist scenarios where the LLL applies but finding the desired output is computationally hard under standard cryptographic assumptions. However, we are not aware of any "natural" LLL application that cannot be made constructive.

chooses which flaw to address at each step; e.g. it is enough for the algorithm to choose the flaw with the lowest index according to some permutation. There, the input is the algorithm to be analyzed and a probability measure μ over the state space of the algorithm, that is either inherited from an application of the probabilistic method, as in the classical LLL, or introduced by the algorithm designer. At a high level, the role of the measure is to gauge how efficiently the algorithm gets rid of flaws, by quantifying the trade-off between the probability that a flaw is present at some state of the execution of the algorithm and the number of other flaws each flaw can possibly introduce when the algorithm addresses it. In particular, the quality of the convergence condition is affected by the *compatibility* between the measure and the algorithm.

To make this rigorous, for any fixed probability distribution $\mu > 0$ on Ω, we define the *charge* of flaw f_i with respect to μ to be

$$\gamma_i := \max_{\tau \in \Omega} \frac{1}{\mu(\tau)} \sum_{\sigma \in f_i} \mu(\sigma) \rho_i(\sigma, \tau) \ . \tag{3.5}$$

That is, the charge γ_i is an upper bound on the ratio between the ergodic flow into a state τ that arrives through transitions that address flaw f_i and the probability $\mu(\tau)$ of τ.

As we will see in the next section, we can interpret charges as bounds on certain conditional probabilities of flaws with respect to probability distribution μ, i.e., the quantities of interest in condition (2.2). Additionally though, charges can be seen as a measure of compatibility between the actions of the algorithm and μ. To see this, for a fixed flaw f_i consider the probability $\nu_i(\tau)$ of ending up in state τ after (i) sampling a state $\sigma \in f_i$ according to μ; and then (ii) addressing f_i at σ. Define the *distortion* associated with f_i as

$$d_i := \max_{\tau \in \Omega} \frac{\nu_i(\tau)}{\mu(\tau)} \geq 1 \ , \tag{3.6}$$

i.e., the maximum possible inflation of a state probability incurred by addressing f_i (relative to its probability under μ, and averaged over the initial state $\sigma \in f_i$ according to μ). Now observe from (3.5) that

$$\gamma_i = \max_{\tau \in \Omega} \frac{1}{\mu(\tau)} \sum_{\sigma \in f_i} \mu(\sigma) \rho_i(\sigma, \tau) = d_i \cdot \mu(f_i) \ . \tag{3.7}$$

An algorithm for which $d_i = 1$ is called a *resampling oracle* [57] for f_i, and notice that it perfectly removes the conditional of the addressed flaw. For instance, note that the Moser-Tardos algorithm defines resampling oracles for flaws in the variable setting.

Remarkably, Harvey and Vondrák have shown that the assumptions of condition (2.1) imply the existence of resampling oracles for addressing the flaws that correspond to bad events, as well as an associated causality graph that is a subgraph of the dependency graph for the bad events. Although their proof does not guarantee that these oracles are efficiently implementable, it does implies a new non-constructive proof of condition (2.1), revealing a strong connection between the LLL and stochastic local search algorithms.

Nonetheless, designing resampling oracles for sophisticated measures can be impossible by local search. This is because small, but non-vanishing, correlations can travel arbitrarily far in Ω. Moreover, the assumptions of the weaker condition (2.2) do not imply the existence of resampling

oracles. Thus, allowing for non-trivial distortion can be very helpful, especially in cases where correlations decay with distance.

Recall now that θ denotes the probability distribution according to which the algorithm chooses the first state of its trajectory, and that $U(\sigma)$ denotes the set of flaw indices present in state σ. We denote by $\mathrm{Span}(\theta)$ the set of flaw indices that may be present in the initial state, i.e., $\mathrm{Span}(\theta) = \bigcup_{\sigma\in\Omega:\theta(\sigma)>0} U(\sigma)$.

The main result of this section is the following.

Theorem 3.15. *Assume that, at each step, the algorithm chooses to address the lowest indexed flaw according to an arbitrary, but fixed, permutation of* $[m]$*. If there exist positive real numbers* $\{\psi_i\}$ *for* $1 \leq i \leq m$ *such that*

$$\zeta_i := \frac{\gamma_i}{\psi_i} \sum_{S\subseteq\Gamma(i)} \prod_{j\in S} \psi_j < 1 \quad \textit{for every } i \in [m], \tag{3.8}$$

then the algorithm reaches a flawless object within $(T_0 + s)/\log_2(1/(1-\epsilon))$ *steps with probability at least* $1 - 2^{-s}$*, where* $\epsilon = 1 - \max_{i\in[m]} \zeta_i > 0$*, and*

$$T_0 = \log_2\left(\max_{\sigma\in\Omega} \frac{\theta(\sigma)}{\mu(\sigma)}\right) + \log_2\left(\sum_{S\subseteq\mathrm{Span}(\theta)} \prod_{j\in S} \psi_j\right) \quad .$$

As we will see very soon, Theorem 3.15 can be seen as the constructive analogue of condition (2.2), which we recall that is the weakest known condition in the general setting. We will prove Theorem 3.15 in Sections 3.4.3 and 3.4.5. In Section 3.4.3 we will also discuss how we can obtain constructive analogues of the cluster expansion and Shearer's conditions via algorithms that adopt more sophisticated flaw choice strategies. Later, in Chapters 6 and 7, we will use Theorem 3.15 to analyze Molloy's algorithm [76] for coloring triangle-free graphs and to give algorithms for the seminal results of Kahn [65, 66] that establish (non-constructively) that the Goldberg-Seymour and List-Edge-Coloring conjectures are true asymptotically.

3.4.2 Causality, Lopsidependency and Approximate Resampling Oracles

In this section we show a connection between Theorem 3.15 and condition (2.2) that was established by Iliopoulos and Sinclair [60]. While this section is not essential to neither the proof nor to the applications of Theorem 3.15, it does provide useful intuition since it implies the following natural approach to making applications of the lopsided LLL algorithmic: To design a local search algorithm for addressing the flaws that correspond to bad events, we consider transition probability distributions $\{\rho_i(\sigma,\cdot)\}_{i\in[m],\sigma\in f_i}$ whose supports induce a causality graph that coincides with the lopsidependency graph of the lopsided LLL application of interest. Identifying this family of distributions is typically an automated task. The key to successful implementation is our ability to make the way in which the algorithm addresses flaws sufficiently *compatible* with the underlying probability measure μ. To make this precise, we prove Theorem 3.16 which shows that Theorem 3.15 can be seen as the algorithmic counterpart of condition (2.2).

Theorem 3.16. *Given a family of flaws $F = \{f_1, \ldots, f_m\}$ over a state space Ω, an algorithm $\mathcal{A}$ with causality graph C with neighborhoods $\Gamma(\cdot)$, and a measure μ over Ω, then for each $S \subseteq F \setminus \Gamma(i)$ we have*

$$\mu\Big(f_i \mid \bigcap_{j \in S} \overline{f_j}\Big) \leq \gamma_i \ , \tag{3.9}$$

where the γ_i are the charges of the algorithm as defined in (3.5)*.*

Proof. Let $F_S := \bigcap_{j \in S} \overline{f_j}$. Observe that

$$\begin{aligned} \mu(f_i \mid F_S) &= \frac{\mu(f_i \cap F_S)}{\mu(F_S)} \\ &= \frac{\sum_{\sigma \in f_i \cap F_S} \mu(\sigma) \sum_{\tau \in a(i,\sigma)} \rho_i(\sigma, \tau)}{\mu(F_S)} \\ &= \frac{\sum_{\sigma \in f_i \cap F_S} \mu(\sigma) \sum_{\tau \in F_S} \rho_i(\sigma, \tau)}{\mu(F_S)} \ , \end{aligned} \tag{3.10}$$

where the second equality holds because each $\rho_i(\sigma, \cdot)$ is a probability distribution and the third by the definition of causality and the fact that $S \subseteq F \setminus \Gamma(i)$. Now notice that changing the order of summation in (3.10) gives

$$\begin{aligned} \frac{\sum_{\tau \in F_S} \sum_{\sigma \in f_i \cap F_S} \mu(\sigma) \rho_i(\sigma, \tau)}{\mu(F_S)} &= \frac{\sum_{\tau \in F_S} \mu(\tau) \sum_{\sigma \in f_i \cap F_S} \frac{\mu(\sigma)}{\mu(\tau)} \rho_i(\sigma, \tau)}{\mu(F_S)} \\ &\leq \frac{\sum_{\tau \in F_S} \mu(\tau) \left(\max_{\tau' \in \Omega} \sum_{\sigma \in f_i} \frac{\mu(\sigma)}{\mu(\tau')} \rho_i(\sigma, \tau') \right)}{\mu(F_S)} \\ &= \gamma_i \ . \end{aligned}$$

□

In words, Theorem 3.16 shows that causality graph C is a lopsidependency graph with respect to measure μ with $b_i = \gamma_i$ for all $i \in [m]$. Thus, when designing an algorithm for an application of condition (2.2) using Theorem 3.16, we have to make sure that the induced causality graph coincides with the lopsidependency graph, and that the measure distortion induced when addressing flaw f_i is sufficiently small so that the resulting charge γ_i is bounded above by b_i.

3.4.3 Forward-Looking Witness Structures and the Role of the Flaw Choice Strategy

Recall that a focused local search algorithm $\mathcal{A}$ amounts to a flaw choice mechanism driving a random walk with transition probabilities $\{\rho_i\}_{i \in [m]}$ and starting state distribution θ.

The high level idea of forward-looking analysis is to bound the probability that $\mathcal{A}$ runs for t or more steps by partitioning the set of all t-trajectories into equivalence classes, bounding the total probability of each class, and summing the bounds for the different classes. Specifically, the partition is according to the t-sequence of the first t flaws addressed.

Definition 3.17. *For any integer $t \geq 1$, let $\mathcal{W}_t(\mathcal{A})$ denote the set containing all t-sequences of flaws that have positive probability of being the first t flaws addressed by $\mathcal{A}$.*

In general, the content of $\mathcal{W}_t(\mathcal{A})$ is an extremely complex function of the flaw choice strategy. An essential idea of the analysis is to estimate it by *syntactic* considerations capturing the following necessary condition for $W \in \mathcal{W}_t(\mathcal{A})$: while the very first occurrence of any flaw f_j in W may be attributed to $f_j \ni \sigma_1$, every subsequent occurrence of f_j must be preceded by a distinct occurrence of a flaw f_i that "assumes responsibility" for f_j. Definition 3.18 below establishes a framework for bounding $\mathcal{W}_t(\mathcal{A})$ by relating flaw choice with responsibility by (i) requiring that the flaw choice mechanism is such that the elements of $\mathcal{W}_t(\mathcal{A})$ can be unambiguously represented by forests with t vertices; while on the other hand (ii) generalizing the subsets of flaws for which a flaw f_i may be responsible from subsets of $\Gamma(i)$ to arbitrary subsets of flaws, thus enabling responsibility shifting.

Definition 3.18. *We will say that algorithm $\mathcal{A}$ is* traceable *if there exist sets $\mathrm{Roots}(\theta) \subseteq 2^{[m]}$ and $\mathrm{List}(1) \subseteq 2^{[m]}, \ldots, \mathrm{List}(m) \subseteq 2^{[m]}$ such that for every $t \geq 1$, there is an injection from the flaw sequences in $\mathcal{W}_t(\mathcal{A})$ to the set of unordered rooted forests with t vertices that have the following properties:*

1. *Each vertex of the forest is labeled by an integer $i \in [m]$.*
2. *The labels of the roots of the forest are distinct and form an element of $\mathrm{Roots}(\theta)$.*
3. *The indices labeling the children of each vertex are distinct.*
4. *If a vertex is labelled by $i \in [m]$, then the labels of its children form an element of $\mathrm{List}(i)$.*

In Section 3.4.5 we demonstrate that algorithms whose flaw choice strategy corresponds to fixing an arbitrary permutation π on $[m]$ and, in each step, addressing the lowest indexed flaw according to π, are traceable. Specifically, for this class of algorithms the set $\mathcal{W}_t$ can be mapped injectively into a family of forests, satisfying Definition 3.18, with $\mathrm{Roots}(\theta) = 2^{\mathrm{Span}(\theta)}$ and $\mathrm{List}(i) = 2^{\Gamma(i)}$. Moreover, and as we show in Appendix 8.2, recursive algorithms akin to Algorithm 2 are also traceable and can be used to make constructive applications of the cluster expansion condition (2.3). In this case, and assuming the input causality graph is symmetric, $\mathcal{W}_t$ can be injected into so-called "Recursive Forests" with $\mathrm{Roots}(\theta) = \mathrm{Ind}(\mathrm{Span}(\theta))$ and $\mathrm{List}(i) = \mathrm{Ind}(\Gamma(i))$. Thus, the aforementioned convergence conditions follow readily from Theorem 3.19 below.

Theorem 3.19. *If algorithm $\mathcal{A}$ is traceable and there exist positive real numbers $\{\psi_i\}_{i\in[m]}$ such that for every $i \in [m]$,*

$$\zeta_i := \frac{\gamma_i}{\psi_i} \sum_{S \in \mathrm{List}(i)} \prod_{j \in S} \psi_j < 1 \ , \tag{3.11}$$

then $\mathcal{A}$ reaches a flawless object within $(T_0 + s)/\log_2(1/(1-\epsilon))$ steps with probability at least $1 - 2^{-s}$, where $\epsilon = 1 - \max_{i\in[m]} \zeta_i$ and

$$T_0 = \log_2 \left(\max_{\sigma \in \Omega} \frac{\theta(\sigma)}{\mu(\sigma)} \right) + \log_2 \left(\sum_{S \in \mathrm{Roots}(\theta)} \prod_{j \in S} \psi_j \right) \ .$$

Moreover, Theorem 3.19 also implies the "Left-Handed Random Walk" result of [3] and extends it to non-uniform transition probabilities, since that algorithm is also traceable. Notably, in the left-handed LLL introduced by Pegden [87] and which inspired the algorithm, the flaw order π can be chosen in a *provably* beneficial way, unlike in the algorithms of Theorem 3.15, which are indifferent to π. Establishing this goodness, though, entails attributing responsibility very differently from what is suggested by the causality digraph, making full use of the power afforded by traceability and Theorem 3.19. We present the details in Appendix 8.2.

Finally, assuming the input causality graph is symmetric and using a sophisticated flaw choice strategy known as "MaximalSetResample" [57], one can establish the constructive analogue of Shearer's condition (2.4). This result is out of the scope of Theorem 3.19, but only in that it requires a counting argument that exploits global properties of the induced witness structures. The reader is referred to [57, 71] for more details.

3.4.4 Proof of Theorem 3.19

In this Section we present the proof of Theorem 3.19.

Bounding the Probabilities of Trajectories

To bound the probability that an algorithm $\mathcal{A}$ runs for t or more steps we partition its t-trajectories into equivalence classes, bound the total probability of each class, and sum the bounds for the different classes. Formally, for a trajectory $\Sigma = \sigma_1 \xrightarrow{w_1} \sigma_2 \xrightarrow{w_2} \cdots$ we let $W(\Sigma) = w_1, w_2, \ldots$ denote its *witness* sequence, i.e., the sequence of indices of flaws addressed along Σ. (Note that Σ determines $W(\Sigma)$ as the flaw choice strategy is deterministic.) We let $W_t(\Sigma) = \perp$ if Σ has fewer than t steps, otherwise we let $W_t(\Sigma)$ be the t-prefix of $W(\Sigma)$. Finally, recall that $\mathcal{W}_t = \mathcal{W}_t(\mathcal{A})$ denotes the range of W_t for algorithm $\mathcal{A}$ except for $\perp$, i.e., $\mathcal{W}_t(\mathcal{A})$ is the set of t-sequences of flaws that have positive probability of being the first t flaws addressed by $\mathcal{A}$, as per Definition 3.17. Thus,

$$\Pr[\text{Algorithm } \mathcal{A} \text{ takes } t \text{ or more steps}] = \sum_{W \in \mathcal{W}_t(\mathcal{A})} \Pr[W_t = W] \ .$$

Key to our analysis will be the derivation of an upper bound for $\Pr[W_t = W]$ that holds for *arbitrary* t-sequences of flaws, i.e., not necessarily elements of $\mathcal{W}_t(\mathcal{A})$, and which factorizes over the flaws in W. For an arbitrary sequence of flaws $A = (a_1, \ldots, a_t)$, let us denote by (i) the index $j \in [m]$ such that $a_i = f_j$.

Lemma 3.20. *Let $\xi = \xi(\theta, \mu) = \max_{\sigma \in \Omega}\{\theta(\sigma)/\mu(\sigma)\}$. For every sequence of flaws $W = w_1, \ldots, w_t$,*

$$\Pr[W_t = W] \leq \xi \prod_{i=1}^{t} \gamma_{(i)} \ .$$

Proof. We claim that for every $t \geq 0$, every t-sequence of flaws W, and every state $\tau \in \Omega$,

$$\Pr[(W_t = W) \wedge (\sigma_{t+1} = \tau)] \leq \xi \cdot \prod_{i=1}^{t} \gamma_{(i)} \cdot \mu(\tau) \ . \tag{3.12}$$

Summing (3.12) over all $\tau \in \Omega$ proves the lemma.

To prove our claim (3.12) we proceed by induction on $|W|$ after recalling that for every $i \in [m]$ and $\tau \in \Omega$, by the definition of $\{\gamma_i\}_{i\in[m]}$,

$$\sum_{\sigma \in f_i} \mu(\sigma)\rho_i(\sigma,\tau) \leq \gamma_i \cdot \mu(\tau) \ . \tag{3.13}$$

For $|W| = 0$, (3.12) holds because $\Pr[\sigma_1 = \tau] = \theta(\tau) \leq \xi\mu(\tau)$ for all $\tau \in \Omega$, by the definition of ξ.

For the inductive step, assume that (3.12) holds for all s-sequences of flaws, for some $s \geq 0$. Let $A' = A$, f_i be any sequence of $s+1$ flaws and let $\tau \in \Omega$ be arbitrary. The first inequality below is due to the fact that, since f_i is the last flaw in A', a necessary (but not sufficient) condition for the event $W_{s+1} = A'$ to occur is that f_i is present in the state that results after the flaws in A have been addressed. (It is not sufficient as $\mathcal{A}$ may choose to address a flaw other than f_i.) Then we have

$$\begin{aligned}
\Pr[W_{s+1} = A' \cap \sigma_{s+2} = \tau] &\leq \sum_{\sigma\in f_i} \rho_i(\sigma,\tau)\Pr[W_s = A \cap \sigma_{s+1} = \sigma] \\
&\leq \xi \cdot \prod_{i=1}^{s} \gamma_{(i)} \cdot \sum_{\sigma\in f_i} \mu(\sigma)\cdot\rho_i(\sigma,\tau) \\
&\leq \xi \cdot \prod_{i=1}^{s+1} \gamma_{(i)} \cdot \mu(\tau) \ ,
\end{aligned}$$

where the second inequality follows from the inductive hypothesis and the third from (3.13). □

Bounding the Sum

Per the hypothesis of Theorem 3.19, the sequences in $\mathcal{W}_t$ can be injected into a set of rooted forests with t vertices that satisfy the properties of Definition 3.18. Let $\mathcal{F}_t$ be the set of *all* forests with t vertices that satisfy the properties of Definition 3.18. By Lemma 3.20, to prove the theorem it suffices to prove that $\max_{\sigma\in\Omega} \frac{\theta(\sigma)}{\mu(\sigma)} \sum_{\phi\in\mathcal{F}_t} \prod_{v\in V(\mathcal{F}_t)} \gamma_{(v)}$ is exponentially small in s for $t = T_0 + s$, where $(v) \in [m]$ denotes the label of vertex v.

To proceed, we use ideas similar to the ones in the proof of Lemma 3.13. Specifically, we introduce a branching process produces every forest in $\mathcal{F}_t$ with positive probability and bound $\sum_{\phi\in\mathcal{F}_t} \prod_{i=1}^{t} \gamma_{(i)}$ by analyzing it. Let us write $\mathrm{Roots}(\theta) = \mathrm{Roots}$ to simplify notation, and recall that neither the trees in each forest, nor the nodes inside each tree are ordered. To start the process we produce the roots of the labeled forest by picking a set $S \in \mathrm{Roots}$ with probability proportional to $\prod_{j\in S} \psi_j$. In each subsequent round we follow a very similar procedure. Specifically, at each step, each leaf node u with label ℓ "gives birth" to a set of nodes whose set of (distinct) labels is a set $S \in \mathrm{List}((u))$ with probability proportional to $\prod_{j\in S} \psi_j$.

It is not hard to see that this process creates every forest in $\mathcal{F}_t$ with positive probability. Specifically, for a vertex labeled by ℓ, every set $S \notin \mathrm{List}(\ell)$ receives probability 0, while every set $S \in \mathrm{List}(\ell)$ receives probability proportional to

$$Q(S) := \prod_{g\in S} \psi_g \ .$$

In particular, letting $Z_\ell = \sum_{S' \in \mathrm{List}(\ell)} Q(S')$, we see that each $S \in \mathrm{List}(\ell)$ receives probability exactly equal to $\frac{Q(S)}{Z_\ell}$. Similarly, each set $R \in \mathrm{Roots}$ receives probability equal to $Q(R)\left(\sum_{R' \in \mathrm{Roots}} Q(R')\right)^{-1}$.

For each forest $\phi \in \mathcal{F}_t$ and each node v of ϕ, let $N(v)$ denote the set of labels of the children of v and let $\mathrm{List}(v) = \mathrm{List}(\ell)$ and $\psi_v = \psi_\ell$, where $\ell = (v)$ is the label of v.

Lemma 3.21. *The branching process described above produces every forest $\phi \in \mathcal{F}_t$ with probability*

$$p_\phi = \left(\sum_{S \in \mathrm{Roots}} \prod_{i \in S} \psi_i\right)^{-1} \prod_{v \in \phi} \frac{\psi_v}{\sum_{S \in \mathrm{List}(v)} Q(S)} \ .$$

Proof. Let R denote the set of roots of ϕ. According to our discussion above,

$$\begin{aligned} p_\phi &= \frac{Q(R)}{\sum_{S \in \mathrm{Roots}} Q(S)} \prod_{v \in V(\phi)} \frac{Q(N(v))}{\sum_{S \in \mathrm{List}(v)} Q(S)} \\ &= \frac{Q(R)}{\sum_{S \in \mathrm{Roots}} Q(S)} \cdot \frac{\prod_{v \in V(\phi) \setminus R} \psi_v}{\prod_{v \in V(\phi)} \sum_{S \in \mathrm{List}(v)} Q(S)} \\ &= \left(\sum_{S \in \mathrm{Roots}} Q(S)\right)^{-1} \prod_{v \in V(\phi)} \frac{\psi_v}{\sum_{S \in \mathrm{List}(v)} Q(S)} \ . \end{aligned}$$

□

Notice now that

$$\begin{aligned} \sum_{\phi \in \mathcal{F}_t} \prod_{v \in V(\phi)} \gamma_{(v)} &= \sum_{\phi \in \mathcal{F}_t} \prod_{v \in V(\phi)} \frac{\zeta_{(v)}\, \psi_{(v)}}{\sum_{S \in \mathrm{List}((v))} Q(S)} && (3.14) \\ &\leq \left(\max_{i \in [m]} \zeta_i\right)^t \sum_{\phi \in \mathcal{F}_t} \prod_{v \in V(\phi)} \frac{\psi_{(v)}}{\sum_{S \in \mathrm{List}((v))} Q(S)} \\ &= \left(\max_{i \in [m]} \zeta_i\right)^t \sum_{\phi \in \mathcal{F}_t} \left(p_\phi \sum_{S \in \mathrm{Roots}} Q(S)\right) && (3.15) \\ &= \left(\max_{i \in [m]} \zeta_i\right)^t \sum_{S \in \mathrm{Roots}} Q(S) \ , && (3.16) \end{aligned}$$

where (3.14) follows by the definition of ζ_i in (3.11), (3.15) follows from Lemma 3.21 and (3.16) follows by the fact that $\mathcal{F}_t$ is a subset of the support of the probability distribution over forests induced by the branching process.

Using (3.16) we see that the binary logarithm of the probability that the walk does not encounter a flawless state within t steps is at most $t \log_2 (\max_{i \in F} \zeta_i) + T_0$, where

$$T_0 \ = \ \log_2 \left(\max_{\sigma \in \Omega} \frac{\theta(\sigma)}{\mu(\sigma)}\right) + \log_2 \left(\sum_{S \in \mathrm{Roots}} \prod_{i \in S} \psi_i\right) \ .$$

Therefore, if $t = (T_0 + s)/\log_2(1/\max_{i \in F} \zeta_i)$, the probability that the algorithm does not reach a flawless state within t steps is at most 2^{-s}.

3.4.5 Proof of Theorem 3.15

In this section we present the proof of Theorem 3.15. Specifically, we show that algorithms whose flaw choice strategy corresponds to fixing an arbitrary permutation π on $[m]$ and, in each step, addressing the lowest indexed flaw according to π, are traceable. Specifically, we show that for these algorithms the sets $\mathcal{W}_t$ can be injected into a family of forests satisfying Definition 3.18 with $\mathrm{Roots}(\theta) = 2^{\mathrm{Span}(\theta)}$ and $\mathrm{List}(i) = 2^{\Gamma(i)}$. Combining this fact with Theorem 3.19 concludes the proof of Theorem 3.15.

Fix a permutation π on $[m]$. For every $S \subseteq [m]$, let $\pi(S) = \min_{j \in S} \pi(j)$, i.e., the lowest index in S according to π. Recall also that $U(\sigma)$ is the set of indices of flaws present in state σ, and note that we sometimes write $\pi(\sigma)$ to abbreviate $\pi(U(\sigma))$.

Let B_i be the set of indices of flaws "introduced" by the i-th step of the walk, where a flaw f_k is said to "introduce itself" if it remains present after an action from $\alpha(k, \sigma_i)$ is taken. Formally, we define B_i inductively as follows:

Definition 3.22. *Let* $B_0 = U(\sigma_1)$. *For* $1 \leq i \leq t-1$, *let* $B_i = U(\sigma_{i+1}) \setminus (U(\sigma_i) \setminus \pi(\sigma_i))$.

Let $B_i^* \subseteq B_i$ comprise the indices of those flaws addressed in the course of the trajectory. Thus, $B_i^* = B_i \setminus \{O_i \cup N_i\}$, where O_i comprises any flaws in B_i that were eradicated "collaterally" by an action taken to address some other flaw, and N_i comprises any flaws in B_i that remained present in every subsequent state after their introduction without being addressed. Formally,

Definition 3.23. *Let* $\Sigma = (\sigma_1 \xrightarrow{w_1} \ldots \sigma_t \xrightarrow{w_t} \sigma_{t+1})$ *be any bad* t*-trajectory and define the sequence* $(B_0^*, B_1^*, \ldots, B_{t-1}^*)$, *where for* $0 \leq i \leq t-1$,

$$\begin{aligned} O_i &= \{k \in B_i \mid \exists j \in [i+1, t] : k \notin U(\sigma_{j+1}) \wedge \forall \ell \in [i+1, j] : k \neq w_\ell\} \\ N_i &= \{k \in B_i \mid \forall j \in [i+1, t] : k \in U(\sigma_{j+1}) \wedge \forall \ell \in [i+1, t] : k \neq w_\ell\} \\ B_i^* &= B_i \setminus \{O_i \cup N_i\} \ . \end{aligned}$$

Given $(B_0^*, B_1^*, \ldots, B_{i-1}^*)$, we can determine $(w_1, w_2, \ldots, w_i)$ inductively as follows. Define $E_1 = B_0^*$, while for $i \geq 1$,

$$E_{i+1} = (E_i - w_i) \cup B_i^* \ . \tag{3.17}$$

By construction, the set $E_i \subseteq U(\sigma_i)$ is guaranteed to contain $w_i = \pi(\sigma_i)$. Therefore, it must be that $\pi(E_i) = w_i$. We note that this is the only place we make use of the fact that the algorithm picks the lowest indexed flaw present to address at each step, thus guaranteeing that for every $i \in [m]$ and $S \subseteq [m]$, if $\pi(S) \neq i$ then $\pi(S \setminus \{i\}) = \pi(S)$.

We next give another 1-to-1 map, mapping each sequence $(B_0^*, B_1^*, \ldots, B_{t-1}^*)$ to a vertex-labeled rooted forest. Specifically, the *Witness Forest* of a bad t-trajectory Σ has $|B_0^*|$ trees and t vertices, each vertex labeled by an element of $W(\Sigma)$. To construct it, we first lay down $|B_0^*|$ vertices as roots and then process the sets $B_1^*, B_2^*, \ldots$ in order, each set becoming the progeny of an already existing vertex (empty sets giving rise to leaves).

Witness Forest Construction

1: Lay down $|B_0^*|$ vertices, each labeled by a different element of B_0^*, and let V_1 consist of these vertices
2: **for** $i = 1$ to $t - 1$ **do**
3: Let v_i be the vertex in V_i with lowest label according to π
4: Add $|B_i^*|$ children to v_i, each labeled by a different element of B_i^*
5: Construct V_{i+1} from V_i by adding the children of v_i and removing v_i

Observe that even though neither the trees, nor the nodes inside each tree of the Witness Forests are ordered, we can still reconstruct $W(\Sigma)$ since the set of labels of the vertices in V_i equals E_i for all $0 \leq i \leq t - 1$. Note also that the set of roots of every witness forest is in $\mathrm{Span}(\theta)$, and that if a vertex is labeled by $i \in [m]$, then the labels of its children form an element of $\Gamma(i)$, as required. This concludes the proof of Theorem 3.15. $\square$

Chapter 4

Point-to-Set Correlations and a Linear Algebraic Perspective

As we have already discussed, following the groundbreaking algorithm of Moser and Tardos, there has been a plethora of results analyzing local search algorithms for various constraint satisfaction problems. These algorithms fall into two broad categories: *resampling* algorithms, analyzed via various algorithmic LLL conditions; and *backtracking* algorithms, analyzed via entropy compression arguments. In this chapter we present a new convergence condition due to Achlioptas, Iliopoulos and Sinclair [5] that seamlessly handles resampling, backtracking, and *hybrid* algorithms, i.e., algorithms that perform both resampling and backtracking steps. Unlike all past LLL work, this condition replaces the notion of a dependency or causality graph by quantifying *point-to-set correlations* between bad events. As a result, this condition captures the most general algorithmic LLL condition known as a special case; simplifies the analysis of entropy compression applications; relates backtracking algorithms, which are conceptually very different from resampling algorithms, to the LLL; and, most importantly, allows for the analysis of hybrid algorithms, which were outside the scope of previous LLL conditions. We will give several applications of this condition, including a new hybrid vertex coloring algorithm that extends the recent important result of Molloy [76] for coloring triangle-free graphs to arbitrary graphs (see Chapter 6).

Our key insight is that LLL-inspired convergence arguments can be seen as a method for bounding the spectral radius of a matrix specifying the algorithm. This linear algebraic perspective allows us to view charges of flaws (as defined in (3.5)) as norms of transition matrices induced by the algorithm, giving us a new handle on these key quantities. It also allows us to generalize Kolmogorov's notion of commutative algorithms [71], casting it as matrix commutativity, which affords much simpler proofs both of the original results and of recent extensions. We will introduce the linear algebraic perspective and give some illustrative applications in this chapter. We will present its applications to commutative algorithms in Chapter 5.

4.1 The Lovász Local Lemma as a Spectral Condition

We start by sketching our linear algebraic viewpoint [5] for the analysis of stochastic local search algorithms.

As in the previous chapter, let Ω be a (large) finite set of objects and let $\Omega^* \subseteq \Omega$ be the "bad"

part of Ω, comprising the flawed objects; e.g., for a CNF formula on n variables $\Omega = \{0,1\}^n$ and Ω^* comprises all non-satisfying assignments. Imagine a particle trying to escape Ω^* by following a Markov chain[1] on Ω with transition matrix P. Our task is to develop conditions under which the particle eventually escapes, thus establishing in particular that $\Omega^* \neq \Omega$. (Motivated by this view, we also refer to objects as states.) Letting A be the $|\Omega^*| \times |\Omega^*|$ submatrix of P that corresponds to transitions from Ω^* to Ω^*, and B the submatrix that corresponds to transitions from Ω^* to $\Omega \setminus \Omega^*$, we see that, after a suitable permutation of its rows and columns, P can be written as

$$P = \left[\begin{array}{c|c} A & B \\ \hline 0 & I \end{array}\right] \quad .$$

Here I is the identity matrix, since we assume that the particle stops after reaching a flawless state.

Let $\theta = [\theta_1 \mid \theta_2]$ be the row vector that corresponds to the probability distribution of the starting state, where θ_1 and θ_2 are the vectors that correspond to states in Ω^* and $\Omega \setminus \Omega^*$, respectively. Then, the probability that after t steps the particle is still inside Ω^* is exactly $\|\theta_1 A^t\|_1$. Therefore, for any initial distribution θ, the particle escapes Ω^* if and only if the spectral radius, $\rho(A)$, of A is strictly less than 1. Moreover, the rate of convergence is dictated by $1 - \rho(A)$. Unfortunately, since A is huge and defined implicitly by an algorithm, the magnitude of its largest eigenvalue, $\rho(A)$, is not readily available.

In linear systems analysis, to sidestep the inaccessibility of the spectral radius, $\rho(A)$, one typically bounds instead some *operator norm* $\|\cdot\|$ of the matrix A, since $\rho(A) \leq \|A\|$ for any such norm. (For brief background on matrix norms see Appendix 8.1.) Moreover, instead of bounding an operator norm of A itself, one often first performs a "change of basis" $A' = MAM^{-1}$ and bounds $\|A'\|$, justified by the fact that $\rho(A) = \rho(A') \leq \|A'\|$, for any invertible matrix M. The purpose of the change of basis is to cast A "in a good light" in the eyes of the chosen operator norm, in the hope of minimizing the cost of replacing the spectral norm with an operator norm. To demonstrate this approach in action, we start by showing how it captures the classical *potential function argument*.

Consider any function ϕ on Ω such that $\phi(\sigma) > 0$ for $\sigma \in \Omega^*$, while $\phi(\sigma) = 0$ for $\sigma \notin \Omega^*$. In our k-SAT example, $\phi(\sigma)$ could be the number of violated clauses under σ. The potential argument asserts that eventually $\phi = 0$ (i.e., the particle escapes Ω^*) if ϕ is always reduced in expectation, i.e., if for every $\sigma \in \Omega^*$,

$$\sum_{\sigma' \in \Omega} P[\sigma, \sigma']\phi(\sigma') < \phi(\sigma) \quad . \tag{4.1}$$

To express this argument via matrix norms, let $A' = MAM^{-1}$ where M is the diagonal $|\Omega^*| \times |\Omega^*|$ matrix $\mathrm{diag}(1/\phi(\sigma))$. Thus, $A'[\sigma, \sigma'] = A[\sigma, \sigma']\phi(\sigma')/\phi(\sigma)$. Recalling that $\|\cdot\|_\infty$ is the maximum row sum of a matrix, we see that the potential argument's condition (4.1) is nothing other than $\|A'\|_\infty < 1$.

Our starting point is the observation that all entropy compression arguments, and indeed all arguments in the algorithmic LLL literature, can be seen as *dual* to the potential function argument. That is, after a suitable change of basis $A' = MAM^{-1}$, they bound not $\|A'\|_\infty$, as the potential argument, but the dual norm $\|A'\|_1$. As a concrete demonstration, let us consider the Moser-Tardos algorithm for a k-CNF formula on n variables with clauses $c_1, \ldots, c_m$, under the uniform

[1]The framework does not require the state evolution to be Markovian, but we make this assumption here to simplify exposition.

measure on $\Omega = \{0,1\}^n$. For simplicity, assume that the lowest indexed violated clause is always resampled, so that the state evolves as a Markov chain.

For each clause c_i, let A_i be the $|\Omega^*| \times |\Omega^*|$ submatrix of A comprising all rows (states) where the resampled clause is c_i. (All other rows of A_i are 0). For $t \geq 1$, let $\mathcal{W}_t$ contain every t-sequence of (indices of) clauses that has non-zero probability of comprising the first t clauses resampled by the algorithm. In other words, $\mathcal{W}_t$ is the set of all t-sequences of indices from $[m]$ corresponding to non-vanishing t-products of matrices from $\{A_1, \ldots, A_m\}$, i.e., $\mathcal{W}_t = \{W = (w_i) \in [m]^t : \prod_{i=1}^t A_{w_i} \neq 0\}$. With these definitions, the first inequality below follows from the fact that $\rho(A) \leq \|A\|$ for any operator norm $\|\cdot\|$, the triangle inequality gives the second inequality and, crucially, the submultiplicativity of operator norms gives the third:

$$\rho(A)^t = \rho(A^t) \leq \left\|A^t\right\| = \left\| \left(\sum_{i \in [m]} A_i \right)^t \right\| = \left\| \sum_{W \in \mathcal{W}_t} \prod_{i=1}^t A_{w_i} \right\| \leq \sum_{W \in \mathcal{W}_t} \left\| \prod_{i=1}^t A_{w_i} \right\| \leq \sum_{W \in \mathcal{W}_t} \prod_{i=1}^t \|A_{w_i}\| . \tag{4.2}$$

Observe that (4.2) holds for every operator norm. To get a favorable bound here, we will apply (4.2) with the norm $\|\cdot\|_1$, i.e., the maximum column sum. We see that for all $j \in [m]$, every column of A_j has at most one non-zero entry, since $A_j(\sigma, \sigma') > 0$ only if σ is the mutation of σ' so that c_j is violated. Recalling that all non-zero entries of A equal 2^{-k}, we conclude $\|A_j\|_1 = 2^{-k}$ for all $j \in [m]$. Therefore, $\|A^t\|_1 \leq |\mathcal{W}_t| 2^{-kt}$. To bound $|\mathcal{W}_t|$ we use a simple necessary condition for membership in $\mathcal{W}_t$ which, by a standard counting argument, implies that if each clause shares variables with at most Δ other clauses, then $|\mathcal{W}_t| \leq 2^m (\mathrm{e}\Delta)^t$. Therefore, $\rho(A)^t \leq 2^m (\mathrm{e}\Delta 2^{-k})^t$ implying that if $\Delta < 2^k/\mathrm{e}$, then $1 > \|A\|_1 \geq \rho(A)$ and the algorithm terminates within $O(m)$ steps with high probability.

A very similar argument can be used to capture even the most general existing versions of the algorithmic LLL [3, 57, 4], which are described by arbitrary flaws and, for each flaw f_i, an arbitrary corresponding transition matrix A_i for addressing the flaw. Note that (4.2) is in essence a weighted counting of witness sequences, the weight of each sequence being the product of the norms $\|A_{w_i}\|_1$. Observe also that in our k-SAT example above, the only probabilistic notion was the transition matrix A and we did not make any implicit or explicit reference to a probability measure μ. To cast general algorithmic LLL arguments in this same form, any measure μ is incorporated as a *change of basis* for the transition matrix A, i.e., we bound $\|A'\|_1 = \|MAM^{-1}\|_1$ as $\sum_{W \in \mathcal{W}_t} \prod_{i=1}^t \|MA_{w_i}M^{-1}\|_1$, where M is the diagonal $|\Omega^*| \times |\Omega^*|$ matrix $\mathrm{diag}(\mu(\sigma))$, similarly to the potential function argument. (Recall definition (3.5) and note that $\|MA_{w_i}M^{-1}\|_1 = \gamma_{w_i}$.) We thus see that the measure μ is nothing other than a tool for analyzing the progress of the algorithm.

4.2 Point to Set Correlations

As we saw in Chapter 3, Moser and Tardos [81] showed that a simple local search algorithm can be used to make the LLL constructive for product probability spaces, and following this work, a large amount of effort has been devoted to making different variants of the LLL constructive [70, 69, 22, 88], and to analyzing sophisticated *resampling* algorithms that extend the Moser-Tardos techniques

to non-product probability spaces [55, 3, 57, 6, 4, 76, 60]. Indeed, intimate connections have been established between resampling algorithms and the LLL (see Section 3.4.2 and [57, 4, 60].)

Moreover, and as we already discussed in Section 3.2, the entropy compression method has been used to analyze *backtracking* algorithms, e.g., [47, 34, 38, 45, 46, 84], which recall that is a class of local search algorithms that have a very different flavor from resampling algorithms: they operate on *partial non-violating* assignments, starting with the empty assignment, and try to extend to a complete, satisfying assignment. To do this, at each step they assign a (random) value to a currently unassigned variable; if this leads to the violation of one or more constraints, they backtrack to a partial non-violating assignment by unassigning some set of variables (typically including the last assigned variable).

While there have been efforts to treat certain classes of backtracking algorithms systematically [38, 46], the analysis of such algorithms via the entropy compression method in general requires ad hoc technical machinery. Moreover, before the work of Achlioptas, Iliopoulos and Sinclair [5] there was no known connection between backtracking algorithms and any known LLL condition, either existential or algorithmic. The main reason for this is that backtracking steps induce non-trivial correlations among bad events, which typically result in very dense dependency graphs that are not amenable to currently known LLL conditions.

The main byproduct of the linear algebraic viewpoint we presented in the previous section is a new algorithmic LLL condition for analyzing *hybrid* algorithms, i.e., algorithms that (potentially) use both resampling and backtracking steps. Such algorithms combine the advantages of both approaches by using resampling to explore the state space, while detecting and backing away from unfavorable regions using backtracking steps. Unlike all past work, this new condition replaces the notion of a dependency or causality graph by quantifying *point-to-set correlations* between bad events. Notably, it captures the most general algorithmic LLL condition known so far, i.e., the one corresponding to Theorem 3.15, and moreover, unifies the analysis of all entropy compression applications, connecting backtracking algorithms to the LLL in the same fashion that the analyses we described in Chapter 3 connect resampling algorithms to the LLL.

More concretely now, recall our discussion in Section 3.4.1 where, given a stochastic local search algorithm and a probability distribution μ over its state space, we defined the charge of a flaw f_i to be its probability with respect to μ times a distortion factor that captures the compatibility between μ and the algorithm. In particular, recall that "distortion" is a worst-case notion, and that when designing stochastic local search algorithms there is a trade-off between the number of flaws each flaw can possibly introduce when the algorithm addresses it, and minimizing distortion. The algorithmic LLL condition of [5] allows for a more refined way of capturing the compatibility between μ and the algorithm by defining a set of charges, $\{\gamma_i^S\}_{(i,S)\in[m]\times 2^{[m]}}$, one for each pair of a flaw f_i and a set of (indices of) flaws S, such that $\gamma_i^S = 0$ whenever $S \in [m] \setminus \Gamma(i)$. From a technical standpoint, and recalling the discussion in Section 4.1, this is a consequence of an (overlapping) decomposition of matrix A into a sum of exponentially many matrices, one for each (i, S) pair, which enables us to perform a more detailed analysis.

We informally discuss the condition of [5] forthwith, and give the full details in Section 4.3.

4.2.1 Informal Discussion

Recall that we say that a transition $\sigma \to \tau$, made to address flaw f_i, introduces flaw f_j if $\tau \in f_j$ and either $\sigma \notin f_j$, or $j = i$. For an arbitrary state $\tau \in \Omega$, flaw f_i, and set of flaws S, let

$$\mathrm{In}_i^S(\tau) := \{\sigma \in f_i : \text{the set of flaws introduced by the transition } \sigma \to \tau \text{ includes } S\} \ . \tag{4.3}$$

For any fixed probability distribution $\mu > 0$ on Ω, we define the *charge* of the pair (i, S) with respect to μ to be

$$\gamma_i^S := \max_{\tau \in \Omega} \left\{ \frac{1}{\mu(\tau)} \sum_{\sigma \in \mathrm{In}_i^S(\tau)} \mu(\sigma)\rho_i(\sigma, \tau) \right\} \ . \tag{4.4}$$

That is, the charge γ_i^S is an upper bound on the ratio between the ergodic flow into a state that arrives through transitions that introduce every flaw in S (and perhaps more), and the probability of the state under μ.

