Topological Optimization in Network Dynamical Systems

Dissertation

for the award of the degree

"Doctor of Philosophy" (Ph.D.)

Division of Mathematics and Natural Sciences
of the Georg-August-Universität Göttingen

submitted by

Frank Van Bussel
of London, Ontario, Canada

Göttingen 2010

Thesis Committee:

Prof. Dr. Marc Timme (thesis supervisor and 1st reviewer)
Network Dynamics Group, MPI for Dynamics and Self-Organization
Prof. Dr. Annette Zippelius (2nd reviewer)
Dept. of Theoretical Physics, Georg August University
Prof. Dr. Fred Wolf
Dept. of Non-linear Dynamics, MPI for Dynamics and Self-Organization

Date of disputation: August 25, 2010

I hereby declare that this dissertation was completed independently and without any unauthorized support.

Göttingen, June 29 2010 Frank Van Bussel

To my wife Zeina, without whom none of this would have been possible.

(... also want to thank my Mom and family in Canada for their support and encouragement, my defense committee, and the staff at GGNB; and give a big shout-out to the gang at MPIDS/NDG: Marc (my Sup), Birgit, Harold (honorary), Carsten, Fabio & Lishma, Niels and the rest ... plus NLD and associates: Ollie, Holger, Hecke, Kai, Michael (K/M), Mick, Katja, Tatjana and the others who started with me ... Jan the Man, Eule, Annette (Witt), Dirk in χ-town, Fred, Ragnar and the other senior researchers, and Sara Solla & Rob Shaw (like family to us) ... my officemate Andres, Olav, Mirko and the PhDNet crew (beers on me at Salamanca) ... Denny (my unofficial other mentor here), Yorke, Marcus (the one-man Golden Horde) ... Tobias, Katharina, Regina, Tanja, and Monika at Complex Fluids who's helped out Zeina and me a lot over the years ... and to Theo: "keep on jazzin' in the free world!")

Contents

Papers submitted for this thesis 8

1 Introduction **9**

1.1 Preliminaries 9

1.1.1 Summary of Topics 10

1.2 Network Reconstruction 11

1.2.1 Functional connectivity 11

1.2.2 Explicit reconstruction methods for oscillator networks 12

1.2.3 The response dynamics framework 13

1.2.4 Explicit reconstruction methods for pulse-coupled networks 14

1.2.5 Leaky integrate-and-fire neuron model 15

1.3 Chromatic Polynomials 16

1.3.1 The Potts model 17

1.3.2 Chromatic roots 18

1.3.3 Computational complexity 20

1.3.4 Random graphs 21

1.3.5 Collecting the threads 22

2 Article: *Inferring Synaptic Connectivity from Spatio-Temporal Spike Patterns* **23**

3 Article: *Counting Complex Disordered States by Efficient Pattern Matching: Chromatic Polynomials and Potts Partition Functions* **39**

4 Article: *Chromatic Polynomials of Random Graphs* **51**

5 Ongoing and Future Work **65**

5.1 Network Reconstruction 65

5.1.1 Logarithmic versus square-Root scaling of $M_{q,\alpha}$ . . . 66
5.1.2 Reconstruction of noisy LIF networks . . . 67
5.1.3 Parallelization and subnetwork reconstruction . . . 69
5.1.4 A comprehensive overview . . . 70
5.2 Chromatic Polynomials . . . 71
5.2.1 Lattice calculations . . . 72
5.2.2 Random graphs . . . 75
5.2.3 Other graph polynomials . . . 78
5.3 General Prospects and Conclusions . . . 81

References **83**

Statement of Contributions to Articles **91**

Curriculum Vitae **93**

A Network Reconstruction: Additional Material **97**
A.1 Overview of Network Reconstruction Procedures . . . 97
A.2 Selected Code . . . 102
A.2.1 Specialized (fast) reconstruction method for LIF networks . . . 102
A.2.2 General reconstruction method for LIF networks . . . 103
A.2.3 Sparse system solver . . . 105
A.2.4 Barrowdale-Roberts L_1-norm minimizer . . . 106

B Chromatic Polynomials: Additional Material **109**
B.1 Overview of Chromatic Polynomial Procedures . . . 109
B.2 Selected Code . . . 113
B.2.1 FORM code without optimizations . . . 116
B.2.2 FORM code with basic optimizations . . . 117
B.2.3 FORM production code . . . 119

C Some Remaining Issues **121**
C.1 Graph Generation and Manipulation Programs . . . 121

C.2 Graph Diagnostic and Analysis Programs 123
C.3 Subsidiary Non-Graph Techniques 124
C.4 Specialized Chromatic Polynomial Techniques 129
C.5 Computational Complexity 133
C.5.1 Complexity classes 133
C.5.2 Graph width metrics 139

Papers submitted for this thesis

1. **Inferring Synaptic Connectivity from Spatio-Temporal Spike Patterns**

   1st author; with Birgit Kriener and Marc Timme; under review at: *Frontiers in Computational Neuroscience*

2. **Counting Complex Disordered States by Efficient Pattern Matching: Chromatic Polynomials and Potts Partition Functions**

   2nd author; with Marc Timme, Denny Fliegner, and Sebastian Stolzenberg; published: *New Journal of Physics* **11** 023001 (2009)

3. **Chromatic Polynomials of Random Graphs**

   1st author; with Christoph Ehrlich, Denny Fliegner, Sebastian Stolzenberg, and Marc Timme; published: *Journal of Physics A: Mathematical and Theoretical* **43** 175002 (2010)

Previous publications

A Theorem on the Quantum Evaluation of Weight Enumerators for a Certain Class of Cyclic Codes with a Note on Cyclotomic Cosets; 2nd author, with Joe Geraci; *CERN Document Server* `cs.IT/0703129` (2007)

Rapid Mixing for Lattice Colourings with Fewer Colours; 4th author, with Dimitris Achlioptas, Mike Molloy, and Cristopher Moore; *Journal of Statistical Mechanics*, P10012 (2005)

Sampling Grid Colorings with Fewer Colors; 4th author; with Dimitris Achlioptas, Mike Molloy, and Cristopher Moore; in *LATIN 2004: Theoretical Informatics*, Springer Lecture Notes in Computer Science 2976, pp. 80–89 (2004)

0-Centred and 0-Ubiquitously Graceful Trees; sole author; *Discrete Mathematics*, **277** 193–218 (2004)

Relaxed Graceful Labellings of Trees; sole author; *Electronic Journal of Combinatorics*, **9** (2002)

Chapter 1

Introduction

1.1 Preliminaries

Networks are everywhere in nature and society; for more than a century they have been recognized in diverse contexts such as the nervous system (Meynert, 1884), food chains (Camerano, 1880), and kinship relations (Radcliffe-Brown, 1922). Despite the fact that networks and network effects have been noted in the above as well as genetics, criminology, epidemiology, and elsewhere, investigation into how network interaction affects both collective and individual dynamics has been somewhat piecemeal until the last two decades, when relatively inexpensive large scale computational resources have become available to researchers.