The condition of [5] may now be stated informally as follows:

Theorem 4.1. *If there exist positive real numbers $\{\psi_i\}_{i=1}^m$ such that for all $i \in [m]$,*

$$\frac{1}{\psi_i} \sum_{S \subseteq [m]} \gamma_i^S \prod_{j \in S} \psi_j \ < 1 \ , \tag{4.5}$$

then a local search algorithm reaches a flawless object quickly with high probability.

The phrase "quickly with high probability" essentially means that the running time has expectation linear in the number of flaws and an exponential tail; we spell this out more formally in Section 4.3.

A key feature of Theorem 4.1 is the absence of a causality/dependency graph, present in all previous LLL conditions. This is because considering point-to-set correlations, i.e., how each flaw interacts with every other set of flaws, frees us from the traditional hard view of dependencies between individual events. In this new condition, every flaw may interact with every other flaw, as long as the interactions are sufficiently weak. Notably, this is achieved without any compromise in the traditional setting of a causality/dependency graph. To see this, note that if S contains any flaw that is never introduced by addressing flaw f_i, then $\gamma_i^S = 0$. Thus, in the presence of a causality/dependency graph, the only terms contributing to the summation in (4.5) are those that correspond to subsets of the graph neighborhood of flaw f_i, recovering the traditional setting.

Besides relaxing the traditional notion of dependence, condition (4.5) is also quantitatively more powerful, even in the traditional setting. To get a feeling for this, observe that the previously most powerful algorithmic LLL condition, i.e., Theorem 3.15, can be derived from (4.5) by replacing γ_i^S by $\gamma_i^\emptyset$ for every S (and restricting S to subsets of the neighborhood of flaw f_i, as discussed in the previous paragraph). Indeed, recall Definition 3.5 and observe that $\gamma_i = \gamma_i^\emptyset$. Note though that since the charges γ_i^S are decreasing in S, replacing $\gamma_i^\emptyset$ with γ_i^S can lead to significant improvement. For example, if the flaws in S are never introduced simultaneously when addressing flaw f_i, then $\gamma_i^S = 0$ and S does not contribute to the sum in (4.5); in contrast, S contributes $\gamma_i^\emptyset$, i.e., the maximum possible charge, to the corresponding sum in [4].

A natural question is whether Theorem 4.1 can be improved by replacing the word "includes" with the word "equals" in (4.3), thus shrinking the sets $\mathrm{In}_i^S(\tau)$. The short answer is "No," i.e.,

such a change invalidates the theorem. The reason for this is that in resampling algorithms we must allow for the possibility that flaws introduced when addressing f_i may later be fixed "collaterally," i.e., as the result of addressing other flaws rather than by being specifically addressed by the algorithm. While it may seem that such collateral fixes cannot possibly be detrimental, they are problematic from an analysis perspective as they can potentially increase the intensity of correlations between flaw f_i and S. Perhaps more convincingly, tracking collateral fixes and taking them into account also appears to be a bad idea in practice [96, 97, 12, 13]: for example, local search satisfiability algorithms which select which variable to flip (among those in the targeted violated clause) based *only* on which clauses will become violated, fare much better than algorithms that weigh this damage against the benefit of the collaterally fixed clauses.

Motivated by the above considerations, Achlioptas, Iliopoulos and Sinclair introduce the notion of *primary* flaws. These are flaws which, once present, can only be eradicated by being addressed by the algorithm, i.e., they cannot be fixed collaterally. Primary flaws allow us to change the definition of the sets $\mathrm{In}_i^S(\tau)$ in the desired direction. Specifically, say that a set of flaws T *covers* a set of flaws S if:

1. the set of primary flaws in T *equals* the set of primary flaws in S; and
2. the set of non-primary flaws in T *includes* the set of non-primary flaws in S.

In other words, we demand equality at least for the primary flaws.

Theorem 4.2. *Theorem 4.1 continues to hold if in* (4.3) *"includes" is replaced by "covers".*

The notion of primary flaws is one of the main conceptual contributions of [5]. Crucially for the applications, backtracking steps always introduce only primary flaws and thus, for such steps, we achieve an ideal level of control. The full version of our new algorithmic LLL condition, incorporating primary flaws, is spelled out formally in Theorem 4.6 in the next section.

4.3 A New Algorithmic LLL Condition

In this section we state the main result of this chapter, which is a formal version of Theorem 4.2 discussed in Section 4.2.1.

Recall that θ denotes the probability distribution of the starting state and that we denote by $\mathrm{Span}(\theta)$ the set of flaw indices that may be present in the initial state, i.e., $\mathrm{Span}(\theta) = \bigcup_{\sigma\in\Omega:\theta(\sigma)>0} U(\sigma)$.

Let π be an arbitrary permutation over $[m]$. We say that an algorithm follows the π-*strategy* if at each step it picks to address the flaw corresponding to the element of $U(\sigma)$ of lowest index according to π.

We now formalize the definitions of primary flaws and charges introduced informally in the previous section.

Definition 4.3. *A flaw f_i is* primary *if for every $\sigma \in f_i$ and every $j \neq i$, addressing f_j at σ always results in some $\tau \in f_i$, i.e., f_i is never eradicated collaterally. For a given set $S \subseteq [m]$, we write S^P and S^N to denote the indices that correspond to primary and non-primary flaws in S, respectively.*

Definition 4.4. *We say that a set of flaws T* covers *a set of flaws S if $T^P = S^P$ and $T^N \supseteq S^N$.*

Definition 4.5. *For a state $\tau \in \Omega$, flaw f_i, and set of flaws S, let*

$$\mathrm{In}_i^S(\tau) = \{\sigma \in f_i : \textit{the set of flaws introduced by the transition } \sigma \to \tau \textit{ covers } S\} \ .$$

Let $\mu > 0$ be an arbitrary measure on Ω. For every $i \in [m]$ and $S \subseteq [m]$, the charge *of (i, S) with respect to μ is,*

$$\gamma_i^S = \max_{\tau \in \Omega} \left\{ \frac{1}{\mu(\tau)} \sum_{\sigma \in \mathrm{In}_i^S(\tau)} \mu(\sigma)\rho_i(\sigma, \tau) \right\} \ . \tag{4.6}$$

We now state the formal version of the new algorithmic LLL condition, Theorem 4.1 of the previous section.

Theorem 4.6. *If there exist positive real numbers $\{\psi_i\}_{i\in[m]}$ such that for every $i \in [m]$,*

$$\zeta_i := \frac{1}{\psi_i} \sum_{S \subseteq [m]} \gamma_i^S \prod_{j \in S} \psi_j \ < 1 \ , \tag{4.7}$$

then, for every permutation π over $[m]$, the probability that an algorithm following the π-strategy fails to reach a flawless state within $(T_0 + s)/\log_2(1-\epsilon)$ steps is 2^{-s}, where $\epsilon = 1 - \max_{i\in[m]} \zeta_i$, and

$$T_0 = \log_2 \mu_{\min}^{-1} + m \log_2 \left(\frac{1 + \psi_{\max}}{\psi_{\min}} \right) \ ,$$

with $\mu_{\min} = \min_{\sigma\in\Omega} \mu(\sigma)$, $\psi_{\max} = \max_{i\in[m]} \psi_i$ and $\psi_{\min} = \min_{i\in[m]} \psi_i$.

Remark 4.7. *In typical applications $\mu, \{\psi_i\}_{i\in[m]}$ are such that $T_0 = O(\log|\Omega| + m)$ and the sum in* (4.7) *is easily computable, as $\gamma_i^S = 0$ for the vast majority of subsets S.*

Remark 4.8. *For any fixed permutation π, the charges γ_i^S can be reduced by removing from $\mathrm{In}_i^S(\tau)$ every state for which i is not the lowest indexed element of $U(\sigma)$ according to π.*

Remark 4.9. *Theorem 4.6 holds also for algorithms using flaw choice strategies other than π-strategies. We discuss some such strategies in Section 4.4.4. However, there is good reason to expect that it does* not *hold for* arbitrary *flaw choice strategies (see Chapter 5 and [71]).*

Finally, we state a refinement of our running time bound that will be important in order to get the best convergence guarantees in the applications of pure backtracking algorithms in Section 4.5.

Remark 4.10. *The upper bound on T_0 in Theorem 4.6 can be replaced by the more refined bound:*

$$T_0 = \log_2 \left(\max_{\sigma\in\Omega} \frac{\theta(\sigma)}{\mu(\sigma)} \right) + \log_2 \left(\sum_{S \subseteq \mathrm{Span}(\theta)} \prod_{j\in S} \psi_j \right) + \log_2 \left(\max_{S\subseteq[m]} \frac{1}{\prod_{j\in S} \psi_j} \right) \ .$$

Moreover, if (as in pure backtracking algorithms) every flaw is primary, and (as is typical in pure backtracking algorithms) every flaw is present in the initial state, and if $\psi_i \in (0, 1]$ for all i, then $T_0 = \log_2 \mu_{\min}^{-1}$.

4.4 Proof of Theorem 4.6

In Sections 4.4.2 and 4.4.3 we present the proof of Theorem 4.6. In Section 4.4.4 we show how to extend the theorem to allow flaw choice strategies other than following a fixed permutation over flaws.

Throughout this section we use standard facts about operator norms, summarized briefly in Appendix 8.1.

4.4.1 Charges as Norms of Transition Matrices

We will first show how charges can be seen as the norms of certain transition matrices.

Recall that for any $S \subseteq [m]$, we denote by S^P and S^N the subsets of S that correspond to primary and non-primary flaws, respectively.

Definition 4.11. *For every $i \in [m]$ and every set of flaw indices $S \subseteq [m]$, let A_i^S be the $|\Omega| \times |\Omega|$ matrix where $A_i^S[\sigma, \tau] = \rho_i(\sigma, \tau)$ if the set of flaws introduced by $\sigma \to \tau$ covers S, i.e., the set of primary flaws introduced by the transition $\sigma \to \tau$* equals *S^P and the set of non-primary flaws introduced by $\sigma \to \tau$ contains S^N; otherwise $A_i^S[\sigma, \tau] = 0$.*

Let $\|\cdot\|_1$ denote the matrix norm induced by the L^1-vector-norm, and recall that it is equal to the max column sum. Let also $M = \mathrm{diag}(\mu(\sigma))$ denote the $|\Omega| \times |\Omega|$ diagonal matrix whose entries correspond to the probability measure μ. Our key observation is that the charges γ_i^S introduced in (4.6) can be expressed as

$$\gamma_i^S = \|M A_i^S M^{-1}\|_1 \ . \tag{4.8}$$

The reader is encouraged to verify this equivalence, which is an immediate consequence of the definitions.

Remark 4.12. *Although we are specializing here to the $\|\cdot\|_1$ and matrix $M = \mathrm{diag}(\mu(\sigma))$, Theorem 4.6 holds for any choice of matrix norm and invertible matrix M. It is an interesting research direction whether using other norms can be useful in applications.*

4.4.2 Tracking the Set of Current Flaws

We say that a trajectory $\Sigma = (\sigma_1, \sigma_2, \ldots, \sigma_{t+1})$ followed by the algorithm is a *bad t-trajectory* if every state σ_i, $i \in [t+1]$, is flawed. Thus, our goal is to bound the probability that the algorithm follows a bad t-trajectory.

Given a bad trajectory, intuitively, we track the flaws introduced into the state in each step, where a flaw is said to "introduce itself" whenever addressing it fails to remove it. Of the flaws introduced in each step, we disregard those that later get eradicated collaterally, i.e., by an action addressing some other flaw. The rest form the "witness sequence" of the trajectory, i.e., a sequence of sets of flaws.

Fix any permutation π on $[m]$. For any $S \subseteq [m]$, let $\pi(S) = \min_{j \in S} \pi(j)$, i.e., the lowest index in S according to π. Recalling that $U(\sigma)$ is the set of indices of flaws present in σ, in the following we assume that the index of the flaw addressed in state σ is $\pi(U(\sigma))$, which we sometimes abbreviate as $\pi(\sigma)$. Also, to lighten notation, we will denote $A \setminus \{\pi(B)\}$ by $A - \pi(B)$.

Definition 4.13. *Let $\Sigma = (\sigma_1, \sigma_2, \ldots, \sigma_{t+1})$ be any bad t-trajectory. Let $B_0 = U(\sigma_1)$. For $1 \leq i \leq t$, let*

$$B_i = U(\sigma_{i+1}) \setminus [U(\sigma_i) - \pi(\sigma_i)] \ ,$$

i.e., B_i comprises the indices of the flaws introduced in the i-th step. For $0 \leq i \leq t$, let

$$C_i = \{k \in B_i \mid \exists j \in [i+1, t] : k \notin U(\sigma_{j+1}) \wedge \forall \ell \in [i+1, j] : k \neq \pi(\sigma_\ell)\} \ ,$$

i.e., C_i comprises the indices of the flaws introduced in the i-th step that get eradicated collaterally. The witness sequence *of bad t-trajectory Σ is the sequence of sets*

$$w(\Sigma) = (B_0 \setminus C_0, B_1 \setminus C_1, \ldots, B_t \setminus C_t) \ .$$

A crucial feature of witness sequences is that they allow us to recover the sequence of flaws addressed.

Definition 4.14. *Given an* arbitrary *sequence $S_0, \ldots, S_t$, let $S_1^* = S_0$, while for $1 \leq i \leq t$, let*

$$S_{i+1}^* = \begin{cases} [S_i^* - \pi(S_i^*)] \cup S_i & \text{if } S_i^* \neq \emptyset \ , \\ \emptyset & \text{otherwise} \ . \end{cases}$$

If $S_i^* \neq \emptyset$ for all $1 \leq i \leq t$, then we say that $(S_i)_{i=0}^t$ is plausible *and write $\pi(S_i^*) = (i)$.*

Lemma 4.15. *If $\Sigma = (\sigma_1, \sigma_2, \ldots, \sigma_{t+1})$ is any bad t-trajectory, then $w(\Sigma) = (S_0, \ldots, S_t)$ is plausible, $\pi(\sigma_i) = \pi(S_i^*) = (i)$ for all $1 \leq i \leq t$, and for every flaw index $z \in [m]$, the number of times z occurs in the multiset $\bigcup_{i=0}^t S_i$ minus the number of times it occurs in the multiset $\bigcup_{i=1}^t (i)$ equals $\mathbf{1}_{z \in S_{t+1}^*}$.*

Proof. Recall that $S_i = B_i \setminus C_i$. For $1 \leq i \leq t+1$, let L_i comprise the elements of $U(\sigma_i)$ eradicated collaterally during the i-th step and let H_i comprise the elements of $U(\sigma_i)$ eradicated collaterally during any step $j \geq i$. Observe that $H_{i+1} = (H_i \setminus L_i) \cup C_i$. We will prove, by induction, that for all $1 \leq i \leq t+1$,

$$S_i^* \subseteq U(\sigma_i) \tag{4.9}$$

$$U(\sigma_i) \setminus S_i^* = H_i \ . \tag{4.10}$$

Observe that if (4.9), (4.10) hold for a given i, then $\pi(\sigma_i) = \pi(S_i^*)$, since $\pi(\sigma_i) \notin H_i$ by the definition of H_i, and $\pi(A) = \pi(A \setminus B)$ whenever $\pi(A) \notin B$. Moreover, $S_i^* \neq \emptyset$, because otherwise $U(\sigma_i) = H_i$, an impossibility. To complete the proof it suffices to note that for any $z \in [m]$, the difference in question equals $\mathbf{1}_{z \in U(\sigma_{t+1})}$ and that $U(\sigma_{t+1}) = S_{t+1}^*$ since, by definition, $H_{t+1} = \emptyset$. The inductive proof is as follows.

For $i = 1$, (4.9), (4.10) hold since $S_1^* = B_0 \setminus C_0$, while $U(\sigma_1) = B_0$. If (4.9), (4.10) hold for some $i \geq 1$, then $S_{i+1}^* = [S_i^* - \pi(\sigma_i)] \cup S_i$ while, by definition, $U(\sigma_{i+1}) = [(U(\sigma_i) - \pi(\sigma_i)) \setminus L_i] \cup B_i$. Thus, the fact that $S_i^* \subseteq U(\sigma_i)$ trivially implies $S_{i+1}^* \subseteq U(\sigma_{i+1})$, while

$$U(\sigma_{i+1}) \setminus S_{i+1}^* = ((U(\sigma_i) \setminus S_i^*) \setminus L_i) \cup (B_i \setminus S_i) = (H_i \setminus L_i) \cup C_i = H_{i+1} \ .$$

□

The first step in our proof of Theorem 4.6 is to give an upper bound on the probability that a given witness sequence occurs in terms of the charges γ_i^S. In particular, and in order to justify Remark 4.8, we will use an arbitrary norm $\|\cdot\|$ and invertible matrix M.

Recall that $\|\cdot\|_*$ denotes the *dual* of norm $\|\cdot\|$ and let $\theta^\top \in [0,1]^{|\Omega|}$ denote the row vector expressing the probability distribution of the initial state σ_1. Moreover, for a state σ, let e_σ denote the indicator vector of σ, i.e., $e_\sigma[\sigma] = 1$ and $e_\sigma[\tau] = 0$ for all $\tau \in \Omega \setminus \{\sigma\}$.

Lemma 4.16. *Fix any integer $t \geq 0$ and let Σ be the random variable $(\sigma_1, \ldots, \sigma_{t+1})$. Fix any arbitrary invertible matrix M and operator norm $\|\cdot\|$, and let $\lambda_i^S = \|MA_i^S M^{-1}\|$. For any plausible sequence $\phi = (S_0, \ldots, S_t)$,*

$$\Pr[w(\Sigma) = \phi] \leq \|\theta^\top M^{-1}\|_* \left(\sum_{\tau\in\Omega} \|Me_\tau\|\right) \prod_{i=1}^{t} \lambda_{(i)}^{S_i} \ . \tag{4.11}$$

Proof. By Definition 4.13 and Lemma 4.15, a necessary condition for $w(\Sigma) = \phi$ to occur is that $(i) \in U(\sigma_i)$ and $S_i \subseteq B_i$, for every $1 \leq i \leq t$.

Recall that for any $S \subseteq [m]$, we denote by S^P and S^N the subsets of S that correspond to primary and non-primary flaws, respectively. By Definition 4.13 and Lemma 4.15, a necessary condition for $w(\Sigma) = \phi$ to occur is that $(i) \in U(\sigma_i)$ and $S_i \subseteq B_i$, for every $1 \leq i \leq t$. Moreover, since primary flaws are never eradicated collaterally, i.e., $C_i^P = \emptyset$ always, it must also be that $S_i^P = B_i^P$ for $1 \leq i \leq t$. Fix any state $\tau \in \Omega$. The probability that $(1) \in U(\sigma_1) \wedge S_1^P = B_1^P(\Sigma) \wedge S_1^N \subseteq B_1^N(\Sigma) \wedge \sigma_2 = \tau$ equals the τ-column (coordinate) of the row-vector $\theta^\top A_{(1)}^{S_1}$. More generally, we see that for any $t \geq 1$,

$$\Pr\left[\bigwedge_{i=1}^{t} ((i) \in U(\sigma_i)) \bigwedge_{i=1}^{t} \left(S_i^P = B_i^P\right) \bigwedge_{i=1}^{t} \left(S_i^N \subseteq B_i^N\right) \bigwedge \sigma_{t+1} = \tau\right] = \theta^\top \prod_{i=1}^{t} A_{(i)}^{S_i} e_\tau \ . \tag{4.12}$$

Consider now any vector norm $\|\cdot\|$ and the corresponding operator norm. By (8.1),

$$\theta^\top \prod_{i=1}^{t} A_{(i)}^{S_i} e_\tau = \theta^\top M^{-1} \left(\prod_{i=1}^{t} MA_{(i)}^{S_i} M^{-1}\right) Me_\tau \leq \left\|\theta^\top M^{-1} \left(\prod_{i=1}^{t} MA_{(i)}^{S_i} M^{-1}\right)\right\|_* \|Me_\tau\| \ . \tag{4.13}$$

Summing (4.13) over all $\tau \in \Omega$ we conclude that

$$\Pr[w(\Sigma) = \phi] = \sum_{\tau\in\Omega} \Pr[w(\Sigma) = \phi \wedge \sigma_{t+1} = \tau] \leq \left\|\theta^\top M^{-1} \prod_{i=1}^{t} MA_{(i)}^{S_i} M^{-1}\right\|_* \sum_{\tau\in\Omega} \|Me_\tau\| \ . \tag{4.14}$$

Applying (8.3) and then (8.2) to (4.14) and recalling the definition of $\lambda_{(i)}^{S_i}$ we conclude that

$$\Pr[w(\Sigma) = \phi] \leq \|\theta^\top M^{-1}\|_* \left(\sum_{\tau\in\Omega} \|Me_\tau\|\right) \prod_{i=1}^{t} \|MA_{(i)}^{S_i} M^{-1}\| = \|\theta^\top M^{-1}\|_* \left(\sum_{\tau\in\Omega} \|Me_\tau\|\right) \prod_{i=1}^{t} \lambda_{(i)}^{S_i} \ ,$$

as claimed. □

Let $\mathcal{F}_t = \{w(\Sigma) : \Sigma \text{ is a bad } t\text{-trajectory of the algorithm}\}$. Since $\mathcal{F}_t$ contains only plausible sequences, an immediate corollary of Lemma 4.16 is a bound on the probability that the algorithm fails in t steps.

Corollary 4.17. *The probability that the algorithm fails to reach a flawless state within t steps is at most*

$$\left(\max_{\sigma\in\Omega}\frac{\theta(\sigma)}{\mu(\sigma)}\right)\cdot\sum_{\phi\in\mathcal{F}_t}\prod_{i=1}^{t}\gamma_{(i)}^{S_i} \ . \tag{4.15}$$

Proof. We apply Lemma 4.16 with $M = \mathrm{diag}(\mu(\sigma))$ and the $\|\cdot\|_1$-norm. The proof is concluded by noticing that the dual norm of $\|\cdot\|_1$ is $\|\cdot\|_\infty$ and, thus, $\|\theta^\top M^{-1}\|_\infty = \max_{\sigma\in\Omega}\frac{\theta(\sigma)}{\mu(\sigma)}$, and that $\sum_{\tau\in\Omega}\|Me_\tau\|_1 = 1$.

□

Thus, to complete the proof of Theorem 4.6 we are left with the task of bounding the sum in (4.15).

4.4.3 Bounding the Sum

Given $\psi_1,\ldots,\psi_m > 0$ and $S\subseteq[m]$, let $\Psi(S) = \prod_{j\in S}\psi_j$, with $\Psi(\emptyset) = 1$. For each $i\in[m]$, let

$$\zeta_i = \frac{1}{\psi_i}\sum_{S\subseteq[m]}\gamma_i^S\Psi(S) \ .$$

Finally, for each $i\in[m]$ consider the probability distribution on $2^{[m]}$ assigning to each $S\subseteq[m]$ probability

$$p(i,S) = \frac{\gamma_i^S\Psi(S)}{\sum_{S\subseteq[m]}\gamma_i^S\Psi(S)} = \frac{\gamma_i^S\Psi(S)}{\zeta_i\psi_i} \ .$$

For any $S_0\subseteq[m]$, let $\mathcal{F}_t(S_0)$ comprise the witness sequences in $\mathcal{F}_t$ whose first set is S_0. Consider the probability distribution on sequences of subsets of $[m]$ generated as follows: $R_1 = S_0$; for $i\geq 1$, if $R_i\neq\emptyset$, then $R_{i+1} = (R_i - \pi(R_i))\cup S_i$, where $\Pr[S_i = S] = p(\pi(R_i),S)$, for any $S\subseteq[m]$. Under this distribution, by Lemma 4.15, each $\phi = (S_0,\ldots,S_t)\in\mathcal{F}_t(S_0)$ receives probability $p_\phi = \prod_{i=1}^t p((i),S_i)$, while $\sum_{\phi\in\mathcal{F}_t(S_0)}p_\phi\leq 1$. At the same time, by the last claim in Lemma 4.15,

$$p_\phi = \prod_{i=1}^{t}p((i),S_i) = \left(\prod_{i=1}^{t}p((i),S_i)\frac{\psi_{(i)}}{\Psi(S_i)}\right)\frac{\Psi(S_{t+1}^*)}{\Psi(S_0)} = \frac{\Psi(S_{t+1}^*)}{\Psi(S_0)}\prod_{i=1}^{t}\frac{\gamma_{(i)}^{S_i}}{\zeta_{(i)}} \ . \tag{4.16}$$

Combining (4.16) with the fact $\sum_{\phi\in\mathcal{F}_t(S_0)}p_\phi\leq 1$ it follows that

$$\sum_{\phi\in\mathcal{F}_t(S_0)}\prod_{i=1}^{t}\frac{\gamma_{(i)}^{S_i}}{\zeta_{(i)}}\leq\max_{S\subseteq[m]}\frac{\Psi(S_0)}{\Psi(S)} \ . \tag{4.17}$$

Let $\zeta = \max_{i\in[m]} \zeta_i$. Then, summing equation (4.17) over all possible sets S_0 yields

$$\sum_{\phi\in\mathcal{F}_t} \prod_{i=1}^{t} \gamma_{(i)}^{S_i} = \sum_{S_0\subseteq \mathrm{Span}(\theta)} \sum_{\phi\in\mathcal{F}_t(S_0)} \prod_{i=1}^{t} \gamma_{(i)}^{S_i} \leq \zeta^t \sum_{S_0\subseteq \mathrm{Span}(\theta)} \sum_{\phi\in\mathcal{F}_t(S_0)} \prod_{i=1}^{t} \frac{\gamma_{(i)}^{S_i}}{\zeta_{(i)}} \leq \max_{S\subseteq[m]} \sum_{S_0\subseteq \mathrm{Span}(\theta)} \frac{\Psi(S_0)}{\Psi(S)} \ . \tag{4.18}$$

Proofs of Theorem 4.6 and Remark 4.10 . Combining (4.18) with Corollary 4.17 we see that the binary logarithm of the probability that the algorithm does not encounter a flawless state within t steps is at most $t \log_2 \zeta + T_0$, where

$$T_0 \ = \ \log_2 \left(\max_{\sigma\in\Omega} \frac{\theta(\sigma)}{\mu(\sigma)} \right) + \log_2 \left(\sum_{S\subseteq \mathrm{Span}(\theta)} \Psi(S) \right) + \log_2 \left(\max_{S\subseteq[m]} \frac{1}{\Psi(S)} \right) \ .$$

Therefore, if $t = (T_0 + s)/\log_2(1/\zeta) \leq (T_0 + s)/\delta$, the probability that the algorithm does not reach a flawless state within t steps is at most 2^{-s}. This concludes the proofs of the first part of Remark 4.10 and Theorem 4.6 since $\max_{\sigma\in\Omega} \theta(\sigma) \leq 1$ and

$$\log_2 \left(\sum_{S\subseteq \mathrm{Span}(\theta)} \Psi(S) \right) + \log_2 \left(\max_{S\subseteq[m]} \frac{1}{\Psi(S)} \right) \leq \log_2 \frac{\prod_{i=1}^{m} (1+\psi_i)}{(\psi_{\min})^m} \leq m \log_2 \left(\frac{1+\psi_{\max}}{\psi_{\min}} \right) \ .$$

To see the second part of Remark 4.10, let $\mathcal{I}(\theta)$ denote the set comprising the sets of flaw-indices that may be present in a state selected according to θ. Recall now that when every flaw is primary, the only equivalence classes of $\mathcal{F}_t$ that contribute to the sum in (4.18) are those for which $S_0 \in \mathcal{I}(\theta)$. Thus, for backtracking algorithms the sum over $S \subseteq \mathrm{Span}(\theta)$ in the definition of T_0 can be restricted to $S \in \mathcal{I}(\theta)$. Finally, if every flaw is always present in the initial state and $\psi_i \in (0, 1]$ for every $i \in [m]$, then $I(\theta) = \{F\}$ and $\log \left(\frac{1}{\max_{S\subseteq[m]} \prod_{j\in S} \psi_j} \right) = -\log_2 \prod_{j\in[m]} \psi_j$. This implies that the second and third term in the expression of T_0 in Remark 4.10 cancel out, concluding its proof.

□

4.4.4 Other Flaw Choice Strategies

The only place where we used the fact that the flaw choice is based on a fixed permutation was to assert, in Lemma 4.15, that the witness sequence of a trajectory determines the sequence of addressed flaws. Thus, our analysis is in fact valid for every flaw choice strategy that shares this property.

A first example of such a strategy is "pick a random occurring flaw and address it". To implement this, we can fix a priori an infinite sequence of uniformly random permutations $\pi_1, \pi_2, \ldots$ and at the i-th step address the lowest indexed flaw present according to π_i. It is straightforward to see that Lemma 4.15 still holds if we replace π with π_i therein and in Definition 4.14.

As a second example, consider the following recursive way to chose which flaw to address at each step (which makes the algorithm non-Markovian). The algorithm now maintains a stack. The flaws present in σ_1, ordered according to some permutation π, comprise the initial stack content.

The algorithm starts by addressing the flaw at the top of the stack, i.e., $\pi(\sigma_1)$, as before. Now, though, any flaws introduced in the i-th step, i.e., the elements of B_i, go on the top of the stack (ordered by π), while all eradicated flaws are removed from the stack. The algorithm terminates when the stack empties. It is not hard to see that, by taking S_0 to be the initial stack content, popping the flaw at the top of the stack at each step, and adding S_i to the top of the stack (ordered by π), the sequence of popped flaws is the sequence of addressed flaws.

4.5 Applications to Backtracking Algorithms

An important class of algorithms naturally devoid of "collateral fixes" are *backtracking* algorithms, as discussed in Section 4.2. In this section we give three representative applications of Theorem 4.6 to this family of algorithms.

In particular, consider a Constraint Satisfaction Problem (CSP) over a set of variables $V = \{v_1, \ldots, v_n\}$, each variable v_i taking values in a domain $\mathcal{D}_i$, with a set of constraints $\mathcal{C} = \{c_1, c_2, \ldots, c_m\}$ over these variables. The backtracking algorithms we consider operate as follows. (Note that in Step 1, we can always take θ to be the distribution under which all variables are unassigned; this does not affect the convergence condition (4.7) but may have a mild effect on the running time.)

Generic Backtracking Algorithm

1: Sample a partial non-violating assignment σ_0 according to a distribution θ and set $i = 0$
2: **while** unassigned variables exist **do**
3: Let v be the lowest indexed unassigned variable in σ_i
4: Choose a new value for v according to a state-dependent probability distribution
5: **if** one or more constraints are violated **then**
6: Remove the values from enough variables so that no constraint is violated
7: Let σ_{i+1} be the resulting assignment
8: $i \leftarrow i + 1$

Let Ω be the set of partial assignments to V that do not violate any constraint in $\mathcal{C}$. For each variable $v_i \in V$, let flaw $f_i \subseteq \Omega$ comprise the partial assignments in which v_i is unassigned. Clearly, each flaw f_i can only be removed by addressing it, as addressing any other flaw can only unassign v_i. Thus, every flaw is primary and a flawless state is a complete satisfying assignment.

Recently, Bissacot and Doin [20] showed that backtracking algorithms can make LLL applications in the variable setting constructive, using the entropy compression method. However, their result applies only to the uniform measure and their algorithms are relatively complicated. Here we show that condition (4.7) makes applications of the LLL in the variable setting [81], with any measure, constructive via a single, simple backtracking algorithm, i.e., an algorithm of very different flavor from the Moser-Tardos resampling algorithm. For example, in the case of k-SAT the algorithm takes the following form:

Randomized DPLL with single-clause backtracking

while unassigned variables exist **do**
  Assign to the lowest indexed unassigned variable x a value $v \in \{0,1\}$ with probability p_x^v
  if one or more clauses become violated **then**
    Unassign all k variables in the lowest indexed violated clause

We also show that applying our condition to the algorithm of Esperet and Parreau [38] for *acyclic edge coloring* recovers, in a simple, black-box fashion, the same bound of 4Δ as their highly non-trivial, problem-specific analysis via entropy compression, while guaranteeing an improved running time bound.

Finally, we make constructive in an effortless manner an existential result of Bernshteyn [16] showing improved bounds for the acyclic chromatic index of graphs that do not contain an arbitrary bipartite graph H.

4.5.1 The Variable Setting

In this section we show how we can use Theorem 4.6 to employ backtracking algorithms in order to capture applications in the variable setting, i.e., the setting considered by Moser and Tardos. In particular, we consider a product measure over variables V and define a bad event for each constraint $c \in \mathcal{C}$ being violated and show the following corollary of Theorem 4.6.

Theorem 4.18. *Let P be any product measure over a set of variables V and let A_c be the event that constraint c is violated. If there exist positive real numbers $\{\psi_v\}_{v\in V}$ such that for every variable $v \in V$,*

$$\frac{1}{\psi_v}\left(1+\sum_{c\ni v} P(A_c)\prod_{u\in c}\psi_u\right) < 1 \ , \tag{4.19}$$

then there exists a backtracking algorithm that finds a satisfying assignment after an expected number of $O\left(\log(P_{\min}^{-1}) + |V|\log_2\left(\frac{1+\psi_{\max}}{\psi_{\min}}\right)\right)$ steps.

We now use Theorem 4.18 to capture a well-known application of the Lovász Local Lemma to sparse k-SAT formulas when P is the uniform measure. For a k-SAT formula Φ we will be denoting its maximum degree as $\Delta \equiv \Delta(\Phi)$, i.e., each variable of Φ is contained in at most Δ clauses.

Theorem 4.19. *Every k-SAT formula Φ with maximum degree $\Delta < \frac{2^k}{\mathrm{e}k}$ is satisfiable. Moreover, there exists a backtracking algorithm that finds a satisfying assignment of Φ efficiently.*

Proof. Setting $\psi_v = \psi = 2\alpha > 0$ we see that it suffices to find a value $\alpha > 0$ such that

$$\frac{1}{\psi} + \frac{1}{2^k}\Delta\psi^{k-1} = \frac{1}{2\alpha} + \frac{1}{2}\Delta\alpha^{k-1} < 1 \ ,$$

which is feasible whenever

$$\Delta < \max_{\alpha>0}\frac{2\alpha-1}{\alpha^k} = \frac{2^k}{k}\cdot\left(1-\frac{1}{k}\right)^{k-1} \le \frac{2^k}{\mathrm{e}k} \ .$$

□

Remark 4.20. *In [42] it is shown that using a non-uniform product measure P one can improve the bound of Theorem 4.19 to $\Delta < \frac{2^{k+1}}{\mathrm{e}(k+1)}$ and that this is asymptotically tight. We note that we can achieve the same bound using Theorem 4.18 with the same P but since this an involved LLL application we will not explicitly present it here.*

Proof of Theorem 4.18

We consider a very simple backtracking algorithm: We start with each variable unassigned. Then, at each state σ we choose the lowest indexed unassigned variable v and sample a value for it according to the product measure P. If one or more constraints become violated, as a result we remove the value from each variable of the lowest indexed violated constraint.

Let Ω be the set of partial non-violating assignments. Let $\mu : \Omega \to \mathbb{R}$ be the probability measure that assigns to each state $\sigma \in \Omega$ the value $\mu(\sigma) \propto \prod_{\substack{v \in V \\ v \notin U(\sigma)}} P(\sigma(v))$, where we abuse notation for brevity by letting $P(\sigma(v))$ denote the event that variable v is assigned value $\sigma(v)$.

Theorem 4.18 will follow immediately from the following lemma. (For brevity, we will index flaws with variables instead of integers.)

Lemma 4.21. *For each vertex v and set of variables $S \neq \emptyset$:*

$$\gamma_v^S = \begin{cases} 1 \text{ if } S = \emptyset \\ P(A_c) \text{ if } S = c, \text{ where } c \text{ is a constraint containing } v \\ 0 \text{ otherwise}, \end{cases} ,$$

Proof. Notice that the actions related to flaw f_v can only remove the value from sets of of variables that correspond to constraints that contain v. Thus, $\gamma_v^S = 0$ for every set $S \neq \emptyset$ that does not correspond to a constraint containing v. Recalling the definition of charges and $U(\sigma)$, we have

$$\gamma_v^S = \max_{\tau \in \Omega} \sum_{\substack{\sigma \in f_v \\ S = U(\tau) \setminus (U(\sigma) \setminus \{v\})}} \frac{\mu(\sigma)}{\mu(\tau)} \rho_v(\sigma, \tau) \ . \tag{4.20}$$

To see the claim for the case of the empty set, notice that given a state τ there exists at most one state σ such that $\rho_v(\sigma, \tau) > 0$ and that $U(\tau) \setminus (U(\sigma) \setminus \{v\}) = \emptyset$. This is because we can uniquely reconstruct σ from τ by removing the value from v at τ. Then we have

$$\frac{\mu(\sigma)}{\mu(\tau)} \rho_v(\sigma, \tau) = \frac{\prod_{u \in V \setminus U(\sigma)} P(\sigma(u))}{\prod_{u \in V \setminus U(\tau)} P(\tau(u))} P(\tau(v)) = \frac{1}{P(\tau(v))} P(\tau(v)) = 1 \ .$$

To see the claim for the case where $S = c$, consider the set $\mathrm{viol}(c)$ consisting of the set of value assignments of the variables of c that violate c. Notice now that for every state $\tau \in \Omega$ there is an injection from the set of states σ such that that $\rho_v(\sigma, \tau) > 0$ and $S = U(\tau) \setminus (U(\sigma) \setminus \{v\})$ to $\mathrm{viol}(c)$. This is because c should be violated at each such state σ and, thus, it should be that each state σ should be of the form $\sigma = \tau_\alpha$ for $\alpha \in \mathrm{viol}(c)$, where τ_α is the state induced by τ when assigning α to the variables of c. Observe further that for every state of the form $\tau_\alpha, \alpha \in \mathrm{viol}(c)$, we have that

$$\frac{\mu(\tau_\alpha)}{\mu(\tau)} \rho_v(\tau_\alpha, \tau) = \left(\prod_{u \in c \setminus \{v\}} P\left(X_u = \tau_\alpha(u)\right) \right) P(X_v = \tau(v)) = P\left(A_c^\alpha\right) \ , \tag{4.21}$$

where $P(A_c^\alpha)$ is the probability of the event that the variables of c receive assignment α. Combining (4.21) with (4.20) and the fact that $P(A_c) = \sum_{\alpha \in \text{viol}(c)} P(A_c^\alpha)$ concludes the proof of Lemma 4.21. □

Plugging Lemma 4.21 into Theorem 4.6 concludes the proof of Theorem 4.18.

4.5.2 Acyclic Edge Coloring

An edge-coloring of a graph is *proper* if all edges incident to each vertex have distinct colors. A proper edge coloring is *acyclic* if it has no bichromatic cycles, i.e., no cycle receives exactly two (alternating) colors. The smallest number of colors for which a graph G has an acyclic edge-coloring is denoted by $\chi'_a(G)$.

Acyclic Edge Coloring was originally motivated by the work of Coleman et al. [29, 28] on the efficient computation of Hessians and, since then, there has been a series of papers [9, 78, 82, 49, 69, 38] that upper bound $\chi'_a(G)$ for graphs with bounded degree. The current best result was given recently by Giotis et al. in [44] who showed that $\chi'_a(G) \leq 3.74\Delta$ in graphs with maximum degree Δ.

A Simple Backtracking Algorithm

We show how one can apply Theorem 4.6 to recover the main application of the framework of [38] with a much simpler proof.

Let G be a graph with m edges $E = \{e_1, \ldots, e_m\}$ and suppose we have q available colors.

Definition 4.22. *Given a graph $G = (V, E)$ and a (possibly partial) edge-coloring of G, say that color c is* 4-forbidden *for $e \in E$ if assigning c to e would result in either a violation of proper-edge-coloration, or a bichromatic 4-cycle containing e. Say that c is* 4-available *if it is not 4-forbidden.*

Lemma 4.23 ([38])**.** *In any proper edge-coloring of G at most $2(\Delta - 1)$ colors are 4-forbidden for any $e \in E$.*

Proof. The 4-forbidden colors for $e = \{u, v\}$ can be enumerated as: (i) the colors on edges adjacent to u, and (ii) for each edge e_v adjacent to v, either the color of e_v (if no edge with that color is adjacent to u), or the color of some edge e' which together with e, e_v and an edge adjacent to u form a cycle of length 4. □

Consider the following backtracking algorithm for Acyclic Edge Coloring with $q = 2(\Delta - 1) + Q$ colors. At each step, choose the lowest indexed uncolored edge e and attempt to color it choosing uniformly at random among the 4-available colors for e. If one or more bichromatic cycles are created, then choose the lowest indexed one of them, say $C = \{e_{i_1}, e_{i_2}, \ldots, e_{i_{2\ell}} = e\}$, and remove the colors from all its edges except e_{i_1} and e_{i_2}.

The main result of [38] states that every graph G admits an acyclic edge coloring with $q > 4(\Delta - 1)$ colors. Moreover, such a coloring can be found in $O\left(|E||V|\Delta^2 \ln \Delta\right)$ time with high probability.

We prove the following theorem, which improves the running time bound when the graph is dense.

Theorem 4.24. *Every graph G admits an acyclic edge coloring with $q > 4(\Delta - 1)$ colors. Moreover, such a coloring can be found in $O(|E||V|\Delta)$ time with high probability.*

Proof. Let Ω be the set of partial acyclic edge colorings of G. For each edge e let f_e be the subset (flaw) of Ω that contains the partial acyclic edge colorings of G in which e is uncolored. We will apply Theorem 4.6 using the $\|\cdot\|_1$ norm and $M = \frac{1}{|\Omega|}$.

We first compute the charges γ_e^S for each edge e and set of edges S. Notice that for γ_e^S to be non-zero, it should either be that $S = \emptyset$, or that S contains e and there exists a cycle $C = \{e_{i_1}, e_{i_2}\} \cup S$ so that, when a recoloring of e makes C bichromatic, the backtracking step uncolors precisely the edges in S. With that in mind, for each edge e and each set S that contains e, let $\mathcal{C}_e(S)$ denote the set of cycles with the latter property.

Lemma 4.25. *For each edge e, let*

$$\gamma_e^S = \begin{cases} \frac{1}{Q} & \text{if } S = \emptyset \\ \frac{|\mathcal{C}_e(S)|}{Q} & \text{if } e \in S \\ 0 & \text{otherwise} \end{cases} .$$

Proof. Notice that

$$\gamma_e^S = \max_{\tau \in \Omega} \sum_{\substack{\sigma \in f_e \\ S = U(\tau) \setminus (U(\sigma) \setminus \{e\})}} \rho_e(\sigma, \tau) \leq \max_{\tau \in \Omega} \sum_{\substack{\sigma \in f_e \\ S = U(\tau) \setminus (U(\sigma) \setminus \{e\})}} \frac{1}{Q} ,$$

since according to Lemma 4.23 $\rho_e(\sigma, \tau) \leq \frac{1}{Q}$ for each pair $(\sigma, \tau) \in f_e \times \Omega$. The proof follows by observing that for each state τ:

- If $S = \emptyset$ then there exists at most one state σ such that $\rho_e(\sigma, \tau) > 0$ and $U(\tau) \setminus (U(\sigma) \setminus \{\mathrm{e}\}) = \emptyset$ (we can reconstruct σ from τ by uncoloring e).
- If $S \ni e$ and $|S| = 2\ell - 2$ then there exist at most $|\mathcal{C}_e(S)|$ states such that $\rho_e(\sigma, \tau) > 0$ and $S = U(\tau) \setminus (U(\sigma) \setminus \{\mathrm{e}\})$. Given a cycle $C = S \cup \{e_{i_1}, e_{i_2}\}$ we reconstruct σ from τ by finding the colors of edges in $S \setminus \{e\}$ from $\tau(e_{i_1}), \tau(e_{i_2})$, exploiting the fact that the backtracking step corresponds to an uncoloring of a bichromatic cycle; e is uncolored; and every other edge has the same color as in τ.
- For all other S there exists no state σ such that $\rho(\sigma, \tau) > 0$ and $S = U(\tau) \setminus (U(\sigma) \setminus \{\mathrm{e}\})$.

□

Observe that there are at most $(\Delta - 1)^{2\ell - 2}$ cycles of length 2ℓ containing a specific edge e. In other words, there exist at most $(\Delta - 1)^{2\ell - 3}$ sets of edges S of size $2\ell - 2$ that contain e and such that $\gamma_e^S > 0$ and, in addition, note that we always have $|\mathcal{C}_e(S)| \leq \Delta - 1$.

Thus, if $Q = c(\Delta - 1)$ for some constant c, setting $\psi_e = \alpha\gamma_\emptyset^e = \frac{\alpha}{Q}$, where α is a constant in $(1, c)$, Lemma 4.25 implies:

$$\begin{aligned}\frac{1}{\psi_e}\left(\sum_{S\subseteq E}\gamma_e^S\prod_{e\in S}\psi_j\right) &\leq \min_{\alpha\in(1,c)}\left(\frac{1}{\alpha}+\sum_{i=3}^{\infty}\left(\frac{\Delta-1}{Q}\right)^{2i-2}\alpha^{2i-3}\right)\\ &\leq \min_{\alpha\in(1,c)}\left(\frac{1}{\alpha}+\frac{1}{c}\sum_{i=3}^{\infty}\left(\frac{\alpha}{c}\right)^{2i-3}\right)\\ &= \min_{\alpha\in(1,c)}\left(\frac{1}{\alpha}+\frac{\alpha^3}{c^2(c^2-\alpha^2)}\right)=\frac{2}{c}\end{aligned}$$

for $\alpha^* = c\left(\frac{\sqrt{5}-1}{2}\right)$. Thus, if $c > 2$ the probability that the algorithm fails to find an acyclic edge coloring within $\frac{T_0+s}{\delta}$ steps is 2^{-s}, where $\delta = 1 - \frac{2}{c}$, and, according to Remark 4.10,

$$T_0 = \log_2|\Omega| = O(|E|) \ .$$

The proof is concluded by observing that each step can be performed in time $O(|V|\Delta)$ (the time it takes to find a 2-colored cycle containing a given edge, if such a cycle exists, in a graph with a proper edge-coloring). □

An Application of the Local Cut Lemma

Bernshteyn [18] introduced a non-constructive generalized LLL condition, called the "Local Cut Lemma", with the aim of to drawing connections between the LLL and the entropy compression method. He later applied it in [16] to the problem of Acyclic Edge Coloring giving improved bounds assuming further constraints on the graph besides sparsity. For example he proved the following.

Theorem 4.26 ([16])**.** *Let G be a graph with maximum degree Δ and let H be a fixed bipartite graph. If G does not contain H as a subgraph, then there exists an acyclic edge coloring of G using at most $3(\Delta + o(1))$ colors.*

We now show how to use our framework to give a constructive proof of Theorem 4.26. This will follow immediately from the following structural lemma in [16].

Lemma 4.27 ([16])**.** *There exist positive constants γ, δ such that the following holds. Let G be a graph with maximum degree Δ that does not contain H as a subgraph. Then for any edge $e \in E(G)$ and for any integer $k \geq 4$, the number of cycles of length k in G that contain e is at most $\gamma\Delta^{k-2-\delta}$.*

Constructive Proof of Theorem 4.26. Notice that in this case, making almost identical calculations to those in the proof of Theorem 4.24, invoking Lemma 4.27 to upper bound the number of cycles that contain e and setting $\alpha = \frac{c}{\beta}$ we obtain

$$\frac{1}{\psi_e}\left(\sum_{S\subseteq E}\gamma_e^S\prod_{h\in S}\psi_h\right) \leq \min_{\alpha\in(1,c)}\left(\frac{1}{\alpha}+\frac{(\alpha)^3\gamma\Delta^{-\delta}}{c^2(c^2-\alpha^2)}\right)=\frac{1}{c}\min_{\beta>1}\left(\beta+\frac{\beta\gamma\Delta^{-\delta}}{\beta(\beta^2-1)}\right) \ .$$

Thus, as Δ grows, the value of c required for the algorithm to terminate approaches 1, concluding the proof. □

Chapter 5

Commutative Algorithms

Besides conditions for existence of and fast convergence to perfect objects, one may naturally ask further questions regarding properties of focused search algorithms. For instance, "are they parallelizable ?", "how many solutions can they output?", "what is the expected "weight" of a solution?", etc. These questions and more have been answered for the Moser-Tardos (MT) algorithm in a long series of parpers [22, 27, 49, 48, 53, 56, 70, 81]. As a prominent example, the result of Haeupler, Saha and Srinivasan [49], as well as follow-up work of Harris and Srinivasan [56, 51], allows one to argue about the dynamics of the MT process, resulting in several new applications such as estimating the entropy of the output distribution, partially avoiding bad events, dealing with super-polynomially many bad events, and even new frameworks [54, 23].

Unfortunately, most of these follow-up results that further enhance, or exploit, our understanding of the MT process are not transferable to the general setting we described in Section 3.1. Mainly, this is because they are byproducts of the backward-looking analysis we saw in Chapter 3 and, in particular, of a key technical result of the original analysis of Moser and Tardos, *the witness tree lemma*, which is known not to hold under the most general assumptions [57]. Roughly, the witness tree lemma states that any tree of bad events growing backwards in time from a certain root bad event A_i, with the children of each node A_j being bad events that are adjacent to A_j in the dependency graph, is consistent with the trajectory of the algorithm with probability bounded by the product of the probabilities of all events in this tree. (Recall that we have already seen this statement it in Chapter 3, Lemma 3.12.) The witness tree lemma and its variations [70, 48] have been used for several other purposes besides those already mentioned, such as designing deterministic, parallel and distributed algorithms for the LLL [81, 22, 27, 48, 53].

On the other hand, Harris and Srinivasan [55] did manage to prove the witness tree lemma for their algorithm for the LLL on the space of permutations, which lies outside the variable framework of Moser-Tardos, via an analysis that is tailored specifically to this setting. Although their proof does not seem to be easily generalizable to other spaces, their success makes it natural to ask if we can impose mild assumptions in the general setting under which the witness tree lemma and most of its byproducts can be established.

This question was answered positively by present author in [59], where it was shown that it is possible to prove the witness tree lemma in the *commutative setting*. The latter was introduced by Kolmogorov [71], who showed that under its assumptions one can obtain parallel versions of stochastic local search algorithms, as well as the flexibility of removing the restrictions on the flaw choice strategy imposed by the general LLL condition (3.8). We note that the commutative setting

captures the vast majority of LLL applications, including but not limited to both the variable and the permutation settings.

Subsequently to [59], Achlioptas, Iliopoulos and Sinclair [5] gave a simpler proof of the witness tree lemma under a more general notion of commutativity (essentially matrix commutativity) at the mild cost of slightly restricting the flaw choice strategy. In this chapter we will present their proof, as well as several results from [59] regarding properties of commutative algorithms and their applications.

Distributional properties. As already mentioned, one of the most important applications of the witness tree lemma is given in the paper of Haeupler, Saha and Srinivasan [49], who study properties of the *MT-distribution*, the output distribution of the MT algorithm. Their main result is that the MT-distribution well-approximates the *LLL-distribution*, i.e., the distribution obtained by conditioning on all bad events being avoided. One immediate consequence of this fact is that one can argue about the *expected weight* of the output of the MT algorithm, given a weighting function over the space Ω. Furthermore, as shown in the same paper [49] and follow-up papers by Harris and Srinivasan [56, 51], one can also lower bound the entropy of the MT distribution, relax the LLL conditions (if one is willing to only partially avoid bad events), and deal with applications with super-polynomially many bad events.

Here we extend the result of [49] to the commutative setting: Given a commutative algorithm that is perfectly compatible with the underlying probability measure, its output well-approximates the LLL-distribution in the same sense as the MT-distribution does in the variable setting. For arbitrary commutative algorithms, the quality of the approximation depends additionally on the compatibility of the algorithm with the measure on the event(s) of interest.

Moreover, we quantitatively improve the bounds of [49] under the weaker assumption of Shearer's condition (2.4); recall from Section 2.3 that this is the most general LLL criterion in the setting where the dependency graph is undirected. This allows us to study distributional properties of commutative algorithms using criteria that lie between the general LLL and Shearer's condition such as the Clique LLL [69]. Finally, in Section 5.3 we discuss several byproducts of the fact that commutative algorithms approximate the LLL-distribution.

Algorithmic LLL without a slack and improved running time bounds As we saw in Chapter 3, the general LLL condition (3.8) requires a multiplicative slack in order to establish fast convergence to a perfect object. On the other hand, Harvey and Vondrák [57] dispense with this requirement in the important case of algorithms that are perfectly compatible with the underlying measure under the mild assumption that the dependency graph is undirected. As we will see, using the witness tree lemma, we are able to dispense with the multiplicative slack requirement for commutative algorithms. Moreover, we are able to improve the running time bounds of Harvey and Vondrák [57] in the commutative setting, matching those of Kolipaka and Szegedy [70] for the MT algorithm. This was posed as an open question in [57].

Concrete applications. In Section 5.6 we study a commutative algorithm for finding *rainbow matchings*, which is a problem in the space of matchings of a complete graph. We use this problem as an example that allows us to show how several byproducts of approximating the LLL-distribution can be applied in a black-box manner to a setting that is not captured either by the

variable or the permutation setting, and for which we know [3, 57, 71] how to design commutative algorithms that are perfectly compatible with the uniform measure over the state space.

In Chapter 6, among other things, we will show that Molloy's algorithm [76] for finding proper colorings in triangle-free graphs with maximum degree Δ using $(1+\epsilon)\frac{\Delta}{\ln \Delta}$ colors, can actually output exponentially many such colorings. We do this by showing that this is a commutative algorithm, a fact that gives us access to properties of its output distribution.

Finally, in Chapter 7 we give an algorithm for the non-constructive result of Kahn [65], showing that the Goldberg-Seymour conjecture is true asymptotically. This is a stochastic local search algorithm for addressing a family of super-polynomially many flaws, and we will need to show that it is commutative in order to establish its efficiency.

5.1 Commutativity and the Witness Tree Lemma

Throughout this chapter we consider algorithms with the property that f_i causes f_j if and only if f_j causes f_i. We will thus view the causality graph as an *undirected* graph G which may contain self-loops. We also write $i \sim j$ to denote that $j \in \Gamma(i)$ (or equivalently, $i \in \Gamma(j)$).

Kolmogorov's [71] notion of commutativity requires that for every $i \nsim j \in [m]$, every sequence of state transitions of the form $\sigma_1 \xrightarrow{i} \sigma_2 \xrightarrow{j} \sigma_3$ can be mapped to a distinct sequence of state transitions of the form $\sigma_1 \xrightarrow{j} \sigma_2' \xrightarrow{i} \sigma_3$, so that $\rho_i(\sigma_1,\sigma_2)\rho_j(\sigma_2,\sigma_3) = \rho_j(\sigma_1,\sigma_2')\rho_i(\sigma_2',\sigma_3) > 0$.

The matrix framework of [5] allows us to introduce a more natural notion of algorithmic commutativity, essentially matrix commutativity, that is also more general than the notion of [71]. For $i \in [m]$, let A_i denote the $|\Omega| \times |\Omega|$ matrix where $A_i[\sigma,\sigma'] = \rho_i(\sigma,\sigma')$ for $\sigma \in f_i$, and 0 otherwise. Recall the definition of $\Gamma(i)$.

Definition 5.1. *An algorithm is* commutative *with respect to a symmetric binary relation $\bowtie$ if*

(a) $A_iA_j = A_jA_i$*, for every* $i,j \in [m]$ *such that* $i \not\bowtie j$.

(b) $\Gamma(i) \subseteq \{j : i \sim j\}$.

Remark 5.2. *In most applications $A_iA_j \neq A_jA_i$ when $i \in \Gamma(j)$, in which case a implies b. For this reason, and to simplify notation, in the following we will always consider commutative algorithms with respect to the causality relation $\sim$ induced by their corresponding (undirected) causality graph.*

Under this new notion, we recover all the results of [71, 59] with much simpler proofs, at the mild cost of restricting the family of flaw choice strategies to *canonical* ones, per Definition 5.3 below. (In [71, 59] the flaw choice strategy can be arbitrary.) Note that in the commutative setting, canonical flaw choice strategies suffice to capture the optimal convergence results, so that the restriction to such strategies is indeed mild. Recall the definition of $U(\sigma)$.

Definition 5.3. *Fix an arbitrary sequence of (possibly stochastic) functions $(s_i)_{i\geq 1} : \Omega \to [m]$, each s_i mapping $\sigma \in \Omega$ to an element of $U(\sigma)$. A flaw choice strategy is* canonical *if the flaw addressed in the i-th step is $s_i(\sigma_i)$, where $\sigma_i \in \Omega$ is the state after i steps.*

In particular, we establish the Witness Tree Lemma, which we describe forthwith, and from which all the other results follow. (In fact, we prove a more general version of the Witness Tree Lemma that takes as input an operator norm $\|\cdot\|$ and diagonal matrix M. Using the $\|\cdot\|_1$-norm and $M = \mathrm{diag}(\mu(\sigma))$, where μ is a probability measure over Ω, recovers the standard Witness Tree Lemma.)

For $i \in [m]$, let A_i denote the $|\Omega| \times |\Omega|$ matrix defined by $A_i[\sigma, \sigma'] = \rho_i(\sigma, \sigma')$ if $\sigma \in f_i$, and $A_i[\sigma, \sigma'] = 0$ otherwise, and define

$$\gamma_i = \|MA_iM^{-1}\| \ , \tag{5.1}$$

where M is a fixed invertible matrix such that $\sum_{\sigma\in\Omega} \|M\mathrm{e}_\sigma\| = 1$. Here e_σ denotes the indicator vector of σ, i.e., $e_\sigma[\sigma] = 1$ and $e_\tau[\tau] = 0$ for all $\tau \in \Omega \setminus \{\sigma\}$.

Remark 5.4. *Recall equation* (3.5)*, which defines the charge of a flaw with respect to a distribution μ, and notice that is equivalent to* (5.1) *when $\|\cdot\| = \|\cdot\|_1$ and $M = \mathrm{diag}(\mu(\sigma))$.*

Recall the definition of witness trees from Section 3.3.1 and, in particular, that $\Pr[\tau]$ denotes the probability that a given witness tree τ occurs during the execution of the algorithm. Recall also that θ denotes the initial distribution of the algorithm.

Theorem 5.5 (Witness Tree Lemma)**.** *Assume that algorithm $\mathcal{A}$ is commutative with respect to binary relation $\sim$ and follows a canonical flaw choice strategy. Then, for every witness tree τ,*

$$\Pr[\tau] \leq \lambda_{\text{init}} \prod_{v\in V(\tau)} \gamma_{(v)} \ ,$$

where $\lambda_{\text{init}} = \|\theta M^{-1}\|_*$ and $\|\cdot\|_*$ denotes the dual norm of $\|\cdot\|$.

5.2 Approximating the LLL-distribution

Assuming that the LLL condition (2.1) holds, the *LLL-distribution*, which we denote by μ_{LLL}, is defined as the distribution induced by the measure μ conditional on no bad event occurring. The following proposition relates the LLL-distribution to measure μ making it a powerful tool that can be used to argue about properties of flawless objects. The idea is that if an (not necessarily bad) event E is independent from most bad events, then its probability under the LLL-distribution is not much larger than its probability under the probability measure μ.

Proposition 5.6 ([49])**.** *If the LLL condition* (2.1) *holds, then for any event E:*

$$\mu_{\mathrm{LLL}}(E) \leq \mu(E) \sum_{S\subseteq D(E)} \prod_{j\in S} \psi_j \ , \tag{5.2}$$

where $D(E) \subseteq [m]$ is such that $\mu(E \mid \bigcap_{j\in S} \overline{A}_j) = \mu(E)$ for all $S \subseteq [m] \setminus D(E)$.

The main result of Haeupler, Saha and Srinivasan [49] is that the Moser-Tardos algorithm approximates well the LLL-distribution, in the sense that the left-hanside of (5.2) bounds the probability that it ever reaches a subspace $E \subseteq \Omega$ during its execution. Building on this fact, [49] and followup works [56, 51] manage to show several new applications.

Here we extend the latter result to arbitrary commutative algorithms. Given an arbitrary set $E \subseteq \Omega$ and a commutative algorithm $\mathcal{A}$, consider an extension, $\mathcal{A}_E$, of $\mathcal{A}$ by defining an extra flaw $f_{m+1} \equiv E$ with its own set of probability distributions $\rho_{m+1}(\sigma, \cdot), \sigma \in E$. If $\mathcal{A}$ is commutative with respect to $\sim$, we will say that $\mathcal{A}_E$ is a *commutative extension* of $\mathcal{A}$ if $\mathcal{A}_E$ is also commutative with respect to $\sim$; that is, if the conditions of Definition 5.1 also hold for matrix $A_E \equiv A_{m+1}$ and set $\Gamma(E) \equiv \Gamma(m+1)$.

As we will see, commutative extensions should be thought of as a tool to bound the probability that $\mathcal{A}$ ever reaches a subset E of the state space. That is, they are defined only for the purposes of the analysis and, typically in applications, they are a natural extension of the algorithm. For example, in the case of the Moser-Tardos algorithm applied to k-SAT, if one would like to bound the probability that the algorithm ever reaches a state such that variables x_1, x_2, x_3 of the formula are all set to true, then one could define $f_{m+1} = \{\sigma \in \Omega \text{ s.t. } \sigma(x_1) = \sigma(x_2) = \sigma(x_3) = 1\}$ along with the corresponding commutative extension of the Moser-Tardos algorithm that addresses f_{m+1} by resampling variables x_1, x_2, x_3 according to the product measure over the variables of the formula that the Moser-Tardos algorithm uses whenever it needs to resample a violated clause. Indeed, commutative extensions of this form are implicitly defined in the analysis of [49] for the Moser-Tardos algorithm.

We will use the notation $\Pr[\cdot] = \Pr_{\mathcal{A}}[\cdot]$ to refer to the probability of events in the probability space induced by the execution of algorithm $\mathcal{A}$. For example, the probability that $\mathcal{A}$ reaches a set $E \subseteq \Omega$ of the state space during its execution will be denoted by $\Pr[E]$. Finally, recall that for a set $S \subseteq [m]$, $\mathrm{Ind}(S) = \mathrm{Ind}_G(S)$ denotes the set of independent subsets of S with respect to G.

The main result of this chapter is Theorem 5.7 below, which we prove in Section 5.4.

Theorem 5.7. *Give a symmetric, binary relation $\sim$ and a commutative algorithm $\mathcal{A}$ with respect to $\sim$, if there exist positive real numbers $\{\psi_i\}_{i \in [m]}$ such that for every $i \in [m]$,*

$$\frac{\gamma_i}{\psi_i} \sum_{S \in \mathrm{Ind}(\Gamma(i))} \prod_{j \in S} \psi_j \ \leq 1 \ ,$$

then

1. *for each $i \in [m]$, $\mathbb{E}[T_i] \leq \lambda_{\text{init}} \psi_i$;*
2. *for each $E \subseteq \Omega$, $\Pr[E] \leq \lambda_{\text{init}} \gamma_E \sum_{S \in \mathrm{Ind}(\Gamma(E))} \prod_{j \in S} \psi_j$;*

where T_i is the number of times flaw f_i is addressed, $\lambda_{\text{init}} = \|\theta^\top M^{-1}\|_*$, and $\Gamma(E)$, γ_E are defined with respect to a fixed commutative extension $\mathcal{A}_E$.

Corollary 5.8. *Algorithm $\mathcal{A}$ terminates after $O(\lambda_{\text{init}} \sum_{i \in [m]} \psi_i)$ steps in expectation.*

Remark 5.9. *Theorem 5.7 corresponds to the cluster expansion condition* (2.3)*. If the Shearer's condition is satisfied, then one can replace ψ_i in Theorem 5.7 with $\frac{q^{\{i\}}(b)}{q^{\emptyset}(b)}$, where $b = (\gamma_1, \ldots, \gamma_m)$.*

We note that the first part of Theorem 5.7 allows us to guarantee fast convergence of $\mathcal{A}$ to a perfect object without having to assume a "slack" in the cluster expansion and Shearer's conditions (unlike the works of [4, 71]) and, moreover, improves upon the (roughly quadratically worse) running bound of [57], matching the one of [70].

5.3 Byproducts of Theorem 5.7

In this section we discuss three important byproducts of Theorem 5.7 which we will use in our applications.

5.3.1 Entropy of the Output Distribution

A useful application of the known bounds for the Moser-Tardos distribution is estimating its randomness. In particular, Harris and Srinivasan [56] show that one can give lower bounds on the *Rényi entropy* of the output of the Moser-Tardos algorithm.

Definition 5.10 ([26])**.** *Let ν be a probability measure over a finite set S. The* Rényi entropy with parameter *ρ of ν is defined to be*

$$H_\rho[\nu] = \frac{1}{1-\rho} \ln \sum_{s \in S} \nu(s)^\rho \ .$$

The min-entropy *H_∞ is a special case defined as $H_\infty[\nu] = \lim_{\rho \to \infty} H_\rho[\nu] = -\ln \max_{s \in S} \nu(s)$.*

Given Theorem 5.7, we can show the analogous result in our setting, using a proof that is akin to the one in [56], and which can be found in Appendix 8.3.

Theorem 5.11. *Let $\sim$ be a symmetric, binary relation, and $\mathcal{A}$ be a commutative algorithm with respect to $\sim$. Assume the conditions of Theorem 5.7 are satisfied, and let ν be the output distribution of $\mathcal{A}$. Then, for $\rho > 1$,*

$$H_\rho[\nu] \geq H_\rho[\mu] - \frac{\rho}{\rho - 1} \ln \left(\sum_{S \in \mathrm{Ind}([m])} \prod_{j \in S} \psi_j \right) - \frac{\rho}{\rho - 1} \ln \lambda_{\mathrm{init}} \ .$$

A straightforward application of having a lower bound on $H_\rho[\nu]$ (for any ρ), where ν is the output distribution of the algorithm, is that there exist at least $\exp(H_\rho[\nu])$ flawless objects. Before [56], the authors in [74] also used the (existential) LLL for enumeration of combinatorial structures by exploiting the fact that it guarantees a small probability p of avoiding all flaws when sampling from the uniform measure (and, thus, their number is at least $p|\Omega|$).

5.3.2 Partially Avoiding Flaws

One of the main results of [49, 56] are LLL conditions for the existence of objects that avoid a large portion of the bad events. For instance, given a sufficiently sparse k-SAT formula which, nonetheless, violates the LLL conditions (see Theorem 2.1), one can still find an assignment that satisfies many clauses. Using Theorem 5.7, we are able to extend the (most general) result of Harris and Srinivasan [56] to the commutative setting.

Given a sequence of positive numbers $\{\psi_i\}_{i=1}^m$, for each $i \in [m]$ define:

$$\eta_i := \sum_{S \in \mathrm{Ind}(\Gamma(i))} \prod_{j \in S} \psi_j \ ,$$

and notice that the cluster expansion condition can be expressed as requiring that for each $i \in [m]$ we have that $\gamma_i \eta_i \leq \psi_i$.

Theorem 5.12. *Let $\sim$ be a symmetric, binary relation, and $\mathcal{A}$ be a commutative algorithm with respect to $\sim$. Assume that the set of charges $\{\gamma_i\}_{i\in[m]}$ are defined with respect to the $\|\cdot\|_1$-norm and the diagonal matrix $M = \mathrm{diag}(\mu(\sigma))$, where μ is a probability distribution. Let also $\{\psi_i\}_{i=1}^m$ be a sequence of positive real numbers and assume $\theta = \mu$. There is an algorithm $\mathcal{A}'$ (which is a modification of $\mathcal{A}$) and whose output distribution ν has the property that for each $i \in [m]$*

$$\nu(f_i) \leq \max\{0, \gamma_i\eta_i - \psi_i\} \ .$$

Furthermore, the expected number of times a flaw f_i is addressed is at most ψ_i.

Given Theorem 5.7, the proof of Theorem 5.12 is akin to the one of [56] and can be found in Appendix 8.3.

5.3.3 Dealing with Super-Polynomially Many Flaws

Here we discuss how one can deal with problems where the number of flaws is super-polynomial in the natural size of the problem using commutative algorithms.

In such a setting, there are two issues to be resolved. The first issue is that one should be able to show that the expected number of steps until convergence is polynomial, and thus, much less than $\Theta(|F|)$. The second issue is that one should have an efficient procedure for finding a flaw that is present in the current state, or decide that no such flaw exists.

Polynomial-Time Convergence. As far as the issue of polynomial-time convergence is concerned, there are at least three approaches one can follow.

A first approach is to start the algorithm at a state σ_1 in which the set of flaws present is of polynomial size, and then employ the main results from [4] which guarantee that the algorithm will terminate after $O\left(|U(\sigma_1)| + \max_{\sigma\in\Omega} \log_2 \frac{1}{\mu(\sigma)}\right)$ steps with high probability. This approach does not require the algorithm to be commutative, but it does require that the LLL condition is satisfied with a slack in order establish quick termination.

A second approach, which was first applied in the context of the Moser-Tardos algorithm by Haeupler, Saha and Srinivsan [49], is to find a *core* set of flaws of polynomial size and apply a modified version of the algorithm that effectively ignores any non-core flaw. The hope is that non-core flaws will never occur during the execution of this modified algorithm. Extended to our setting, one uses the following result which is a straightforward corollary of Theorem 5.7.

Corollary 5.13. *Let $\mathcal{A}$ be a commutative algorithm with respect to a symmetric, binary relation $\sim$. Let $I \subseteq [m]$ be a set of indices that corresponds to a core subset of F and assume there exist positive real numbers $\{\psi_i\}_{i=1}^m$ such that for each $i \in [m]$*

$$\gamma_i \sum_{S\in \mathrm{Ind}(\Gamma(i)\cap I)} \prod_{j\in S} \psi_j \leq \psi_i \ .$$

Then there exists a modification of $\mathcal{A}$ that terminates in an expected number of $O\left(\sum_{i\in I} \psi_i\right)$ steps and outputs a flawless element with probability at least $1 - \sum_{i\in[m]\setminus I} \lambda_{\mathrm{init}}\gamma_i \sum_{S\in\mathrm{Ind}(\Gamma(i)\cap I)} \prod_{j\in S} \psi_j$.

Finally, a third approach is to show that the causality graph can be decomposed into a set of cliques of polynomial size and then apply a result of [49] which states that, in this case, the running time of the algorithm is polynomial (roughly quadratic) in the size of the decomposition. To be more precise, we note that in [49] the latter result is shown for the Moser-Tardos algorithm in the variable setting and assuming condition (2.1), where the clique decomposition considered is induced by the family of independent random variables that form the probability space (one clique per variable). However, the proof for the general case is identical. Using Theorem 5.7 and recalling Remark 2.2 we can extend this result to our setting to get:

Theorem 5.14. *Let $\mathcal{A}$ be a commutative algorithm with respect to a symmetric, binary relation $\sim$. Assume there exist real numbers $\{x_i\}_{i\in[m]}$ in $(0,1)$ such that*

$$\gamma_i \leq x_i \prod_{j\in\Gamma(i)} (1-x_j) \text{ for every } i \in [m] \quad . \tag{5.3}$$

Assume further that the causality graph induced by $\sim$ can be partitioned into n cliques, with potentially further edges between them. Setting $\delta := \min_{i\in[m]} x_i \prod_{j\in\Gamma(i)}(1-x_j)$, the expected number of steps performed by $\mathcal{A}$ is at most $t = O\left(\lambda_{\text{init}} \cdot \frac{n}{\epsilon} \log \frac{n\log(1/\delta)}{\epsilon}\right)$, and for any parameter $\eta \geq 1$, $\mathcal{A}$ terminates within ηt resamplings with probability $1 - \mathrm{e}^{-\eta}$.

Remark 5.15. *In [49] it is argued that in the vast majority of applications $\delta = O(n \log n)$ and in many cases even linear in n.*

Following Theorem 5.7, the proof of Theorem 5.14 is identical to the analogous result of Haeupler, Saha and Srinivasan [49] for the Moser-Tardos algorithm and hence we omit it.

Fast Search for Flaws. Searching for occurring flaws efficiently can be a major obstacle in getting polynomial time algorithms, even in the case where convergence is guaranteed after a polynomial number of steps. Again, there is more than one approach one can follow to deal with this issue.

A first approach was introduced in [49] where it is shown that Corollary 5.13 and Theorem 5.14, in the context of the variable setting, can be combined into a single theorem that guarantees the existence of a Monte Carlo algorithm which runs in polynomial time, even in the presence of super-polynomially many flaws. The theorem assumes the existence of a polynomial size decomposition of the causality graph into cliques and, moreover, that the LLL condition holds with an exponential slack. Using Theorem 5.7, we can extend this result in a straightforward way to our setting to get the following theorem.

Theorem 5.16. *Let $\mathcal{A}$ be a commutative algorithm with respect to a symmetric, binary relation $\sim$. Assume there exist real numbers $\{x_i\}_{i\in[m]}$ and ϵ in $(0,1)$ such that*

$$\gamma_i^{1-\epsilon} \leq x_i \prod_{j\in\Gamma(i)} (1-x_j) \text{ for every } i \in [m] \quad . \tag{5.4}$$

Assume further that the causality graph induced by $\sim$ can be partitioned into n cliques, with potentially further edges between them, and let $\delta := \min_{i\in[m]} x_i \prod_{j\in\Gamma(i)}(1-x_j)$. If we furthermore have that $\log 1/\delta \leq \mathrm{poly}(n)$, then for every $\gamma \geq \frac{1}{\mathrm{poly}(n)}$ the set $\{i \in [m] \text{ s.t. } \gamma_i \geq \gamma\}$ has size at most $\mathrm{poly}(n)$. There also exists a Monte Carlo algorithm that terminates after $O(\lambda_{\text{init}} \frac{n}{\epsilon} \log \frac{n}{\epsilon^2})$ steps and returns a perfect object with probability at least $1-n^{-c}$, where c is any desired constant.

In a follow-up work [56], Harris and Srinivasan describe a general technique that yields efficient procedures for searching for flaws. The main building blocks of their technique is a "witness tree lemma for internal states" and problem-specific, possibly randomized, data-structures that contain the flaws that are present in each state. We refer the reader to [56] for more details, but we note that combining the proof of [56] with the proof of Theorems 5.5 and 5.7, one can show that the "witness tree lemma for internal states" holds for commutative algorithms.

5.4 Proof of Theorem 5.7

Recall the definition of witness sequences and witness trees we saw in Section 3.3.1. The first part follows by observing that if W is the witness sequence corresponding to the trajectory of the algorithm, then T_i is the number of occurrences of flaw f_i in W, and according to Proposition 3.8, also the number of distinct witness trees occurring that have their root labeled i. Therefore, letting $\mathcal{W}_i$ denote the set of witness trees whose root node is labelled by i, one can bound the expectation of T_i by summing the bounds in Lemma 5.5. In particular, the following lemma, whose proof is identical to the one of Lemma 3.13, and therefore omitted, concludes the proof of the first part.

Lemma 5.17. *Under the assumptions of Theorem 5.7:*

$$\sum_{\tau\in\mathcal{W}_i} \prod_{v\in V(\tau)} \gamma_{(i)} \leq \psi_i \ .$$

To see the second part of Theorem 5.7, consider the new set of flaws $F' = F \cup \{f_{m+1}\}$, where $f_{m+1} = E$, as well as a "truncated" commutative extension $\mathcal{A}'$ of $\mathcal{A}$ with the following properties:

(i) For each state $\sigma \notin f_{m+1}$ algorithm $\mathcal{A}'$ invokes $\mathcal{A}$ to choose its next state;

(ii) $\gamma_E := \gamma_{(m+1)}(\mathcal{A}')$;

(iii) f_{m+1} is always of the highest priority: when at a state $\sigma \in f_{m+1}$, $\mathcal{A}'$ chooses to address f_{m+1};

(iv) $\mathcal{A}'$ stops after it addresses f_{m+1} for the first time.

By coupling $\mathcal{A}$ and $\mathcal{A}'$ we see that $\Pr_{\mathcal{A}}[E] = \Pr_{\mathcal{A}'}[f_{m+1}]$. Let $\mathcal{W}_E$ be the set of witness trees that can occur in an execution of $\mathcal{A}'$ and whose root is labelled by $m+1$. Notice that, due to property (iv) of $\mathcal{A}'$, every tree $\tau \in \mathcal{W}_E$ contains exactly one node (the root) labelled by $m+1$, while every other node is labelled by elements in $[m]$. Furthermore, the set of labels of the children of the root of τ is an element of $\mathrm{Ind}(\Gamma(E))$. Finally, if v is a node that corresponds to a child of the root in τ, then the subtree τ_v that is rooted at v is an element of $\mathcal{W}_{(v)}$. Using Theorem 5.5, Lemma 5.17 and the fact that $\mathcal{A}'$ is commutative we get:

$$\Pr_{\mathcal{A}}[E] \leq \sum_{\tau \in \mathcal{W}_E} \Pr_{\mathcal{A}'}[\tau] \leq \lambda_{\mathrm{init}} \gamma_E \sum_{S \in \mathrm{Ind}(\Gamma(E))} \left(\prod_{j \in S} \sum_{\tau \in \mathcal{W}_j} \prod_{v \in \tau} \gamma_{(v)} \right) \leq \lambda_{\mathrm{init}} \gamma_E \sum_{S \in \mathrm{Ind}(\Gamma(E))} \prod_{j \in S} \psi_j \ ,$$

where the last equality follows from Lemma 5.17.

As a final note, Remark 5.9 follows from the fact that, under the assumptions of Shearer's condition, it can be established hat $\sum_{\tau \in \mathcal{W}_i} \prod_{v \in V(\tau)} \gamma_{(v)} \leq \frac{q^{\{i\}}(b)}{q^{\emptyset}(b)}$. (See [70, 57, 71, 59] for more details.)

5.5 Proof of Theorem 5.5

We will assume without loss of generality that our algorithm follows a deterministic canonical flaw choice strategy. This is because randomized flaw choice strategies can equivalently be interpreted as convex combination of deterministic ones, and hence their analysis is equivalent to taking an expectation over deterministic strategies.

Theorem 5.5 will follow from Lemmas 5.18 and 5.19.

Lemma 5.18. *Let $\mathcal{A}$ be a commutative algorithm with respect to $\sim$. Then for each witness $W = (w_1, \ldots, w_t)$,*

$$\Pr\left[\textit{The witness of the trajectory of } \mathcal{A} \textit{ has } W \textit{ as prefix}\right] \leq \left\| \theta^{\top} \prod_{i=1}^{t} A_{w_i} \right\|_1 \ .$$

For a given witness tree τ and an arbitrary permutation π of $\{1, 2, \ldots, m\}$, let $\chi_\pi(\tau)$ be the *ordered* witness tree that is induced by ordering the children of each node in τ from left to right, increasingly according to π.

Lemma 5.19. *Let $\mathcal{A} = \mathcal{A}(\mathcal{S})$ be a commutative algorithm with respect to $\sim$ that follows a canonical flaw choice strategy $\mathcal{S}$ and let τ be a witness tree. Then there exists a canonical flaw choice strategy $\mathcal{S}'$ such that, for algorithm $\mathcal{A}' = \mathcal{A}(\mathcal{S}')$,*

$$\Pr_{\mathcal{A}}[\tau] \leq \Pr_{\mathcal{A}'}[\textit{The witness of the trajectory of } \mathcal{A}' \textit{ has } ((v_1), (v_2), \ldots, (v_{|\tau|})) \textit{ as prefix}] \ ,$$

where $v_1, v_2, \ldots v_{|\tau|}$ are the vertices of $\chi_\pi(\tau)$ in backward breadth first order.

Proof of Theorem 5.5. Define $q = \theta^{\top} \prod_{i=1}^{|\tau|} A_{[v_i]}$ and let $\mathcal{A}'$ be the algorithm promised by Lemma 5.19 when applied to $\mathcal{A}$. Applying Lemmata 5.18 and 5.19 we obtain

$$\Pr[\tau] \leq \Pr\left[\text{The witness of the trajectory of } \mathcal{A}' \text{ has } ((v_1), \ldots, (v_{|\tau|})) \text{ as prefix}\right] \leq \sum_{\sigma' \in \Omega} q(\sigma') \ . \tag{5.5}$$

Now the righthand side of (5.5) can be written as

$$\sum_{\sigma' \in \Omega} \theta^\top M^{-1} \left(\prod_{i=1}^{|\tau|} M A_{(v_i)} M^{-1} \right) M e_{\sigma'}^\top \leq \left\| \theta^\top M^{-1} \prod_{i=1}^{|\tau|} M A_{(v_i)} M^{-1} \right\|_* \sum_{\sigma' \in \Omega} \left\| M e_{\sigma'}^\top \right\| \quad (5.6)$$

$$\leq \|\theta^\top M^{-1}\|_* \prod_{i=1}^{|\tau|} \|M A_{(v_i)} M^{-1}\| \quad (5.7)$$

$$= \lambda_{\text{init}} \prod_{i=1}^{|\tau|} \gamma_{(v_i)} \ , \quad (5.8)$$

where to get (5.6) we apply (8.3), to get (5.7) we first apply (8.3) and then (8.2) and that $\sum_{\sigma' \in \Omega} \|M e_{\sigma'}\| = 1$, and, finally, (5.8) holds by the definitions of λ_{init} and γ_i. Note that (5.8) concludes the proof. □

5.5.1 Proof of Lemma 5.18

Given a deterministic flaw choice strategy $\mathcal{S} = (s_1, s_2 \ldots,)$, an integer $t \geq 1$ and $i \in \{1, 2, \ldots, m\}$, define matrix A_{i,s_t} by $A_{i,s_t}[\sigma, \sigma'] = A_i[\sigma, \sigma'] = \rho_i(\sigma, \sigma')$ if $s_t(\sigma) = i$, and $A_{i,s_t}[\sigma, \sigma'] = 0$ otherwise. Moreover, let $\mathcal{P}(W)$ denote the set of witnesses that can occur in an execution of $\mathcal{A}$ and have $W = (w_1, w_2, \ldots, w_t)$ as prefix.

With this notation, we can express the probability that the witness of the trajectory of $\mathcal{A}$ has W as a prefix as

$$\left\| \theta^\top \prod_{i=1}^{t} A_{w_i,s_i} \sum_{W' \in \mathcal{P}(W)} \prod_{j=t+1}^{|W'|} A_{w'_j,s_j} \right\|_1 \ . \quad (5.9)$$

Define $q = \theta^\top \prod_{i=1}^{t} A_{w_i,s_i}$, and for any state σ let $\mathcal{P}_\sigma(W)$ denote the subset of witnesses of $\mathcal{P}(W)$ that have $s_{t+1}(\sigma)$ as their $(t+1)$ element. With this notation we can write (5.9) as

$$\begin{aligned}
\sum_{\sigma \in \Omega} \sum_{\sigma' \in \Omega} q(\sigma') e_{\sigma'} \sum_{W' \in \mathcal{P}(W)} \prod_{j=t+1}^{|W'|} A_{w_j,s_j} e_\sigma &= \sum_{\sigma \in \Omega} \sum_{\sigma' \in \Omega} q(\sigma') \sum_{W' \in \mathcal{P}_{\sigma'}(W)} e_{\sigma'} \prod_{j=t+1}^{|W'|} A_{w'_j,s_j} e_\sigma \\
&\leq \sum_{\sigma' \in \Omega} q(\sigma') \sum_{\sigma \in \Omega} e_{\sigma'} \sum_{W' \in \mathcal{P}_{\sigma'}(W)} \prod_{j=t+1}^{|W'|} A_{w'_j,s_j} e_\sigma \\
&\leq \sum_{\sigma' \in \Omega} q(\sigma') \leq \|\theta^\top \prod_{i=1}^{t} A_{w_i}\|_1 \ ,
\end{aligned}$$

where the first equality follows from the fact that at step $t+1$ the algorithm addresses flaw $f_{s_{t+1}(\sigma')}$, and the last inequality follows from the fact that

$$\sum_{\sigma \in \Omega} e_{\sigma'} \sum_{W' \in \mathcal{P}_{\sigma'}(W)} \prod_{j=t+1}^{|W'|} A_{w'_j,s_j} e_\sigma = \left\| e_{\sigma'} \sum_{W' \in \mathcal{P}_{\sigma'}(W)} \prod_{j=t+1}^{|W'|} A_{w'_j,s_j} \right\|_1$$

is at most the probability that $\mathcal{A}$ starts from state σ' and ends at some state in Ω and is therefore at most 1.

5.5.2 Proof of Lemma 5.19

Recall the definition of canonical flaw choice strategies $\mathcal{S} = (s_1, s_2, \ldots)$, as well as the definition of A_{i,s_t} for $i \in \{1, 2, \ldots, m\}$ and $t \geq 1$. Let $\mathcal{W}(\tau)$ denote the set of witnesses that can occur in a trajectory of $\mathcal{A}(\mathcal{S})$ and for which τ occurs. That is, for each $W \in \mathcal{W}(\tau)$ there exists $t \in \{1, \ldots, |W|\}$ so that $\tau_W(t) = \tau$. We can express the probability of τ occurring in an execution of $\mathcal{A}$ as

$$\Pr[\tau] = \left\| \sum_{W \in \mathcal{W}(\tau)} \theta^\top \prod_{i=1}^{|W|} A_{w_i,s_i} \right\|_1 .$$

The idea now will be to gradually transform $\mathcal{S}$ and $\mathcal{W}(\tau)$ to the canonical flaw choice strategy $\mathcal{S}' = (s'_1, s'_2, \ldots)$ and set $\mathcal{W}'(\tau)$, respectively, so that every witness in $\mathcal{W}'(\tau)$ has $((v_1), (v_2), \ldots, (v_{(|\tau|)}))$ as a prefix, and

$$\Pr[\tau] = \left\| \sum_{W \in \mathcal{W}(\tau)} \theta^\top \prod_{i=1}^{|W|} A_{w_i,s_i} \right\|_1 \leq \left\| \sum_{W' \in \mathcal{W}'(\tau)} \theta^\top \prod_{i=1}^{|W'|} A_{w'_i,s'_i} \right\|_1 . \tag{5.10}$$

This will suffice to prove the lemma since if $\mathcal{A}' = \mathcal{A}(\mathcal{S}')$ we have that

$$\left\| \sum_{W' \in \mathcal{W}'(\tau)} \theta^\top \prod_{i=1}^{|W'|} A_{w'_i,s'_i} \right\|_1 \leq \Pr_{\mathcal{A}'} \left[\text{The witness of the trajectory of } \mathcal{A}' \text{ has } ((v_1), (v_2), \ldots, (v_{|\tau|})) \text{ as prefix}\right] .$$

We first define the elementary operation for transforming $(\mathcal{S}, \mathcal{W}(\tau))$ to $(\mathcal{S}', \mathcal{W}'(\tau))$. For $S \subseteq \{1, \ldots, m\}$ recall that we define $\pi(S)$ to be the lowest indexed integer according to π in S. Given an integer $1 \leq i \leq m$ we define $p_i : \Omega \to \{1, 2, \ldots, m\}$ to be the following function. For every state σ:

$$p_i(\sigma) = \begin{cases} i & \text{if } \sigma \in f_i \\ \pi(U(\sigma)) & \text{otherwise} \end{cases} .$$

In words, $p_i(\sigma)$ always gives priority to flaw f_i, unless f_i is not present in σ in which case it selects the flaw corresponding to the lowest index in $U(\sigma)$. We also define function Swap that takes as input a canonical flaw choice strategy $\mathcal{S}_1 = (s_1, \ldots, s_{i-1}, s_i, \ldots)$, a set of witnesses $\mathcal{W}_1$, a witness $W = (w_1, w_2 \ldots w_t) \in \mathcal{W}_1$ and an integer $i \in \{1, 2, \ldots, |W|\}$ such that $w_{i-1} \nsim w_i$, and outputs $(\mathcal{S}_2, \mathcal{W}_2)$ defined as follows:

- $\mathcal{S}_2 = (s_1, \ldots, p_{w_i}, p_{w_{i-1}}, s_{i+1}, \ldots)$;
- $\mathcal{W}_2$ is obtained from $\mathcal{W}_1$ by changing every witness $W' = (w'_1, w'_2, \ldots, w'_{i-1}, w'_i, \ldots) \in \mathcal{W}_1$ such that $w'_i = w_i$ and $w'_{i-1} = w_{i-1}$ to $W'' = (w'_1, w'_2, \ldots, w'_i, w'_{i-1}, \ldots)$.