This new, computationally aided focus on dynamical interaction in networks has, however, brought up new computational issues. In particular, overall dynamics can often depend explicitly and sensitively on the presence or absence of an individual connection [77, 112, 127]. This affects not only outcomes for particular instances, but our ability to calculate these outcomes as well. The introduction of a presence/absence distinction as an important factor may mean that exact computation requires exhaustive enumeration of discrete objects, discrete optimization subject to constraints, or some similar operation, which puts us in the realm of **NP**-complete problems [49].

Graph theory is the branch of mathematics that studies abstract and, one could say, static networks. It has been around in its own right for more than a century, and has made useful contributions to the natural sciences [52], though most graph theoretical applications have been to fields such as computer science, algorithmics, operations research, and industrial engineering. Results from graph theory have been instrumental in the understanding of one class of networks supporting an interaction of sorts that were already being studied intently before the last two decades: human-made communication, traffic, and flow networks. Canonical algorithms for determining the shortest path between two sites (Dijkstra's algorithm), the minimum cost spanning tree (Kruskal's algorithm), and the minimum cut-set e.g. bottleneck in a flow network (the Ford-Fulkerson algorithm) date from the 1950's and 1960's [31].

Graph theorists have been grappling with **NP**-completeness and related issues in computational complexity theory for as long as anyone; the first inklings that some standard combinatorial optimization problems are fundamentally harder than others came from graph theorists such as Jack Edmonds during the 1960's [36]. In the intervening time they have put together an extensive toolkit for tackling problems at the edge of tractability; this includes optimized brute force (so called "constant shaving"), various heuristics, polynomial time approximation schemes, randomized algorithms, and perturbation/restriction of problem instances.

In the last two decades while we have seen increasing interest in natural and social networks, we have also seen an corresponding increase in interdisciplinary work involving graph theorists and computer scientists on one hand and physicists, biologists, sociologists etc. on the other. This has been very much a two-way street: while the general scientific community has a strong interest in the development of faster and better computational techniques, graph theory and computer science for their part have benefited greatly from such things as the introduction of small world networks [124] and power-law graphs [5, 116], as well as e.g. the discovery of phase-transitions in the difficulty of hard computational problems [46].

1.1.1 Summary of Topics

In this cumulative thesis we look at results concerning two separate topics.

The first topic belongs to the general problem of reconstruction of interaction networks using only per-site information. Section 1.2 in the introduction briefly surveys the field up to this date. Chapter 2 consists of the article "Inferring Synaptic Connectivity from Spatio-Temporal Spike Patterns". In this article a method for exact reconstruction of leaky integrate-and-fire networks from spike time data is presented. This method is capable of reconstructing networks of several hundred neurons in a relatively short time; it is not only a significant advance on the current state of the art with respect to model neuron networks, it stands as a proof in principle that the problem of exact reconstruction of pulse-coupled networks is not inherently intractable due to discrete interactions. In section 5.1 of the conclusion we present several ideas for extending this theoretical result to deal with more realistic neuronal networks.

The second topic is the relation between Potts/Ising models from statistical physics and the chromatic polynomial from graph theory. A general outline of the problem and various mathematical issues related to it is given in section 1.3 of the introduction. Two of the articles presented in this thesis deal with this topic. Chapter 3, "Counting Complex Disordered States by Efficient Pattern Matching: Chromatic Polynomials and Potts Partition Function", presents a new method for computing the chromatic polynomial of a given graph based on algebraic operators that can be implemented with simple match-and-replace rules in any symbolic math system. On strip lattices the running time of the new method is competitive with previous specialized lattice-based methods from statistical physics; however, it is also capable of working on arbitrary graphs without modification, so vastly expands the structural range of graphs that can be accessed. Among the results presented in this article is the chromatic polynomial of the $4 \times 4 \times 4$ simple cubic lattice (free boundary conditions), the first time this has ever been successfully computed; previous research had been restricted to relatively unphysical 2-dimensional systems by feasibility issues. In chapter 4, "Chromatic Polynomials of Random Graphs", the greater flexibility of the new method is exploited to do extensive calculations of chromatic polynomials on random graphs with between 12 and 30 vertices across the entire range of edge-densities. Our finding is that the complex root sets of the chromatic polynomials of random graphs fall into stereotypical locations depending on size and density; in particular, when the average degree is fixed the point at which the complex root set will meet the real line is very predictable, and independent of the total number of vertices. Several implications of this for statistical physics are discussed in the conclusion of the article.

While these two topics are scientifically quite independent of each other, they both arise from ongoing research projects of the MPIDS Networks Dynamics Group. Application of various graph theoretical and combinatorial optimization techniques were an essential part of the work

I have done on these projects; however, I am mindful of the fact that the scientific aims are primary. Hence the emphasis in the articles and introductory material is on "Network Dynamical Systems" rather than "Combinatorial Optimization"; discussion of implementation details and technical computational issues are reserved for the appendix. No specialist knowledge of computer science or graph theory should be required to understand the material here and in the articles. Where concepts or terminology from these fields come up they will be accompanied with brief definitions, explanations, or external references as needed.