We now describe the algorithm that achieves the transformation of $(\mathcal{S}, \mathcal{W})$ to $(\mathcal{S}', \mathcal{W}')$ and then prove its correctness. For a witness sequence $W \in \mathcal{W}(\tau)$ and a vertex v_i of τ, let $p_W(v_i)$ denote the position of the element that corresponds to $[v_i]$ in W. The algorithm is as follows.

1. Set $\mathcal{S}' \leftarrow \mathcal{S}$ and $\mathcal{W}'(\tau) \leftarrow \mathcal{W}(\tau)$
2. For $i = 1$ to $|\tau|$
   - While there exist $W \in \mathcal{W}'(\tau)$ for which $p_W(v_i) \neq i$
     - $(\mathcal{S}', \mathcal{W}'(\tau)) \leftarrow \mathrm{Swap}\left(\mathcal{S}', \mathcal{W}'(\tau), W, p_W(v_i)\right)$

First notice that if the algorithm terminates then each witness $W \in \mathcal{W}'(\tau)$ has $((v_1), (v_2), \ldots, (v_{|\tau|}))$ as a prefix. Moreover, it always terminates because at every step, if $W = (w_1, \ldots, w_t)$ is the input of Swap, then $w_{p_W(i)} \nsim w_{p_W(i)-1}$. To see this, observe that the fact that there exists $t \in \{1, 2, \ldots, |W|\}$ such that $\tau_W(t) = \tau$ implies that $f_{w_{p_W(v_i)-1}}$ is either a flaw that appears in τ and corresponds to a vertex at the same level with $f_{w_{p_W(v_i)}}$ or doesn't appear in τ. Proposition 3.7 and the definition of the algorithm for constructing witness trees guarantee that in both these cases we have $w_{p_W(i)} \nsim w_{p_W(i)-1}$.

Finally, we need to show that (5.10) is true. To that end, we prove the following invariant regarding Swap, concluding the proof.

Lemma 5.20. *Fix any set of witnesses $\mathcal{W}$ and canonical flaw choice strategy $\mathcal{S}$. For some $W^* \in \mathcal{W}$ and $k \in \{1, \ldots, |W^*|\}$, let $(\mathcal{S}', \mathcal{W}') = \mathrm{Swap}(\mathcal{S}, \mathcal{W}, W^*, k)$. Then,*

$$\left\| \sum_{W \in \mathcal{W}(\tau)} \theta^\top \prod_{i=1}^{|W|} A_{w_i, s_i} \right\|_1 \leq \left\| \sum_{W' \in \mathcal{W}'(\tau)} \theta^\top \prod_{i=1}^{|W'|} A_{w'_i, s'_i} \right\|_1 .$$

5.5.3 Proof of Lemma 5.20

We start with a key observation.

Lemma 5.21. *Consider an arbitrary pair of functions $s_1, s_2 : \Omega \to \{1, \ldots, m\}$ that take as input a state σ and output an index in $U(\sigma)$. Let $i \nsim j$ be two arbitrary indices in $\{1, \ldots, m\}$. Then, for every pair of states $\sigma_1, \sigma_2 \in \Omega$ we have that $A_{i,s_1} A_{j,s_2}[\sigma_1, \sigma_2] \leq A_{j,p_j} A_{i,p_i}[\sigma_1, \sigma_2]$.*

Proof.

$$A_{i,s_1} A_{j,s_2}[\sigma_1, \sigma_2] \leq \sum_{\sigma \in \alpha(i, \sigma_1)} \rho_i(\sigma_1, \sigma) \rho_j(\sigma, \sigma_2) = \sum_{\sigma \in \alpha(j, \sigma_1)} \rho_j(\sigma_1, \sigma) \rho_i(\sigma, \sigma_2) = A_{j,p_j} A_{i,p_i}[\sigma_1, \sigma_2] \ .$$

where the first equality holds because $A_i A_j = A_j A_j$ (since $i \nsim j$) and the second equality holds by the definitions of p_i, p_j. □

Let $W^* = (w_1^*, w_2^*, \ldots, w_{i-1}^*, w_i^*, \ldots)$ and define $A(W^*) \subseteq \mathcal{W}$ to be the set of witnesses that will be affected by Swap, i.e., the set of witnesses whose $(i-1)$-th and i-th elements are w_{i-1}^* and w_i^*, respectively. Finally, let $I_A(W^*)$ be the subset of $\mathcal{W}'$ that is the image of $A(W^*)$. We consider two cases.

In the first case, we assume that the mapping from $\mathcal{W}$ to $\mathcal{W}'$ is bijective. Given a witness sequence $W = (w_1, \ldots, w_{i-1}^*, w_i^*, \ldots, w_t) \in A(W^*)$, let $W' = (w_1, \ldots, w_i^*, w_{i-1}^*, \ldots, w_t) \in$

$I_A(W^*)$ be the corresponding witness according to the bijection. By Lemma 5.21 and the definition of $\mathcal{S}'$ we obtain

$$\left\|\theta^\top A_{w_1,s_1}\dots A_{w_{i-1},s_{i-1}}A_{w_i,s_i}\dots\right\|_1 \leq \left\|\theta^\top A_{w_1,s_1}\dots A_{w_i,p_{w_i}}A_{w_{i-1},p_{w_{i-1}}}\dots\right\|_1 \ ,$$

which concludes the proof for this case.

In the second case, the mapping from $\mathcal{W}$ to $\mathcal{W}'$ is not bijective. Observe that this can only be the case when there exist pairs $(W_1, W_2) \in \mathcal{W}$ of the form

$$\begin{aligned} W_1 &= (w_1, w_2, \dots, w^*_{i-1}, w^*_i, \dots, w_t) \ ; \\ W_2 &= (w_1, w_2, \dots, w^*_i, w^*_{i-1}, \dots, w_t) \ , \end{aligned}$$

in which case W_1 is mapped to W_2. Thus, it suffices to show that

$$\begin{aligned} \left\|\theta^\top A_{w_1,s_1}\dots(A_{w^*_{i-1},s_{i-1}}A_{w^*_i,s_i} + A_{w^*_i,s_{i-1}}A_{w^*_{i-1},s_i})\dots A_{w_t,s_t}\right\|_1 \leq \\ \left\|\theta^\top A_{w_1,s_1}\dots A_{w^*_i,p_{w^*_i}}A_{w^*_{i-1},p_{w^*_{i-1}}}\dots A_{w_t,s_t}\right\|_1 \ . \end{aligned} \tag{5.11}$$

To prove (5.11) it suffices to show that

$$A_{w^*_{i-1},s_{i-1}}A_{w^*_i,s_i}[\sigma,\sigma'] + A_{w^*_i,s_{i-1}}A_{w^*_{i-1},s_i}[\sigma,\sigma'] \leq A_{w^*_i,p^*_{w_i}}A_{w^*_{i-1},p_{w^*_{i-1}}}[\sigma,\sigma'] \ , \tag{5.12}$$

for each pair of states (σ, σ'). If $s_{i-1}(\sigma) \notin \{w^*_{i-1}, w^*_i\}$, then (5.12) is trivially true. If, on the other hand, $s_{i-1}(\sigma) \in \{w^*_{i-1}, w^*_i\}$, then either $s_{i-1}(\sigma) = w^*_i$ and $A_{w^*_{i-1},s_{i-1}}A_{w^*_i,s_i}[\sigma,\sigma'] = 0$, or $s_{i-1}(\sigma) = w^*_{i-1}$ and $A_{w^*_i,s_{i-1}}A_{w^*_{i-1},s_i}[\sigma,\sigma'] = 0$. The proof of (5.12) in the first case follows immediately by the definitions of $p_{w_{i-1}}, p_{w_i}$ and in the second case from Lemma 5.21.

5.6 An Example: Rainbow Matchings

In an edge-colored graph $G = (V, E)$, say that $S \subseteq E$ is *rainbow* if its elements have distinct colors. In this section we consider the problem of finding rainbow matchings in complete graphs of size $2n$, where each color appears a limited amount of times.

Applying the cluster expansion condition, it can be shown [3, 57] that any edge-coloring of a complete graph of size $2n$ in which each color appears on at most $\frac{27}{128}n \approx 0.211n$ edges admits a rainbow perfect matching that can be found efficiently. Furthermore, in [71] it is shown that the resampling oracles defined by [57] for the space of matchings in a clique of even size, and which are used in this particular application, induce commutative algorithms. The latter fact will allow us to use the results of this section to further study this problem.

5.6.1 Finding Rainbow Perfect Matchings

We first formulate the problem to fit our setting and use Theorem 5.7 to show that the algorithm of [3, 57] finds a perfect rainbow matching efficiently. Assuming a multiplicative slack in the cluster expansion conditions, a running time (number of steps) of $O(n)$ can be given using the results of [3, 57, 71]. However, the best known upper bound without this assumption was given in [57] to be $O(n^2)$. Here we improve the latter to $O(n)$.

Let ϕ be any edge-coloring of K_{2n} in which each color appears on at most λn edges. Let $P = P(\phi)$ be the set of all pairs of vertex-disjoint edges with the same color in ϕ, i.e., $P = \{\{e_1, e_2\} : \phi(e_1) = \phi(e_2)\}$. Let Ω be the set of all perfect matchings of K_{2n}. For each $\{e_i, e_j\} \in P$ let

$$f_{i,j} = \{M \in \Omega : \{e_i, e_j\} \subset M\} \ .$$

Thus, an element of Ω is flawless iff it is a rainbow perfect matching. The algorithm that finds a rainbow perfect matching starts at a state of Ω chosen uniformly at random and, in every subsequent step, it chooses a flaw to address according to an arbitrary, but fixed, canonical flaw choice strategy. Algorithm 5 below describes the probability distributions $\rho_{i,j}(M, \cdot)$, where $M \in f_{i,j}$. This a special case of the implementation of a general resampling oracle with respect to the uniform measure over Ω for perfect matchings described in [57]. For the problem of rainbow matchings, the latter implies that $\gamma(f_{i,j}) = \mu(f_{i,j}) = \frac{1}{(2n-1)(2n-3)}$, where we write $\gamma(f_{i,j}) := \|MA_{(i,j)}M^{-1}\|_1$, $M = \mathrm{diag}(\mu(\sigma))$, to lighten the notation.

Algorithm 5 Probability Distribution $\rho_{i,j}(M, \cdot)$

$M' := M$, $A := \{e_1, e_2\}$, $A' := A$.
while $A' \neq \emptyset$ **do**
    Pick $(u, v) \in A'$ arbitrarily
    Pick $(x, y) \in M' \setminus A'$ uniformly at random, with (x, y) randomly ordered;
    With probability $1 - \frac{1}{2|M' \setminus A'|+1}$,
      Add $(u, y), (v, x)$ to M' and remove $(u, v), (x, y)$ from M';
    Remove (u, v) from A';
Output M'.

For a vertex v let $\Gamma(v)$ denote the set of indices of flaws that correspond to edges adjacent to v. By observing the algorithm it's not hard to verify (and is also proved in [3, 57, 71]) that the graph C over indices of flaws such that for each $(e_i = (v_1, v_2), e_j = (v_3, v_4)) \in P$ we have that

$$\Gamma(i, j) = \bigcup_{i=k}^{4} \Gamma(v_k)$$

is a causality graph. Furthermore, if $S \in \mathrm{Ind}(\Gamma(i, j))$, then for each $k \in \{1, 2, 3, 4\}$ we have that $|S \cap \Gamma(v_k)| \leq 1$. This means that $|S| \leq 4$ and, moreover, for each $j \in \{0, 1, 2, 3, 4\}$ there are at most $\binom{4}{j}(2n-1)^j(\lambda n-1)^j$ subsets $S \in \mathrm{Ind}(\Gamma(i, j))$ of size j. Choosing parameters $\psi_{i,j} = \psi = \frac{3}{4n^2}$ we have that:

$$\gamma(f_{i,j})\eta(f_{i,j}) := \gamma(f_{i,j}) \sum_{S \in \mathrm{Ind}(\Gamma(i,j))} \psi^{|S|} \leq \frac{1}{(2n-3)(2n-1)} \left(1 + (2n-1)(\lambda n - 1)\right)^4 \ ,$$

from which it can be seen that whenever $\lambda \leq \frac{27}{128}$ we have that $\gamma(f_{i,j})\eta(f_{i,j}) \leq 1$ and so the cluster expansion condition is satisfied.

Since $|P| \leq (2n)^2 \cdot (\lambda n - 1) < 4\lambda n^3$, Theorem 5.7 implies that the algorithm terminates after an expected number of $3\lambda n$ steps. Overall, we have showed the following theorem.

Theorem 5.22. *For any $\lambda \leq \frac{27}{128}$, given any edge-coloring of the complete graph on $2n$ vertices in which each color appears on at most λn edges, there exists an algorithm that terminates in an expected number of at most $3\lambda n$ steps and outputs a rainbow perfect matching.*

5.6.2 Number of Rainbow Perfect Matchings

In this subsection we use Theorem 5.11 to give an exponential lower bound on the number of perfect matchings when each color appears at most λn times, where $\lambda \leq \frac{27}{128}$, by bounding the entropy of the output distribution of the algorithm described in the previous subsection.

Theorem 5.23. *For any $\lambda \leq \frac{27}{128}$, given any edge-coloring of the complete graph on $2n$ vertices in which each color appears on at most λn edges, there exist at least [1] $e^{-3\lambda n} \cdot (2n-1)!!$ rainbow perfect matchings. Furthermore, there exists an algorithm that outputs each one of them with positive probability.*

Proof. To apply Theorem 5.11, we will need to give an upper bound for $\sum_{S \in \mathrm{Ind}([m])} \psi^{|S|}$. Similarly to applications in [56], we will find useful the following crude, but general upper bound:

$$\sum_{S \in \mathrm{Ind}([m])} \prod_{j \in S} \psi_j \leq \sum_{S \subseteq [m]} \prod_{j \in S} \psi_j \leq \prod_{i \in [m]} (1 + \psi_i) \leq \exp\left(\sum_{i \in [m]} \psi_i \right) \ . \tag{5.13}$$

Since the number of perfect matching in K_{2n} is $(2n-1)!!$ and also $|P| < 4\lambda n^3$, Theorem 5.11 and (5.13) imply that the number of rainbow perfect matchings is at least

$$\exp\left(\ln|\Omega| - \sum_{i \in [m]} \psi \right) \geq \exp\left(\ln\left((2n-1)!!\right) - 3\lambda n \right) = \frac{(2n-1)!!}{e^{3\lambda n}} \ ,$$

concluding the proof. □

5.6.3 Low Weight Rainbow Perfect Matchings

Consider an arbitrary weighting function $W : E \to \mathbb{R}$ over the edges of K_{2n}. Here we consider the problem of finding rainbow perfect matchings of low weight, where the weight of a matching is defined as the sum of weights of its edges. Clearly, there is a selection of n edges of K_{2n} whose total weight is at most $\frac{1/2}{2n-1} \sum_{e \in K_{2n}} W(e)$. We use Theorem 7.3 to show that, whenever $\lambda \leq \frac{27}{128}$, the algorithm of subsection 5.6.1 outputs a rainbow perfect matching of similar expected weight.

Theorem 5.24. *For any $\lambda \leq \frac{27}{128}$, given any edge-coloring of the complete graph on $2n$ vertices in which each color appears on at most λn edges, there exists an algorithm that outputs a perfect rainbow matching M such that*

$$\mathbb{E}[W(M)] \leq \frac{\left(1 + \frac{3}{2}\lambda\right)^2}{2n-1} \sum_{e \in K_{2n}} W(e) \ .$$

[1] Recall that $(2n-1)!! = 1 \cdot 3 \cdot \ldots \cdot (2n-1) = \frac{(2n)!}{2^n n!}$.

Proof. Let A_e be the subset of Ω that consists of the matchings that contain e. It is proven in [57], and it's also not hard to verify, that Algorithm 5 with $A = \{e\}$ is a resampling oracle for this type of flaw. Moreover, using an identical counting argument to the one in subsection 5.6.1 we get that:

$$\sum_{S \in \text{Ind}(\Gamma(A_e))} \psi^{|S|} \leq \left(1 + (2n-1)(\lambda n - 1)\psi\right)^2 \quad .$$

Applying Theorem 5.7 we get that:

$$\begin{aligned} \mathbb{E}[W(M)] &\leq \sum_{e \in K_{2n}} W(e) \Pr[A_e] \\ &\leq \sum_{e \in K_{2n}} W(e) \mu\left(A_e\right) \left(1 + (2n-1)(\lambda n - 1)\psi\right)^2 \\ &< \frac{\left(1 + \frac{3}{2}\lambda\right)^2}{2n-1} \sum_{e \in K_{2n}} W(e) \ , \end{aligned}$$

concluding the proof. □

5.6.4 Finding Rainbow Matchings with many edges

In this subsection we use Theorem 5.12 to show that whenever $\lambda < 0.5$ we can find rainbow matchings with a linear number of edges.

Theorem 5.25. *Given any edge-coloring of the complete graph on $2n$ vertices in which each color appears on at most λn edges, where $\lambda < 0.5$ and n is sufficiently large, there exists an algorithm that terminates within $O(n)$ steps in expectation and finds a rainbow matching with an expected number of edges that is at least $n \min\left(1, 0.94\sqrt[3]{\frac{2}{\lambda}} - 1\right)$.*

Proof. Let ϕ be any edge-coloring of K_{2n} in which each color appears on at most λn edges and recall the definitions of $P = P(\phi)$, Ω, and $f_{i,j} = \{M \in \Omega : \{e_i, e_j\} \subset M\}$ from the proof of Theorem 5.23.

The idea is to apply Theorem 5.12 that guarantees that we can come up with a "truncated version" $\mathcal{A}'$ of our algorithm for finding perfect rainbow matchings. In particular, if ν is the output probability distribution of $\mathcal{A}'$ and for each flaw $f_{i,j}$ we set $\psi_{i,j} = \alpha$ then:

$$\begin{aligned} \nu(f_{i,j}) &\leq \max\left(0, \gamma(f_{i,j})\eta_{i,j} - \alpha\right) \\ &\leq \max\left(0, \frac{\left(1 + (2n-1)(\lambda n - 1)\alpha\right)^4}{(2n-3)(2n-1)} - \alpha\right) \quad . \end{aligned} \tag{5.14}$$

Consider now the following strategy: We first execute algorithm $\mathcal{A}'$ to get a perfect, possibly non-rainbow, matching M of K_{2n}. Then, for each flaw $f_{i,j}$ that appears in M, we choose arbitrarily one of its corresponding edges and remove it from M, to get a non-perfect, but rainbow, matching M'. If $S = S(M')$ is the random variable that equals the size (number of edges) of M' then by setting

$$\alpha = \frac{1}{(2n-1)(\lambda n - 1)} \left(\sqrt[3]{\frac{2n-3}{4(\lambda n - 1)}} - 1\right)$$

we get:

$$\begin{aligned}
\mathbb{E}[S] &= n - \sum_{(e_i,e_j)\in P} \nu(f_{i,j}) \\
&\geq n - \max\left(0, |P|\left(\frac{(1+(2n-1)(\lambda n-1)\alpha)^4}{(2n-3)(2n-1)} - \alpha\right)\right) \\
&= n\min\left(1, 1 - \frac{4n}{2n-1}\left(1 - \frac{3}{4\cdot 2^{2/3}}\sqrt[3]{\frac{2n-3}{\lambda n-1}}\right)\right) \quad .
\end{aligned}$$

For large enough n, the latter is $\min\left(1, 0.94\sqrt[3]{\frac{2}{\lambda}} - 1\right)$. Finally, notice that (for large n) α is positive whenever $\lambda < 0.5$. □

Chapter 6

Coloring Graphs with Sparse Neighborhoods

In this chapter we present two applications of the tools we developed in the previous chapters to graph coloring. First, we show that Molloy's algorithm [76] for coloring triangle-free graphs can be analyzed using the main result of Achlioptas, Iliopoulos and Kolmogorov [4], and that it is commutative. This further allows us to prove that it can output exponentially many colorings of the input graph, a fact which was established by Iliopoulos in [59]. Second, we use the algorithmic LLL condition of Achlioptas, Iliopoulos and Sinclair [5] to give a new vertex coloring algorithm for graphs with a bounded number of triangles in the neighborhood of a vertex that uses a number of colors that matches the algorithmic barrier [1] for random graphs.

6.1 Triangle-Free Graphs

In graph coloring one is given a graph $G = (V, E)$ and the goal is to find a mapping of V to a set of q colors so that no edge in E is monochromatic. The *chromatic number*, $\chi(G)$, of G is the smallest integer q for which this is possible. Given a set $\mathcal{L}_v$ of colors for each vertex v (called a *list*), a list-coloring maps each $v \in V$ to a color in $\mathcal{L}_v$ so that no edge in E is monochromatic. A graph is q-list-colorable if it has a list-coloring no matter how one assigns a list of q colors to each vertex. The *list chromatic number*, $\chi_\ell(G)$, is the smallest q for which G is q-list-colorable. Clearly $\chi_\ell(G) \geq \chi(G)$. A celebrated result of Johansson [62] established that there exists a large constant $C > 0$ such that every *triangle-free* graph with maximum degree $\Delta \geq \Delta_0$ can be list-colored using $C\Delta/\ln\Delta$ colors. Recently, using the entropy compression method, Molloy [76] improved Johansson's result, replacing C with $(1+\epsilon)$ for any $\epsilon > 0$ and all $\Delta \geq \Delta_\epsilon$. Soon thereafter, Bernshteyn [17] established the same bound for the list chromatic number, non-constructively, via the lopsided LLL, and Iliopoulos [59] showed that the algorithm of Molloy can be analyzed using the algorithmic LLL condition of [4], avoiding the need for a problem-specific entropy compression argument.

Here we present the result of Iliopoulos and show that Molloy's algorithm can output exponentially many list-colorings of the input graph per the following theorem.

Theorem 6.1. *For every $\epsilon > 0$ there exists Δ_ϵ such that every triangle-free graph G with maximum degree $\Delta \geq \Delta_\epsilon$ has list-chromatic number $\chi_\ell(G) \leq (1+\epsilon)\frac{\Delta}{\ln\Delta}$. Furthermore, if G is a graph on*

n vertices then, for every $\eta > 0$, there exists an algorithm $\mathcal{A}$ that constructs such a coloring in polynomial time with probability at least $1 - \frac{1}{n^\eta}$. In addition, $\mathcal{A}$ is able to output e^{cn} distinct list-colorings with positive probability, where $c > 0$ is a constant that depends on ϵ and Δ.

6.1.1 The Algorithm

For each vertex $v \in V$, let N_v denote the neighbors of v. Recall that the color-list of v is denoted by $\mathcal{L}_v$. It will be convenient to treat Blank also as a color. Indeed, the algorithm will operate in the space of partial proper colorings of G and, thus,

$$\Omega \subseteq \prod_{v \in V} \{\mathcal{L}_v \cup \{\mathrm{Blank}\} \quad .$$

Let $L = \Delta^{\frac{\epsilon}{2}}$ and assume Δ is sufficiently large so that $L \geq 10$.

The initial distribution. The initial distribution θ, which is important in order to get a good lower bound on the number colorings the algorithm can output, is chosen to be the following: Fix an independent set S of G of size at least $n/(\Delta + 1)$. (This is trivial to find efficiently via a greedy algorithm). Choose one color from $\mathcal{L}_u \setminus \mathrm{Blank}$, $u \in S$, uniformly at random, and assign it to u.

The flaws. We let $L_v(\sigma) \subseteq (\mathcal{L}_v \cup \{\mathrm{Blank}\})$ be the set of colors we can assign to v in state σ without creating any monochromatic edge. We call these the *available colors for v in σ* and note that Blank is always available. For each $v \in V$, we define a flaw expressing that there are "too few available colors for v," namely

$$B_v = \{\sigma \in \Omega : |L_v(\sigma)| < L\} \quad .$$

For each color c other than Blank, let $T_{v,c}(\sigma)$ be the set of Blank neighbors of v for which c is available in σ, i.e., the vertices that may "compete" with v for color c. For each $v \in V$, we define a flaw expressing that there is "too much competition for v's available colors," namely

$$Z_v = \left\{ \sigma \in \Omega : \sum_{c \in L_v(\sigma) \setminus \mathrm{Blank}} |T_{v,c}(\sigma)| > \frac{L}{10} |L_v(\sigma)| \right\} \quad .$$

Let $F_v = B_v \cup Z_v$ and let $\Omega^+ = \Omega - \cup_{v \in V} F_v$.

Lemma 6.2 (The Second Phase [76])**.** *Given $\sigma \in \Omega^+$, a complete list-coloring of G can be found efficiently.*

Lemma 6.2 was proved in [76] via a fairly straightforward application of the Lovász Local Lemma, and can be made constructive via the Moser-Tardos algorithm. We also present its proof in Appendix 8.4.1, as it will be useful in our analysis.

Flaw choice Strategy. The algorithm can use any π-strategy, i.e., it chooses to address the lowest indexed flaw according to an arbitrary, but fixed, ordering π of the set of flaws.

The actions. To address either B_v or Z_v in σ, the algorithm performs the following, i.e., the set of actions and the distribution on them as induced by the following procedure.

1: **procedure** RESAMPLE(v, σ)
2: Assign to each colored vertex u in N_v a uniformly random color from $L_u(\sigma)$

6.1.2 Proving Termination

To prove fast converge we use Theorem 3.15. The measure μ we use for the analysis is the the uniform measure over Ω. The following key lemma is proved in Section 6.1.4.

Lemma 6.3. *For each vertex v and flaw $f \in \{B_v, Z_v\}$: $\gamma_f \leq 2\Delta^{-4}$.*

For any two vertices u, v, let $\mathrm{dist}(u, v)$ denote the length of a shortest path between u and v. Consider the symmetric causality digraph for which, for any $f_v \in \{B_v, Z_v\}$ and $f_u \in \{B_u, Z_u\}$, we have that $f_u \in \Gamma(f_v)$ iff $\mathrm{dist}(u, v) \leq 3$, and observe that its maximum degree is at most $2(\Delta^3 + 1)$. Setting $\psi_f = \psi = \frac{1}{2(\Delta^3+1)}$ for every flaw $f \in \{B_v, Z_v\}$ and applying (3.8), we obtain

$$\gamma_f \sum_{S \subseteq \Gamma(f)} \prod_{g \in S} \psi_g \leq \frac{2}{\Delta^4} \cdot 2(\Delta^3 + 1) \cdot \mathrm{e} < 4\mathrm{e} \left(\frac{1}{\Delta} + \frac{1}{\Delta^4} \right) < 1 \ ,$$

for large enough Δ. The proof of fast convergence is concluded by noticing that

$$T_0 = \log_2 \left(\max_{\sigma \in \Omega} \frac{\theta(\sigma)}{\mu(\sigma)} \right) + \log_2 \left(\sum_{S \subseteq \mathrm{Span}(\theta)} \prod_{j \in S} \psi_j \right) \leq \log_2 |\Omega| + 2n \log_2(1 + \psi) = O(n \log n) \ .$$

6.1.3 A Lower Bound on the Number of Possible Outputs

In this section it will be convenient to assume that the list of each vertex has size exactly q.

Let $\mathcal{A}_1$, $\mathcal{A}_2$ denote the first and second phase of our algorithm, respectively. The bound regarding the number of list-colorings the algorithm can output with positive probability follows almost immediately from the two following lemmas.

Lemma 6.4. *Algorithm $\mathcal{A}_1$ can output at least* $\exp\left(n\left(\frac{\ln q}{\Delta+1} - \frac{1}{\Delta^3}\right)\right)$ *flawless partial colorings with positive probability.*

Proof. It is not hard to verify that algorithm $\mathcal{A}_1$ is commutative with respect to the causality relation $\sim$ induced by neighborhoods $\Gamma(\cdot)$. To see this, notice that for any two flaws f_v, f_u and any $\sigma \in f_v \cap f_u$, invoking procedure RESAMPLE(v, σ) does not change the list of available colors of the neighbors of u. Applying Theorem 5.11 (using the crude bound we saw in (5.13)) we get that the algorithm can output at least

$$\exp\left(\ln \frac{|\Omega|}{\lambda_{\mathrm{init}}} - \sum_{f \in F} \psi_f \right) = \exp\left(\ln \frac{1}{\max_{\sigma \in \Omega} \theta(\sigma)} - \frac{2n}{2(\Delta^3 + 1)} \right) > \exp\left(n \left(\frac{\ln q}{\Delta + 1} - \frac{1}{\Delta^3} \right) \right) \tag{6.1}$$

flawless partial colorings. □

Lemma 6.5. *Suppose $\mathcal{A}_1$ can output N flawless partial colorings with positive probability. Suppose further that among these partial colorings, the ones with the lowest number of colored vertices have exactly αn vertices colored, where $\alpha \in (0,1)$. Then, $\mathcal{A}_2$ can output at least $\max\left(Nq^{-(1-\alpha)n}, \left(\frac{8L}{11}\right)^{(1-\alpha)n}\right)$ list-colorings with positive probability.*

The proof of Lemma 6.5 can be found in Appendix 8.4.

Proof of Theorem 6.1. According to Lemma 6.4 algorithm $\mathcal{A}_1$ can output at least $N := \exp\left(n\left(\frac{\ln q}{\Delta+1} - \frac{1}{\Delta^3}\right)\right)$ flawless partial colorings. Moreover, according to Lemma 6.5, algorithm $\mathcal{A}_2$ can output at least

$$\min_{\alpha\in(0,1)}\left\{\max\left(Nq^{-(1-\alpha)n}, \left(\frac{8L}{11}\right)^{(1-\alpha)n}\right)\right\}$$

distinct full-list colorings. Since $Nq^{-(1-\alpha)n}$, $\left(\frac{8L}{11}\right)^{(1-\alpha)n}$ are increasing and decreasing as functions of α, respectively, the value of α that minimizes our lower bound is the one that makes them equal, which can be seen to be

$$\alpha^* := 1 - \frac{\ln N}{n\ln(\frac{8Lq}{11})} = 1 - \frac{\frac{\ln q}{\Delta+1} - \frac{1}{\Delta^3}}{\ln q + \ln(\frac{8L}{11})} \ .$$

Therefore, algorithm $\mathcal{A}_2$ can output at least

$$\exp\left(n\left(\frac{\ln q}{\Delta+1} - \frac{1}{\Delta^3}\right)\right)\cdot q^{-(1-\alpha^*)n} = \exp\left(n\left(\frac{q}{\Delta+1} - \frac{1}{\Delta^3}\right)(1-\delta)\right) \ ,$$

list-colorings, where $\delta := \frac{1}{1+\frac{\ln(8L/11)}{\ln q}} \in (0,1)$, concluding the proof.

□

6.1.4 Proof of Lemma 6.3

It will be convenient to extend the operation of "addressing a flaw f in a state σ" to arbitrary states $\sigma \in \Omega$, i.e., not necessarily elements of f, meaning that we recolor the vertices associated with f in σ in the same way we would do it if the constraint corresponding to f was indeed violated. Consider the following random experiments.

- Address B_v in an arbitrary state $\sigma \in \Omega$ to get a state τ. Let $\Pr_\sigma[B_v]$ denote the probability that $\tau \in B_v$.
- Address Z_v in an arbitrary state $\sigma \in \Omega$ to get a state τ. Let $\Pr_\sigma[Z_v]$ denote the probability that $\tau \in Z_v$.

Our claim now is that for every $f \in \{B_v, Z_v\}$

$$\gamma_f \leq \max_{\tau\in\Omega} \Pr_\tau[f] \ .$$

To see this, observe that

$$\gamma_f = \max_{\tau\in\Omega} \sum_{\sigma\in f} \frac{\mu(\sigma)}{\mu(\tau)} \rho_f(\sigma,\tau) = \max_{\tau\in\Omega} \sum_{\sigma\in \mathrm{In}_f(\tau)} \frac{1}{|\Lambda(\sigma)|} \ ,$$

where $\Lambda(\sigma) := \prod_{u\in N_v} L_u(\sigma)$ is the cartesian product of the lists of available colors of each vertex $u \in N_v$ at state σ, and $\mathrm{In}_f(\tau)$ is the set of states $\sigma \in f$ such that $\tau \in \alpha(f,\sigma)$.

The key observation now is that $\Lambda(\tau) = \Lambda(\sigma)$ for each state $\sigma \in \mathrm{In}_f(\tau)$. This is because any transition of the form $\sigma \xrightarrow{f} \tau$ does not alter the lists of available colors of vertices $u \in N_v$, since the graph is triangle-free. Thus,

$$\gamma_f = \max_{\tau\in\Omega} \frac{|\mathrm{In}_f(\tau)|}{\Lambda(\tau)} = \max_{\tau\in\Omega} \Pr_\tau[f] \ ,$$

where the second equality follows from the fact that there is a natural bijection between $\mathrm{In}_f(\tau)$ and the set of color assignments from $\Lambda(\tau)$ to the vertices of N_v that violate the constraint related to flaw f.

The following lemma, proved in [76] concludes the proof. (We omit the proof of Lemma 6.6 here because we will show the proof of more general lemma in Section 6.2.4.)

Lemma 6.6 ([76])**.** *For every state $\sigma \in \Omega$ and every $f \in \{B_v, Z_v\}$: $\Pr_\sigma[f] < \Delta^{-4}$.*

6.2 General Graphs

In this section we present the main application of Achlioptas, Iliopoulos and Sinclair [5], who generalized Molloy's algorithm by establishing a bound on the chromatic number of arbitrary graphs, as a function of the maximum number of triangles in the neighborhood of a vertex, and giving an algorithm that produces such a coloring.

Theorem 6.7 (Informal Statement)**.** *Let G be any graph with maximum degree Δ in which the neighbors of every vertex span at most $T \geq 0$ edges between them. For every $\epsilon > 0$, if $\Delta \geq \Delta_\epsilon$ and $T \lesssim \Delta^{2\epsilon}$ then*

$$\chi(G) \leq (1+\epsilon) \frac{\Delta}{\ln\Delta - \frac{1}{2}\ln(T+1)} \ , \tag{6.2}$$

and such a vertex coloring can be found efficiently. Moreover, the theorem holds for any $T \geq 0$ if the leading constant $(1+\epsilon)$ is replaced with $(2+\epsilon)$.

As we will see, the bound (6.2) *matches* the algorithmic barrier for random graphs [1]. This implies that any improvement on the guarantee of the algorithm of [5] for $T \lesssim \Delta^{2\epsilon}$ would amount to an unexpected breakthrough in random graph theory. (Random graphs are only informative in the regime $T \lesssim \Delta^{2\epsilon}$.) For arbitrary graphs our bound is within a factor of 4 of the chromatic number, improving upon a classical result of of Alon, Krivelevich and Sudakov [10] who showed (6.2) with an unspecified (large) leading constant.

We establish Theorem 6.7 in two steps. First, we present a hybrid local search algorithm that is at the heart of Theorem 6.7, and which we analyze using Theorem 4.6. Molloy's result (and

resampling algorithms based on it) breaks down immediately in the presence of triangles; the key to our algorithm is to allow backtracking steps in order to avoid undesirable portions of the search space. Specifically, in Section 6.2.1 we establish the following theorem which is a key ingredient in the proof of Theorem 6.7. Note that, in order to comply with the standard notation used in results in the area, we express the bound on the number of triangles as Δ^2/f; the triangle-free case then corresponds to $f = \Delta^2 + 1$.

Theorem 6.8. *Let G be any graph with maximum degree Δ in which the neighbors of every vertex span at most Δ^2/f edges. For all $\epsilon > 0$, there exists Δ_ϵ such that if $\Delta \geq \Delta_\epsilon$ and $f \in [\Delta^{\frac{2+2\epsilon}{1+2\epsilon}}(\ln \Delta)^2, \Delta^2 + 1]$, then*

$$\chi_\ell(G) \leq (1+\epsilon)\Delta/\ln\sqrt{f} \ .$$

Furthermore, if G is a graph on n vertices then, for every $c > 0$, there exists an algorithm that constructs such a coloring in polynomial time with probability at least $1 - \frac{1}{n^c}$.

Theorem 6.8 is interesting for several reasons. First, random graphs suggest that it is sharp, i.e., that no efficient algorithm can color graphs satisfying the conditions of the theorem with $(1-\epsilon)\Delta/\ln\sqrt{f}$ colors. More precisely, Proposition 6.9 below, proved in Appendix 8.4.6, implies that any such algorithm would entail coloring random graphs using fewer than twice as many colors as their chromatic number. This would be a major (and unexpected) breakthrough in random graph theory, where beating this factor of two has been an elusive goal for over 30 years. Besides the lack of progress, further evidence for the optimality of this factor of two is that it corresponds precisely to a phase transition in the geometry of the set of colorings [1], known as the *shattering threshold*. Second, Theorem 6.8 establishes that it is possible to have an algorithm that is *robust* enough to apply to worst-case graphs, while at the same time matching the performance of the best known (and highly tuned) algorithms for random graphs.

Proposition 6.9. *For every $\epsilon > 0$ and $d \in (d_\epsilon \ln n, (n \ln n)^{\frac{1}{3}})$, there exist $\Delta = \Delta(d,\epsilon)$ and $f = f(d,\epsilon)$ such that with probability tending to 1 as $n \to \infty$, a random graph $G = G(n, d/n)$ satisfies the conditions of Theorem 6.8 and $\chi(G) \geq (\frac{1}{2} - \epsilon)\Delta/\ln\sqrt{f}$.*

Third, armed with Theorem 6.8, we are able to prove the following result concerning the chromatic number of arbitrary graphs. (This is a formal restatement of the second of part of Theorem 6.7.)

Theorem 6.10. *Let G be a graph with maximum degree Δ in which the neighbors of every vertex span at most Δ^2/f edges. For all $\epsilon > 0$, there exist Δ_ϵ, f_ϵ such that if $\Delta \geq \Delta_\epsilon$ and $f \in [f_\epsilon, \Delta^2+1]$, then*

$$\chi(G) \leq (2+\epsilon)\Delta/\ln\sqrt{f} \ . \tag{6.3}$$

Furthermore, if G is a graph on n vertices then, for every $c > 0$, there exists an algorithm that constructs such a coloring in polynomial time with probability at least $1 - \frac{1}{n^c}$.

As we have already said, Theorem 6.10 improves a classical result of Alon, Krivelevich and Sudakov [10] which established (6.3) with an unspecified (large) constant in place of $2 + \epsilon$. The main idea in their analysis is to break down the input graph into triangle-free subgraphs, and color each one of them separately using distinct sets of colors by applying the result of Johansson [62].

Note that even if one used Molloy's [76] recent result in place of Johansson's in this scheme, the corresponding constant would still be in the thousands. Instead, we break down the graph into subgraphs with *few* triangles per neighborhood, and use Theorem 6.8 to color the pieces. The proof of Theorem 6.10 can be found in Appendix 8.4.3.

As final remark, we note that Vu [103] proved the analogue of the main result of [10] (again with a large constant) for the list chromatic number. While we don't currently see how to sharpen Vu's result to an analogue of Theorem 6.10 for the list chromatic number using our techniques, we note that our Theorem 6.8 improves over [103] for all $f \geq \Delta^{\frac{2+2\epsilon}{1+2\epsilon}}(\ln \Delta)^2$.

6.2.1 A Hybrid Algorithm

To prove Theorem 6.8 we will generalize the algorithm of Molloy [76] for coloring triangle-free graphs. The main issue we have to address is that in the presence of triangles, the natural generalization of Molloy's algorithm introduces monochromatic edges when the neighborhood of a vertex is recolored. As a result, the existing analysis fails completely even if each vertex participates in just one triangle. To get around this problem, we introduce backtracking steps into the algorithm, whose analysis is enabled by Theorem 4.6.

Recall that for each vertex $v \in V$, N_v denotes the neighbors of v and let $E_v = \{\{u_1, u_2\} : u_1, u_2 \in N_v\}$ denote the edges spanned by them. Recall that the color-list of v is denoted by $\mathcal{L}_v$. Again, it will be convenient to treat Blank also as a color. Indeed, the initial distribution θ of our algorithm assigns all its probability mass to the state where every vertex is colored Blank. Whenever assigning a color to a vertex creates monochromatic edges, the algorithm will immediately uncolor enough vertices so that no monochromatic edge remains. Edges with two Blank endpoints are not considered monochromatic. To uncolor a vertex v, the algorithm picks a monochromatic edge e incident to v and assigns e to v instead of a color, thus also creating a record of the reason for the uncoloring. Thus,

$$\Omega \subseteq \prod_{v \in V} \{\mathcal{L}_v \cup \{\mathrm{Blank}\} \cup E_v\} \ .$$

Let $L = (1+\epsilon)\frac{\Delta}{\ln f} f^{-\frac{1}{2+2\epsilon}}$ and assume Δ is sufficiently large so that $L \geq 10$.

The flaws. We let $L_v(\sigma) \subseteq (\mathcal{L}_v \cup \{\mathrm{Blank}\})$ be the set of colors we can assign to v in state σ without creating any monochromatic edge. We call these the *available colors for v in σ* and note that Blank is always available. As in Molloy's algorithm, for each $v \in V$, we define a flaw expressing that there are "too few available colors for v," namely

$$B_v = \{\sigma \in \Omega : |L_v(\sigma)| < L\} \ ,$$

and, for each color c other than Blank, we let $T_{v,c}(\sigma)$ be the set of Blank neighbors of v for which c is available in σ, i.e., the vertices that may "compete" with v for color c. For each $v \in V$, we define a flaw expressing that there is "too much competition for v's available (real) colors," namely

$$Z_v = \left\{ \sigma \in \Omega : \sum_{c \in L_v(\sigma) \setminus \mathrm{Blank}} |T_{v,c}(\sigma)| > \frac{L}{10} |L_v(\sigma)| \right\} \ .$$

Finally, for each $v \in V$ and $e \in E$ we define a flaw expressing that v is uncolored (because of e), namely

$$f_v^e = \{\sigma \in \Omega : \sigma(v) = e\} \quad .$$

Let $F_v = B_v \cup Z_v \cup_{e \in E} f_v^e$ and let $\Omega^+ = \Omega - \cup_{v \in V} F_v$.