1.2 Network Reconstruction

Relevant Paper:

1. Frank Van Bussel, Birgit Kriener, and Marc Timme, Inferring Synaptic Connectivity from Spatio-Temporal Spike Patterns, under review at *Frontiers in Computational Neuroscience*

Inverse problems lie at the heart of science: we know outcomes, and we want to know what lead to these outcomes. While natural scientists were always aware of the complex network interactions that underlie much of the phenomena they studied, it is only in the last couple of decades, with access to inexpensive large-scale digital computation, that they could study such interactions on a realistic scale. The problem of recovering networks based on their observed dynamics has therefore attracted growing attention [2, 24, 133, 134].

Perhaps because predicting dynamics of complex networks based on known connectivity is itself usually very difficult, explicit inverse methods that give the exact strength of pair-wise connections have not been the focus of most of this attention; instead, it has been directed to methods based on correlations in activity between different areas of the network. Much work has been devoted to establishing *functional* and/or *effective* connectivity, particularly in neuroscience, but more recently in other areas of biology such as genetics and ecology [27, 60].

1.2.1 Functional connectivity

Within neuroscience the study of functional connectivity comes directly out of the long tradition of research in neurophysiology devoted to modular specialization of function in specific areas of the brain. A variety of techniques and tools are available, but for the most part they involve application of cross-correlations and other covariance based statistics to PET, EEG, fMRI, and other data [38, 50, 70]. The advantages are obvious: it ties into a large existing body of knowledge on area specialization; no strong assumptions about the nature of the interactions or the underlying neuron model need be made; and it can work with data that is noisy, large grained, or imprecise. On the other hand, it has the usual problems of broadly targeted statistical techniques, such as determining direction of connectivity, or making distinction between direct and indirect connectivity. As well, almost by definition it yields results that are on a much different scale than the neurons that make up the network.

This last point, I think, is important. While large-scale statistical methods will remain useful for the foreseeable future in establishing connectivity between regions of the brain, ultimately we also want to isolate small-scale functional subassemblages such as timers, gates, tags, and masks (discrete filters). These are what graph theorists call "gadgets", small subgraphs with

a consistent topology that encode some generic information or purpose. A plausible approach to doing this is to scan connectivity tables of slightly larger-scaled regions of interest for these subassemblages; which of course requires accurate connectivity tables in the first place. Work in this vein has already begun with respect to establishing relative frequency of various small graph motifs in real neuronal networks [108]. So far the networks involved have been limited in size (roughly 100 neurons or less) and acquired only with extensive experimental effort.

There has been some work using more sophisticated statistical techniques (PCA, Grainger causality) to determine effective connectivity of neurons on an individual basis from simulation data, but it has had mixed success. Statistical methods pursued on the micro scale have the same limitations they do on the macro scale; as well, such methods tend to require relatively large amounts of data, which must furthermore fulfill not entirely realistic conditions (e.g. constant statistical properties, no mutual couplings, entrainment cannot occur) [70].

A note on effective connectivity. It seems that many researchers regard the terms "effective connectivity" and "functional connectivity" as roughly interchangeable, while others attempt to draw a subtle but real distinction between them; the nature of this distinction is a matter of debate among (some) neuroscientists [38, 50]. Here I only want to register that I am aware that effective connectivity may be a separate issue that will have to be addressed in future papers; for obvious reasons I would prefer not to wade into any open controversies surrounding the issue at the present time.

1.2.2 Explicit reconstruction methods for oscillator networks

While eventual application to neuronal networks has been the motivation for much reconstruction research, the general network inverse problem is of independent mathematical interest, and has possible applications in diverse fields such as chemistry, sociology, and genetics [41, 78]. Hence much of the work on explicit reconstruction and similar problems has focused on continuously interacting oscillator networks e.g. Kuramoto, Lorenz, or Rössler oscillators, arising out of the general study of dynamics of oscillator networks [113, 133, 134].

We do not have space to do more than touch upon the large body of proto-reconstructive work that has been done. Noteworthy for our purposes are studies such as those by Zanette [135], and Kori and Mikhailov [62], which look at propagation of perturbations through the network emanating from a single node. In such systems it is possible to directly relate the degree and timing of the perturbation experienced by nodes to their topological distance from the source. Following from these results Radicchi and Ortmann [90] considered how quickly parts of a network synchronized to pacemaker cells. Here as well distance between nodes was a major influence, though other aspects of the network topology such as the presence or absence of closed cycles had an effect. Timme [112] showed that, independent of distance metrics, the way a network is partitioned by its strongly connected components has a determinative effect on its ability to synchronize after random perturbation, and that in reverse one can measure the network's ability to synchronize to learn about the its connectivity.

While it does not address topological issues, the autosynchronization technique of Parlitz [86] should be mentioned here since it stands as a precursor to connectivity reconstruction methodology that has followed. This technique is able to recover multiple parameter values simultaneously, up to complete systems of ODE's, by synchronizing a model system to the dynamics of the original system as given by time series data.

The method of Yu, Righero, and Kocarev [134], for example, follows directly from Parlitz's approach: network topology is recovered by synchronization of a model network to the original network's dynamics by a process of continuous error minimization. A later approach given by Yu and Parlitz [133] dispenses with the entire comparison network i.e. the second connectivity matrix, though it does feature a set of comparison values for the dynamics and output functions of the individual nodes in the network. In this case, an external driving that is proportional to the difference between the node's current phase value and its comparison value is applied, and the network is brought into a steady state by adjustment of the comparison values. Once the network achieves its steady state the connectivity can then be recovered from the comparison values and the total driving strength.

All the full and partial reconstruction approaches mentioned above depend on networks achieving synchronization or stable dynamics; so far the only method (that I know of) to reconstruct oscillator networks which does not is that of Łęski and Wójcik [67]. Their method features a time consuming statistical comparison of the transient portion of time series, so it can only reconstruct one connection at a time. For comparison, the Yu, Righero, and Kocarev method has been used to reconstruct networks of 16 nodes, and the Yu and Parlitz method has been used for networks with at least 50 nodes [133, 134].

1.2.3 The response dynamics framework

We will take a closer look at the reconstruction approach for phase-coupled networks developed by Timme [113], since it has a direct relevance on my subsequent work on pulse-coupled networks. The basic idea is that measurements of stable response dynamics under different external driving conditions are used to construct linear systems that can be solved to obtain the incoming connections for each node. We start with a network of N oscillators with dynamics described by the differential equations

$$\dot{\phi}_i = \omega_i + \sum_{j=1}^{N} J_{ij}\, f_{ij}(\phi_j - \phi_i) + I_i \tag{1.1}$$

where $\phi_i(t)$ is the measured phase of oscillator i at time t, ω_i is its natural frequency, J_{ij} is the coupling strength, f_{ij} is the coupling function, and I_i is an external driving force. If such a system is able to synchronize when undriven ($I_i = 0$) it will phase lock under a small enough perturbation. If the systems is perturbed M times by using different drivings $I_{i,m}$, $m = 1, 2, ..., M \leq N$, we obtain

$$D_{i,m} = \Omega_m - \Omega_0 - I_{i,m} = \sum_{j=1}^{N} J_{ij}\,[f_{ij}(\phi_{j,m} - \phi_{i,m}) - f_{ij}(\phi_{j,0} - \phi_{i,0})] \tag{1.2}$$

where Ω_m is the system's collective frequency under driving m and Ω_0 is the undriven collective frequency. Via linearization we get the following approximation:

$$D_{i,m} = \sum_{j=1}^{N} \widehat{J}_{ij}(\phi_{j,m} - \phi_{j,0}) \quad \text{where} \quad \widehat{J}_{ij} = \begin{cases} J_{ij}\, f'_{ij}(\phi_{j,0} - \phi_{i,0}) & \text{for } i \neq j \\ -\sum_{k,k\neq j} J_{ik}\, f'_{ik}(\phi_{j,0} - \phi_{i,0}) & \text{for } i = j\,. \end{cases} \tag{1.3}$$

If we let $\theta_{j,m} = \phi_{j,m} - \phi_{j,0}$ this gives us the linear system $D = \widehat{J}\theta$, which can be directly solved to obtain the off-diagonal elements of the connectivity matrix J (the diagonal elements are assumed to be zero to begin with).

This method gives both the topology and the connection strengths with high degree of accuracy. It has been used with success on Kuramoto oscillator networks ($f_{ij} = \sin$) [113], and preliminarly results indicate that it is applicable to higher dimensional systems. It is important to note that the response dynamics (multiple driving) framework is detachable from the linearization aspect of the solution. Here linearization is necessary to approximate the system, so the dynamics need to be stable; networks with entirely different interactions (e.g. discrete, repulsive, non-oscillatory) displaying irregular or chaotic dynamics can in principle also be reconstructed using response dynamics, as long as some way exists of solving or estimating the ODE's.

Another feature of the method is that it is amenable to an optimization that can reduce the number of different drivings M required (and hence experimental or simulation runs) to significantly less than N. This optimization, previously presented by Collins and various collaborators in [41, 132], makes use of singular value decomposition plus an L_1-norm minimizing heuristic based on the naturalistic assumption that the network in question is sparse. This same optimization is used as well in chapter 2, where it is described in some detail; since there are aspects of its application that I am continuing to work on, further discussion of this issue is left to section 5.1.1 below.

1.2.4 Explicit reconstruction methods for pulse-coupled networks

A voluminous literature exists on the dynamics of neuronal and general pulse-coupled networks. It should also be noted that while biological neuron models are the most common subject or proposed application of these studies, many other natural phenomena can also be considered as discrete-interaction networks [75], while abstract versions such as "chip-firing games" are as well studied by mathematicians and graph theorists [11, 16]. While there is often a topological aspect to this research, due to space and time constraints I will restrict myself here to previous/ contemporary research that has an actual reverse-engineering component.

As with reconstruction of continuous oscillator networks, there was some work done in the 1990's with regard to recovering model parameter information from the dynamics displayed by a single node. Noteworthy in this regard is the prediction/correction method used by Sauer [97] to reconstruct the attractor of single integrate-and-fire neuron from time series data. The premise is that, due to synchronizing influences, multiple interactions from unknown neighbours could be modeled as input from a single low-dimensional chaotic attractor; hence the demonstration of the method used spike trains generated by applying a threshold-with-spike to Lorenz and Rössler attractors.

Work in earnest on multiple neuron network reconstruction has begun only in the last decade. The most direct attack on the problem has been given by Makarov et al [70]; they provide a method that recovers not only topology and connection weights, but also the other model parameters for a generic integrate-and-fire neuron network. The basis of the method is a fit with continual error minimization of each individual neuron's phase values (i.e. potential) to oncoming spike data, until the parameters and connection weights are consistent with the spiking that is seen. The method has been tested on biological as well as simulation data with some success; however, the largest network they report as having reconstructed this way consists of only 5 neurons.

An approach coming from the opposite direction is given by Memmesheimer and Timme [73, 74]. They give mathematical criteria which any network must fulfill in order to generate

a specified precisely timed periodic spike pattern. In principle, a particular periodic spike pattern may be producible by infinitely many different neuronal networks, though there are also impossible patterns which cannot be generated by any (the methodology can detect both cases). The criteria can be then applied to design a network to generate the pattern in question.

Even relatively simple model neurons can display behavioural quirks that have no analogue in continuous interaction networks (these quirks are of course built into the models to mimic the behaviour of real neurons). Any eventual practical reconstruction technique will have to either handle all such eventualities, or consist of a repertoire of specialized subroutines for different spiking regimes. The beginnings of an approach to neurons displaying bursting behaviour has been made by Shen, Hou, and Xin [101, 102]; after extensively studying the effect of adding or strengthening network couplings on burstiness, they show that the degree distribution of the network can be obtained via a mean-field method.

1.2.5 Leaky integrate-and-fire neuron model

Before we move on to the second half of this introduction I would like to briefly discuss the leaky integrate-and-fire neuron model, and in particular the mathematically equivalent but notationally different versions used during the reconstruction project. I hope this removes any potential confusion that might be caused by close comparison of chapter 2 and e.g. the reconstruction code provided in appendix A.2.