Lemma 6.11 (The Second Phase [76])**.** *Given $\sigma \in \Omega^+$, a complete list-coloring of G can be found efficiently.*

The proof of Lemma 6.11 is an a fairly standard application of the (algorithmic) LLL, showing that σ can be extended to a complete list-coloring by coloring all Blank vertices with actual colors. (It is identical to the one of Lemma 6.2 and can be found in Appendix 8.4.) Thus, the heart of the matter is reaching a state i.e., (a partial coloring) not suffering from any of the flaws specified above.

Flaw choice. The algorithm can use any π-strategy in which every B-flaw has priority over every f-flaw.

The actions. To address f_v^e at σ, i.e., to color v, the algorithm simply chooses a color from $L_v(\sigma)$ uniformly and assigns it to v. The fact that B-flaws have higher priority than f-flaws implies that there are always at least L choices.

Addressing B- and Z- flaws is significantly more sophisticated. For each vertex v, for each vertex $u \in N_v$, let $R_u^v(\sigma) \supseteq L_u(\sigma)$ comprise those colors having the property that assigning them to u in state σ creates no monochromatic edge except, perhaps, in E_v. To address either B_v or Z_v in σ, the algorithm performs the following, i.e., the set of actions and the distribution on them as induced by the following procedure.

1: **procedure** RECOLOR(v, σ)
2: Assign to each colored vertex u in N_v a uniformly random color from $R_u^v(\sigma)$
3: **while** monochromatic edges exist **do**
4: Let u be the lowest indexed vertex participating in a monochromatic edge
5: Let e be the lowest indexed monochromatic edge with u as an endpoint
6: Uncolor u by assigning e to u

Lemma 6.12. *Let $S'(v, \sigma)$ be the set of colorings that can be reached at the end of Step 2 of* RECOLOR(v, σ) *and let $S''(v, \sigma)$ be the set of possible final colorings. Then $|S'(v, \sigma)| = |S''(v, \sigma)|$.*

Proof. Since Steps 4–6 are deterministic, $|S''(v, \sigma)| \leq |S'(v, \sigma)|$. To prove that $|S''(v, \sigma)| \geq |S'(v, \sigma)|$, we will prove that if $u \in N_v$ has distinct colors in $\sigma_1', \sigma_2' \in S'$, then there exists $z \in V$ such that $\sigma_1''(z) \neq \sigma_2''(z)$. Imagine that in Step 6, we also oriented e to point away from u. Then, in the resulting partial orientation, every vertex would have outdegree at most 1 and there would be no directed cycles. Consider the (potentially empty) oriented paths starting at u in σ_1'' and σ_2'' and let z be their last common vertex. If z is uncolored, then $\sigma_1''(z) = e_1$ and $\sigma_2''(z) = e_2$, where $e_1 \neq e_2$; if z is colored, then $\sigma_i''(z) = \sigma_i'(u)$. □

6.2.2 Proving Termination

Let D_v be the set of vertices at distance 1 or 2 from v and let

$$S_v = \{B_u\}_{u\in D_v} \cup \{Z_u\}_{u\in D_v} \cup \{f_u^{\{u,w\}}\}_{u,w\in N_v} \ .$$

To lighten notation, in the following we write $\gamma^S(f)$ instead of γ_f^S. Let $q = (1+\epsilon)\frac{\Delta}{\ln\sqrt{f}} \geq 1$.

Lemma 6.13. *For every vertex $v \in V$ and edge $e \in E$,*

1. *if $S \not\subseteq S_v$, then $\gamma^S(B_v) = \gamma^S(Z_v) = \gamma^S(f_v^e) = 0$;*
2. *if $S \supseteq \{f_{u_1}^{\{u_1,u_2\}}, f_{u_2}^{\{u_1,u_2\}}\}$, then $\gamma^S(B_v) = \gamma^S(Z_v) = \gamma^S(f_v^e) = 0$.*
3. $\max_{S\subseteq F} \gamma^S(f_v^e) \leq \frac{1}{L} =: \gamma(f_v^e)$;
4. $\max_{S\subseteq F} \gamma^S(B_v) \leq 2\mathrm{e}^{-\frac{L}{6}} =: \gamma(B_v)$;
5. $\max_{S\subseteq F} \gamma^S(Z_v) \leq 3q\mathrm{e}^{-\frac{L}{60}} =: \gamma(Z_v)$,

where the charges are computed with respect to the uniform measure over Ω.

We note that while we give uniform bounds on the charges corresponding to each flaw, the analysis of our algorithm cannot be captured by the result of [4]. This is because, crucially, we exploit to our advantage the existence of primary flaws.

Before we give the proof, we first use Lemma 6.13 to derive Theorem 6.8.

Proof of Theorem 6.8. For every flaw $f \in F$, we will take $\psi_f = \gamma(f)\psi$, where $\psi > 0$ will be chosen later.

For any vertex $v \in V$, flaw $f \in \{B_v, Z_v, f_v^e\}$, and set of flaws $S \subseteq F$, Lemma (6.13) implies that $\gamma^S(f) = 0$ unless all B- and Z-flaws in S correspond to vertices in D_v, per part (1), and every edge $e \in E_v$ contributes at most one flaw to S, per part (2). Therefore, for $f \in \{B_v, Z_v, f_v^e\}$,

$$\frac{1}{\psi_f}\sum_{S\subseteq F}\gamma^S(f)\prod_{g\in S}\psi_g \leq \frac{1}{\psi}\prod_{u\in D_v}(1+\gamma(B_u)\psi)(1+\gamma(Z_u)\psi)\prod_{e=\{u_1,u_2\}\in E_v}\left(1+\gamma(f_{u_1}^e)\psi+\gamma(f_{u_2}^e)\psi\right) \ . \tag{6.4}$$

To bound the right hand side of (6.4) we use parts (3)–(5) of Lemma 6.13 along with the facts $|D_v| \leq \Delta^2+1$ and $|E_v| \leq \Delta^2/f$ to derive (6.5) below. To derive (6.6), we use the fact that $2\mathrm{e}^{-\frac{L}{6}} \leq 3q\mathrm{e}^{-\frac{L}{60}}$, since $q \geq 1$, and that $1+x \leq \mathrm{e}^x$ for all x. Thus, for $f \in \{B_v, Z_v, f_v^e\}$, we conclude

$$\begin{aligned}\frac{1}{\psi_f}\sum_{S\subseteq F}\gamma^S(f)\prod_{g\in S}\psi_g &\leq \frac{1}{\psi}\left(1+2\mathrm{e}^{-\frac{L}{6}}\psi\right)^{\Delta^2+1}\left(1+3q\mathrm{e}^{-\frac{L}{60}}\psi\right)^{\Delta^2+1}\left(1+\frac{2\psi}{L}\right)^{\frac{\Delta^2}{f}} &(6.5)\\ &\leq \frac{1}{\psi}\exp\left(\frac{2\psi\Delta^2}{fL}+6q\mathrm{e}^{-\frac{L}{60}}\psi(\Delta^2+1)\right) := \frac{1}{\psi}\exp(Q) \ . &(6.6)\end{aligned}$$

Setting $\psi = (1+\epsilon)$, we see that the right hand side of (6.6) is strictly less than 1 for all $\Delta \geq \Delta_\epsilon$, since $Q \xrightarrow{\Delta\to\infty} 0$ for all $f \in [\Delta^{\frac{2+2\epsilon}{1+2\epsilon}}(\ln\Delta)^2, \Delta^2+1]$. To see this last claim, recall that

$L = (1+\epsilon)\frac{\Delta}{\ln f} f^{-\frac{1}{2+2\epsilon}}$ and $q = (1+\epsilon)\frac{\Delta}{\ln \sqrt{f}}$, and note that $\ln f < 3 \ln \Delta$ and $f^{\frac{1+2\epsilon}{2+2\epsilon}} \geq \Delta (\ln \Delta)^{\frac{2+4\epsilon}{2+2\epsilon}}$. Thus,

$$\frac{2\psi\Delta^2}{fL} = \frac{2\Delta^2}{f\frac{\Delta}{\ln f} f^{-\frac{1}{2+2\epsilon}}} = \frac{2\Delta \ln f}{f^{\frac{1+2\epsilon}{2+2\epsilon}}} \leq \frac{2 \ln f}{(\ln \Delta)^{\frac{2+4\epsilon}{2+2\epsilon}}} \leq \frac{6 \ln \Delta}{(\ln \Delta)^{\frac{2+4\epsilon}{2+2\epsilon}}} = \frac{6}{(\ln \Delta)^{\frac{\epsilon}{1+\epsilon}}} \xrightarrow{\Delta \to \infty} 0 \ , \quad (6.7)$$

while the facts $L = \Omega(\Delta^{\frac{\epsilon}{1+2\epsilon}})$ and $q \leq (1+\epsilon)\Delta$ imply that $6qe^{-\frac{L}{60}}\psi(\Delta^2+1) \xrightarrow{\Delta \to \infty} 0$.

□

6.2.3 Proof of Lemma 6.13

Proof of part (1). Addressing B_v or Z_v by executing RECOLOR$(v, \cdot)$ only changes the color of vertices in N_v, with any resulting uncolorings being due to edges in E_v. Thus, only flaws in S_v may be introduced. Addressing f_v^e, by coloring v, trivially, can only introduce flaws B_u, Z_u, where $u \in N_v$. □

Proof of part (2). Since addressing an f-flaw never introduces another f-flaw, we only need to discuss procedure RECOLOR. Therein, vertices are uncolored serially in time, so that any time a vertex w is uncolored there exists, at the time of w's uncoloring, a monochromatic edge $e = \{w, u\}$. Therefore, an edge $e = \{u_1, u_2\}$ can never be the reason for the uncoloring of both its endpoints, i.e., $f_{u_1}^e \cap f_{u_2}^e = \emptyset$.

□

Proof of part (3). If addressing f_v^e results in τ, then the previous state σ must be the mutation of τ that results by assigning e to v. Since $\pi(\sigma) = f_v^e$ implies $\sigma \notin B_v$, it follows that $|L_v(\sigma)| \geq L$. Since colors are chosen uniformly from $L_v(\sigma)$, it follows that $\gamma(f_v^e) \leq 1/L$.

□

Proof of parts (4) and (5). Observe that every flaw corresponding to an uncolored vertex is primary since procedure RECOLOR never colors an uncolored vertex and addressing f_v^e only colors v. Thus, when computing $\gamma^S(f)$, for $f \in \{B_v, Z_v\}$ and $S \subseteq F$, we can restrict to pairs (σ, τ) such that the set of uncolored vertices in τ is exactly the union of the set of uncolored vertices in σ and the set $\{u \in N_v : f_u^e \in S\}$. Fixing $f \in \{B_v, Z_v\}$, $S \subseteq F$, and τ, let us denote by $\mathrm{In}_f^S(\tau)$ the candidate set of originating states and by $\mathcal{U}_f^S(\tau)$ their common set of uncolored vertices. Then, for any $f \in \{B_v, Z_v\}$ and any $S \subseteq F$,

$$\gamma^S(f) = \max_{\tau \in \Omega} \sum_{\sigma \in \mathrm{In}_f^S(\tau)} \rho_f(\sigma, \tau) \ . \quad (6.8)$$

To bound $\rho_f(\sigma, \tau)$ in (6.8) we recall that RECOLOR assigns to each colored vertex $u \in N_v$ a random color from $R_u^v(\sigma)$ and invoke Lemma 6.12 to derive the first equality in (6.9). For the second equality we observe that for every $u \in N_v$, the set R_u^v is determined by the colors of the vertices in $V \setminus N_v$. Since RECOLOR only changes the color of vertices in N_v, it follows that $R_u^v(\sigma) = R_u^v(\tau)$, yielding

$$\rho_f(\sigma, \tau) = \frac{1}{\prod_{u \in N_v \setminus \mathcal{U}_f^S(\tau)} |R_u^v(\sigma)|} = \frac{1}{\prod_{u \in N_v \setminus \mathcal{U}_f^S(\tau)} |R_u^v(\tau)|} := \frac{1}{\Lambda_f^S(\tau)} \ . \quad (6.9)$$

Next we bound $|\mathrm{In}_f^S(\tau)|$, as follows. First we observe that if $\sigma \in \mathrm{In}_f^S(\tau)$, then $\sigma(u) \neq \tau(u)$ implies $u \in N_v \setminus \mathcal{U}_f^S(\tau)$ and, therefore, $\sigma(u) \in R_u^v(\tau)$ since $\sigma(u) \in L_u(\sigma) \subseteq R_u^v(\sigma) = R_u^v(\tau)$. Thus, the set of τ-mutations that result by recoloring each vertex in $N_v \setminus \mathcal{U}_f^S(\tau)$ with a color from $R_u^v(\tau)$ so that the resulting state belongs in f is a superset of $\mathrm{In}_f^S(\tau)$. Denoting this last set by $\mathrm{Viol}(f,\tau)$, we conclude that

$$\gamma^S(f) = \max_{\tau\in\Omega} \frac{|\mathrm{In}_f^S(\tau)|}{\Lambda_f^S(\tau)} \leq \max_{\tau\in\Omega} \frac{|\mathrm{Viol}(f,\tau)|}{\Lambda_f^S(\tau)} = \max_{\tau\in\Omega} \Pr[\textsc{recolor}(v,\tau) \in f] \ , \tag{6.10}$$

where for the last equality we use the definition of $\Lambda_f^S(\tau)$.

Remark 6.14. *We note that expressing the sum of the transition probabilities into a state in terms of a random experiment as we do in* (6.10) *was the key technical idea of [76] in order to apply the entropy compression method. It is also the one that breaks down if we allow our algorithm to go through improper colorings.*

To conclude the proof of Lemma 6.13 we prove the following in Section 6.2.4.

Lemma 6.15. *For each vertex v and $\sigma \in \Omega$:*

1. $\Pr[\textsc{recolor}(v,\sigma) \in B_v] \leq 2\mathrm{e}^{-\frac{L}{6}}$.
2. $\Pr[\textsc{recolor}(v,\sigma) \in Z_v] \leq 3q\mathrm{e}^{-\frac{L}{60}}$.

□

6.2.4 Proof of Lemma 6.15

Our computations are similar to the ones in [76]. The following version of Chernoff Bounds will be useful:

Lemma 6.16. *Suppose $\{X_i\}_{i=1}^m \in \{0,1\}$ are boolean variables, and set $Y_i = 1 - X_i$, $X = \sum_{i=1}^m X_i$. If $\{Y_i\}_{i=1}^m$ are negatively correlated, then for any $0 < t \leq \mathbb{E}[X]$*

$$\Pr[|X - \mathbb{E}[X]| > t] < 2\exp\left(-\frac{t^2}{3\mathbb{E}[X]}\right) \ .$$

Proof of part (1). Let $v \in V$ and $\sigma \in \Omega$ be arbitrary and let $\tau \in \Omega$ be the (random) output (state) of $\textsc{recolor}(v,\sigma)$. For each color $c \in \mathcal{L}_v$, let $P_v^c = \{u \in N_v : c \in R_u^v(\sigma)\}$ and define

$$\rho(c) = \sum_{u\in P_v^c} \frac{1}{|R_u^v(\sigma)| - 1} \ .$$

Since $c \in R_u^v(\sigma)$ implies $|R_u^v(\sigma)| \geq 2$, and since $1 - 1/x > \exp(-1/(x-1))$ for $x \geq 2$, we see that

$$\mathbb{E}[|L_v(\tau)|] = 1 + \sum_{c\in\mathcal{L}_v} \prod_{u\in P_v^c} \left(1 - \frac{1}{|R_u^v(\sigma)|}\right) > \sum_{c\in\mathcal{L}_v} \prod_{u\in P_v^c} \exp\left(-\frac{1}{|R_u^v(\sigma)| - 1}\right) = \sum_{c\in\mathcal{L}_v} \mathrm{e}^{-\rho(c)} \ . \tag{6.11}$$

Also, since each $R_u^v(\sigma)$ has $|R_u^v(\sigma)| - 1$ non-Blank colors, we see that

$$Z_v := \sum_{c \in \mathcal{L}_v} \rho(c) \leq \sum_{u \in N_v} \sum_{c \in R_u^v(\sigma) \setminus \text{Blank}} \frac{1}{|R_u^v(\sigma)| - 1} \leq \Delta \ . \tag{6.12}$$

The fact that e^{-x} is convex implies that the right hand side of (6.11) is at least $|\mathcal{L}_v| \exp(-Z_v/|\mathcal{L}_v|)$. Recalling that $|\mathcal{L}_v| = q = (1+\epsilon)\frac{\Delta}{\ln \sqrt{f}}$ and combining (6.11) with (6.12) yields

$$\mathbb{E}[|L_v(\tau)|] > q\mathrm{e}^{-Z_v/q} \geq (1+\epsilon)\frac{\Delta}{\ln \sqrt{f}}\mathrm{e}^{-\Delta/q} = 2(1+\epsilon)\frac{\Delta}{\ln f} f^{-\frac{1}{2(1+\epsilon)}} = 2L \ .$$

Let X_c be the indicator variable that $c \in L_v(\tau)$ so that $|L_v(\tau)| = 1 + \sum_{c \in \mathcal{L}_v(\tau)} X_c$. It is not hard to see that the variables $Y_c = 1 - X_c$ are negatively correlated, so that applying Lemma 6.16 with $t = \frac{1}{2}\mathbb{E}[|L_v(\tau)|] > L$ yields

$$\Pr\left[|L_v(\tau)| < \tfrac{1}{2}\mathbb{E}\left[|L_v(\tau)|\right]\right] \leq 2\mathrm{e}^{-\mathbb{E}[|L_v(\tau)|]/12} < 2\mathrm{e}^{-L/6} \ .$$

□

Proof of part (2). Let $\Psi = \{c \in L_v(\sigma) : \rho(c) \geq L/20\} \setminus \text{Blank}$. The probability that $L_v(\tau)$ contains at least one color from Ψ is at most

$$\mathbb{E}\left[|L_v(\tau) \cap \Psi|\right] = \sum_{c \in \Psi} \prod_{u \in P_v^c} \left(1 - \frac{1}{|R_u^v(\sigma)|}\right) < \sum_{c \in \Psi} \prod_{u \in P_v^c} \exp\left(-\frac{1}{2(|R_u^v(\sigma)| - 1)}\right) < \sum_{c \in \Psi} \mathrm{e}^{-\rho(c)/2} \ ,$$

where we used that $c \in R_u^v(\sigma)$ implies $|R_u^v(\sigma)| \geq 2$, and that $1 - 1/x < \exp(-1/(2(x-1)))$ for $x \geq 2$. Finally note that $\sum_{c \in \Psi} \mathrm{e}^{-\rho(c)/2} \leq q\mathrm{e}^{-L/40}$ by the definition of the set Ψ.

Recall that $T_{v,c}(\tau) = \{u \in N_v : \tau(u) = \text{Blank} \text{ and } c \in L_u(\tau)\}$. Since $L_u(\tau) \subseteq R_u(\tau) = R_u(\sigma)$, it follows that $T_{v,c}(\tau) \subseteq P_v^c$ and, therefore, $\mathbb{E}\left[|T_{v,c}(\tau)|\right] \leq \sum_{u \in P_v^c} 1/|R_u^v(\sigma)| \leq \rho(c)$. Since the vertices in P_v^c are colored (and thus become Blank) independently and since $\rho(c) < L/20$ for $c \notin \Psi$, applying Lemma 6.16 with $t = L/20$ yields $\Pr\left[|T_{v,c}(\tau)| > \mathbb{E}\left[|T_{v,c}(\tau)|\right] + L/20\right] < 2\mathrm{e}^{-L/60}$. Applying the union bound over all q colors, we see that the probability there is at least one $c \notin \Psi$ for which $|T_{v,c}(\tau)| > L/10$ is at most $2q\mathrm{e}^{-L/60}$. Thus, with probability at least $1 - 3q\mathrm{e}^{-L/60}$,

$$\sum_{c \in L_v(\tau) \setminus \text{Blank}} |T_{v,c}(\tau)| = \sum_{c \in L_v(\tau) \setminus (\Psi \cup \text{Blank})} |T_{v,c}(\tau)| < \frac{L}{10}|L_v(\tau)| \ .$$

□

Chapter 7

Efficiently List-Edge Coloring Multigraphs Asymptotically Optimally

In this chapter we present the polynomial time algorithms of Iliopoulos and Sinclair [5] for the seminal results of Kahn [65, 66], who showed that the Goldberg-Seymour and List-Coloring conjectures for (list-) edge coloring multigraphs hold asymptotically. Kahn's arguments are based on the probabilistic method and are non-constructive. The key insight is to use the main result of Achlioptas, Iliopoulos and Kolmogorov [4] (see also Section 3.4) to design algorithms that exploit the fact that correlations in the probability spaces on matchings used by Kahn decay with distance.

7.1 Statement of Results and Technical Overview

In graph edge coloring one is given a (multi)graph $G(V, E)$ and the goal is to find an assignment of one of q colors to each edge $e \in E$ so that no pair of adjacent edges share the same color. The *chromatic index*, $\chi_e(G)$, of G is the smallest integer q for which this is possible. In the more general *list-edge coloring* problem, a list of q allowed colors is specified for each edge. A graph is q-list-edge colorable if it has a list-coloring no matter how the lists are assigned to each edge. The *list chromatic index*, $\chi_e^\ell(G)$, is the smallest q for which G is q-list-edge colorable.

Edge coloring is one of the most fundamental and well-studied coloring problems with various applications in computer science (e.g., [25, 39, 64, 65, 66, 83, 90, 92, 93, 94, 102]). To give just one representative example, if edges represent data packets then an edge coloring with q colors specifies a schedule for exchanging the packets directly and without node contention. In this paper we are interested in designing algorithms for efficiently edge coloring and list-edge coloring multigraphs. To formally describe our results, we need some notation.

For a multigraph G let $\mathcal{M}(G)$ denote the set of matchings of G. A *fractional edge coloring* is a set $\{M_1, \ldots, M_\ell\}$ of matchings and corresponding positive real weights $\{w_1, \ldots, w_\ell\}$, such that the sum of the weights of the matchings containing each edge is one. I.e., $\forall e \in E$, $\sum_{M_i : e \in M_i} w_i = 1$. A fractional edge coloring is a *fractional edge c-coloring* if $\sum_{M \in \mathcal{M}(G)} w_M = c$. The *fractional chromatic index* of G, denoted by $\chi_e^*(G)$, is the minimum c such that G has a fractional edge c-coloring.

Let $\Delta = \Delta(G)$ be the maximum degree of G and define $\Gamma := \max_{H \subseteq V, |H| \geq 2} \frac{|E(H)|}{\lfloor |H|/2 \rfloor}$. Both of these quantities are obvious lower bounds for the chromatic index and it is known [35] that

$\chi_e^*(G) = \max(\Delta, \Gamma)$. Furthermore, Padberg and Rao [86] show that the fractional chromatic index of a multigraph, and indeed an optimal fractional edge coloring, can be computed in polynomial time.

Goldberg and Seymour independently stated the now famous conjecture that every multigraph G satisfies $\chi_e(G) \leq \max(\Delta + 1, \lceil \chi_e^*(G) \rceil)$. In a seminal paper [65], Kahn showed that the Goldberg-Seymour conjecture holds asymptotically:

Theorem 7.1 ([65]). *For multigraphs G, $\chi_e(G) \leq (1 + o(1))\chi_e^*(G)$.*

(Here $o(1)$ denotes a term that tends to zero as $\chi_e(G) \to \infty$.) He later [66] proved the analogous result for list-edge coloring, establishing that the List Coloring Conjecture, which asserts that $\chi_e^\ell(G) = \chi_e(G)$ for any multigraph G, also holds asymptotically:

Theorem 7.2 ([66]). *For multigraphs G, $\chi_e^\ell(G) \leq (1 + o(1))\chi_e^*(G)$.*

The proofs of Kahn use the probabilistic method and are not constructive. The main contribution of this paper is to provide polynomial time algorithms for the above results, as follows:

Theorem 7.3. *For every $c > 0$, there exists an algorithm that, given a multigraph G on n vertices, constructs a $(1 + o(1))\chi_e^*(G)$-edge coloring of G with probability at least $1 - \frac{1}{n^c}$ in expected polynomial time.*

Theorem 7.4. *For every $c > 0$, there exists an algorithm that, given a multigraph G on n vertices and an arbitrary list of $q = (1+o(1))\chi_e^*(G)$ colors for each edge, constructs a q-list-edge coloring of G with probability at least $1 - \frac{1}{n^c}$ in expected polynomial time.*

Clearly, Theorem 7.4 subsumes Theorem 7.3. Moreover, in a very recent breakthrough [24], Chen, Jing and Zang proved the (non-asymptotic) Goldberg-Seymour conjecture without exploiting the arguments of Kahn. Even before this work, the results of Sanders and Steurer [92] and Scheide [94] already give polynomial time algorithms for edge coloring multigraphs asymptotically optimally, again without exploiting the arguments of Kahn. Nonetheless, we choose to present the proof of Theorem 7.3 for three reasons. First and most importantly, its proof is significantly easier than that of Theorem 7.4, while it contains many of the key ideas required for proving Theorem 7.4. Second, our algorithms and techniques are very different from those of [24, 92, 94]. Finally, as we will see, we will need to show that the algorithm of Theorem 7.3 is commutative. This fact may be of independent interest since, as shown in [71, 59] (see also Chapter 5), commutative algorithms have several nice properties: they are typically parallelizable, their output distribution has high entropy, etc.

As a final remark, we note that, to the best of our knowledge, Theorem 7.4 is the first result to give an asymptotically optimal polynomial time algorithm for list-edge coloring multigraphs.

7.1.1 Technical Overview

The proofs of Theorems 7.1 and 7.2 are based on a very sophisticated variation of what is known as the *semi-random method* (also known as the "naive coloring procedure"), which is the main technical tool behind some of the strongest graph coloring results, e.g., [62, 64, 68, 77]. The idea is to gradually color the graph in iterations, until we reach a point where we can finish the coloring

using a greedy algorithm. In its most basic form, each iteration consists of the following simple procedure (using vertex coloring as a canonical example): Assign to each vertex a color chosen uniformly at random; then uncolor any vertex which receives the same color as one of its neighbors. Using the Lovász Local Lemma (LLL) [36] and concentration inequalities, one typically shows that, with positive probability, the resulting partial proper coloring has useful properties that allow for the continuation of the argument in the next iteration. For a nice exposition of both the method and the proofs of Theorems 7.1 and 7.2, the reader is referred to [79].

The key new ingredient in Kahn's arguments is the method of assigning colors. For each color c, we choose a matching M_c from some *hard-core* distribution on $\mathcal{M}(G)$ and assign the color c to the edges in M_c. The idea is that by assigning each color exclusively to the edges of one matching, we avoid conflicting color assignments and the resulting uncolorings.

The existence of such hard-core distributions is guaranteed by the characterization of the matching polytope due to Edmonds [35] and a result by Lee [73] (also shown independently by Rabinovich et al. [91]). The crucial fact about them is that they are endowed with very useful approximate stochastic independence properties, as was shown by Kahn and Kayll in [67]. In particular, for every edge e, conditioning on events that are determined by edges far enough from e in the graph does not effectively alter the probability of e being in the matching.

The reason why this property is important is because it enables the application of the lopsided Lovász Local Lemma, i.e., condition (2.2). Notably, the breakthrough result of Moser and Tardos [80, 81] that made the LLL constructive for the vast majority of its applications does not apply in this case, as it lies outside the variable setting (recall our discussion in Section 3.3.2.) The lack of an efficient algorithm for lopsided LLL applications is the primary obstacle to making the arguments of Kahn constructive.

Our main technical contribution is the design and analysis of such algorithms. Towards this goal, we use the flaws-actions framework and, in particular, the algorithmic LLL condition (3.8) for the analysis of stochastic local search algorithms developed by Achlioptas, Iliopoulos and Kolmogorov in [4]. As we saw in Section 3.4.2, there is a connection between this condition and the lopsided LLL, in the sense that the former can be seen as the constructive counterpart of the latter. However, this observation by itself is not sufficient, since the result of [4] is a tool for analyzing a *given* stochastic local search algorithm. Thus, we are still left with the task of designing the algorithm before using it. Nonetheless, this connection provides valuable intuition on how to realize this task.

To get a feeling for the nature of our algorithms it is helpful to recall the intuition behind condition (3.8). There, the input is the stochastic local search algorithm to be analyzed and a probability measure μ over the state space of the algorithm. The goal of the algorithm is to reach a state that avoids every flaw. At a high level, the role of the measure is to gauge how efficiently the algorithm rids the state of flaws, by quantifying the trade-off between the probability that a flaw is present at some inner state of the execution of the algorithm and the number of other flaws each flaw can possibly introduce when the algorithm addresses it. In particular, the quality of the convergence criterion is affected by the *compatibility* between the measure and the algorithm.

Roughly, the states of our algorithm will be matchings in a multigraph (corresponding to color classes) and the goal will be to construct matchings that avoid certain flaws. To that end, our algorithm will locally modify each flawed matching by (re)sampling matchings in subgraphs of G according to distributions induced by the hard-core distributions used in Kahn's proof. The fact that correlations decay with distance in these distributions allows us to prove that, while the

changes are local, and hence not many new flaws are introduced at each step, the compatibility of our algorithms with these hard-core distributions is high enough to allow us to successfully apply condition (3.8).

7.2 Hard-Core Distributions on Matchings

In this section we present the necessary background on hard-core distributions on matchings.

A probability distribution ν on the matchings of a multigraph G is *hard-core* if it is obtained by associating to each edge e a positive real $\lambda(e)$ (called the *activity* of e) so that the probability of any matching M is proportional to $\prod_{e\in M}\lambda(e)$. Thus, recalling that $\mathcal{M}(G)$ denotes the set of matchings of G, and setting $\lambda(M)=\prod_{e\in M}\lambda(e)$ for each $M\in\mathcal{M}(G)$, we have

$$\nu(M)=\frac{\lambda(M)}{\sum_{M'\in\mathcal{M}(G)}\lambda(M')} \ .$$

The characterization of the matching polytope due to Edmonds [35] and a result of Lee [73] (which was also shown independently by Rabinovich et al. [91]) imply the following connection between fractional edge colorings and hard-core probability distributions on matchings. Before describing it, we need a definition.

For any probability distribution ν on the matchings of a multigraph G, we refer to the probability that a particular edge e is in the random matching as the *marginal* of ν at e. We write $(\nu_{e_1},\ldots,\nu_{e_{|E(G)|}})$ for the collection of marginals of ν at all the edges $e_i\in E(G)$.

Theorem 7.5. *There is a hard-core probability distribution ν with marginals $(\frac{1}{c},\ldots,\frac{1}{c})$ if and only if there is a fractional c'-edge coloring of G with $c'<c$, i.e., if and only if $\chi_e^*<c$.*

Kahn and Kayll [67] proved that the probability distribution promised by Theorem 7.5 is endowed with very useful approximate stochastic independence properties.

Definition 7.6. *Suppose we choose a random matching M from some probability distribution. We say that an event Q is t-*distant *from a vertex v if Q is completely determined by the choice of all matching edges at distance at least t from v. We say that Q is t-distant from an edge e if it is t-distant from both endpoints of e.*

Theorem 7.7 ([67])**.** *For any $\delta>0$, there exists a $K=K(\delta)$ such that for any multigraph G with fractional chromatic number c there is a hard-core distribution ν with marginals $(\frac{1-\delta}{c},\ldots,\frac{1-\delta}{c})$ such that*

(a) for every $e\in E(G)$, $\lambda(e)\le\frac{K}{c}$ and hence $\forall v\in V(G)$, $\sum_{e\ni v}\lambda(e)\le K$.

(b) for every $\epsilon\in(0,1)$, if we choose a matching M according to ν then, for any edge e and event Q which is t-distant from e,

$$(1-\epsilon)\Pr[e\in M]\le\Pr[e\in M\mid Q]\le(1+\epsilon)\Pr[e\in M] \ ,$$

where $t=t(\epsilon)=8(K+1)^2\epsilon^{-1}+2$.

We conclude this subsection with the result of Jerrum and Sinclair [61] for sampling from hard-core distributions on matchings. The algorithm works by simulating a rapidly mixing Markov chain on matchings, whose stationary distribution is the desired hard-core distribution ν, and outputting the final state.

Theorem 7.8 ([61], Corollary 4.3)**.** *Let G be a multigraph, $\{\lambda(e)\}_{e\in E(G)}$ a vector of activities associated with the edges of G, and ν the corresponding hard-core distribution. Let $n = |V(G)|$ and define $\lambda' = \max\{\max_{u,v\in V(G)} \sum_{e\ni\{u,v\}} \lambda(e), 1\}$. There exists an algorithm that, for any $\epsilon > 0$, runs in time $\mathrm{poly}(n, \lambda', \log \epsilon^{-1})$ and outputs a matching in G drawn from a distribution ν' such that $\|\nu - \nu'\|_{\mathrm{TV}} \le \epsilon$.*

Remark 7.9. *[61] establishes this result for matchings in (simple) graphs. However, the extension to multigraphs is immediate: make the graph simple by replacing each set of multiple edges $e_1, \ldots, e_\ell$ between a pair of vertices u, v by a single edge e of activity $\lambda(e) = \sum_i \lambda(e_i)$; then use the algorithm to sample a matching from the hard-core distribution in the resulting simple graph; finally, for each edge $e = \{u, v\}$ in this matching, select one of the corresponding multiple edges $e_i \ni \{u, v\}$ with probability $\lambda(e_i)/\sum_i \lambda(e_i)$. Note that the running time will depend polynomially on the maximum activity λ' in the simple graph, as claimed.*

7.3 Edge Coloring Multigraphs: Proof of Theorem 7.3

We follow the exposition of the proof of Kahn in [79]. The key to the proof of Theorem 7.3 is the following lemma.

Lemma 7.10. *For all $\epsilon > 0$, there exists $\chi_0 = \chi_0(\epsilon)$ such that if $\chi_e^*(G) \ge \chi_0$ then we can find $N = \lfloor \chi_e^*(G)^{\frac{3}{4}} \rfloor$ matchings in G whose deletion leaves a multigraph G' with $\chi_e^*(G') \le \chi_e^*(G) - (1+\epsilon)^{-1}N$ in expected $\mathrm{poly}(n, \ln \frac{1}{\epsilon})$ time with probability at least $1 - \frac{1}{n^c}$, for any constant $c > 0$.*

Using the algorithm of Lemma 7.10 recursively, for every $\epsilon > 0$ we can efficiently find an edge coloring of G using at most $(1+\epsilon)\chi_e^* + \chi_0$ colors as follows. First, we compute $\chi_e^*(G)$ using the algorithm of Padberg and Rao. If $\chi_e^* \ge \chi_0$, then we apply Lemma 7.10 to get a multigraph G' with $\chi_e^*(G') \le \chi_e^*(G) - (1+\epsilon)^{-1}N$. We can now color G' recursively using at most $(1+\epsilon)\chi_e^*(G') + \chi_0 \le (1+\epsilon)\chi_e^*(G) - N + \chi_0$ colors. Using one extra color for each one of the N matchings promised by Lemma 7.10, we can then complete the coloring of G, proving the claim. In the base case where $\chi_e^*(G) < \chi_0$, we color G greedily using $2\Delta - 1$ colors. The fact that $2\Delta - 1 \le 2\chi_e^* - 1 < \chi_e^* + \chi_0$ concludes the proof of Theorem 7.3 since the number of recursive calls is at most n.

7.3.1 The Algorithm

Observe that we only need to prove Lemma 7.10 for $\epsilon < \frac{1}{10}$ since, clearly, if it holds for ϵ then it holds for all $\epsilon' > \epsilon$. So we fix $\epsilon \in (0, 0.1)$ and let $c^* = \chi_e^* - (1+\epsilon)^{-1}N$. Our goal will be to delete N matchings from G to get a multigraph G' which has fractional chromatic index at most c^*.

The flaws. Let $\Omega = \mathcal{M}(G)^N$ be the set of possible N-tuples of matchings of G. For a state $\sigma = (M_1, \ldots, M_N) \in \Omega$ let G_σ denote the multigraph induced by deleting the N matchings $M_1, \ldots, M_N$ from G. For a vertex $v \in V(G_\sigma)$ we define $d_{G_\sigma}(v)$ to be the degree of v in G_σ. We now define the following flaws. For every vertex $v \in V(G)$ let

$$f_v = \left\{\sigma \in \Omega : d_{G_\sigma}(v) > c^* - \frac{\epsilon}{4}N\right\} \ .$$

For every connected subgraph H of G with an odd number of vertices, let

$$f_H = \left\{\sigma \in \Omega : H \subseteq G_\sigma, |V(H)| \leq \frac{\Delta}{(\epsilon/4)N} \text{ and } |E(H)| > \left(\frac{|V(H)|-1}{2}\right) c^*\right\} \ .$$

The following lemma states that it suffices to find a flawless state.

Lemma 7.11 ([65])**.** *Any flawless state σ satisfies $\chi_e^*(G_\sigma) \leq c^*$.*

Proof. Edmonds' characterization [35] of the matching polytope implies that the chromatic index of G_σ is less than c^* if

1. $\forall v : d_{G_\sigma}(v) \leq c^*$; and
2. $\forall H \subseteq G_\sigma$ with an odd number of vertices: $E(H) \leq \frac{|V(H)|-1}{2} c^*$.

Now clearly, addressing every flaw of the form f_v establishes condition 1. By summing degrees this also implies that for every subgraph F with an even number of vertices $|E(F)| \leq \left(\frac{|V(F)|}{2}\right) c^*$.

Moreover, any odd subgraph H can be split into a connected component H' with an odd number of vertices, and a subgraph F with an even number of vertices. Thus, in the absence of f_v flaws, it suffices to prove condition 2 for connected H. Again by summing degrees, we see that if no f_v flaw is present, then condition 2 can fail only for H with fewer than $\frac{\Delta}{(\epsilon/4)N}$ vertices, concluding the proof. □

To describe an efficient algorithm for finding flawless states we need to (i) determine the initial distribution of the algorithm and show that is efficiently samplable; (ii) show how to address each flaw efficiently; (iii) show that the expected number of steps of the algorithm is polynomial; and finally (iv) show that we can search for flaws in polynomial time, so that each step is efficiently implementable.

The initial distribution. Let $\delta = \frac{\epsilon}{4}$ and apply Theorem 7.7. Let ν be the promised hard-core probability distribution, $\lambda = \{\lambda(e)\}$ the vector of activities associated with it, and K the corresponding constant. Note that the activities $\lambda(e)$ defining ν are not readily available. However, the next lemma says that we can efficiently compute a set of activities that gives an arbitrarily good approximation to the desired distribution ν.

Lemma 7.12. *For every $\eta > 0$, there exists a $\mathrm{poly}(n, \ln\frac{1}{\eta}, \ln\frac{1}{\delta})$-time algorithm that computes a set of edge activities $\{\lambda'(e)\}_{e \in E(G)}$ such that the corresponding hard-core distribution ν' satisfies $\|\nu - \nu'\|_{\mathrm{TV}} \leq \eta$.*

Proof. Lemma 7.12 is a straightforward corollary of the main results of Singh and Vishnoi [99] and Jerrum and Sinclair [61]. Briefly, the main result of [99] states that finding a distribution that apprxoximates ν can be seen as the solution of a max-entropy distribution estimation problem which can be efficiently solved given a "generalized counting oracle" for ν. The latter is provided by [61]. □

For a parameter $\eta > 0$ and a distribution p, we say that we *η-approximately sample* from p to express that we sample from a distribution $\tilde{p}$ such that $\|p - \tilde{p}\|_{\mathrm{TV}} \leq \eta$. Set $\eta = \frac{1}{n^C}$, where C is a sufficiently large constant to be specified later, and let ν' be the distribution promised by Lemma 7.12. The initial distribution of our algorithm, θ, is obtained by η-approximately sampling N random matchings (independently) from ν'. Observe that $\|\theta - \mu\|_{\mathrm{TV}} \leq 2\eta N$, where μ denotes the probability distribution over Ω induced by taking N independent samples from ν.

Addressing flaws. For an integer $d > 0$ and a connected subgraph H let $S_{<d}(H)$ be the set of vertices within distance strictly less than d of a vertex $u \in V(H)$.

We consider the procedure RESAMPLE below which takes as input a connected subgraph H, a state σ and a positive integer $d \leq n$, and which will be used to address flaws.

---

procedure RESAMPLE(H, σ, d)
  Let $\sigma = (M_1, M_2, \ldots, M_N)$
  for $i = 1$ to N **do**
    Let $E_{i,\geq d}$ be the set of edges of M_i that do not belong to the multigraph induced by $S_{<d+1}(H)$
    Let $E_{i,=d}$ be the set of edges of M_i whose both endpoints are in distance d from H
    Let $V_{i,d}$ be the set of vertices of $S_{<d+1}(H)$ that belong to edges in $E_{i,\geq d} \cup E_{i,=d}$
    Let $G_{i,<d+1}$ be the multigraph induced by $S_{<d+1}(H) \setminus V_{i,d}$
    Let p be the hard-core distribution induced by $\{\lambda'(e)\}_{e \in E(G_{i,<d+1})}$.
    η-approximately sample a matching M from p
    Let $M'_i = (M_i \cap E_{i,\geq d}) \cup M$ ▷ By definition, M'_i is a matching
  Output $\sigma' = (M'_1, M'_2, \ldots, M'_N)$

Notice that Theorem 7.8 implies that procedure RESAMPLE (H, σ, d) terminates in $\mathrm{poly}(n, \ln \frac{1}{\eta})$ time.

Set $t = 8(K+1)^2\delta^{-1} + 2$. To address f_v, f_H at state σ, we invoke procedures RESAMPLE $(\{v\}, \sigma, t)$ and RESAMPLE (H, σ, t), respectively.

Searching for flaws. Notice that we can compute c^* in polynomial time using the algorithm of Padberg and Rao [86]. Therefore, given a state $\sigma \in \Omega$ and c^*, we can search for flaws of the form f_v in polynomial time. However, the flaws of the form f_H are potentially exponentially many, so a brute-force search does not suffice for our purposes.

Fortunately, the result of Padberg and Rao essentially provides a polynomial time oracle for this problem as well. Recall Edmonds' characterization used in the proof of Lemma 7.11. The constraints over odd subgraphs H are called *matching constraints*. Recall further that in the proof of Lemma 7.11 we showed that, in the absence of f_v-flaws, the only matching constraints that

could possibly be violated correspond to f_H flaws. On the other hand, the oracle of Padberg and Rao, given as input $(\frac{1}{c}, \ldots, \frac{1}{c})$ and a multigraph G, can decide in polynomial time whether G has a fractional c-coloring or return a violated matching constraint. Hence, if our algorithm prioritizes f_v flaws over f_H flaws, this oracle can be used to detect the latter in polynomial time.

7.3.2 Proof of Lemma 7.10

We are left to show that the expected number of steps of the algorithm is polynomial and that each step can be executed in polynomial time. To that end, we will show that both of these statements are true assuming that the initial distribution θ is μ instead of approximately μ, and that in Lines 8, 9 of the procedure RESAMPLE(H, σ, d) we perfectly sample from the hard-core probability distribution induced by activities $\{\lambda(e)\}_{e \in E(G_{i,<d}(H))}$ instead of η-approximately sampling from p. Observe that, since we will prove that in this case the expected running time of the ideal algorithm is polynomial, we can maximally couple the approximate and ideal distributions, and then take the constant C in the definition of the approximation parameter η to be sufficiently large. The latter implies that the probability that the coupling will fail during the execution of the algorithm is negligible (i.e., at most $\frac{1}{n^c}$), establishing that the algorithm converges even if we use approximate distributions.