The version of the leaky integrate-and-fire neuron that I myself used for the initial planning phase of the reconstruction project was:

$$C_i \frac{dV_i}{dt} = I_i - \frac{1}{R_i} V_i + S_i(t) \tag{1.4}$$

where C_i, I_i, R_i, and V_i are the capacitance, external current, membrane resistance, and membrane potential respectively, in their standard units. S_i is the interaction term that for discrete pulse interactions would be $\pm\infty$ at the instant that a spike arrived, and be zero otherwise. This model is based on the one given in Koch and Segev's *Methods in Neuronal Modeling* [40].

Since we did not expect be fiddling with the value of the capacitance while fixing the other parameters, for simplicity the following "working model" was used throughout the development phase (i.e. coding of the simulation program, derivation of the solution, and coding of the reconstruction routines):

$$\frac{dV_i}{dt} = \widetilde{I}_i - \gamma V_i + S_i(t) \tag{1.5}$$

where $\widetilde{I}_i = I_i/C_i$ in volts per second, and $\gamma = 1/(C_i R_i)$ is the "leakage" in units of 1/seconds. Since the interaction is instantaneous the capacitance cannot affect it; as well, whatever the interaction contributes when the RHS is integrated is immediately deposited into the value of V_i, where the capacitance works through γ. It should be noted that large scale testing of the reconstruction programs was implemented through a wrapper that took as parameters those used in the (1.4) model, and then converted to the (1.5) working values.

Admittedly, in reports, program code, and other documents that were generated throughout the project a common form of notational abuse was committed, where I_i or I was written without any indication that this was not in fact an electrical current in the ordinary sense

(i.e. we did not bother with the tilde). To my knowledge this never once caused any mistakes to be made in any analysis, coding, or testing. However, when it came time to write the results up for publication we realized we would need to use a notation that was unambiguous and not vulnerable to misunderstanding due to typographical omissions. After it was decided to submit to *Frontiers in Computational Neuroscience* we adopted a version of the LIF model that was more in line with the usage in the neuroscience literature:

$$\frac{dV_i(t)}{dt} = \gamma \left(R_i I_i - V_i(t) \right) + S(t) \,, \tag{1.6}$$

where $1/\gamma$ is the *membrane time constant* (γ still having the same units as in (1.5)), and R_i and I_i have their familiar meanings as in (1.4).

1.3 Chromatic Polynomials

Relevant Papers:

1. Marc Timme, Frank Van Bussel, Denny Fliegner, and Sebastian Stolzenberg, Counting Complex Disordered States by Efficient Pattern Matching: Chromatic Polynomials and Potts Partition Functions, *New Journal of Physics* **11** 023001 (2009)

2. Frank Van Bussel, Christoph Ehrlich, Denny Fliegner, Sebastian Stolzenberg, and Marc Timme, Chromatic Polynomials of Random Graphs *Journal of Physics A: Mathematical and Theoretical* **43** 175002 (2010)

The history of the chromatic polynomial gives a nice example of the convergence of entirely separate streams of inquiry that can sometimes happen in science and mathematics. In 1912 the American mathematician G. Birkhoff proposed the chromatic polynomial as a way of using complex analysis to prove the then 4-color conjecture in graph theory [14]. Mathematical work in this vein resulted in much of worth, though in the end the 4-color theorem itself was proven by more straightforwardly combinatorial means [8, 117, 128]. In recent decades, however, the chromatic polynomial has become the focus of attention from statistical physicists due to its connection to the Potts model of interacting spins on a lattice [29, 103, 104, 105]; the nature of this connection will be outlined below. Chapter 4 of this thesis contributes further to the convergences: while our study of the chromatic polynomials of random graphs is at base motivated by statistical physics concerns, it also ties into the study of random graphs pioneered by the Hungarian mathematician P. Erdös in the 1950's.

Before we proceed, we will need some definitions. Given a graph G with vertex set V and edge set E, where $N = |V|$ and $M = |E|$, a *proper coloring* of G is defined as an assignment of values (colors) to the vertices of G such that no two vertices connected by an edge share the same value. G is *q-colorable* if there is a proper coloring of G using q or fewer colors; the *chromatic number* $\chi(G)$ is then the minimum q such that G is q-colorable. The *chromatic polynomial* $P(G, q)$ is the associated counting function for proper q-colorings of G, that is, it tells us how many ways we can color G with q colors. Hence for integer q such that $0 \leq q < \chi(G)$ we have $P(G, q) = 0$.

There are many different formulas for the chromatic polynomial, though the simplest is probably:

$$P(G,q) = \sum_{k=0}^{N} a_k q^k = \sum_{E' \subseteq E} (-1)^{|E'|} q^{\kappa(E')} \tag{1.7}$$

where $\kappa(E')$ is the number of connected components of the graph induced by the edge subset E'. This formula, a straightforward application of the combinatorial inclusion-exclusion principle, is not of much use for computational purposes, but it does tell us that $P(G,q)$ is a polynomial of order N, that the signs of the coefficients alternate, and that $a_N = 1$, $a_{N-1} = -M$, and $a_0 = 0$. A more elegant expression is based on the *Deletion-Contraction* recurrence: for any edge $e \in E$,

$$P(G,q) = P(G-e,q) - P(G \cdot e,q) \tag{1.8}$$

where $G - e$ is G with the edge e removed, and $G \cdot e$ is G with the endpoints of e contracted to a single vertex. This simple formula is the basis for almost all general purpose programs for calculating chromatic polynomials [83, 91]. Similarities between this recurrence and recurrences for various other graph invariants where noted by the English/Canadian mathematician W. Tutte in the 1940's; these prompted him to generalize the chromatic polynomial into the bivariate Tutte polynomial [117]:

$$T(G,x,y) = \sum_{E' \subseteq E} (x-1)^{\kappa(E')-\kappa(E)} (y-1)^{|E'|+\kappa(E')-N}. \tag{1.9}$$

Tutte himself early realized that this generalized polynomial had uses in statistical physics as well as graph theory, though the full extent of the connection was not clear until the work of Fortuin and Kasteleyn in the early 1970's [39].

1.3.1 The Potts model

The *standard q-state Potts model* was introduced in 1952 as a generalization of the 2-state Ising model for interactions on a crystal lattice [88, 131]. It describes systems in which sites can take one of q different values, and the interactions occur only between neighbouring sites on a lattice that are in the same state. The total energy in the global state $\boldsymbol{\sigma} = \{\sigma_1, \ldots, \sigma_N\}$ is given by the Hamiltonian

$$H(\boldsymbol{\sigma}) = -J \sum_{(i,j) \in E} \delta_{\sigma_i \sigma_j} \tag{1.10}$$

where J is the interaction strength, and E is the set of connections.

For the ferromagnetic ($J > 0$) Potts model, phase diagrams and critical exponents are now known for various infinite lattices [7, 104, 131]. The model in this case is amenable to mean field solutions as well as numerical methods such as Monte Carlo or renormalization group techniques. While the model was originally thought to be of theoretical interest only, by the 1980's several real systems were found to correspond closely enough to it to allow experimental confirmation of theory based predictions; these include the cubic ferromagnet $DyAl_2$ ($q = 3$) and O_2 adsorbed on nickel ($q = 4$) [131]. The antiferromagnetic ($J < 0$) case is not so well understood. It is known that for large enough q the system is disordered: it has positive ground state entropy, with $T = 0$ belonging to the high temperature regime. As well, for countably infinite G with maximum degree k, when $q > 2k$ there are no first order phase transitions [96, 104].

The partition function for this system at positive temperature $T = (k_B\beta)^{-1}$ is

$$\begin{aligned} Z(G,q,T) &= \sum_{\boldsymbol{\sigma}} e^{-\beta H(\boldsymbol{\sigma})} \\ &= \sum_{\boldsymbol{\sigma}} \prod_{(i,j)\in E} (1 + (e^{\beta J} - 1)\delta_{\sigma_i\sigma_j}) \end{aligned} \tag{1.11}$$

where k_B = Boltzmann constant and G is the lattice or structure corresponding to the edge-set E. The relation between the Potts model and the Tutte polynomial comes out of the second of these representations [39, 126]; if we set $y = e^{\beta J}$ then evaluating the Tutte polynomial along the hyperbola $(x-1)(y-1) = q$ gives us the Potts partition function for that value of q. The partition function for ferromagnetic Potts model lies on the branch of the hyperbola with both x and $y > 1$, with temperature increasing from 0 to ∞ as x is increased; the antiferromagnetic Potts partition function is in the part of the other branch where $y > 0$, with the remainder of the branch not corresponding to anything physical.

While the Tutte polynomial generalizes both the Potts partition function and the chromatic polynomial, the above does not give their relation directly. The chromatic polynomial exists along the line $y = 0$, meeting the edge of every antiferromagnetic hyperbola as x decreases from 1 to $-\infty$. When $J < 0$ we have $e^{\beta J} \to 0$ as $T \to 0$, so in the zero-temperature limit the partition function for the antiferromagnetic case simplifies to:

$$\lim_{T\to 0} Z(G,q,T) = \sum_{\boldsymbol{\sigma}} \prod_{(i,j)\in E} (1 - \delta_{\sigma_i\sigma_j}). \tag{1.12}$$

The product here will equal 1 if all pairs of adjacent nodes are in different states, and 0 otherwise; it functions as an indicator that the assignment of states given by $\boldsymbol{\sigma}$ is equivalent to a proper coloring of G. Since $Z(G,q,T)$ just sums over these indicators it is itself then equivalent to the chromatic polynomial $P(G,q)$.

For this reason the statistical physics community has taken a serious interest in chromatic polynomials, with the location of their roots in the complex plane being a particular focus. In the 1950's Lee and Yang [66] showed that for some ferromagnetic systems it was possible to bound critical values, if all zeros of the partition functions for such systems had a non-zero imaginary part. On the other hand, if the complex root set approaches arbitrarily close to the real line as the system size N is increased, then in the $N \to \infty$ limit it is possible for the (real-valued) partition function to go to zero, leading to singularities in the system which might indicate phase transitions or critical points. This second situation has come up repeatedly in the study of antiferromagnetic systems; since no general *Lee-Yang*-type theorem exists here, for the zero-temperature limit of the antiferromagnetic Potts model results have been established on a lattice by lattice basis [96, 104]. This has led some authors to refer to the principles involved in the plural, as *Lee-Yang theorems* [104].

1.3.2 Chromatic roots

Most statistical physics research on the chromatic polynomial has been done on thin sections of 2-dimensional lattices [28, 92, 96, 103], since their repetitive layered structure lend themselves to specialized computational and analytic techniques. In terms of the system size N these lattice strips are among the largest graphs for which chromatic polynomials have so far been computed. For thin-enough strips it is sometimes possible to calculate the $N \to \infty$ limit

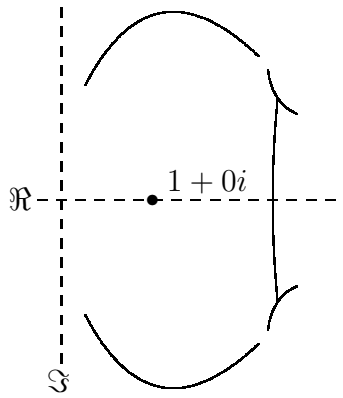


Figure 1.1: General form of the limiting curve for the chromatic root set of a lattice strip. An example of such a limiting curve for a specific lattice ($4 \times n$, square, FBC) is shown in figure 1 of chapter 4

of the chromatic root set to obtain a continuous curve, which allows one to invoke the Lee-Yang theorems mentioned in the previous subsection. The solutions obtained so far have been "highly lattice-dependent" compared to what is found for ferromagnets [96]; even so, there are some common features [28, 92, 96, 103, 105]. Typically root sets trace out a somewhat elongated backwards "C" curling around the point $1 + 0i$. Gaps and small horns often show up in characteristic locations (see figure 1.1). For 2-dimensional lattices the imaginary values fall between $2.5i$ and $-2.5i$; these bounds increase to around $\pm 4i$ for 3-dimensional lattices. Similarly, real values for most 2-dimensional lattices are less than 3, but can reach higher for 3-dimensional lattices and 2-dimensional triangular lattices. Negative real values are not unknown but not often encountered, and so far the magnitudes involved have been very small.