For an integer $d > 0$ and a vertex v, let $S^*_d(v)$ be the set of flaws indexed by a vertex of $S_{<d}(v)$ or a set H intersecting $S_{<d}(v)$. For each set H for which we have defined f_H we let $S^*_d(H) = \bigcup_{v \in V(H)} S^*_d(v)$. For each flaw f_v we define the causality neighborhood $\Gamma(f_v) = S^*_{t+2}(v)$ and for each flaw f_H we define $\Gamma(f_H) = S^*_{t+2}(H)$, where t is as defined in the previous subsection. Notice that this is a valid choice because flaw f_v can only cause flaws in $S^*_{t+1}(v)$ and flaw f_H can only cause flaws in $S^*_{t+1}(H)$. The reason why we choose these neighborhoods to be larger than seemingly necessary is because, as we will see, with respect to this causality graph our algorithm is commutative, allowing us to apply Theorem 5.14. (We apply Theorem 5.14 using the $\|\cdot\|_1$-norm and a diagonal matrix that corresponds to a probability distribution, so that condition (5.14) is equivalent to (3.8)).

Lemma 7.13. *Let $f \in \{f_v, f_H\}$ for a vertex v and a connected subgraph H of G with an odd number of vertices and let $D = \Delta^{t+\Delta^{\frac{1}{3}}+4}$. For every $\zeta > 0$ there exists Δ_ζ such that if $\Delta \geq \Delta_\zeta$ then*

(a) $\gamma_f \leq \frac{1-\zeta}{\mathrm{e}D}$;

(b) $|\Gamma(f)| \leq D$,

where the charges are computed with respect to the measure μ and the algorithm that samples from the ideal distributions.

The proof of Lemma 7.13 can be found in Section 7.3.3. Lemma 7.14 establishes that our algorithm is commutative with respect to the causality relation $\sim$ induced by neighborhoods $\Gamma(\cdot)$. Its proof can be found in Section 7.3.4.

Lemma 7.14. *For each pair of flaws $f \nsim g$, the matrices A_f, A_g commute.*

Setting $x_f = \frac{1}{1+\max_{f' \in F} |\Gamma(f')|}$ for each flaw f, we see that condition (5.3) with $\epsilon = \zeta/2$ is implied by

$$\gamma_f \cdot \big(1 + \max_{f' \in F}(|\Gamma(f')|\big) \cdot \mathrm{e} \leq 1 - \zeta/2 \text{ for every flaw } f \ , \tag{7.1}$$

which is true for large enough Δ according to Lemma 7.13. Notice further that the causality graph induced by $\sim$ can be partitioned into n cliques, one for each vertex of G, with potentially further edges between them. Indeed, flaws indexed by subgraphs that contain a certain vertex of G form a clique in the causality graph. Combining Lemma 7.14 with the latter observation, we are able to apply Theorem 5.14 which implies that our algorithm terminates after an expected number of at most $O\big(\max_{\sigma \in \Omega} \frac{\theta(\sigma)}{\mu(\sigma)} \cdot \frac{n}{\zeta} \log \frac{n \log(1/\delta)}{\zeta}\big) = O(n \log n)$ steps. (This is because we assume that $\theta = \mu$ per our discussion above.)

This completes the proof of Lemma 7.10 and hence, as explained at the beginning of Section 7.3, Theorem 7.3 follows. It remains, however, to go back and prove Lemmas 7.13 and 7.14, which we do in the next two subsections.

7.3.3 Proof of Lemma 7.13

In this section we prove Lemma 7.13. Given a state $\sigma = (M_1, \ldots, M_N)$, a subgraph H, and $d > 0$ let

$$Q_H(d, \sigma) = (M_1 - S_{<d}(H), M_2 - S_{<d}(H), \ldots, M_N - S_{<d}(H)) \ ,$$

where we define $M - X = M \cap E(G - X)$. Moreover, let $Q^i_H(d, \sigma) = M_i - S_{<d}(H)$ denote the i-th entry of $Q_H(d, \sigma)$. Finally, let $G_{<d+1}(H)$ be the multigraph induced by $S_{<d+1}(H)$ and $\mathcal{M}^i_{d+1}(H, \sigma)$ be the set of matchings of $G_{<d+1}(H)$ that are compatible with $Q^i_H(d, \sigma)$. That is, for any matching M in $\mathcal{M}^i_{d+1}(H, \sigma)$ we have that $M \cup Q^i_H(d, \sigma)$ is also a matching of G.

Remark 7.15. *Recall the definition of the multigraph $G_{i,<d+1}$ in Line 7 of procedure* RESAMPLE *and observe that the set of matchings $\mathcal{M}^i_{d+1}(H, \sigma)$ is exactly the set of matchings of this multigraph. As we saw earlier, this implies that any hard-core distribution over $\mathcal{M}^i_{d+1}(H, \sigma)$ is efficiently samplable via the algorithm of [61]. We introduce this equivalent definition of $\mathcal{M}^i_{d+1}(H, \sigma)$ here because it will be convenient in defining events with respect to the probability space induced by μ.*

Proof of part (a). We will need the following key lemma, which was essentially proved in [65]. Its proof can be found in Appendix 8.5.1. Recall that μ is the distribution over Ω induced by taking N independent samples from ν.

Lemma 7.16. *For every $\zeta > 0$ there exists Δ_ζ such that if $\Delta \geq \Delta_\zeta$ then for any random state σ distributed according to μ,*

(i) for every flaw f_v and state $\tau \in \Omega$: $\mu(\sigma \in f_v \mid Q_v(t, \sigma) = Q_v(t, \tau)) \leq \frac{1-\zeta}{eD}$, and

(ii) for every flaw f_H and state $\tau \in \Omega$: $\mu(\sigma \in f_H \mid Q_H(t, \sigma) = Q_H(t, \tau)) \leq \frac{1-\zeta}{\mathrm{e}D}$.

We show the proof of part (a) of Lemma 7.13 only for the case of f_v- flaws, as the proof for f_H- flaws is very similar. Specifically, our goal will be to prove that

$$\gamma_{f_v} = \max_{\tau \in \Omega} \mu(\sigma \in f_v \mid Q_v(t, \sigma) = Q_v(t, \tau)) \ . \tag{7.2}$$

Lemma 7.16 then concludes the proof.

Let $x_v(\sigma) = (x_{v,1}(\sigma), \ldots, x_{v,N}(\sigma))$ denote the vector such that $x_{v,i}(\sigma) = |M_i \cap E_v|$, where E_v is the set of edges adjacent to v. Notice that $x_{v,i}(\sigma) \leq 1$ since M_i is a matching. For a vector $x \in \{0,1\}^N$ define $O(x) := \{i \in [N] : x_i = 1\}$ and observe that $\sigma \in f_v$ iff $|O(x_v(\sigma))| < d_G(v) - c^* + \frac{\epsilon}{4}N$. Define the set $X_v = \{x \in \{0,1\}^N : x = x_v(\sigma) \text{ for some } \sigma \in f_v\}$ and notice that the latter observation implies that $\sigma \in f_v$ iff $x_v(\sigma) \in X_v$. (In other words, the elements of X_v induce a partition of f_v.) Hence, for a fixed state $\tau \in \Omega$ and a random sample σ from μ, we have

$$\mu(\sigma \in f_v \mid Q_v(t,\sigma) = Q_v(t,\tau)) = \sum_{x \in X_v} \prod_{i=1}^{N} \nu\left(x_{v,i}(\sigma) = x_i \mid Q_v^i(t,\sigma) = Q_v^i(t,\tau)\right) \quad , \tag{7.3}$$

since μ corresponds to N independent samples from ν. Recall that ν is associated with a set of activities $\{\lambda(e)\}_{e \in E}$. Thus, for any vector $x \in X_v$, we obtain

$$\begin{aligned} \nu\left(x_{v,i}(\sigma) = x_i \mid Q_v^i(t,\sigma) = Q_v^i(t,\tau)\right) &= \frac{\nu\left((x_{v,i}(\sigma) = x_i) \cap (Q_v^i(t,\sigma) = Q_v^i(t,\tau))\right)}{\nu\left(Q_v^i(t,\sigma) = Q_v^i(t,\tau)\right)} \\ &= \frac{\sum_{M:|M\cap E_v|=x_i,(M-S_{<t}(v))=Q_v^i(t,\tau)} \lambda(M)}{\sum_{M:(M-S_{<t}(v))=Q_v^i(t,\tau)} \lambda(M)} \\ &= \frac{\sum_{M\in\mathcal{M}_{t+1}^i(v,\tau),|M\cap E_v|=x_i} \lambda(M)}{\sum_{M\in\mathcal{M}_{t+1}^i(v,\tau)} \lambda(M)} \quad , \end{aligned} \tag{7.4}$$

where recall that $\mathcal{M}_{t+1}^i(v,\tau)$ denotes the set of matchings of $G_{<t+1}(v)$ that are compatible with $Q_v^i(t,\tau)$. To get (7.4) we used the form of $\lambda(M)$ to cancel the contributions of edges in $Q_v^i(t,\tau)$.

We will use (7.3) and (7.4) to prove that, for σ distributed according to μ, and any state $\tau \in \Omega$,

$$\sum_{\omega \in f_v} \frac{\mu(\omega)}{\mu(\tau)} \rho_{f_v}(\omega,\tau) = \mu(\sigma \in f_v \mid Q_v(t,\sigma) = Q_v(t,\tau)) \ . \tag{7.5}$$

According to the definition of γ_{f_v}, maximizing (7.5) over $\tau \in \Omega$ yields (7.2) and completes the proof.

Fix $\tau = (M_1, M_2, \ldots, M_N) \in \Omega$. To compute the sum on the left-hand side of (7.5) we need to determine the set of states $\mathrm{In}_v(\tau) \subseteq f_v$ for which $\rho_{f_v}(\omega,\tau) > 0$. To do this, recall that given as input a state $\omega = (M_1^\omega, M_2^\omega, \ldots, M_N^\omega) \in f_v$, procedure RESAMPLE$(v,\omega,t)$ modifies one by one each matching M_i, $i \in [N]$, "locally" around v. In particular, observe that the support of the distribution for updating M_i is exactly the set $\mathcal{M}_{t+1}^i(v,\omega)$ and, hence, it has to be that $Q_v^i(t,\omega) = Q_v^i(t,\tau)$ for every $i \in [N]$ and state $\omega \in \mathrm{In}_v(\tau)$. This also implies that, for every such ω,

$$\frac{\mu(\omega)}{\mu(\tau)} = \prod_{i=1}^{N} \frac{\nu(M_i^\omega)}{\nu(M_i)} = \prod_{i=1}^{N} \frac{\lambda(M_i^\omega \cap E(G_{<t+1}(v)))}{\lambda(M_i \cap E(G_{<t+1}(v)))} \ . \tag{7.6}$$

Recall now that we have assumed that the hard-core distribution in Lines 8, 9 of RESAMPLE (v,ω,t) is induced by the ideal vector of activities λ. In particular, we have

$$\rho_{f_v}(\omega,\tau) = \prod_{i=1}^{N} \frac{\lambda(M_i \cap E(G_{<t+1}(v)))}{\sum_{M\in\mathcal{M}_{t+1}^i(v,\omega)} \lambda(M)} = \prod_{i=1}^{N} \frac{\lambda(M_i \cap E(G_{<t+1}(v)))}{\sum_{M\in\mathcal{M}_{t+1}^i(v,\tau)} \lambda(M)} \tag{7.7}$$

since $Q_v^i(t,\omega) = Q_v^i(t,\tau)$, which combined with (7.6) yields

$$\frac{\mu(\omega)}{\mu(\tau)}\rho_{f_v}(\omega,\tau) = \prod_{i=1}^{N} \frac{\lambda(M_i^\omega \cap E(G_{<t+1}(v)))}{\sum_{M\in\mathcal{M}_{t+1}^i(v,\tau)}\lambda(M)} \ . \tag{7.8}$$

Finally, recall that $X_v = \{x \in [0,1]^N : x = x_v(\omega) \text{ for some } \omega \in f_v\}$, and specifically that $\omega \in f_v$ iff $x_v(\omega) \in X_v$. For $x \in X_v$, let $\Omega_{v,x} = \{\omega : x_v(\omega) = x\}$. We now have

$$\begin{aligned}
\sum_{\omega\in f_v}\frac{\mu(\omega)}{\mu(\tau)}\rho_{f_v}(\omega,\tau) &= \sum_{x\in X_v}\sum_{\omega\in\Omega_{v,x}}\frac{\mu(\omega)}{\mu(\tau)}\rho_{f_v}(\omega,\tau) \\
&= \sum_{x\in X_v}\sum_{\omega\in\Omega_{v,x}}\prod_{i=1}^{N}\frac{\lambda(M_i^\omega \cap E(G_{<t+1}(v)))}{\sum_{M\in\mathcal{M}_{t+1}^i(v,\tau)}\lambda(M)} &(7.9)\\
&= \sum_{x\in X_v}\prod_{i=1}^{N}\sum_{\substack{\omega\in\Omega_{v,x}\\ x_{v,i}(\omega)=x_i}}\frac{\lambda(M_i^\omega \cap E(G_{<t+1}(v)))}{\sum_{M\in\mathcal{M}_{t+1}^i(v,\tau)}\lambda(M)} &(7.10)\\
&= \sum_{x\in X_v}\prod_{i=1}^{N}\frac{\sum_{M\in\mathcal{M}_{t+1}^i(v,\tau),|M\cap E_v|=x_i}\lambda(M)}{\sum_{M\in\mathcal{M}_{t+1}^i(v,\tau)}\lambda(M)} \ . &(7.11)
\end{aligned}$$

To get (7.9) we used (7.8). For (7.10) we used the fact that Ω is the product space $\mathcal{M}(G)^N$, so that the choices per matching are independent, while for (7.11) we used the definition of $x_{v,i}(\omega)$.

Combining (7.11) with (7.3) and (7.4) establishes (7.5), concluding the proof. □

Proof of part (b). To see part (b) of Lemma 7.13, first notice that every set $S_{<t+2}(v)$ has at most Δ^{t+2} elements. Moreover, the fact that $N = \lfloor \chi_e^*(G)^{3/4} \rfloor = \Theta(\Delta^{3/4})$ implies that $\frac{\Delta}{(\epsilon/4)N} \leq \Delta^{\frac{1}{3}}$ for sufficiently large Δ. So, every vertex u is in at most $\Delta^{\Delta^{\frac{1}{3}}}$ sets H corresponding to a flaw f_H. Hence, every $S_{t+2}^*(v)$ has at most $\Delta^{t+\Delta^{\frac{1}{3}}+3}$ elements. Thus, since every H for which we define $S_{t+2}^*(H)$ has fewer than Δ vertices, every $S_{t+2}^*(H)$ has less than $D = \Delta^{t+\Delta^{\frac{1}{3}}+4}$ elements. □

7.3.4 Proof of Lemma 7.14

Fix $\sigma_1 = (M_1, M_2, \ldots, M_N) \in f$ and $\sigma_2 = (M_1', M_2', \ldots, M_N') \in g$ such that $f \not\sim g$. To prove that the matrices A_f, A_g commute, we need to show that for every such pair

$$\sum_\tau \rho_f(\sigma_1,\tau)\rho_g(\tau,\sigma_2) = \sum_\tau \rho_g(\sigma_1,\tau)\rho_f(\tau,\sigma_2) \ . \tag{7.12}$$

To that end, let H_f, H_g be the subgraphs (which may consist only of a single vertex) associated with flaws f and g, respectively. Since $f \nsim g$ we have that $\min_{u\in V(H_f),v\in V(H_g)} \mathrm{dist}(u,v) \geq t+2$, where $\mathrm{dist}(u,v)$ denotes the length of the shortest path between u and v. Notice that this implies that $S_{<t+2}(H_f) \cap S_{<t+2}(H_g) = \emptyset$.

Consider a pair of transitions $\sigma_1 \xrightarrow{f} \tau$, $\tau \xrightarrow{g} \sigma_2$, where $\tau = (M_1'', \ldots, M_N'')$, and so that $\rho_f(\sigma_1,\tau) > 0$, $\rho_g(\tau,\sigma_2) > 0$. The facts that procedure RESAMPLE (σ, f, t) only modifies the input

set of matchings locally within $S_{<t+1}(H_f)$, that $\rho_g(\tau, \sigma_2) > 0$, and that $S_{<t+2}(H_f) \cap S_{<t+2}(H_g) = \emptyset$ imply that (i) $\sigma_1 \in g$; and (ii) for every $i \in [N]$, $M_i \cap (S_{<t+2}(H_g)) = M_i'' \cap (S_{<t+2}(H_g))$. Notice now that the probability distribution $\rho_g(\tau, \cdot)$ depends only on $(M_1'' \cap S_{<t+2}(H_g), \ldots, M_N'' \cap S_{t+2}(H_g))$. Hence, (i) and (ii) imply that the probability distribution $\rho_g(\sigma_1, \cdot)$ is well defined and, in addition, there exists a natural bijection b_g between the action set $a(g, \tau)$ and the action set $a(g, \sigma_1)$ so that $\rho_g(\tau, \tau') = \rho_g(\sigma_1, b_g(\tau'))$ for every $\tau' \in a(g, \tau)$. This is because both distributions are implemented by sampling from the set of matchings of the same multigraph, according to the same probability distribution.

Now let $\tau' = b_g(\sigma_2)$. A symmetric argument implies that $\tau' \in f$ and that there exists a natural bijection b_f between $a(f, \sigma_1)$ and $a(f, \tau')$ so that $\rho_f(\sigma_1, \sigma) = \rho_f(\tau', b_f(\sigma))$ for every $\sigma \in a(f, \sigma_1)$. In particular, notice that $\sigma_2 = b_f(\tau)$ and that

$$\rho_f(\sigma_1, \tau)\rho_g(\tau, \sigma_2) = \rho_g(\sigma_1, \tau')\rho_f(\tau', b_f(\tau)) = \rho_g(\sigma_1, \tau')\rho_f(\tau', \sigma_2) \ . \tag{7.13}$$

Overall, what we have shown is a bijective mapping that sends any pair of transitions $\sigma_1 \xrightarrow{f} \tau, \tau \xrightarrow{g} \sigma_2$ to a pair of transitions $\sigma_1 \xrightarrow{g} \tau', \tau' \xrightarrow{f} \sigma_2$ and which satisfies (7.13). This establishes (7.12), concluding the proof. $\square$

7.4 List-Edge Coloring Multigraphs: Proof of Theorem 7.4

In this section we review the proof of Theorem 7.2 and then prove its constructive version, Theorem 7.4.

7.4.1 A High Level Sketch of the Existential Proof

As we explained in the introduction, the non-constructive proof of Theorem 7.2 is a sophisticated version of the semi-random method and proceeds by partially coloring the edges of the multigraph in iterations, until at some point the coloring can be completed greedily. (More accurately, the method establishes the *existence* of such a sequence of desirable partial colorings.)

We will follow the exposition in [79]. In each iteration, we have a list L_e of acceptable colors for each edge e. We assume that each L_e originally has C colors for some $C \geq (1+\epsilon)\chi_e^*(G)$, where $\epsilon > 0$ is an arbitrarily small constant. For each color i, we let G_i be the subgraph of G formed by the edges for which i is acceptable. Since $G_i \subseteq G, \chi_e^*(G_i) \leq \chi_e^*(G)$. Thus, Theorem 7.7 implies that we can find a hard-core distribution on the matchings of G_i with marginals $(\frac{1}{C}, \ldots, \frac{1}{C})$ whose activity vector λ_i satisfies $\lambda_i(e) \leq \frac{K}{C}$ for all e, where $K = K(\epsilon)$ is a constant.

In each iteration, we will use the *same* activity vector λ_i to generate the random matchings assigned to color i. Of course, in each iteration we restrict our attention to the subgraph of G_i obtained by deleting the set E^* of edges colored (with any color) in previous iterations, and the endpoints of the set of edges E_i^* colored i in previous iterations. (Thus, although we use the same activity vector for each color in each iteration, the induced hard-core distributions may vary significantly.) Further, we will make sure that our distributions have the property that for each edge e, the expected number of matchings containing e is very close to 1.

We apply the lopsided LLL in the following probability space. For each color i, we choose a matching $M_i \in G_i$ from the corresponding distribution, with these choices made independently.

Next, we *activate* each edge in M_i independently with probability $\alpha := \frac{1}{\log \Delta(G)}$; we assign colors only to activated edges in order to ensure that very few edges are assigned more than one color. We then update the multigraph by deleting the colored edges, and update the lists L_e by deleting any color assigned to an edge incident to e. We give a more detailed description below.

Notice that our argument needs to ensure that (i) at the beginning of each iteration the induced hard-core distributions are such that, for each uncolored edge e, the expected number of random matchings containing e is very close to 1; and (ii) after some number of iterations, we can complete the coloring greedily.

As far as the latter condition is concerned, notice that if (i) holds throughout then, in each iteration, the probability that an edge retains a color remains close to the activation probability α. This allows us to prove that the maximum degree in the uncolored multigraph drops by a factor of about $1 - \alpha$ in each iteration. Hence, after $\log_{\frac{1}{1-\alpha}} 3K$ iterations, the maximum degree in the uncolored multigraph will be less than $\frac{\Delta}{2K}$. Furthermore, for each e and i, the probability that e is in the random matching of color i is at most $\lambda_i(e) \leq \frac{K}{C}$. Since (i) continues to hold, this implies there are at least $\frac{C}{K} > \frac{\Delta}{K}$ colors available for each edge, and so the coloring can be completed greedily. (Recall that the $C > \chi_e^*(G) \geq \Delta$.)

An Iteration.

1. For each color i, pick a matching M_i according to a hard-core probability distribution μ_i on $\mathcal{M}(G_i)$ with activities λ_i such that for some constant K:

   (a) $\forall e \in E(G), \sum_i \mu_i(e \in M_i) \approx 1$

   (b) $\forall i, e \in E(G), \lambda_i(e) \leq \frac{K}{C}$ and hence $\forall v \in V(G), \sum_{L_e \ni i} \lambda_i(e) \leq K$.

2. For each i, activate each edge of M_i independently with probability $\alpha = \frac{1}{\log \Delta(G)}$, to obtain a matching F_i. We color the edges of F_i with color i and delete $V(F_i)$ from G_i. We also delete from G_i every edge not in M_i which is in F_j for some $j \neq i$. We do not delete edges of $(M_i - F_i) \cap F_j$ from G_i. (Note that this may result in edges receiving more than one color, which is not a problem since we can always pick one of them arbitrarily at the end of the iterative procedure.)

3. Note that the expected number of edges that are both colored and removed from G_i in Step 2 is less than $\alpha|E(G_i)|$ because, although the expected number of colors retained by an edge is very close to α, some edges may be assigned more than one color. As is standard in this kind of proof, we will perform an *equalizing coin flip* for each edge e of G_i so that the probability that e is both colored and removed from G_i in either Step 2 or Step 3 is exactly α.

The *outcome* of an iteration is defined to be the choices of matchings, activations, and equalizing coin flips. Let $Q = Q_\ell$ denote the random variable that equals the outcome of the ℓ-th iteration. (In what follows, we will focus on a specific iteration ℓ and so we will omit the subscript.)

For each edge $e = (u, v)$, we define a bad event A_e as follows. Let G_i' be the multigraph obtained after carrying out the modifications to G_i in Steps 2 and 3 of the above iteration. Let $t' = 8(K+1)^2(\log \Delta)^{20} + 2$ and recall the definition of $S_{<t'}(H)$ for subgraph H. Let Z_i be a

random matching in $G'_i \cap S_{<t'}(\{u,v\})$ sampled from the hard-core probability distribution induced by activity vector λ_i. Let A_e be the event that

$$\Big|\sum_{i:G'_i \ni e} \Pr(e \in Z_i \mid Q) - \sum_{i:G_i \ni e} \Pr(e \in M_i)\Big| > \frac{1}{2(\log\Delta)^4} \ . \tag{7.14}$$

To get some intuition behind the definition of event A_e, let M'_i be a random matching in G'_i chosen according to the hard-core distribution with activities λ_i. Since correlations decay with distance, one can show that $\Pr(e \in M'_i \mid Q)$ is within a factor of $1 + \frac{1}{(\log\Delta)^{20}}$ of $\Pr(e \in Z_i \mid Q)$. Thus, according to (7.14), avoiding bad event A_e implies that $\sum_i \Pr(e \in M'_i) \approx \sum_i \Pr(e \in M_i) \approx 1$, which is what is required in order to maintain property (i) at the beginning of the next iteration. In particular, it is straightforward to see that avoiding all bad events $\{A_e\}_{e \in E(G)}$ guarantees that

$$\Big|\sum_{i:G'_i \ni e} \Pr(e \in M'_i \mid Q) - \sum_{i:G_i \ni e} \Pr(e \in M_i)\Big| \leq \frac{1}{(\log\Delta)^4} \ , \tag{7.15}$$

for sufficiently large Δ, which is what we really need. The reason we consider Z_i and not M'_i is that events defined with respect to the former are mildly negatively correlated with most other bad events, making it possible to apply the lopsided LLL.

Further, for each vertex v we define A_v to be the event that the proportion of edges incident to v which are colored in the iteration is less than $\alpha - \frac{1}{(\log\Delta)^4}$.

It can be formally shown that, if we avoid all bad events, then (i) holds, i.e., at the beginning of the next iteration we can choose new probability distributions so that for each uncolored edge e we maintain the property that the expected number of random matchings containing e is very close to 1, and, moreover, after $\log_{\frac{1}{1-\alpha}} 3K$ iterations we can complete the coloring greedily.

Theorem 7.17 ([66])**.** *Assume that* (7.15) *holds for the edge marginals of the matching distributions of iteration ℓ. Then, with positive probability, the same is true for the matching distributions of iteration $\ell + 1$.*

Theorem 7.18 ([66])**.** *If we can avoid the bad events of the first $\log_{\frac{1}{1-\alpha}} 3K$ iterations, then we can complete the coloring greedily.*

Proving Theorems 7.17, 7.18 is the heart of the proof of Theorem 7.2. The most difficult part is proving that for any $x \in V \cup E$ the probability of event A_x is very close to 0 conditioned on any choice of outcomes for distant events. (This is needed in order to apply the lopsided LLL.) Below we state the key lemma that is proven in [66], and which we will also use in the analysis of our algorithm.

Recall the definition of t' and let $t = (t')^2$. For a subgraph H, we let R_H be the random outcome of our iteration in $G - S_{<t}(H)$, i.e., R_H consists of $\bigcup_i (M_i - S_{<t}(H))$, together with the choices of the activated edges in $G - S_{<t}(H)$ which determine the $\bigcup_i (F_i - S_{<t}(H))$, and the outcomes of the equalizing coin flips for edges in this subgraph.

Lemma 7.19 ([66])**.** *For every $x \in E \cup V$ and possible choice R^*_x for R_x, there exists Δ_0 such that if $\Delta \geq \Delta_0$, then $\Pr(A_x \mid R_x = R^*_x) \leq \frac{1}{\Delta^{3(t+t'+2)}}$.*

In the next sections we will focus on providing an efficient algorithm for Theorem 7.17 which, combined with Theorem 7.18, will imply the proof of Theorem 7.4.

As a final remark, we note that detecting whether bad events $\{A_e\}_{e\in E(G)}$ are present in a state is not a tractable task since it entails the exact computation of edge marginals of hardcore distributions over matchings. In order to overcome this obstacle, we will define flaws $\{f_e\}_{e\in E(G)}$ whose absence provides somewhat weaker guarantees than ridding of bad events $\{A_e\}_{e\in E(G)}$, but nonetheless implies (7.15) for every edge. To decide whether flaw a f_e is present in a state, we will we use the results of [61] to estimate the corresponding edge marginals of random variables M_i and Z_i for every color i. Note that since we will only perform an approximation, there is the possibility to deduce that f_e is not present while in reality it is. However, our approximation will be tight enough so that, even in this case, (7.15) will still hold for every edge. We give the details forthwith.

7.4.2 The Algorithm

Let $\mathcal{U}$ denote the set of uncolored edges and $N = |\bigcup_{e\in\mathcal{U}} L_e|$, the cardinality of the set of colors that appear in the list of available colors of some uncolored edge. For a color $i \in [N]$, recall that G_i denotes the subgraph of uncolored edges that contain i in their list of available colors. Finally, let $E_i = |E(G_i)|$.

Define $\Omega = \prod_{i\in[N]} \left(\mathcal{M}(G_i) \times \{0,1\}^{E_i} \times \{0,1\}^{E_i}\right)$. We consider an arbitrary but fixed ordering over $\mathcal{U}$, so that each state $\sigma \in \Omega$ can be represented as $\sigma = ((M_1, a_1, h_1), \ldots, (M_N, a_N, h_N))$, where M_i, a_i, h_i are the matching, activation and equalizing coin flip vectors, respectively, that correspond to color i, so that edge e is activated in G_i if $a_i(e) = 1$ and is marked to be removed if $h_i(e) = 1$.

Recall that for color i we choose a matching according to probability distribution μ_i and we define $\mathrm{Eq}_i(e)$ to be the probability of success of the equalizing coin flip that corresponds to edge e and color i. Note that, given access to the marginals of μ_i, the value of $\mathrm{Eq}_i(e)$ can be computed efficiently. (Of course, we will have only (arbitrarily good) estimates of the marginals of μ_i, but as in the proof of Theorem 7.3, this suffices for our purposes.)

We let μ be the probability distribution over Ω that is induced by the product of the μ_i's, activation flips, and equalizing coin flips for each color i. In other words, μ is the probability distribution over Ω induced by the iteration.

The initial distribution. Recall that each edge e initially has a list L_e of size at least $(1+\epsilon)\chi^*_e(G)$. As we have already seen in Lemma 7.12, the results of [61, 99] imply that for every color i and parameter $\eta > 0$, there exists a $\mathrm{poly}(n, \ln\frac{1}{\eta}, \ln\frac{1}{\epsilon})$-algorithm that computes a vector λ'_i such that the induced hard-core distribution η-approximates in variation distance the hard-core distribution induced by vector λ_i. Setting $\eta = \frac{1}{n^\beta}$ for β sufficiently large, let μ' be the distribution obtained in an identical way to μ but using vectors λ'_i instead of vectors λ_i. The initial distribution θ of our algorithm is obtained by η-approximately sampling from μ'. Theorem 7.8 implies that this can be done in polynomial time.

Finding and addressing flaws. We define a flaw f_v for each bad event A_v. Moreover, for each edge e we define flaw f_e to be the set of states $\sigma \in \Omega$ such that

$$\Big| \sum_{i:G'_i \ni e} \Pr(e \in Z_i \mid \sigma) - \sum_{i:G_i \ni e} \Pr(e \in M_i) \Big| > \frac{2}{3(\log \Delta)^4} \ . \tag{7.16}$$

We fix an arbitrary ordering π over $V \cup E$. In each step, the algorithm finds the lowest indexed flaw according to π that is present in the current state and addresses it.

Clearly, checking if vertex-flaws A_v are present in the current state can be done efficiently.

For edge indexed flaws, we use the results of [61] to approximate the edge marginals of the corresponding distributions within a factor $(1+\eta)$ with probability at least $1-\eta$, in time $\mathrm{poly}(n, \ln \frac{1}{\eta})$). Recalling that $\eta = \frac{1}{n^\beta}$ and taking β to be a sufficient large constant, we can subsume this error probability into the probability that our algorithm fails.

Moreover, since, as we have already mentioned, $\Pr(e \in M'_i \mid \sigma)$ is within a factor $1 + \frac{1}{(\log \Delta)^{20}}$ of $\Pr(e \in Z_i \mid \sigma)$, for Δ and β sufficiently large, deducing that flaw f_e is not present in a state σ using our estimates for the edge marginals implies that (7.15) holds for edge e at state σ. In other words, if our algorithm decides that it has fixed every flaw, we are guaranteed that (7.15) holds for its output, even if some flaws are in fact still present.

In the opposite direction, there is the possibility that our algorithm decides that a flaw f_e is present while in reality it is not. In particular, there is a danger that, due to approximation errors, our algorithm effectively attempts to get rid of supersets $\widetilde{f}_e \supseteq f_e$ of the original flaws we defined and, as a result, fails to converge efficiently. Nonetheless, using Lemma 7.19, together with the facts that our approximations can be made arbitrarily accurate and that $A_e \subseteq f_e$ for all $e \in E$, we can still conclude that $\mu(\widetilde{f}_e \mid R_e = R^*_e) \leq \Delta^{-3(t+t'+2)}$.

Summarizing, we may and will assume without loss of generality that we are able to accurately and efficiently search for edge-flaws f_e, and that their probability with respect to measure μ is bounded above by $\Delta^{-3(t+t'+2)}$ conditional on any instantiation of R_e.

Recall the definition of t and the procedure RESAMPLE described in Section 7.3.1. Below we describe procedure FIX that takes as input a subgraph H and a state σ. In the description of FIX below we invoke procedure RESAMPLE with an extra parameter, namely an activity vector λ'_i for each color i. By that we mean that in Lines 8, 9 of RESAMPLE we use the vector λ'_i to define p.

1: **procedure** FIX(H, σ)
2: Let $\sigma = ((M_1, b_1, h_1), (M_2, b_2, h_2) \ldots, (M_N, b_N, h_N))$
3: $(M'_1, M'_2, \ldots, M'_N) \leftarrow$ RESAMPLE$(H, (M_1, M_2, \ldots, M_N), t, \lambda'_i)$
4: **for** $i = 1$ to N **do**
5: Update a_i to a'_i by activating independently each edge in $G_i \cap S_{<t+1}(H)$ with probability α
6: Update h_i to h'_i by flipping the corresponding equalizing coin for each edge in $G_i \cap S_{<t+1}(H)$
7: Output $\sigma = ((M'_1, a'_1, h'_1), (M'_2, a'_2, h'_2), \ldots, (M'_N, a'_N, h'_N))$

Theorem 7.8 implies that procedure FIX runs in polynomial time for any input subgraph H and state σ. To address flaws $f_v, f_{\{u_1,u_2\}}$ in a state σ we invoke FIX$(\{v\}, \sigma)$ and FIX$(\{u_1, u_2\}, \sigma)$, respectively.

7.4.3 Proof of Theorem 7.4

Similarly to the proof of Theorem 7.3, for our analysis we will assume that our algorithm samples from the "ideal" distributions, i.e., the ones induced by the vectors λ_i, rather than by the approximate ones λ'_i. An identical argument shows that this is sufficient if we take the exponent β in the definition of η to be large enough.

For two flaws f_{x_1}, f_{x_2}, where $x_1, x_2 \in V \cup E$, we consider the causality relation $f_{x_1} \sim f_{x_2}$ iff $\operatorname{dist}(x_1, x_2) \leq t + t' + 2$. By inspecting procedure FIX it is not hard to verify that this is a valid choice for a causality graph in the sense that no flaw f can cause flaws outside $\Gamma(f)$. This is because, in order to determine whether a flaw f_x is present in a state σ, we only need information about σ in $G \cap S_{<t'}(x)$, and procedure FIX locally modifies the state within a radius at most t of the input subgraph H.

The algorithmic proof of Theorem 7.17, which as we explained earlier is the key ingredient in making Kahn's result constructive, follows almost immediately by combining Theorem 3.15 with Lemma 7.20 below, whose proof can be found in Section 7.4.4.

Lemma 7.20. *Let $f \in \{f_e, f_v\}$ for an edge e and a vertex v. There exists Δ_0 such that if $\Delta \geq \Delta_0$ then*

$$\gamma_f \leq \frac{1}{\Delta^{3(t+t'+2)}} \ ,$$

where the charges are computed with respect to measure μ and the algorithm that samples from the ideal distributions.

Constructive Proof of Theorem 7.17. Setting $\psi_f = \frac{1}{\max_{f \in F} |\Gamma(f)|}$ for each flaw f, condition (3.8) with $\epsilon = \zeta/2$ is implied by

$$\max_{f \in F} \gamma_f \cdot \left(1 + \max_{f \in F} |\Gamma(f)|\right) \cdot \mathrm{e} \leq 1 - \zeta/2 \ . \tag{7.17}$$

Clearly, for each flaw f, $|\Gamma(f)| = O(\Delta^{2(t+t'+2)})$ so, by Lemma 7.20, condition (7.17) is satisfied for all sufficiently large Δ. Thus, Theorem 3.15 implies that, for every multigraph with large enough degree Δ_0, the algorithm for each iteration terminates after an expected number

$$O\left((m+n)\log_2\left(\frac{1}{1 - 1/\Delta^{2(t+t'+2)}}\right)\right) = O(n^2)$$

steps. □

Finally, the proof of Theorem 7.4 is concluded by combining the algorithm for Theorem 7.17 with the greedy algorithm of Theorem 7.18. It remains only for us to prove Lemma 7.20 stated above. This we do in the next subsection.

7.4.4 Proof of Lemma 7.20

Let $\Omega_1 = \prod_{i=1}^{N} \mathcal{M}(G_i)$ and $\Omega_2 = \Omega_3 = \prod_{i=1}^{N} \{0,1\}^{E_i}$ and note that each state in $\sigma \in \Omega$ can be represented as $\sigma = (\sigma_1, \sigma_2, \sigma_3) \in \Omega_1 \times \Omega_2 \times \Omega_3$. For notational convenience, sometimes we write $\Omega_1^i = \mathcal{M}(G_i)$ and $\Omega_2^i = \Omega_3^i = \{0,1\}^{E_i}$, for $i \in [N]$.

Let ν_1 be the distribution over Ω_1 induced by the product of distributions μ_i, $i \in [N]$. Let also ν_2, ν_3 be the distributions over Ω_2 and Ω_3 induced by the product of activation and equalizing coin flips of each color $i \in [N]$, respectively. Recall that $\mu = \nu_1 \times \nu_2 \times \nu_3$ is a product distribution. Moreover, note that each ν_j, is the product of N distributions ν_j^i, one for each color $i \in [N]$. For example, notice that ν_1^i is another name for μ_i, while ν_2^i is the product measure over the edges of G_i induced by flipping a coin with probability α for each edge.

For $\sigma_1 = (M_1, M_2, \ldots, M_N) \in \Omega_1$, a subgraph H, and an integer $d > 0$, we define $Q_H(d, \sigma_1) = (M_1 - S_{<d}(H), \ldots, M_N - S_{<d}(H))$ and $Q_H^i(d, \sigma_1) = M_i - S_{<d}(H)$, similarly to the proof of Lemma 7.13. Moreover, for $\sigma_2 \in \Omega_2$ that represents the outcome of the activations, we let $A_H(d, \sigma_2)$ denote the restriction of σ_2 in $M_i - S_{<d}(H)$, for each color $i \in [N]$. In the same fashion, for $\sigma_3 \in \Omega_3$ that represents the outcome of the equalizing coin flips, we let $C_H(d, \sigma_3)$ denote the restriction of σ_3 in $M_i - S_{<d}(H)$ for each color $i \in [N]$. For $\sigma_2 \in \Omega_2, \sigma_3 \in \Omega_3$, we also define $A_H^i(d, \sigma_2)$ and $C_H^i(d, \sigma_3)$, $i \in [N]$, similarly to $Q_H^i(d, \sigma_1)$. Finally, for $\sigma = (\sigma_1, \sigma_2, \sigma_3) \in \Omega$, define $R_H(d, \sigma) = (Q_H(d, \sigma_1), A_H(d, \sigma_2), C_H(d, \sigma_3))$.

Our goal will be to show that, for every $x \in V \cup E$,

$$\gamma_{f_x} = \max_{\tau \in \Omega} \mu(\sigma \in f_x \mid R_x(t, \sigma) = R_x(t, \tau)) \ , \tag{7.18}$$

where σ is a random state distributed according to μ. This is because combining (7.18) with Lemma 7.19 concludes the proof.

We only prove (7.18) for f_e-flaws, since the proof for f_v flaws is very similar (and we have actually seen a big part of it in the proof of Lemma 7.13). Observe that whether flaw f_e is present at a state σ is determined by $\bigcup_{i=1}^{N} (G_i \cap S_{<t'}(e))$ and the entries of the activation and equalizing flip vectors of each color $i \in [N]$ that correspond to edges in $G_i \cap S_{<t'}(e)$. With that in mind, for each color i let $M_i(t', e) = M_i \cap E(G_i \cap S_{<t'}(e))$ and $a_i(t', e), h_i(t', e)$ denote the (random) vectors constraining the entries of the activation and equalizing coin flip vectors for color i that correspond to the edges of $G_i \cap S_{<t'}(e)$. Let also $\mathcal{D}_i(t', e)$ denote the domain of possible values of $(M_i(t', e), a_i(t', e), h_i(t', e))$.

The fact that we can determine whether f_e is present in a state by examining local information around e implies that there exists a set $X_e = X_e(t')$ of vectors of size N such that the i-th entry of a vector $x \in X_e$ is an element of $\mathcal{D}_i(t', e)$, and so that

$$f_e = \bigcup_{x \in X_e} \bigcap_{i \in [N]} ((M_i(t', e), a_i(t', e), h_i(t', e)) = x_i) \ . \tag{7.19}$$

For a state $\sigma \in \Omega$, let x_e^σ be the N-dimensional vector whose i-th entry is $(M_i(t', e), a_i(t', e), h_i(t', e))$. According to (7.19), for $\tau \in \Omega$ we have

$$\mu(\sigma \in f_e \mid R_e(t, \sigma) = R_e(t, \tau)) = \sum_{x \in X_e} \prod_{i=1}^{N} \mu(x_{e,i}^\sigma = x_i \mid R_e(t, \sigma) = R_e(t, \tau)) \ , \tag{7.20}$$

since the random choices of matching, activation, and equalizing coin flips for each color are independent. For an N-dimensional vector x whose i-th entry is an element of $\mathcal{D}_i(t', e)$, we write $x_i(j)$ to denote the j-th element of triple x_i. Thus, recalling the definition of the distributions ν_j^i, we have

$$\mu(x_{e,i}^\sigma = x_i \mid R_e(t, \sigma) = R_e(t, \tau)) = \prod_{j=1}^{3} \nu_j^i(x_{e,i}^\sigma(j) = x_i(j) \mid R_e(t, \sigma) = R_e(t, \tau)) \ , \tag{7.21}$$

because, for a fixed color, the random choices of matching, activation and equalizing coin flips are independent.

Recall now that for a subgraph H, multigraph $G_{<d+1}(H)$ is induced by $S_{<d+1}(H)$ and $\mathcal{M}^i_{d+1}(H,\sigma)$ is the set of matchings of $G_{<d+1}(H)$ that are compatible with $Q^i_H(d,\sigma_1)$. Hence,

$$\begin{aligned}\nu^i_1(x^\sigma_{e,i}(1)=x_i(1)\mid R_e(t,\sigma)=R_e(t,\tau)) &= \nu^i_1(x^\sigma_{e,i}(1)=x_i(1)\mid Q^i_e(t,\sigma_1)=Q^i_e(t,\tau_1))\\ &= \frac{\nu^i_1(x^\sigma_{e,i}(1)=x_i(1)\cap Q^i_e(t,\sigma_1)=Q^i_e(t,\tau_1))}{\nu^i_1(Q^i_e(t,\sigma_1)=Q^i_e(t,\tau_1))}\\ &= \frac{\sum_{M\in\mathcal{M}^i_{t+1}(e,\tau_1),M\cap S_{<t'}(e)=x_i(1)}\lambda_i(M)}{\sum_{M\in\mathcal{M}^i_{t+1}(e,\tau_1)}\lambda_i(M)}. \qquad (7.22)\end{aligned}$$

Moreover, we clearly have

$$\nu^i_2(x^\sigma_{e,i}(2)=x_i(2)\mid R_e(t,\sigma)=R_e(t,\tau)) = \nu^i_2(a_i(t',e)=x_i(2))\ , \qquad (7.23)$$

$$\nu^i_3(x^\sigma_{e,i}(3)=x_i(3)\mid R_e(t,\sigma)=R_e(t,\tau)) = \nu^i_3(h_i(t',e)=x_i(3)) \qquad (7.24)$$

We will use (7.20)-(7.24) to show that, for σ distributed according to μ and any state $\tau\in\Omega$,

$$\sum_{\omega\in f_e}\frac{\mu(\omega)}{\mu(\tau)}\rho_{f_e}(\omega,\tau)=\mu(\sigma\in f_e\mid R_e(t,\sigma)=R_e(t,\tau))\ . \qquad (7.25)$$

According to the definition of γ_{f_e}, maximizing (7.25) over $\tau\in\Omega$ yields (7.18).