To give these results some context we should take a look at what is known in general about the root sets of chromatic polynomials. To start, explicit formulas for the chromatic polynomial in terms of its coefficients are known for many simple graph families such as cycles, "wheels", "gears", "helms", "barbells", "books" and the like, as well as some individual graphs of perennial interest such as the Petersen graph [125]. The roots of the cycle C_N are equally spaced around a unit circle centered at $1 + 0i$ and thus represent the limiting case of a non-trivial complex chromatic root set; many of the other families on the above list have chromatic polynomials with many repeated small factors, so they only have a small, usually fixed, number of distinct roots regardless of the number of vertices involved. Chordal graphs (nonplanar triangulations, a larger class that includes trees and complete graphs) have chromatic polynomials with only integer roots, as do certain almost but not quite chordal graphs [34].

More generally, a graph G with N vertices will have no real chromatic roots less than 0 or greater than N. As well, no graph exists with a real chromatic root in either of the intervals $0 < z < 1$ or $1 < z \leq \frac{32}{27}$ [56]. An important result due to Sokal [105] is that over the set of all graphs chromatic roots are dense in the complex plane. If the set is restricted to all planar graphs this result still holds, with the possible exception of the unit disk centered at $1+0i$ (chromatic roots for some planar graphs are known to fall inside this disk but the density question is unresolved).

Loose upper and lower bounds on chromatic root location do exist for parameterized graphs. For a graph with maximum degree Δ and second highest degree Δ_2, the modulus $|z|$ of any

chromatic root z satisfies $|z| \leq 7.963907\Delta$ and $|z| \leq 7.963907\Delta_2 + 1$, with no similar bound possible for third highest degree; this is again due to Sokal [104]. No graph is actually expected to have a chromatic root this distance from the origin, with the true bound conjectured to be closer to 1.6Δ, but this is so far unproven [94]. An even looser but less structure-specific bound is given by Brown [21], who shows that all non-zero chromatic roots of a graph with N vertices and M edges are in the disk $|z - 1| \leq M - N + 1$; he also gives a lower bounds of sort, proving that at least one of these roots must have modulus $|z| \geq \frac{M-1}{N-1}$. The existence of the anomalous non-chordal graphs mentioned above removes the possibility of setting any unqualified non-zero lower bound on the imaginary values. Brown does show, however, that for every N there does exist a graph on N vertices with at least one root z having imaginary value $Im(z) > \frac{\sqrt{N}}{4}$.

1.3.3 Computational complexity

For readers who are unfamiliar with the basic outline of computational complexity theory a brief review is given in appendix C.5.1.

Graph coloring, the problem of determining whether a given graph G has at least one q coloring for arbitrary q, is one of the original **NP**-complete problems. Hence determining the chromatic number is as well **NP**-complete. Since calculating the value of the chromatic polynomial for a given G and q is a counting problem rather than a decision problem it does not belong to the computational complexity class **NP**; determining the chromatic polynomial of a graph is instead **#P**-*complete*, where the class **#P** is the counting problem analogue to **NP**. **#P**-complete problems are as least as hard as any **NP**-complete problem and presumed to be some orders of magnitude harder; even in the case that **P**=**NP** and graph coloring is actually easy, the chromatic polynomial and other **#P**-complete problems would likely stay intractable [31, 85].

Until recently the best general case algorithms for the chromatic polynomial have been based on the deletion-contraction formula [15, 91]. Since this works by recursively removing edges from the graph until there are none left, a naive implementation on a graph with N vertices and M edges would require $2^M \sim 2^{N^2}$ recursive calls. This can be reduced by sophisticated pruning or identification of computational branches, plus a judicious ordering of edges, but pursuing such strategies involves as well their own computational demands, and improved versions do not achieve running times better than exponential in the number of edges anyhow. For certain graph classes there are polynomial time algorithms; these include the chordal graphs mentioned above, P_4-free graphs, and graphs of bounded treewidth [9].

Lattice strips belong to the last of these classes, hence their popularity among researchers despite the limited physical applications of long, very thin systems. It should be noted that the various specialized algorithms, while often linear in the length of the graph, have embedded constants that grow exponentially with the width, so the range of feasible values really is limited. The fastest but most lattice-specific algorithms are those that depend not only on bounded treewidth but also a layered repeating structure; among these are Sokal's transfer matrix approach and Shrock's method of generating functions [28, 96]. These have been used not only to calculate chromatic roots for very long strips but also the limiting curves of the zero set as $N \to \infty$. Dependence on repeating structure, of course, precludes using the algorithms on arbitrary graphs or even arbitrary low treewidth graphs. For such graphs deletion-contraction, as implemented in MATHEMATICA or MAPLE, is still the standard approach [123, 130].

1.3.4 Random graphs

Because of the pioneering work on random graph theory done by Erdös and Rényi in the late 1950's, the standard random graph is commonly denoted as an Erdös-Rényi (ER) random graph [59]. The theory has a close relation to the probabilistic method in combinatorics, also developed by Erdös [3]. Random graphs are everywhere in research, since an instance of a random graph can stand as an initial approximation to an unknown arbitrary system. Hence they are often used for testing algorithms, analytic methods, and theories. As one would expect, there is a degree of overlap between random graph theory and statistical physics; for example, *percolation theory* deals with connectedness in relatively sparse random graphs [99, 100].

The two models of the ER random graph are:

$G(N,p)$ the graph has N vertices, between every pair of vertices an edge exists with uniform probability p ;

$G(N,M)$ the graph is chosen uniformly from all graphs with N vertices and M edges.

In the $N \to \infty$ limit the two models are largely equivalent, though when N is finite (and in particular, when N is relatively small) they can give slightly different results. Because any part of a $G(N,p)$ graph can be considered independently of any other part this model lends itself to pencil-and-paper estimates and situations where global constraints are of no importance. On the other hand, when the global edge-density of the graph is to be precisely specified one must use the $G(N,M)$ model. This, for example, is the case in the *random graph process*, a way of modeling stochastic processes for which connectivity increases with time.