To compute the sum in (7.25) we need to determine the set of states $\mathrm{In}_e(\tau)=\{\omega:\rho_{f_e}(\omega,\tau)>0\}$. We claim that for each $\omega\in\mathrm{In}_e(\tau)$ we have that $R_e(t,\omega)=R_e(t,\tau)$.

To see this, let

$$\begin{aligned}\omega &= (\omega_1,\omega_2,\omega_3)=\big((\omega^1_1,\dots,\omega^N_1),(\omega^1_2,\dots,\omega^N_2),(\omega^1_3,\dots\omega^N_3)\big)\ ,\\ \tau &= (\tau_1,\tau_2,\tau_3)=\big((\tau^1_1,\dots,\tau^N_1),(\tau^1_2,\dots,\tau^N_2),(\tau^1_3,\dots,\tau^N_3)\big)\ ,\end{aligned}$$

where $\omega_j,\tau_j\in\Omega_j$ and $\omega^i_j,\tau^i_j\in\Omega^i_j$. Notice that the probability distribution $\rho_{f_e}(\omega,\cdot)$ can be seen as the product of $3N$ distributions. Namely, for each $i\in[N]$ we have a probability distribution $\rho^{i,1}_{f_e}(\omega^i_1,\cdot)$ corresponding to Line 3 of FIX and color i, and similarly, for ω^i_2,ω^i_3 we have probability distributions $\rho^{i,2}_{f_e}(\omega^i_2,\cdot),\rho^{i,3}_{f_e}(\omega^i_3,\cdot)$, corresponding to Lines 5, 6 of FIX and color i, respectively. Recalling procedure RESAMPLE, we see that the support of $\rho^{i,1}_{f_e}(\omega^i_1,\cdot)$ is $\mathcal{M}^i_{t+1}(e,\omega_1)$ and, thus, it must be the case that $Q^i_e(t,\omega_1)=Q^i_e(t,\tau_1)$ for every $i\in[N]$ and state $\omega\in\mathrm{In}_e(\tau)$. Similarly, by inspecting procedure FIX one can verify that $A^i_e(t,\omega_2)=A^i_e(t,\tau_2)$ and that $C^i_e(t,\omega_3)=C^i_e(t,\tau_3)$ for each $i\in[N]$. Hence, $R_e(t,\omega)=R_e(t,\tau)$, as claimed.

For each $\omega\in f_e$,

$$\frac{\mu(\omega)}{\mu(\tau)}\rho_{f_e}(\omega,\tau)=\prod_{i=1}^N\prod_{j=1}^3\frac{\nu^i_j(\omega^i_j)}{\nu^i_j(\tau^i_j)}\rho^{i,j}_{f_e}(\omega^i_j,\tau)=:\prod_{i=1}^N\prod_{j=1}^3 r_{i,j}(\omega)\ . \qquad (7.26)$$

We will now give an alternative expression for each $r_{i,j}(\omega)$ in order to relate (7.26) to (7.25). We start with $r_{i,1}(\omega)$. The fact that $Q^i_e(t,\omega_1)=Q^i_e(t,\tau_1)$ for each $\omega\in\mathrm{In}_e(\tau)$ implies that

$$\frac{\nu^i_1(\omega^i_1)}{\nu^i_1(\tau^i_1)}=\frac{\lambda_i(\omega^i_1\cap E(G_{<t+1}(e)))}{\lambda_i(\tau^i_1\cap E(G_{<t+1}(e))}\ . \qquad (7.27)$$

Furthermore, since we have assumed that the hard-core distribution in Lines 8, 9 of RESAMPLE is induced by the ideal vector of activities λ_i, we have

$$\rho_{f_e}(\omega_1^i, \tau_1^i) = \frac{\lambda_i(\tau_1^i \cap E(G_{<t+1}(e))}{\sum_{M \in \mathcal{M}_{t+1}^i(e,\omega_1)} \lambda_i(M)} \ . \tag{7.28}$$

Combining (7.27) with (7.28) and the fact that $Q_e^i(t, \omega_1) = Q_e^i(t, \tau_1)$ we obtain

$$r_{i,1}(\omega) = \frac{\lambda_i(\omega_1^i \cap E(G_{<t+1}(e))}{\sum_{M \in \mathcal{M}_{t+1}^i(e,\tau_1)} \lambda_i(M)} \ . \tag{7.29}$$

Recall now the definitions of $a_i(t', e)$ and $h_i(t', e)$. The fact that $A_e^i(t, \omega_2) = A_e^i(t, \tau_2)$ for each $\omega \in \mathrm{In}_e(\tau)$ implies that

$$\frac{\nu_2^i(\omega_2^i)}{\nu_2^i(\tau_2^i)} = \frac{\nu_2^i(a_i(t', e) = x_{e,i}^{\omega}(2))}{\nu_2^i(a_i(t', e) = x_{e,i}^{\tau}(2))} \ . \tag{7.30}$$

Further, since in Line 5 of FIX we simply flip a coin independently with success probability α for each edge of $G_i \cap S_{<t+1}(e)$, we have

$$\rho_{f_e}(\omega_2^i, \tau_2^i) = \frac{\nu_2^i(a_i(t', e) = x_{e,i}^{\tau}(2))}{\sum_a \nu_2^i(a_i(t', e) = a)} \ , \tag{7.31}$$

where the sum in the denominator ranges over all the possible values for $a_i(t', e)$. Thus, combining (7.30) with (7.31) we get

$$r_{i,2}(\omega) = \frac{\nu_2^i(a_i(t', e) = x_{e,i}^{\omega}(2))}{\sum_a \nu_2^i(a_i(t', e) = a)} \ . \tag{7.32}$$

Finally, an identical argument shows that

$$r_{i,3}(\omega) = \frac{\nu_3^i(h_i(t', e) = x_{e,i}^{\omega}(2))}{\sum_h \nu_3^i(h_i(t', e) = h)} \ . \tag{7.33}$$

For $x \in X_e$, let $\Omega_{e,x} = \{\omega : x_e^{\omega} = x\}$. For σ distributed according to μ, the left-hand side of (7.25) can be written as

$$\begin{aligned}
\sum_{x \in X_e} \sum_{\omega \in \Omega_e} \frac{\mu(\omega)}{\mu(\tau)} \rho_{f_e}(\omega, \tau) &= \sum_{x \in X_e} \sum_{\omega \in \Omega_{e,x}} \prod_{i=1}^{N} \prod_{j=1}^{3} r_{i,j}(\omega) \\
&= \sum_{x \in X_e} \prod_{i=1}^{N} \prod_{j=1}^{3} \sum_{\substack{\omega \in \Omega_{e,x} \\ x_{e,i}^{\omega} = x_i(j)}} r_{i,j}(\omega) \qquad (7.34) \\
&= \sum_{x \in X_e} \prod_{i=1}^{N} \prod_{j=1}^{3} \nu_j^i(x_{e,i}^{\sigma}(j) = x_i(j) \mid R_e(t, \sigma) = R_e(t, \tau)) \qquad (7.35) \\
&= \mu(\sigma \in f_e \mid R_e(t, \sigma) = R_e(t, \tau)) \ ,
\end{aligned}$$

concluding the proof of (7.25). Note that (7.34) follows from the fact that Ω is a product space, and (7.35) follows by (7.22) and (7.29) for $j = 1$, (7.23) and (7.32) for $j = 2$, and (7.24) and (7.33) for $j = 3$.

Bibliography

[1] Dimitris Achlioptas and Amin Coja-Oghlan. Algorithmic barriers from phase transitions. In *49th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2008, October 25-28, 2008, Philadelphia, PA, USA*, pages 793–802. IEEE Computer Society, 2008.

[2] Dimitris Achlioptas, Themis Gouleakis, and Fotis Iliopoulos. Local computation algorithms for the Lovász Local Lemma. *CoRR*, abs/1809.07910, 2018.

[3] Dimitris Achlioptas and Fotis Iliopoulos. Random walks that find perfect objects and the Lovász local lemma. *J. ACM*, 63(3):22:1–22:29, July 2016.

[4] Dimitris Achlioptas, Fotis Iliopoulos, and Vladimir Kolmogorov. A local lemma for focused stochastic algorithms. To appear in *SIAM Journal on Computing*. Preprint at *arXiv:1805.02026*.

[5] Dimitris Achlioptas, Fotis Iliopoulos, and Alistair Sinclair. Beyond the Lovász local lemma: Point to set correlations and their algorithmic applications. To appear in *Proceedings of IEEE FOCS*, 2019. Preprint at *arXiv:1805.02026*.

[6] Dimitris Achlioptas, Fotis Iliopoulos, and Nikos Vlassis. Stochastic control via entropy compression. In Ioannis Chatzigiannakis, Piotr Indyk, Fabian Kuhn, and Anca Muscholl, editors, *44th International Colloquium on Automata, Languages, and Programming, ICALP 2017, July 10-14, 2017, Warsaw, Poland*, volume 80 of *LIPIcs*, pages 83:1–83:13. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2017.

[7] Dimitris Achlioptas and Assaf Naor. The two possible values of the chromatic number of a random graph. In László Babai, editor, *Proceedings of the 36th Annual ACM Symposium on Theory of Computing, Chicago, IL, USA, June 13-16, 2004*, pages 587–593. ACM, 2004.

[8] Michael Albert, Alan Frieze, and Bruce Reed. Multicoloured Hamilton cycles. *The Electronic Journal of Combinatorics*, 2(1):R10, 1995.

[9] Noga Alon. A parallel algorithmic version of the local lemma. *Random Struct. Algorithms*, 2(4):367–378, 1991.

[10] Noga Alon, Michael Krivelevich, and Benny Sudakov. Coloring graphs with sparse neighborhoods. *Journal of Combinatorial Theory, Series B*, 77(1):73–82, 1999.

[11] Noga Alon and Joel H. Spencer. *The Probabilistic Method*. Wiley Publishing, 4th edition, 2016.

[12] Adrian Balint and Uwe Schöning. Choosing probability distributions for stochastic local search and the role of make versus break. In *International Conference on Theory and Applications of Satisfiability Testing*, pages 16–29. Springer, 2012.

[13] Adrian Balint and Uwe Schöning. Engineering a lightweight and efficient local search SAT solver. In Lasse Kliemann and Peter Sanders, editors, *Algorithm Engineering - Selected Results and Surveys*, volume 9220 of *Lecture Notes in Computer Science*, pages 1–18. 2016.

[14] Nikhil Bansal, Ola Svensson, and Luca Trevisan. New notions and constructions of sparsification for graphs and hypergraphs. *arXiv preprint arXiv:1905.01495*, 2019.

[15] József Beck. An algorithmic approach to the Lovász local lemma. I. *Random Structures Algorithms*, 2(4):343–365, 1991.

[16] Anton Bernshteyn. New bounds for the acyclic chromatic index. *Discrete Mathematics*, 339(10):2543–2552, 2016.

[17] Anton Bernshteyn. The Johansson–Molloy theorem for DP-coloring. *arXiv preprint arXiv:1708.03843*, 2017.

[18] Anton Bernshteyn. The local cut lemma. *European Journal of Combinatorics*, 63:95–114, 2017.

[19] Armin Biere, Marijn Heule, and Hans van Maaren. *Handbook of Satisfiability*, volume 185. IOS press, 2009.

[20] Rodrigo Bissacot and Luís Doin. Entropy compression method and legitimate colorings in projective planes. *arXiv preprint arXiv:1710.06981*, 2017.

[21] Rodrigo Bissacot, Roberto Fernández, Aldo Procacci, and Benedetto Scoppola. An improvement of the Lovász local lemma via cluster expansion. *Combinatorics, Probability & Computing*, 20(5):709–719, 2011.

[22] Karthekeyan Chandrasekaran, Navin Goyal, and Bernhard Haeupler. Deterministic algorithms for the Lovász local lemma. *SIAM J. Comput.*, 42(6):2132–2155, 2013.

[23] Antares Chen, David G. Harris, and Aravind Srinivasan. Partial resampling to approximate covering integer programs. In Robert Krauthgamer, editor, *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2016, Arlington, VA, USA, January 10-12, 2016*, pages 1984–2003. SIAM, 2016.

[24] Guantao Chen, Guangming Jing, and Wenan Zang. Proof of the Goldberg-Seymour conjecture on edge-colorings of multigraphs. *arXiv preprint arXiv:1901.10316*, 2019.

[25] Guantao Chen, Xingxing Yu, and Wenan Zang. Approximating the chromatic index of multigraphs. *Journal of Combinatorial Optimization*, 21(2):219–246, 2011.

[26] Benny Chor and Oded Goldreich. Unbiased bits from sources of weak randomness and probabilistic communication complexity. *SIAM J. Comput.*, 17(2):230–261, 1988.

[27] Kai-Min Chung, Seth Pettie, and Hsin-Hao Su. Distributed algorithms for the Lovász local lemma and graph coloring. In Magnús M. Halldórsson and Shlomi Dolev, editors, *ACM Symposium on Principles of Distributed Computing, PODC '14, Paris, France, July 15-18, 2014*, pages 134–143. ACM, 2014.

[28] Thomas F Coleman and Jin Yi Cai. The cyclic coloring problem and estimation of spare Hessian matrices. *SIAM J. Algebraic Discrete Methods*, 7(2):221–235, April 1986.

[29] Thomas F. Coleman and Moré Jorge J. Estimation of sparse Hessian matrices and graph coloring problems. *Mathematical Programming*, 28(3):243–270, 1984.

[30] Stephen A Cook. The complexity of theorem-proving procedures. In *Proceedings of the third annual ACM Symposium on Theory of Computing*, pages 151–158. ACM, 1971.

[31] Artur Czumaj and Christian Scheideler. Coloring non-uniform hypergraphs: a new algorithmic approach to the general Lovász local lemma. In *Proceedings of the Eleventh Annual ACM-SIAM Symposium on Discrete Algorithms (San Francisco, CA, 2000)*, pages 30–39, 2000.

[32] Rina Dechter and David Cohen. *Constraint processing*. Morgan Kaufmann, 2003.

[33] Andrzej Dudek, Alan Frieze, and Andrzej Ruciński. Rainbow Hamilton cycles in uniform hypergraphs. *The Electronic Journal of Combinatorics*, 19(1):46, 2012.

[34] Vida Dujmović, Gwenaël Joret, Jakub Kozik, and David R Wood. Nonrepetitive colouring via entropy compression. *Combinatorica*, 36(6):661–686, 2016.

[35] Jack Edmonds. Maximum matching and a polyhedron with 0, 1-vertices. *Journal of research of the National Bureau of Standards B*, 69(125-130):55–56, 1965.

[36] Paul Erdős and László Lovász. Problems and results on 3-chromatic hypergraphs and some related questions. In *Infinite and finite sets (Colloq., Keszthely, 1973; dedicated to P. Erdős on his 60th birthday), Vol. II*, pages 609–627. Colloq. Math. Soc. János Bolyai, Vol. 10. North-Holland, Amsterdam, 1975.

[37] Paul Erdös and Joel Spencer. Lopsided Lovász local lemma and Latin transversals. *Discrete Applied Mathematics*, 30(2-3):151–154, 1991.

[38] Louis Esperet and Aline Parreau. Acyclic edge-coloring using entropy compression. *European Journal of Combinatorics*, 34(6):1019–1027, 2013.

[39] Uriel Feige, Eran Ofek, and Udi Wieder. Approximating maximum edge coloring in multigraphs. In *International Workshop on Approximation Algorithms for Combinatorial Optimization*, pages 108–121. Springer, 2002.

[40] Alan M Frieze and Michael Molloy. Splitting an expander graph. *Journal of Algorithms*, 33(1):166–172, 1999.

[41] RG Gallager. Low-density parity-check codes mit press. *Cambridge, Massachusetts*, 1963.

[42] Heidi Gebauer, Tibor Szabó, and Gábor Tardos. The local lemma is tight for SAT. In Dana Randall, editor, *SODA*, pages 664–674. SIAM, 2011.

[43] Stuart Geman and Donald Geman. Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images. In *Readings in computer vision*, pages 564–584. Elsevier, 1987.

[44] Ioannis Giotis, Lefteris M. Kirousis, Kostas I. Psaromiligkos, and Dimitrios M. Thilikos. Acyclic edge coloring through the Lovász local lemma. *Theor. Comput. Sci.*, 665:40–50, 2017.

[45] Adam Gkagol, Gwenael Joret, Jakub Kozik, and Piotr Micek. Pathwidth and nonrepetitive list coloring. *arXiv preprint arXiv:1601.01886*, 2016.

[46] Daniel Gonçalves, Mickaël Montassier, and Alexandre Pinlou. Entropy compression method applied to graph colorings. *arXiv preprint arXiv:1406.4380*, 2014.

[47] Jarosław Grytczuk, Jakub Kozik, and Piotr Micek. New approach to nonrepetitive sequences. *Random Structures & Algorithms*, 42(2):214–225, 2013.

[48] Bernhard Haeupler and David G Harris. Parallel algorithms and concentration bounds for the Lovász local lemma via witness-DAGs. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1170–1187. SIAM, 2017.

[49] Bernhard Haeupler, Barna Saha, and Aravind Srinivasan. New constructive aspects of the Lovász local lemma. *J. ACM*, 58(6):Art. 28, 28, 2011.

[50] David G. Harris. Lopsidependency in the Moser-Tardos framework: Beyond the lopsided Lovász local lemma. *ACM Trans. Algorithms*, 13(1):17:1–17:26, 2016.

[51] David G Harris. New bounds for the Moser-Tardos distribution. *arXiv preprint arXiv:1610.09653*, 2016.

[52] David G. Harris. Comparison for two convergence criteria for the variable-assignment lopsided Lovasz local lemma. *arXiv preprint arXiv:1610.01926*, 2018.

[53] David G. Harris. Oblivious resampling oracles and parallel algorithms for the lopsided Lovász local lemma. In Timothy M. Chan, editor, *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2019, San Diego, California, USA, January 6-9, 2019*, pages 841–860. SIAM, 2019.

[54] David G. Harris and Aravind Srinivasan. The Moser-Tardos framework with partial resampling. In *54th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2013, 26-29 October, 2013, Berkeley, CA, USA*, pages 469–478. IEEE Computer Society, 2013.

[55] David G. Harris and Aravind Srinivasan. A constructive algorithm for the Lovász local lemma on permutations. In Chandra Chekuri, editor, *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2014, Portland, Oregon, USA, January 5-7, 2014*, pages 907–925. SIAM, 2014.

[56] David G. Harris and Aravind Srinivasan. Algorithmic and enumerative aspects of the Moser-Tardos distribution. In *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2016, Arlington, VA, USA, January 10-12, 2016*, pages 2004–2023, 2016.

[57] Nicholas J. A. Harvey and Jan Vondrák. An algorithmic proof of the Lovász local lemma via resampling oracles. In Venkatesan Guruswami, editor, *IEEE 56th Annual Symposium on Foundations of Computer Science, FOCS 2015, Berkeley, CA, USA, 17-20 October, 2015*, pages 1327–1346. IEEE Computer Society, 2015.

[58] Kun He, Liang Li, Xingwu Liu, Yuyi Wang, and Mingji Xia. Variable-version Lovász local lemma: Beyond Shearer's bound. In *Foundations of Computer Science (FOCS), 2017 IEEE 58th Annual Symposium on*, pages 451–462. IEEE, 2017.

[59] Fotis Iliopoulos. Commutative algorithms approximate the LLL-distribution. In Eric Blais, Klaus Jansen, José D. P. Rolim, and David Steurer, editors, *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2018, August 20-22, 2018 - Princeton, NJ, USA*, volume 116 of *LIPIcs*, pages 44:1–44:20. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2018.

[60] Fotis Iliopoulos and Alistair Sinclair. Efficiently list-edge coloring multigraphs asymptotically optimally. *CoRR*, abs/1812.10309, 2018.

[61] Mark Jerrum and Alistair Sinclair. Approximating the permanent. *SIAM journal on computing*, 18(6):1149–1178, 1989.

[62] A. Johansson. Asympotic choice number for triangle free graphs. *Unpublished manuscript*, 1996.

[63] A. Johansson. The choice number of sparse graphs. *Unpublished manuscript*, 1996.

[64] Jeff Kahn. Asymptotically good list-colorings. *Journal of Combinatorial Theory, Series A*, 73(1):1–59, 1996.

[65] Jeff Kahn. Asymptotics of the chromatic index for multigraphs. *journal of combinatorial theory, Series B*, 68(2):233–254, 1996.

[66] Jeff Kahn. Asymptotics of the list-chromatic index for multigraphs. *Random Structures & Algorithms*, 17(2):117–156, 2000.

[67] Jeff Kahn and P Mark Kayll. On the stochastic independence properties of hard-core distributions. *Combinatorica*, 17(3):369–391, 1997.

[68] Jeong Han Kim. On Brooks' theorem for sparse graphs. *Combinatorics, Probability and Computing*, 4(2):97–132, 1995.

[69] Kashyap Kolipaka, Mario Szegedy, and Yixin Xu. A sharper local lemma with improved applications. In Anupam Gupta, Klaus Jansen, José Rolim, and Rocco Servedio, editors, *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques*,

volume 7408 of *Lecture Notes in Computer Science*, pages 603–614. Springer Berlin Heidelberg, 2012.

[70] Kashyap Babu Rao Kolipaka and Mario Szegedy. Moser and Tardos meet Lovász. In *STOC*, pages 235–244. ACM, 2011.

[71] Vladimir Kolmogorov. Commutativity in the algorithmic Lovász local lemma. In Irit Dinur, editor, *IEEE 57th Annual Symposium on Foundations of Computer Science, FOCS 2016, 9-11 October 2016, Hyatt Regency, New Brunswick, New Jersey, USA*, pages 780–787. IEEE Computer Society, 2016.

[72] Vipin Kumar. Algorithms for constraint-satisfaction problems: A survey. *AI magazine*, 13(1):32–32, 1992.

[73] Carl W Lee. Some recent results on convex polytopes. *Contemporary Math*, 114:3–19, 1990.

[74] Linyuan Lu and Laszlo A Szekely. A new asymptotic enumeration technique: the Lovász local lemma. *arXiv preprint arXiv:0905.3983*, 2009.

[75] Steven Minton, Mark D Johnston, Andrew B Philips, and Philip Laird. Minimizing conflicts: a heuristic repair method for constraint satisfaction and scheduling problems. *Artificial Intelligence*, 58(1-3):161–205, 1992.

[76] Michael Molloy. The list chromatic number of graphs with small clique number. *Journal of Combinatorial Theory, Series B*, 2018.

[77] Michael Molloy and Bruce Reed. A bound on the total chromatic number. *Combinatorica*, 18(2):241–280, 1998.

[78] Michael Molloy and Bruce Reed. Further algorithmic aspects of the local lemma. In *STOC '98 (Dallas, TX)*, pages 524–529. ACM, New York, 1999.

[79] Michael Molloy and Bruce Reed. *Graph colouring and the probabilistic method*, volume 23 of *Algorithms and Combinatorics*. Springer-Verlag, Berlin, 2002.

[80] Robin A. Moser. A constructive proof of the Lovász local lemma. In *STOC'09—Proceedings of the 2009 ACM International Symposium on Theory of Computing*, pages 343–350. ACM, New York, 2009.

[81] Robin A. Moser and Gábor Tardos. A constructive proof of the general Lovász local lemma. *J. ACM*, 57(2):Art. 11, 15, 2010.

[82] Sokol Ndreca, Aldo Procacci, and Benedetto Scoppola. Improved bounds on coloring of graphs. *Eur. J. Comb.*, 33(4):592–609, May 2012.

[83] Takao Nishizeki and Kenichi Kashiwagi. On the 1.1 edge-coloring of multigraphs. *SIAM Journal on Discrete Mathematics*, 3(3):391–410, 1990.

[84] Pascal Ochem and Alexandre Pinlou. Application of entropy compression in pattern avoidance. *arXiv preprint arXiv:1301.1873*, 2013.

[85] Rafail Ostrovsky, Yuval Rabani, and Leonard J Schulman. Error-correcting codes for automatic control. *IEEE Transactions on Information Theory*, 55(7):2931–2941, 2009.

[86] Manfred W Padberg and M Ram Rao. Odd minimum cut-sets and b-matchings. *Mathematics of Operations Research*, 7(1):67–80, 1982.

[87] Wesley Pegden. Highly nonrepetitive sequences: Winning strategies from the local lemma. *Random Struct. Algorithms*, 38(1-2):140–161, 2011.

[88] Wesley Pegden. An extension of the Moser-Tardos algorithmic local lemma. *SIAM J. Discrete Math.*, 28(2):911–917, 2014.

[89] Pavel Pevzner. *Computational molecular biology: an algorithmic approach*. MIT press, 2000.

[90] Michael Plantholt. A sublinear bound on the chromatic index of multigraphs. *Discrete Mathematics*, 202(1-3):201–213, 1999.

[91] Yuri Rabinovich, Alistair Sinclair, and Avi Wigderson. Quadratic dynamical systems. In *Foundations of Computer Science, 1992. Proceedings., 33rd Annual Symposium on*, pages 304–313. IEEE, 1992.

[92] Peter Sanders and David Steurer. An asymptotic approximation scheme for multigraph edge coloring. *ACM Transactions on Algorithms (TALG)*, 4(2):21, 2008.

[93] Diego Scheide. *On the 15/14 edge-colouring of multigraphs*. Institut for Matematik og Datalogi, Syddansk Universitet, 2007.

[94] Diego Scheide. Graph edge colouring: Tashkinov trees and Goldberg's conjecture. *Journal of Combinatorial Theory, Series B*, 100(1):68–96, 2010.

[95] Alexander D Scott and Alan D Sokal. The repulsive lattice gas, the independent-set polynomial, and the Lovász local lemma. *Journal of Statistical Physics*, 118(5-6):1151–1261, 2005.

[96] Bart Selman, Henry A. Kautz, and Bram Cohen. Local search strategies for satisfiability testing. In David S. Johnson and Michael A. Trick, editors, *Cliques, Coloring, and Satisfiability, Proceedings of a DIMACS Workshop, New Brunswick, New Jersey, USA, October 11-13, 1993*, volume 26 of *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, pages 521–532. DIMACS/AMS, 1993.

[97] Bart Selman, Henry A. Kautz, and Bram Cohen. Noise strategies for improving local search. In Barbara Hayes-Roth and Richard E. Korf, editors, *Proceedings of the 12th National Conference on Artificial Intelligence, Seattle, WA, USA, July 31 - August 4, 1994, Volume 1.*, pages 337–343. AAAI Press / The MIT Press, 1994.

[98] J.B. Shearer. On a problem of Spencer. *Combinatorica*, 5(3):241–245, 1985.

[99] Mohit Singh and Nisheeth K Vishnoi. Entropy, optimization and counting. In *Proceedings of the forty-sixth annual ACM symposium on Theory of computing*, pages 50–59. ACM, 2014.

[100] Joel Spencer. Asymptotic lower bounds for ramsey functions. *Discrete Mathematics*, 20(0):69 – 76, 1977.

[101] Aravind Srinivasan. Improved algorithmic versions of the Lovász local lemma. In Shang-Hua Teng, editor, *SODA*, pages 611–620. SIAM, 2008.

[102] Vadim G Vizing. On an estimate of the chromatic class of a p-graph. *Discret Analiz*, 3:25–30, 1964.

[103] Van H Vu. A general upper bound on the list chromatic number of locally sparse graphs. *Combinatorics, Probability and Computing*, 11(1):103–111, 2002.

Chapter 8

Appendices

8.1 Matrices and Norms

Let $\|\cdot\|$ be any norm over vectors in $\mathbb{R}^n$. The *dual* norm, also over vectors in $\mathbb{R}^n$, is defined as

$$\|z\|_* = \sup_{\|x\|=1} |z^\top x| \ .$$

For example, the dual norm of $\|\cdot\|_\infty$ is $\|\cdot\|_1$. It can be seen that $\|\cdot\|_{**} = \|\cdot\|$ and that for any vectors x, z,

$$z^\top x = \|x\| \left(\frac{z^\top x}{\|x\|} \right) \leq \|z\|_* \|x\| \ . \tag{8.1}$$

The corresponding *operator norm*, over $n \times n$ real matrices, is defined as

$$\|A\| \equiv \sup_{\|x\|=1} \|Ax\| \ .$$

For example, if A is a matrix with non-negative entries then $\|A\|_\infty$ and $\|A\|_1$ can be seen to be the maximum *row* and *column* sum of A, respectively. Operator norms are submultiplicative, i.e., for every operator norm $\|\cdot\|$ and any two $n \times n$ matrices A, B,

$$\|AB\| \leq \|A\| \|B\| \ . \tag{8.2}$$

Finally, for any vector norm $\|\cdot\|$, any row vector $x^\top$ and $n \times n$ matrix A we have that

$$\|x^\top A\|_* \leq \|x^\top\|_* \|A\| \ . \tag{8.3}$$

8.2 Recursive Algorithms

In this section we show how we can obtain constructive analogues of the cluster expansion condition (2.3) and the left-handed LLL condition of Pegden [87] via recursive algorithms.

8.2.1 Dense Neighborhoods

In a number of applications the subgraph induced by the neighborhood of each flaw in the causality graph contains several (directed) edges. We improve Theorem 3.15 in such settings by employing a recursive algorithm akin to Algorithm 2. This has the effect that the flaw addressed in each step depends on the entire trajectory up to that point, not just the current state, i.e., the walk is non-Markovian. It is for this reason that we required a non-empty set of actions for every flaw present in a state, and why the definition of the causality digraph does not involve flaw choice. Specifically, for any permutation π on $[m]$ the recursive walk is the non-Markovian random walk on Ω that occurs by invoking procedure ELIMINATE below.

Recall that $U(\sigma)$ denotes the set of indices of flaws present in state σ, that for a set $S \subseteq [m]$, $\pi(S) = \min_{j \in S} \pi(j)$, and that we sometimes abbreviate $\pi(U(\sigma))$ as $\pi(\sigma)$.

Recursive Walk

procedure ELIMINATE
    $\sigma \leftarrow \theta(\cdot)$
    while $U(\sigma) \neq \emptyset$ **do**
        $\sigma \leftarrow$ ADDRESS $(\pi(\sigma), \sigma)$
procedure ADDRESS(i, σ)
    $\sigma' \leftarrow \tau \in \alpha(i, \sigma)$ with probability $\rho_i(\sigma, \tau)$
    while $S = U(\sigma') \cap \Gamma(i) \neq \emptyset$ **do**
        $\sigma' \leftarrow$ ADDRESS$(\pi(B), \sigma')$

Theorem 8.2 below can be used to make constructive applications of the cluster expansion condition (2.3). We prove it in Section 8.2.3 using Theorem 3.19.

Definition 8.1. *For a causality graph C on $[m]$, let $G = G(C) = ([m], E)$ be the* undirected *graph where $\{i, j\} \in E$ iff both $i \to j$ and $j \to i$ exist in C. For $S \subseteq [m]$, let* $\mathrm{Ind}(S) = \{S' \subseteq S : S' \text{ is an independent set in } G\}$.

Theorem 8.2. *If there exist positive real numbers $\{\psi_i\}$ for $1 \leq i \leq m$ such that*

$$\zeta_i := \frac{\gamma_i}{\psi_i} \sum_{S \in \mathrm{Ind}(\Gamma(i))} \prod_{j \in S} \psi_j < 1 \ \text{, for every } i \in [m] \tag{8.4}$$

then for every permutation π, the recursive walk reaches a sink within $(T_0 + s)/\log_2(1/(1-\epsilon))$ steps with probability at least $1 - 2^{-s}$, where $\epsilon = 1 - \max_{i \in [m]} \zeta_i > 0$, and

$$T_0 = \log_2 \left(\max_{\sigma \in \Omega} \frac{\theta(\sigma)}{\mu(\sigma)} \right) + \log_2 \left(\sum_{S \subseteq \mathrm{Ind}(\mathrm{Span}(\theta))} \prod_{j \in S} \psi_j \right) \ .$$

Remark 8.3. *Theorem 8.2 strictly improves Theorem 3.15 since: (i) the summation in* (8.4) *is only over the subsets of $\Gamma(i)$ that are independent in G, instead of all subsets of $\Gamma(i)$ as in* (3.8)*; and (ii) similarly, for T_0 the summation is only over the independent subsets of $\mathrm{Span}(\theta)$, rather than all subsets of $\mathrm{Span}(\theta)$.*

Remark 8.4. *Theorem 8.2 can be strengthened by introducing, for each flaw $f_i \in F$, a permutation π_i of $\Gamma(i)$ and replacing π with π_i in line 8 of the Recursive Walk. With this change in* (8.4) *it suffices to sum only over $S \subseteq \Gamma(i)$ satisfying the following: if the subgraph of R induced by S contains an edge $j \to k$, then $\pi_i(j) \geq \pi_i(k)$. As such a subgraph cannot contain both $j \to k$ and $k \to j$ we see that $S \in \mathrm{Ind}(\Gamma(i))$.*

8.2.2 A Left-Handed Algorithm

While Theorems 3.15 and 8.2 do not care about the permutation π on $[m]$, inspired by the so-called left-handed version of the LLL introduced by Pegden [87], we give a condition under which the flaw order π can be chosen in a *provably* beneficial way. This is done by organizing the flaws in an order akin to an elimination sequence. Specifically, the idea is to seek a permutation π and a "responsibility digraph" R, derived from the causality digraph C so as to "shift responsibility" from flaws failing to satisfy condition (3.8) of Theorem 3.15 to flaws that have slack.

Definition 8.5. *For an ordered set of vertices $v_1 < v_2 < \cdots < v_n$, say that edge $v_i \to v_j$ is* forward *if $i < j$ and* backward *if $i > j$. Given a causality digraph C and a permutation π on $[m]$ ordering the vertices of C, we say that R is a* responsibility digraph for C with respect to π *if:*

1. *Every forward edge and self-loop of C exists in R.*
2. *If a backward edge $v_j \to v_i$ of C does not exist in R, then for each k such that $v_k \to v_j$ exists in R, $v_k \to v_i$ exists in R as well.*

The neighborhood of a flaw f_i in a responsibility graph R is $\Gamma_R(i) = \{j \in [m] : i \to j \text{ exists in } R\}$.

For any permutation π of $[m]$, and any responsibility digraph R with respect to π, the left-handed walk is the random walk induced on Ω by modifying the Recursive Walk as follows.

Left-Handed Walk

For $S \subseteq [m]$, let $\pi(S) = \max_{j \in S} \pi(j)$ denote the *greatest* index in S.
In line 7 of **Recursive Walk** replace Γ with Γ_R.

In Section 8.2.4 we prove the following theorem, again as a corollary of Theorem 3.19.

Theorem 8.6. *For any permutation π on $[m]$ and any responsibility digraph R with respect to π, if there exist positive real numbers $\{\psi_i\}$ for every $1 \leq i \leq m$ such that*

$$\zeta_i := \frac{\gamma_i}{\psi_i} \sum_{S \subseteq \Gamma_R(i)} \prod_{j \in S} \psi_j < 1 \text{ for every } i \in [m] \ ,$$

then the Left-Handed Walk reaches a sink within $(T_0 + s)/\log_2((1/1-\epsilon))$ steps with probability at least $1 - 2^{-s}$, where $\delta = 1 - \max_{i \in [m]} \zeta_i$, and

$$T_0 = \log_2 \left(\max_{\sigma \in \Omega} \frac{\theta(\sigma)}{\mu(\sigma)} \right) + \log_2 \left(\sum_{S \subseteq \mathrm{Span}(\theta)} \prod_{j \in S} \psi_j \right) \ .$$

Remark 8.7. *Since the causality digraph C is, trivially, a responsibility graph, Theorem 8.6 can be seen as a non-Markovian generalization of Theorem 3.15 in which flaw choice is driven by recursion and R.*

8.2.3 Forests of the Recursive Walk (Theorem 8.2)

We will represent each bad t-trajectory, Σ, of the Recursive Walk as a vertex-labeled unordered rooted forest, having one tree per invocation of procedure ADDRESS by procedure ELIMINATE. Specifically, to construct the *Recursive Forest* $\phi = \phi(\Sigma)$ we add a root vertex per invocation of ADDRESS by ELIMINATE and one child to every vertex for each (recursive) invocation of ADDRESS that it makes. As each vertex corresponds to an invocation of ADDRESS (step of the walk) it is labeled by the invocation's flaw-argument. Observe now that (the invocations of ADDRESS corresponding to) both the roots of the trees and the children of each vertex appear in Σ in their order according to π. Thus, given the unordered rooted forest $\phi(\Sigma)$ we can order its trees and the progeny of each vertex according to π and recover $W(\Sigma)$ as the sequence of vertex labels in the preorder traversal of the resulting ordered rooted forest.

Recall the definition of graph G on $[m]$ from Definition 8.1. We will prove that the flaws labeling the roots of a Recursive Forest are independent in G and that the same is true for the flaws labeling the progeny of every vertex of the forest. To do this we first prove the following.

Proposition 8.8. *If* ADDRESS(i, σ) *returns at state* τ*, then* $U(\tau) \subseteq U(\sigma) \setminus (\Gamma(i) \cup \{i\})$.

Proof. Let σ' be any state subsequent to the ADDRESS(i, σ) invocation. If any flaw in $U(\sigma) \cap \Gamma(i)$ is present at σ', the "while" condition in line 7 of the Recursive Walk prevents ADDRESS(i, σ) from returning. On the other hand, if $k \in \Gamma(i) \setminus U(\sigma)$ is present in σ', then there must have existed an invocation ADDRESS(j, σ''), subsequent to invocation ADDRESS(i, σ), wherein addressing j caused k. Consider the last such invocation. If σ''' is the state when this invocation returns, then $k \notin U(\sigma''')$, for otherwise the invocation could not have returned, and by the choice of invocation, k is not present in any subsequent state between σ''' and τ. □

Let (w_i, σ_i) denote the argument of the i-th invocation of ADDRESS by ELIMINATE. By Proposition 8.8, $\{U(\sigma_i)\}_{i \geq 1}$ is a decreasing sequence of sets. Thus, the claim that the root labels form an independent set of $\mathrm{Span}(\theta)$ follows trivially: for each $i \geq 1$, the flaws in $\Gamma(w_i) \cup \{w_i\}$ are not present in σ_{i+1} and, therefore, are not present in $U(\sigma_j)$, for any $j \geq i + 1$. The proof of the claim that the children of each node form an independent set in the subgraph of G induced by the neighborhood of the label of the node is essentially identical. If a node corresponding to an invocation ADDRESS(w, σ) has children corresponding to (recursive) invocations with arguments $\{(g_i, \sigma_i)\}$, then the sequence of sets $\{U(\sigma_i)\}_{i \geq 1}$ is decreasing. Thus, the flaws in $\Gamma(g_i) \cup \{g_i\}$ are not present in σ_{i+1} and, therefore, not present in $U(\sigma_j)$, for any $j \geq i + 1$. Applying Theorem 3.19 concludes the proof.

8.2.4 Forests of the Left-Handed Walk (Theorem 8.6)

Recall that π is an arbitrary permutation on $[m]$ and that the Left-Handed Walk is the Recursive Walk modified by replacing $\Gamma(i)$ with $\Gamma_R(i)$ in line 7, where R is a responsibility graph for the input causality graph C with respect to π. We map the bad trajectories of the Left-Handed Walk into vertex-labeled unordered rooted forests, exactly as we did for the bad trajectories of the Recursive Walk, i.e., one tree per invocation of ADDRESS by ELIMINATE, one child per recursive invocation of ADDRESS, all vertices labeled by the flaw-argument of the invocation. The challenge for the Left-Handed Walk is to prove that the labels of the roots are distinct and, similarly, that

the labels of the children of each node are distinct. (For Witness Forests both properties were true automatically; for Recursive Forests we established the stronger property that each of these sets of flaws is independent.) To do this we first prove the following analogue of Proposition 8.8.

Definition 8.9. *Let S_i denote the set of indices of flaws strictly greater than i according to π. For a state σ and a flaw $i \in U(\sigma)$, let $W(\sigma, i) = U(\sigma) \cap S_i$.*

Proposition 8.10. *If* ADDRESS(i, σ) *returns at state τ, then $\tau \notin f_i$ and $W(\tau, i) \subseteq W(\sigma, i)$.*

Proof. The execution of ADDRESS(i, σ) generates a recursion tree, each node labeled by the index of its flaw-argument. Thus, the root is labeled by i and each child of the root is labeled by a flaw in $\Gamma_R(i)$. Let $S_i^+ = S_i \cup \{i\}$. For a state ω, let $Q(i, \omega)$ be the set of flaws in $S_i^+ \setminus \Gamma_R(i)$ that are present in ω. We claim that if $j \in \Gamma_R(i)$ and ADDRESS(j, ω) terminates at ω', then $Q(i, \omega') \subseteq Q(i, \omega)$. This suffices to prove the lemma as:

- By the claim, any flaw in $Q(i, \tau) \setminus Q(i, \sigma)$ must be introduced by the action $\sigma \xrightarrow{i} \sigma'$ taken by the original invocation ADDRESS(i, σ). Thus, $Q(i, \tau) \subseteq Q(i, \sigma')$.
- All flaws in S_i^+ introduced by $\sigma \xrightarrow{i} \sigma'$ are in $\Gamma_R(i)$, since R contains all forward edges and self-loops of C. Thus, $Q(i, \sigma') \subseteq Q(i, \sigma)$. In particular, f_i can only be present in σ' if $i \in \Gamma_R(i)$.
- No flaw in $\Gamma_R(i)$ can be present in τ since ADDRESS(i, σ) returned at τ.

To prove the claim, consider the recursion tree of ADDRESS(j, ω). If $k \in Q(i, \omega')$ and $k \notin Q(i, \omega)$, then there has to be a path $j_1 = j, j_2, \ldots, j_\ell$ from the root of the recursion tree of ADDRESS(j, ω) to a node j_ℓ such that $k \in \Gamma(j_\ell)$ but $k \notin \Gamma_R(j_h)$ for each $h \in [\ell]$. To see this, notice that since k was absent in ω but is present in ω', it must have been introduced by some flaw j_ℓ addressed during the execution of ADDRESS(j, ω). But if k belonged in the neighborhood with respect to R of any of the flaws on the path from the root to j_ℓ, the algorithm would have not terminated. However, such a path can not exist, as it would require all of the following to be true, violating the definition of responsibility digraphs (let $j_0 = i$ for notational convenience): (i) $k \in \Gamma(j_\ell)$; (ii) $k \notin \Gamma_R(j_\ell)$; (iii) $j_\ell \in \Gamma_R(j_{\ell-1})$, and (iv) $k \notin \Gamma_R(j_{\ell-1})$. □

To establish the distinctness of the root labels, observe that each time procedure ELIMINATE is invoked at a state σ, by the definition of $\pi(\cdot)$, we have $W(\sigma, (\pi(\sigma)) = \emptyset$. By Proposition 8.10, if the invocation returns at state τ, then neither $\pi(\sigma)$ nor any greater flaws are present in τ. Therefore, ELIMINATE invokes ADDRESS at most once for each $i \in [m]$. To see the distinctness of the labels of the children of each node, consider an invocation of ADDRESS(i, σ). Whenever this invocation recursively invokes ADDRESS(j, σ'), where $j \in \Gamma_R(i)$, by definition of $\pi(\cdot)$, every flaw in $S_j \cap \Gamma_R(i)$ is absent from σ'. By Proposition 8.10, whenever each such invocation returns, neither j nor any of the flaws in $S_j \cap \Gamma_R(i)$ are present implying that ADDRESS(i, σ) invokes ADDRESS(j, σ') at most once for each $j \in \Gamma_R(i)$. Applying Theorem 3.19 concludes the proof.

8.3 Proofs Omitted from Chapter 5

8.3.1 Proof of Theorem 5.11

To lighten the notation, let $u := \lambda_{\text{init}} \sum_{S \in \text{Ind}([m])} \prod_{j \in S} \psi_j$. For each $\sigma \in \Omega$, define a flaw $f_\sigma = \{\sigma\}$ and consider the extended algorithm that addresses it by sampling from μ, as well as the extended causality graph that connects f_σ with every flaw in F. Observe that $\gamma(f_\sigma) := \|MA_{f_\sigma}M^{-1}\| = \mu(\sigma)$, where $M = \text{diag}(\mu(\sigma))$. Moreover, if the original algorithm is commutative, so is the extended one since the commutativity condition is trivially true for flaws $\{f_\sigma\}_{\sigma \in \Omega}$. Observe now that for every $\sigma \in \Omega$, Theorem 5.7 yields $\nu(\sigma) \leq \Pr[\sigma] \leq u \cdot \mu(\sigma)$. Thus:

$$H_\rho[\nu] = \frac{1}{1-\rho} \ln \sum_{\sigma \in \Omega} \nu(\sigma)^\rho \geq \frac{1}{1-\rho} \ln \sum_{\sigma \in \Omega} (u\mu(\sigma))^\rho = \frac{1}{1-\rho} \ln \sum_{\sigma \in \Omega} \mu(\sigma)^\rho - \frac{\rho}{\rho - 1} \ln u \ ,$$

concluding the proof.

8.3.2 Proof of Theorem 5.12

For each flaw f_i we define a Bernoulli variable Y_i with probability of success $p_i = \min\left\{1, \frac{\psi_i}{\eta_i \gamma_i}\right\}$. The sequence $\{Y_i\}_{i=1}^m$ and Ω induce a new space $\Omega' = \Omega \times \{0,1\}^m$ which can be thought as a "labelled" version of Ω, where each state σ is labelled with a binary vector of length m whose i-th bit describes the state of Y_i. Similarly, measure μ and $\{Y_i\}_{i=1}^m$ induce a measure μ' over Ω'.

In this new state space we introduce a new family of flaws $F' = \{f'_1, f'_2, \ldots, f'_m\}$, where f'_i is defined as the subset of Ω' where f_i is present and $Y_i = 1$. Consider now the algorithm $\mathcal{A}'$ that is induced by $\mathcal{A}$ as follows: Each time we want to address flaw f'_i we move in Ω by invoking $\mathcal{A}$ to address f_i and also take a sample from Y_i to update the value of the i-th entry of the label-vector.

It is not hard to verify that (i) the charge of each flaw f'_i is $\gamma(f'_i) = \gamma_i p_i$; (ii) any causality graph for $(\Omega, F, \mathcal{A})$ is also a causality graph for $(\Omega', F', \mathcal{A}')$ (and, in particular, so is the one induced by $\sim$); (iii) if $\mathcal{A}$ is commutative then so is $\mathcal{A}'$; and that (iv) the cluster expansion condition with respect to the causality graph induced by $\sim$ is satisfied.

To conclude the proof, consider a flaw f'_i and notice that in order for f_i to be present in the output of $\mathcal{A}'$ it has to be the case that $Y_i = 0$. Notice now that Theorem 5.7 implies:

$$\nu(f_i \cap Y_i = 0) \leq (1 - p_i)\gamma_i \eta_i = \max\{0, \gamma_i \eta_i - \psi_i\} \ .$$

8.4 Proofs Omitted from Chapter 6

8.4.1 Proof of Lemma 6.2

We will use the following LLL condition which is obtained from (2.1) by setting $\psi_i = \frac{2\mu(B_i)}{1-2\mu(B_i)}$. It's proof can be found in [79].

Proposition 8.11. *Let (Ω, μ) be an arbitrary probability space and let $\mathcal{B} = \{B_1, \ldots, B_m\}$ be a set of (bad) events. For each $i \in [m]$ let $D(i) \subseteq ([m] \setminus \{i\})$ be such that $\mu(B_i \mid \cap_{j \in S} \overline{B_j}) = \mu(B_i)$ for*

every $S \subseteq (D(i) \cup \{i\})$. *If*

$$\sum_{j \in D(i) \cup \{i\}} \mu(f_j) < \frac{1}{4} \text{ for each } i \in [m] \quad ,$$

then the probability that none of the events in $\mathcal{B}$ *occurs is strictly positive.*

Let μ be the probability distribution induced by giving each Blank vertex v a color from $L_v(\sigma) \setminus \mathrm{Blank}$ uniformly at random. For any edge e and color $c \in \bigcap_{u \in e} L_u(\sigma) \setminus \mathrm{Blank}$ we define $A_{e,c}$ to be the event that all vertices of e receive c. We also define $\mathrm{Blank}(e)$ to be the set of vertices of e that are Blank in σ. Observe now that

$$\mu\left(A_{e,c}\right) \leq \frac{1}{\prod_{v \in \mathrm{Blank}(e)} \left(|L_v(\sigma)| - 1\right)} \quad .$$

Furthermore, $A_{e,c}$ is mutually independent of all events with which it does not share a vertex. The lemma follows from Proposition 8.11 (and can be made constructive using the Moser-Tardos algorithm) as flaws B_v, Z_v are not present for every vertex $v \in V$ and so

$$\begin{aligned}
\sum_{v \in \mathrm{Blank}(e)} \sum_{c \in L_v(\sigma) \setminus \mathrm{Blank}} \sum_{u \in T_{v,c}(\sigma)} \mu\left(A_{\{u,v\},c}\right) &= \sum_{v \in \mathrm{Blank}(e)} \sum_{c \in L_v(\sigma) \setminus \mathrm{Blank}} \sum_{u \in T_{v,c}(\sigma)} \frac{1}{(|L_v(\sigma)| - 1)(|L_u(\sigma)| - 1)} \\
&\leq 2 \max_{v \in \mathrm{Blank}(e)} \frac{1}{\left(|L_v(\sigma)| - 1\right)(L-1)} \sum_{c \in L_v(\sigma) \setminus \mathrm{Blank}} |T_{v,c'}(\sigma)| \\
&\leq \frac{2}{10} \max_{v \in \mathrm{Blank}(e)} \frac{L \cdot |L_v(\sigma)|}{(L-1) \cdot (|L_v(\sigma)| - 1)} \\
&\leq \frac{1}{5} \left(\frac{L}{L-1}\right)^2 < \frac{1}{4} \quad ,
\end{aligned}$$

for large enough Δ, concluding the proof.

8.4.2 Proof of Lemma 6.5

Recall the description of $\mathcal{A}_2$ from the proof of Lemma 6.2.

First, we show that $\mathcal{A}_2$ is able to output at least $Nq^{-(1-\alpha)n}$ list-colorings with positive probability. Let $\Omega^*_{\mathcal{A}_1}$ denote the set of flawless partial list-colorings algorithm $\mathcal{A}_1$ can output with positive probability, and note that, according to our assumption, $|\Omega^*_{\mathcal{A}_1}| = N$. To see the idea behind the bound, observe that given two colorings $\sigma_1, \sigma_2 \in \Omega^*_{\mathcal{A}_1}$, applying $\mathcal{A}_1$ to each one of them is guaranteed to result in different full list-colorings unless there is a way to start from σ_1 (respectively, from σ_2) and assign colors to Blank vertices so that we reach σ_2 (respectively, to σ_1). In this bad case we write $\sigma_1 \bowtie \sigma_2$. Consider now the graph H over $\Omega^*_{\mathcal{A}_1}$ in which two colorings σ_1, σ_2 are adjacent iff $\sigma_1 \bowtie \sigma_2$, and observe that the size of any independent set of H is a lower bound on the number of list-colorings $\mathcal{A}_2$ can output. Since we have assumed that every coloring in $\Omega^*_{\mathcal{A}_1}$ has at least αn vertices colored, we see that the maximum degree of H is at most $D := q^{(1-\alpha)n} - 1$ and, therefore, there exists an independent set of size at least $\frac{|\Omega^*_{\mathcal{A}_1}|}{D+1} = Nq^{-(1-\alpha)n}$, concluding the proof of the first part of Lemma 6.2.

Second, we show that $\mathcal{A}_2$ is able to output at least $\left(\frac{8L}{11}\right)^{(1-\alpha)n}$ list colorings with positive probability. To do that, we will need the following theorem regarding the output distribution of the Moser-Tardos algorithm that was proved in [56], and which we rephrase here to fit our needs.

Theorem 8.12 ([56])**.** *Consider a constraint satisfaction problem on a set of variables $\mathcal{V}$ and set of constraints $\mathcal{C}$. Assume we have a flaw f_c for each constraint c, comprising the set of states that violate c. We are also given an undirected causality graph such that two constraints are connected with an edge iff they share variables. For each constraint c define*

$$y_c = (1+\psi_c)^{\frac{1}{|\mathrm{var}(c)|}} - 1 \ ,$$

where $\mathrm{var}(c)$ denotes the set of variables that correspond to constraint c. Then:

$$\sum_{S\in \mathrm{Ind}(\mathcal{C})} \prod_{c\in S} \psi_c \leq \prod_{v\in\mathcal{V}} \left(1 + \sum_{\substack{c\in\mathcal{C} \\ v\in\mathrm{var}(c)}} y_c\right) \ ,$$

where $\mathrm{Ind}(\mathcal{C})$ denotes the set of independent sets of $\mathcal{C}$.

Observe that the hypothesis implies that $\mathcal{A}_1$ can output a flawless partial coloring σ where exactly αn vertices are colored with positive probability. We apply $\mathcal{A}_2$ to σ, which recall that is an instantiation of the Moser-Tardos algorithm using the uniform measure μ over the the cartesian product, Ω', of the lists of non-Blank available colors of the Blank vertices of σ, and where we have a bad event $A_{e,c}$ for any edge e and color $c \in \bigcap_{u\in e} L_u(\sigma)\setminus\{\mathrm{Blank}\}$. Recall further that the general (and, thus, also the cluster expansion) LLL condition (2.1) is satisfied with $\psi_{e,c} = \frac{2\mu(A_{e,c})}{1-2\mu(A_{e,c})}$. Thus, we can combine Theorem 5.11 and Theorem 8.12 to get that $\mathcal{A}_2$ can output at least

$$|\Omega'|\left(\prod_{\substack{v\in V \\ \sigma(v)=\mathrm{Blank}}} \left(1 + \sum_{c\in L_v(\sigma)\setminus\mathrm{Blank}} \sum_{u\in T_{v,c}(\sigma)} y_{\{u,v\},c}\right)\right)^{-1} \geq \frac{L^{(1-\alpha)n}}{\left(1+\frac{3}{8}\right)^{(1-\alpha)n}} \geq \left(\frac{8L}{11}\right)^{(1-\alpha)n} \ ,$$

list-colorings with positive probability. To see this, notice that for any $v \in V$ that is Blank in σ, and sufficiently large Δ,

$$\begin{aligned}
\sum_{c\in L_v(\sigma)\setminus\mathrm{Blank}} \sum_{u\in T_{v,c}(\sigma)} y_{\{u,v\},c} &= \sum_{c\in L_v(\sigma)\setminus\mathrm{Blank}} \sum_{u\in T_{v,c}(\sigma)} \left(\sqrt{1+\frac{2\mu(A_{\{u,v\},c})}{1-2\mu(A_{\{u,v\},c})}} - 1\right) \\
&\leq \sum_{c\in L_v(\sigma)\setminus\mathrm{Blank}} \sum_{u\in T_{v,c}(\sigma)} 3\mu(A_{\{u,v\},c}) \\
&\leq 3\cdot\left(\frac{1}{2}\cdot\frac{1}{4}\right) = \frac{3}{8} \ , \qquad (8.5)
\end{aligned}$$

where to obtain (8.5) we perform identical calculations to the ones in Lemma 6.2.

8.4.3 Proof of Theorem 6.10

We will follow closely the approach adopted by the authors in [10]. Throughout the proof we assume that $\epsilon \in (0, \epsilon_0)$, where ϵ_0 is sufficiently small, and that $f_\epsilon > 0$ and $\Delta_\epsilon > 0$ are sufficiently large.

We distinguish two cases, depending on whether $f \geq \Delta^{(2+\epsilon^2)\epsilon}$ or not. To prove Theorem 6.10 for the case $f \geq \Delta^{(2+\epsilon^2)\epsilon}$ we will prove the following.

Theorem 8.13. *For every $\theta, \zeta \in (0,1)$, there exists $\Delta_{\theta,\zeta} > 0$ such that every graph G with maximum degree $\Delta \geq \Delta_{\theta,\zeta}$ in which the neighbors of every vertex span at most $\Delta^{2-(2+\zeta)\theta}$ edges, has chromatic number $\chi(G) \leq (1+\zeta)(1+\theta^{-1})\frac{\Delta}{\ln \Delta}$.*

Proof of Theorem 6.10 for $f \geq \Delta^{(2+\epsilon^2)\epsilon}$. We apply Theorem 8.13 with $\zeta = \epsilon^2$ and $\theta = \frac{\ln f}{(2+\epsilon^2)\ln \Delta} \geq \epsilon$, so that $\Delta^2/f = \Delta^{2-(2+\zeta)\theta}$. Since $\zeta, \theta < 1$, we obtain

$$\begin{aligned}
\chi(G) &\leq (1+\zeta)\left(1 + \frac{(2+\zeta)\ln \Delta}{\ln f}\right)\frac{\Delta}{\ln \Delta} \\
&= (1+\zeta)\frac{\Delta}{\ln \Delta} + (1+\zeta)(1+\zeta/2)\frac{\Delta}{\ln \sqrt{f}} \\
&\leq (1+2\zeta)\frac{\Delta}{\ln \sqrt{f}} + (1+\zeta)(1+\zeta/2)\frac{\Delta}{\ln \sqrt{f}} \\
&= \left(2 + \frac{7\zeta}{2} + \frac{\zeta^2}{2}\right)\frac{\Delta}{\ln \sqrt{f}} \\
&\leq (2+\epsilon)\frac{\Delta}{\ln \sqrt{f}} \ .
\end{aligned}$$

□

Theorem 8.13 follows immediately from the following lemma, whose proof is similar to Lemma 2.3 in [10] and can be found in Section 8.4.4. The proof of Lemma 8.14 uses the standard Local Lemma and Theorem 6.8, so it can be made constructive using the Moser-Tardos algorithm and the algorithm in Theorem 6.8.

Lemma 8.14. *For every $\theta, \zeta \in (0,1)$ there exists $\Delta_{\theta,\zeta} > 0$ such that for every graph $G = (V, E)$ with maximum degree $\Delta \geq \Delta_{\theta,\zeta}$ in which the neighbors of every vertex span at most $\Delta^{2-(2+\zeta)\theta}$ edges, there exists a partition of the vertex set $V = V_1 \cup \ldots \cup V_k$ with $k = \Delta^{1-\theta}$, such that for every $1 \leq i \leq k$,*

$$\chi(G[V_i]) \leq (1+\zeta)(1+\theta^{-1})\frac{\Delta^\theta}{\ln \Delta} \ .$$

Proof of Theorem 8.13. If $V_1, V_2, \ldots, V_k$, $k = \Delta^{1-\theta}$ is the partition promised by Lemma 8.14 then

$$\chi(G) \leq \sum_{i=1}^{\Delta^{1-\theta}} \chi(G[V_i]) \leq (1+\zeta)(1+\theta^{-1})\frac{\Delta}{\ln \Delta} \ .$$

□

To prove Theorem 6.10 for $f \in [f_\epsilon, \Delta^{(2+\epsilon^2)\epsilon})$, we will perform a sequence of random halving steps, as in [10], to partition the graph into subgraphs satisfying the condition of Theorem 6.10 with $f \geq \Delta^{(2+\epsilon^2)\epsilon}$ and color these subgraphs using disjoint sets of colors. To perform the partition we use the following lemma from [10]. As it is proven via the standard LLL, it can be made constructive using the Moser-Tardos algorithm.

Lemma 8.15 ([10])**.** *Let $G(V, E)$ be a graph with maximum degree $\Delta \geq 2$ in which the neighbors of every vertex span at most s edges. There exists a partition $V = V_1 \cup V_2$ such that the induced subgraph $G[V_i], i = 1, 2$, has maximum degree at most $\Delta/2 + 2\sqrt{\Delta \ln \Delta}$ and the neighborhors of every vertex in $G[V_i], i = 1, 2$, span at most $s/4 + 2\Delta^{\frac{3}{2}}\sqrt{\ln \Delta}$ edges.*

We will also use the following lemma whose proof, presented in Section 8.4.5, is almost identical to a similar statement in the proof of Theorem 1.1 of [10].

Lemma 8.16. *Given Δ, f sufficiently large, let the sequences Δ_t and s_t be defined as follows. $\Delta_0 = \Delta, s_0 = \Delta^2/f$ and*

$$\Delta_{t+1} = \Delta_t/2 + 2\sqrt{\Delta_t \ln \Delta_t}, \;\; s_{t+1} = s_t/4 + 2\Delta_t^{\frac{3}{2}}\sqrt{\ln \Delta_t} \;\; .$$

For any $\delta \in (0, 1/100)$ and $\zeta > 0$ such that $\zeta(2+\delta) < 1/10$, let j be the smallest integer for which $f > \left(\frac{(1+\delta)\Delta}{2^j}\right)^{(2+\delta)\zeta}$. Then $\Delta_j \leq (1+\delta)\Delta/2^j$ and $s_j \leq \left((1+\delta)\Delta/2^j\right)^2/f$.

Proof of Theorem 6.10 for $f \in [f_\epsilon, \Delta^{(2+\epsilon^2)\epsilon})$. Let $\epsilon_0 = 1/11$. For $\epsilon \in (0, \epsilon_0]$, let $\delta = \zeta = \epsilon^2$. Since $\zeta(2+\delta) < 1/10$, apply Lemma 8.16 and let $j = j(\Delta, f, \delta, \zeta)$ be the integer described therein. Let S be the process which, given a graph G, does nothing if $\Delta(G) < 2$, and otherwise partitions G as described in Lemma 8.15. Apply S to G to get subgraphs $G[V_1], G[V_2]$. Apply S to $G[V_1], G[V_2]$ to get $G[V_{1,1}], G[V_{1,2}], G[V_{2,1}], G[V_{2,2}]$. And so on, j times, obtaining a partition of G into at most 2^j induced subgraphs. Observe that for each such subgraph H, either $\Delta(G) < 2$ and, thus, $\chi(H) \leq 2$, or, by Lemma 8.16, $\Delta(H) \leq (1+\delta)\Delta/2^j =: \Delta_*$ and the neighbors of every vertex in H span at most Δ_*^2/f edges, where $f \geq \Delta_*^{(2+\delta)\zeta} = \Delta_*^{(2+\zeta)\zeta} \geq \Delta_*^{(2+\zeta^2)\zeta}$. Therefore, by the already established case of Theorem 6.10, either $\chi(H) \leq 2$, or $\chi(H) \leq (2+\zeta)\Delta_*/\ln\sqrt{f}$. Thus,

$$\chi(G) \leq 2^j \max\left\{2, (2+\zeta)\frac{(1+\delta)\Delta/2^j}{\ln\sqrt{f}}\right\} \leq \max\left\{2^{j+1}, (2+\zeta)\frac{(1+\delta)\Delta}{\ln\sqrt{f}}\right\} \;\; .$$

To bound 2^{j+1} from above we first observe that for all f sufficiently large, i.e., for all $f \geq f_\epsilon$,

$$\left(\frac{(1+\delta)\Delta}{\frac{\Delta}{2\ln\sqrt{f}}}\right)^{(2+\delta)\zeta} = \left(2(1+\delta)\ln\sqrt{f}\right)^{(2+\delta)\zeta} < f \;\; . \tag{8.6}$$

Now, since j was defined as the smallest integer for which $\left(\frac{(1+\delta)\Delta}{2^j}\right)^{(2+\delta)\zeta} < f$, we see that (8.6) implies $2^j \leq \frac{\Delta}{2\ln\sqrt{f}}$ and, therefore, $2^{j+1} \leq \frac{\Delta}{\ln\sqrt{f}}$. Finally, we observe that $(2+\zeta)(1+\delta) = (2+\epsilon^2)(1+\epsilon^2) < 2+\epsilon$ for all $\epsilon \in (0, \epsilon_0]$. Therefore, as claimed,

$$\chi(G) \leq (2+\epsilon)\frac{\Delta}{\ln\sqrt{f}} \;\; .$$

□

8.4.4 Proof of Lemma 8.14

We follow an approach similar to the one of Lemma 2.3 in [10] making appropriate modifications as needed. First we partition the vertices of G into $\Delta^{1-\theta}$ parts by coloring them randomly and independently with $\Delta^{1-\theta}$ colors. For a vertex v and a neighbor u adjacent to it, call u a *bad neighbor* of v, if u and v have at least $\Delta^{1-(1+\zeta/2)\theta}$ common neighbors. Otherwise, say that u is a *good neighbor*. Since the neighbors of every vertex span at most $\Delta^{2-(2+\zeta)\theta}$ edges, there are at most $2\Delta^{1-(1+\zeta/2)\theta}$ bad neighbors for any vertex in G.

For any vertex v, define three types of bad event with respect to the random partitioning experiment.

- A_v: more than $(1+\theta)\Delta^\theta$ neighbors of v receive the same color as v.
- B_v: more than $\frac{10}{\theta\zeta}$ bad neighbors of v receive the same color as v.
- C_v: the good neighbors of v that receive the same color as v span more than $\frac{100}{(\theta\zeta)^2}$ edges.

We will use the symmetric version of the Local Lemma [36] to show that we can find a coloring of the graph that avoids all bad events. First, note that each of the bad events A_v, B_v, C_v is independent of all but at most Δ^2 others, as it independent of all events A_u, B_u, C_u corresponding to vertices u whose distance from v is more than 2. Since the degree of any vertex in its colors class is binomially distributed with mean at most Δ^θ, standard Chernoff estimates imply that the probability that v has more than $(1+\theta)\Delta^\theta$ neighbors of the same color as that of v is at most $\mathrm{e}^{-\Omega(\Delta^\theta)}$, which means that $\Pr[A_v] < \Delta^{-3}$ for large enough Δ. Moreover, we also have

$$\Pr[B_v] \leq \binom{2\Delta^{1-(1+\zeta/2)\theta}}{\frac{10}{\theta\zeta}} \left(\frac{1}{\Delta^{1-\theta}}\right)^{\frac{10}{\theta\zeta}} \leq \left(\frac{2}{\Delta^{\frac{\theta\zeta}{2}}}\right)^{\frac{10}{\theta\zeta}} \leq \Delta^{-3} \ ,$$

for large enough Δ. Finally, to bound the probability of C_v we make the following observation. If a graph has at least e^2 edges, then either it has a vertex of degree at least e, or every vertex has degree strictly less than e, implying that the graph can be edge-colored with $\lceil e \rceil$ colors, in which case the largest color class must contain at least $e^2/\lceil e \rceil \geq e-1$ edges. Thus, a graph with more than $100/(\theta\zeta)^2$ edges either has a vertex of degree at least $10/(\theta\zeta) \geq 9/(\theta\zeta)$ or a matching with at least $10/(\theta\zeta) - 1 \geq 9/(\theta\zeta)$ edges, where the inequality follows from the fact that $\theta, \zeta < 1$. Thus, C_v can happen only if there is a good neighbor u of v such that u and v have at least $9/(\theta\zeta)$ common neighbors with the same color as v, or if there is a matching of size at least $9/(\theta\zeta)$ on the good neighbors of v that have the same color as v. The probabilities of the first and second of these events are bounded, respectively, by

$$\Delta\binom{\Delta^{1-(1+\zeta/2)\theta}}{\frac{9}{\theta\zeta}} \left(\frac{1}{\Delta^{1-\theta}}\right)^{\frac{9}{\theta\zeta}} \leq \left(\frac{1}{\Delta^{\frac{\theta\zeta}{2}}}\right)^{\frac{9}{\theta\zeta}} \leq \frac{1}{2}\Delta^{-3} \ ,$$

$$\binom{\Delta^{2-(2+\zeta)\theta}}{\frac{9}{\theta\zeta}} \left(\left(\frac{1}{\Delta^{1-\theta}}\right)^2\right)^{\frac{9}{\theta\zeta}} \leq \left(\frac{1}{\Delta^{\theta\zeta}}\right)^{\frac{9}{\theta\zeta}} \leq \frac{1}{2}\Delta^{-3} \ .$$

Therefore the probability of C_v is at most Δ^{-3}. Thus, the Local Lemma applies since each bad event has probability at most Δ^{-3} and is independent of all but at most Δ^2 other bad events. This means that we can find a partition $V = V_1, \ldots, V_k$, where $k = \Delta^{1-\theta}$, so that in each induced subgraph $G[V_i]$, every vertex: has degree at most $(1+\theta)\Delta^\theta$, has at most $\frac{10}{\theta\zeta}$ bad neighbors, and is contained in at most $\frac{100}{(\theta\zeta)^2}$ triangles in which both other vertices are good. We will show that, given such a partition, each $G[V_i]$ can be colored with at most $\frac{(1+\zeta)(1+\theta^{-1})\Delta^\theta}{\ln \Delta}$ colors, assuming Δ is large enough.

To see this, consider the partition $B_i, V_i \setminus B_i$ of V_i, where B_i is the set of vertices $u \in V_i$ for which there exists a vertex $v \in V_i$, such that u is a bad neighbor of v. We claim that $\chi(G[B_i]) \leq \frac{20}{\theta\zeta} + 1$ and $\chi(G[V_i \setminus B_i]) \leq \frac{(1+\zeta/2)(1+\theta^{-1})\Delta^\theta}{\ln \Delta}$. Assuming this claim, observe that

$$\chi(G[V_i]) \leq \frac{20}{\theta\zeta} + 1 + \frac{(1+\zeta/2)(1+\theta^{-1})\Delta^\theta}{\ln \Delta} \leq \frac{(1+\zeta)(1+\theta^{-1})\Delta^\theta}{\ln \Delta} \ ,$$

where the last inequality holds for all $\Delta \geq \Delta_{\theta,\zeta}$.

To see the first part of the claim, note that it is well-known (and easy to see) that if a graph has an orientation with maximum outdegree d, then it is $(2d+1)$-colorable. Consider the orientation of the graph on B_i that results when every vertex points to its bad neighbors in B_i. Clearly, the maximum outdegree is at most $\frac{10}{\theta\zeta}$ and, thus, $\chi(G[B_i]) \leq \frac{20}{\theta\zeta} + 1$.

To see the second part of the claim, observe that each vertex of $G[V_i \setminus B_i]$ is contained in at most $\frac{100}{(\theta\zeta)^2}$ triangles. Let $\Delta_* = (1+\theta)\Delta^\theta$ and

$$f = \frac{\left((1+\theta)\Delta^\theta\right)^2}{100/(\theta\zeta)^2} = \frac{(\theta\zeta)^2\Delta_*^2}{100} \geq \Delta_*^{\frac{2+\frac{2\zeta}{3}}{1+\frac{2\zeta}{3}}} \left(\ln \Delta_*\right)^2 \ ,$$

where the last inequality holds for $\Delta \geq \Delta_{\theta,\zeta}$. Applying Theorem 6.8 to $G[V_i \setminus B_i]$ (by plugging in $\zeta/3$ for the ϵ in Theorem 6.8) we get that, for all $\Delta \geq \Delta_{\theta,\zeta}$,

$$\begin{aligned}
\chi_\ell(G[V_i \setminus B_i]) &\leq (1+\zeta/3)\frac{\Delta_*}{\ln \sqrt{f}} \\
&= (1+\zeta/3)\frac{(1+\theta)\Delta^\theta}{\ln \frac{(1+\theta)\theta\zeta\Delta^\theta}{10}} \\
&= (1+\zeta/3)\frac{(1+\theta)\Delta^\theta}{\theta \ln \Delta + \ln \frac{(1+\theta)\theta\zeta}{10}} \\
&\leq (1+\zeta/2)\frac{(1+\theta)}{\theta}\frac{\Delta^\theta}{\ln \Delta} \\
&= (1+\zeta/2)\frac{(1+\theta^{-1})\Delta^\theta}{\ln \Delta} \ ,
\end{aligned}$$

as claimed.

8.4.5 Proof of Lemma 8.16

Let $\epsilon' := \zeta(2+\delta)$ and recall that $\epsilon' < \frac{1}{10}$ by hypothesis. By the definition of j, for every $t < j$,

$$\Delta_t \geq \Delta/(2^t) > \frac{f^{\frac{1}{\epsilon'}}}{1+\delta} \ ,$$

and $f^{\frac{1}{\epsilon'}}/(1+\delta)$ can be made arbitrarily large by taking f to be sufficiently large. Hence, we can assume that Δ_t is sufficiently large in order for $\Delta_{t+1} \leq \frac{\Delta_t}{2} + \Delta_t^{\frac{2}{3}} \leq \frac{1}{2}\left(\Delta_t^{\frac{1}{3}} + 1\right)^3$ to hold. Taking cube roots and subtracting $\frac{1}{2^{\frac{1}{3}}-1}$ from both sides we get

$$\Delta_{t+1}^{\frac{1}{3}} - \frac{1}{2^{\frac{1}{3}}-1} \leq \frac{1}{2^{\frac{1}{3}}}(\Delta_t^{\frac{1}{3}} + 1) - \frac{1}{2^{\frac{1}{3}}-1} = \frac{1}{2^{\frac{1}{3}}}\left(\Delta_t^{\frac{1}{3}} - \frac{1}{2^{\frac{1}{3}}-1}\right) \ .$$

Therefore,

$$\Delta_j^{\frac{1}{3}} - \frac{1}{2^{\frac{1}{3}}-1} \leq \frac{1}{2^{j/3}}\left(\Delta_0^{\frac{1}{3}} - \frac{1}{2^{\frac{1}{3}}-1}\right) \ . \tag{8.7}$$

Since $\Delta_0 = \Delta$, $2^{\frac{1}{3}} - 1 > \frac{1}{4}$ and $\Delta/2^{j-1} > \frac{f^{\frac{1}{\epsilon'}}}{1+\delta}$ is large enough, (8.7) implies that

$$\Delta_j^{\frac{1}{3}} \leq \frac{\Delta^{\frac{1}{3}}}{2^{j/3}} + 4 \leq (1+\delta)^{\frac{1}{3}}\frac{\Delta^{\frac{1}{3}}}{2^{j/3}} \ .$$

Therefore, we have shown that $\Delta_j \leq (1+\delta)\frac{\Delta}{2^j}$. Note also that the same proof shows that for every $t \leq j$ we have that $\Delta_t \leq (1+\delta)\frac{\Delta}{2^t}$.

We turn now to the claim regarding s_j. For all $t < j$, we have by definition

$$s_t \geq \frac{s_0}{4^t} = \frac{\Delta^2}{4^t f} = \frac{1}{(1+\delta)^2}\frac{\left(\frac{(1+\delta)\Delta}{2^t}\right)^2}{f} \geq \frac{1}{(1+\delta)^2}\left(\frac{(1+\delta)\Delta}{2^t}\right)^{2-\epsilon'} \geq \frac{1}{(1+\delta)^2}\Delta_t^{2-\epsilon'} \ , \tag{8.8}$$

where in the last inequality we used the fact that $\Delta_t \leq (1+\delta)\frac{\Delta}{2^t}$ for all $t \leq j$. Using (8.8) to bound Δ_t in the expression that defines s_t, we get

$$s_{t+1} \leq \frac{s_t}{4} + 2((1+\delta)^2 s_t)^{\frac{3}{2}\frac{1}{(2-2\epsilon')}} \ . \tag{8.9}$$

To bound the r.h.s. of (8.9) we recall that $\epsilon' < \frac{1}{10}$ implying $\frac{3}{2(2-2\epsilon')} < \frac{3}{2(2-1/5)} = 5/6$. Assuming that f (and, thus, Δ_t) is large enough, we obtain

$$s_{t+1} \leq \frac{s_t}{4} + 3s_t^{5/6} \leq \frac{1}{4}(s_t^{1/6} + 2)^6 = \frac{1}{4}(s_t + 12s_t^{5/6} + 60s_t^{2/3} + \cdots) \ . \tag{8.10}$$

Hence, taking 6-th roots and subtracting $\frac{5}{6^{1/6}-1}$ from both sides, we obtain

$$s_{t+1}^{1/6} - \frac{5}{6^{1/6}-1} \leq \frac{1}{4^{1/6}}(s_t^{1/6} + 2) - \frac{5}{6^{1/6}-1} \leq \frac{1}{4^{1/6}}\left(s_t^{1/6} - \frac{5}{6^{1/6}-1}\right) \ .$$

Therefore

$$s_j^{1/6} - \frac{5}{6^{1/6}-1} \leq \frac{1}{4^{j/6}} \left(s_0^{1/6} - \frac{5}{6^{1/6}-1} \right) \ ,$$

and, since $s_0 = \Delta^2/f$,

$$s_j^{1/6} \leq \frac{(\Delta^2)^{1/6}}{4^{j/6} f^{1/6}} + \frac{5}{6^{1/6}-1} \leq \left(\frac{\Delta^2}{4^j f} \right)^{1/6} + 15 \ .$$

Since

$$\frac{\Delta^2}{4^j f} = \left(\frac{\Delta}{2^j} \right)^2 \frac{1}{f} = \left(\frac{\Delta}{2^{j-1}} \right)^2 \frac{1}{4f} > \left(\frac{f^{1/\epsilon'}}{1+\delta} \right)^2 \frac{1}{4f} = \frac{f^{\frac{2}{\epsilon'}-1}}{4(1+\delta)^2}$$

can be made arbitrarily large by taking f sufficiently large, we see that

$$\left(\frac{\Delta^2}{4^j f} \right)^{1/6} + 15 \leq (1+\delta)^{1/3} \left(\frac{\Delta^2}{4^j f} \right)^{1/6} \ .$$

Thus, $s_j \leq ((1+\delta)\Delta/2^j)^2/f$, completing the proof.

8.4.6 Proof of Proposition 6.9

We use the term "with high probability" to refer to probabilities that tend to 1 as n goes to infinity. Corollary 6.9 follows in a straightforward way from the following lemma.

Lemma 8.17. *For any $\delta \in (0,1)$ there exists a constant d_0 such that, for any $d \in \left(d_0 \ln n, (n \ln n)^{\frac{1}{3}} \right)$, each vertex of the random graph $G = G(n, d/n)$ is contained in at most Δ^δ triangles with high probability, where Δ is the maximum degree of G.*

Proof of Corollary 6.9. According to [7], for a graph $G \in G(n, d/n)$ we know that with high probability

$$\chi(G) = \frac{1}{2} \frac{d}{\ln d} (1 + o(1)) \ . \tag{8.11}$$

Fix $\zeta \in (0,1)$ and $\delta \in (0, \frac{2\zeta}{1+2\zeta})$. According to Lemma 8.17, there exists a constant d_0 such that for any $d \in \left(d_0 \ln n, (n \ln n)^{\frac{1}{3}} \right)$ each vertex of $G = G(n, d/n)$ is contained in at most Δ^δ triangles with probability that tends to 1 as n goes to infinity. Thus, we can apply Theorem 6.8 with parameter $\zeta > 0$ since

$$f = \frac{\Delta^2}{\Delta^\delta} > \Delta^{2 - \frac{2\zeta}{1+2\zeta}} (\ln \Delta)^2,$$

for large enough Δ. This yields an upper bound q on the chromatic number of G that is at most

$$\begin{aligned} q &= (1+\zeta) \frac{\Delta}{\ln \sqrt{f}} \\ &\leq (1+\zeta) \frac{\Delta}{\frac{1+\zeta}{1+2\zeta} \ln \Delta + \ln \ln \Delta} \\ &\leq (1+2\zeta) \frac{\Delta}{\ln \Delta} \ . \end{aligned} \tag{8.12}$$

Moreover, since the expected degree of every vertex of G is d and its distribution is binomial with parameter $\frac{d}{n}$, standard Chernoff bounds and the union bound imply that for any $\eta \in (0,1)$, $\Delta \leq (1+\eta)d$ with high probability, for large enough d_0.

Combining the latter fact with (8.11) and (8.12), we deduce that we can find an arbitrarily small constant $\eta' \in (0,1)$ such that

$$q \leq (2+\eta')\chi(G)$$

by choosing ζ and η sufficiently small. Picking $\eta' = \frac{4\epsilon}{1-2\epsilon}$ we obtain $\chi(G) \geq \frac{q}{2+\eta'} \geq q(\frac{1}{2}-\epsilon)$, concluding the proof of Proposition 6.9. □

8.4.7 Proof of Lemma 8.17

Let Δ_v be the random variable that equals the degree of vertex v of G. Observe that $\Delta_v \sim \mathrm{Binom}(n-1, \frac{d}{n})$ and, therefore, using a standard Chernoff bound and the fact that $d \geq d_0 \log n$ we get that

$$\Pr\left[\Delta_v \notin (1 \pm \frac{1}{10})d\right] \leq \frac{1}{n^2} \ ,$$

for large enough d_0. Thus, by a union bound we get that $\Pr[\Delta \in (1 \pm \frac{1}{10})d] \leq \frac{1}{n}$.

Let T_v be the number of triangles that contain vertex v and B be the event that $\Delta \notin (1 \pm \frac{1}{10})d$. Then,

$$\Pr[T_v > \Delta^\delta] \leq \Pr[T_v > \Delta^\delta \mid \overline{B}] + \Pr[B] \leq \frac{\Pr[\left(T_v > \Delta^\delta\right) \cap \overline{B}]}{1-\frac{1}{n}} + \frac{1}{n} \ .$$

Observe that $T_v \sim \mathrm{Binom}\left(\binom{n-1}{2}, \left(\frac{d}{n}\right)^3\right)$ and $\mathbb{E}[T_v] \leq \frac{d^3}{2n}$. Thus, for any fixed value of $\Delta \in (1 \pm \frac{1}{10})d$, setting $1+\beta = \frac{\Delta^\delta}{d^3/2n}$ and using a standard Chernoff bound we obtain:

$$\Pr[T_v > \Delta^\delta] \leq \mathrm{e}^{-\frac{\beta^2 d^3/2n}{3}} \leq \frac{1}{n^2}$$

since

$$\begin{aligned} \beta &\geq \frac{\left((1-\frac{1}{10})d\right)^\delta - d^3/2n}{d^3/2n} > 0 \ , \\ \frac{1}{3}\beta^2\frac{d^3}{2n} &\geq \frac{1}{3}\frac{\left(\left((1-\frac{1}{10})d\right)^\delta - d^3/2n\right)^2}{d^3/2n} \geq 2\ln n \ , \end{aligned}$$

whenever $d \in [d_0 \ln n, (n \ln n)^{\frac{1}{3}}]$ and for large enough n and d_0. Taking a union bound over v concludes the proof of the lemma.

8.5 Proofs Omitted from Chapter 7

8.5.1 Proof of Lemma 7.16

We will need the following standard concentration bound (see, e.g., Chapter 10, Section 10.1 of [79]).

Lemma 8.18. *Let X be a random variable determined by n independent trials $T_1, \ldots, T_n$, and such that changing the outcome of any one trial can affect X by at most c. Then*

$$\Pr[|X - \mathbb{E}[X]| > \lambda] \leq 2\mathrm{e}^{-\frac{\lambda^2}{2c^2 n}} \ .$$

Proof of Part (a) of Lemma 7.16. Recall that $t = 8(K+1)^2\delta^{-1} + 2$ and that $\delta = \frac{\epsilon}{4}$. Consider a random state σ distributed according to μ and a fixed state $\tau \in \Omega$, and notice that applying Theorem 7.7 with the parameter ϵ instantiated to δ and our choice of t imply that

$$\mu(e \in M_i \mid Q_v^i(t,\sigma) = Q_v^i(t,\tau)) \geq (1-\delta)\frac{1-\delta}{\chi_e^*(G)} \geq \frac{1-\frac{\epsilon}{2}}{\chi_e^*(G)} \ ,$$

for any vertex v, any edge e adjacent to v and any $i \in [N]$. This implies

$$\mathbb{E}[d_{G_\sigma}(v) \mid Q_v^i(t,\sigma) = Q_v^i(t,\tau)] \leq \Delta\left(1 - \frac{1-\frac{\epsilon}{2}}{\chi_e^*(G)}\right)^N \leq \chi_e^*(G)\left(1 - \frac{1-\frac{\epsilon}{2}}{\chi_e^*(G)}\right)^N \ . \tag{8.13}$$

Now, since $N = o(\chi_e^*(G))$, we have

$$\mathbb{E}[d_{G_\sigma}(v) \mid Q_v^i(t,\sigma) = Q_v^i(t,\tau)] \leq \chi_e^*(G)\left(1 - (1+o(1))\frac{(1-\frac{\epsilon}{2})N}{\chi_e^*(G)}\right) \leq \chi_e^*(G) - \left(1 - \frac{9\epsilon}{17}\right)N \ . \tag{8.14}$$

Further, since $c^* = \chi_e^*(G) - (1+\epsilon)^{-1}N$ and $\epsilon \leq \frac{1}{10}$, (8.14) yields

$$\mathbb{E}[d_{G_\sigma}(v) \mid Q_v^i(t,\sigma) = Q_v^i(t,\tau)] \leq c^* - \left(1 - \frac{9\epsilon}{17} - (1+\epsilon)^{-1}\right)N \leq c^* - \frac{\epsilon}{3}N \ . \tag{8.15}$$

As the choices of the M_i are independent and each affects the degree of v in G' by at most 1, we can apply Lemma 8.18 with $\lambda = (\frac{\epsilon}{3} - \frac{\epsilon}{4})N = \frac{\epsilon}{12}N$ to prove part (a). In particular, recalling that $N = \lfloor \chi_e^*(G)^{\frac{3}{4}} \rfloor \sim \Delta^{3/4}$ we have that

$$\mu\left(d_{G_\sigma(v)} > c^* - \frac{\epsilon}{4}N \;\middle|\; Q_v^i(t,\sigma) = Q_v^i(t,\tau)\right) \leq 2\mathrm{e}^{-\frac{\lambda^2}{2N}} \leq \frac{1}{\Delta^{C+\Delta^{\frac{1}{3}}}} \ ,$$

for any constant C for sufficiently large Δ. □

Proof of Part (b) . The proof of part (b) is similar. Consider again a random state σ distributed according to μ and fix a state $\tau \in \Omega$. Theorem 7.7 implies that for each $i \in [N]$, the probability that an edge e with both endpoints in H is in M_i, conditional on $Q_H^i(t,\sigma) = Q_H^i(t,\tau)$, is at least $(1-\delta)\frac{1-\delta}{\chi_e^*(G)} \geq \frac{1-\frac{\epsilon}{2}}{\chi_e^*(G)}$. Moreover, Edmonds' characterization of the matching polytope (which we

have already seen in the the proof of Lemma 7.11) implies that the number of edges in G with both endpoints in H is at most $\chi_e^*(G)\lfloor\frac{V(H)-1}{2}\rfloor$. Similar calculations to the ones in part (a) reveal that

$$\mathbb{E}[|E_\sigma(H)| \mid Q_H^i(t,\sigma) = Q_H^i(t,\tau)] \leq \left(\frac{V(H)-1}{2}\right)(c^* - \frac{\epsilon}{3}N) \ ,$$

where $E_\sigma(H)$ is the set of edges of G_σ induced by H. Since the choices of matchings M_i are independent and each affects $|E_\sigma(H)|$ by at most $\frac{|V(H)|-1}{2}$, we can again apply Lemma 8.18 to prove part (b). □