Large random graphs can possess surprisingly definite properties and precise invariants [59]. For example, for $\epsilon > 0$ and large enough N, we have that

$$\begin{aligned} &\text{if } \; p < \frac{(1-\epsilon)\log(N)}{N} \;\; \text{then } G(N,p) \text{ will almost surely have isolated vertices, and} \\ &\text{if } \; p > \frac{(1+\epsilon)\log(N)}{N} \;\; \text{then } G(N,p) \text{ will almost surely be fully connected.} \end{aligned} \tag{1.13}$$

(this is an extension of Erdös' famous Giant Component result [58]). The situation is similar for many other features, such as clique size, diameter, or whether the graph has a Hamiltonian cycle [3, 59]. Of particular relevance to this paper, since the chromatic number to some extent puts a bound on where chromatic roots can appear, is the result due to Bollobás [19] that is highlighted in the chapter 4: the chromatic number of a $G(N,p)$ random graph is almost always

$$\left(\frac{1}{2} + o(1)\right) \log\left(\frac{1}{1-p}\right) \frac{N}{\log N}\,. \tag{1.14}$$

Here $o(1)$ contains decreasing lower order terms as well as corrections for rounding etc.

Such *sharp thresholding* results have increased the profile of random graph theory, but they have also made clear that ER random graphs might not always be appropriate for modeling systems where little is known as opposed to absolutely nothing. Recent decades have therefore seen various alternatives brought forward, such as small-worlds and scale free networks [69, 79, 81, 124]. These more accurately reflect what we now know about networks in social, economic, biological, transportation and many other contexts. They also, however, involve new complications at every stage of research, starting with the determination of what kind of network we are dealing with in the first place.

1.3.5 Collecting the threads

As mentioned at the beginning of this section, our chromatic polynomial research takes up the various lines of inquiry outlined in the previous subsections: statistical physics, algorithmics, and random graph theory. Here I will briefly outline how they lead into the material of chapters 3 and 4.

To begin with, as with much recent work on chromatic polynomials [92, 96] the motivation and the basic methodology comes from statistical physics. This begins with the formula that underlies the new algorithm presented in chapter 3, which is based on the antiferromagnetic Potts partition function as given in (1.12) in section 1.3.1. Furthermore, while we do maintain an interest in some questions of a graph theoretical nature, our main focus is on the behaviour of entire chromatic root sets for physically relevant graphs, since in infinite size limit these give indication of possible phase transitions and other critical phenomena (cf sections 1.3.1 and 1.3.2). Note that the current state of art/knowledge in this area is still formative, and does not yet support any general scientific conjectures such as "Do all antiferromagnetic X's do Y at the zero temperature limit?"

The main reason for this is twofold. First, as given in section 1.3.3, calculating chromatic polynomials is extremely difficult, enough so to make tackling graphs as small as 30 or 40 vertices a daunting proposition. Second, the behaviour of chromatic root sets seen so far in both the physics and graph theoretic contexts is, as mentioned in section 1.3.2, not consistent enough for us to even guess as to what the general case is. Up to the publication of chapter 3 the largest graphs chromatic polynomials had been calculated for were all thin 2-dimensional lattice strips [28, 96], so nobody was in a position to say how representative the results would be of physically realistic 3-dimensional systems.

Our approach to the problem has likewise been twofold. In chapter 3 we attack it by main force, so to speak, utilizing the new algorithm implemented with additional optimizations in the computer algebra system FORM [121] to calculate chromatic polynomials for the $4 \times 4 \times 4$ simple cubic lattice and various other large 3-dimensional lattices (see appendix B.1 for details about the optimizations). In chapter 4 we come at the problem in a more extensive way, by doing chromatic polynomial calculations for more than a thousand random graphs. While the individual graphs involved are smaller than those featured in chapter 3, they span the entire range of connection densities and are structurally arbitrary; up to this time the fastest algorithms for the problem were only equipped to handle very sparse and highly structured graphs. The results obtained dovetail with those of random graph theory given in section 1.3.4, with the chromatic root sets falling predictably into locations determined by the size of the graph and the connection density.

Chapter 2

Article: *Inferring Synaptic Connectivity from Spatio-Temporal Spike Patterns*

Frank Van Bussel, Birgit Kriener, and Marc Timme
Under review at: *Frontiers in Computational Neuroscience*
Submitted: May 14, 2010

Inferring Synaptic Connectivity from Spatio-Temporal Spike Patterns

Frank Van Bussel*[1,2,3], Birgit Kriener[1,2], Marc Timme[1,2,3]

[1] *Network Dynamics Group, Max Planck Institute for Dynamics and Self-Organization, Göttingen*

[2] *Bernstein Center for Computational Neuroscience Göttingen*

[3] *Fakultät für Physik, Georg-August-Universität Göttingen, Germany*

* **Correspondence:**
Network Dynamics Group, MPIDS,
Bunsenstrasse 10, 37073 Göttingen, Germany
fvb@nld.ds.mpg.de

Abstract

Networks of well-known constituents but unknown interaction topology arise across various fields of biology, including genetics, ecology and neuroscience. The collective dynamics of such networks is often sensitive to the presence (or absence) of individual interactions, but there is usually no direct way to probe for their existence. Here we present an explicit method for reconstructing interaction networks of leaky integrate-and-fire neurons from the spike patterns they exhibit. The approach works well for networks in simple collective states, e.g. close to complete synchrony, but is also applicable to networks exhibiting complex spatio-temporal spike patterns. In particular, stationarity of spiking time series is not required.

Keywords: networks, inverse methods, leaky integrate-and-fire neuron, irregular spiking, chaotic spiking, synchronization

Submission for: *Frontiers in Computational Neuroscience*

1. Introduction

Reconstructing or reverse-engineering the exact interaction topology of coupled dynamical units has become an active area of research in different fields of biology, including genetics, ecology and neuroscience (see Yeung et al., 2002; Makarov et al., 2005; Cadenasso et al., 2006, for recent theoretical results). In neuroscience, such efforts generally focus on establishing the *functional* or *effective* connectivity between individual neurons or different regions of the brain using methods based on correlations in the activity between the constituent parts of the network (Aertsen & Gerstein, 1985; Aertsen et al., 1989). This research has yielded many valuable results (Jirsa & McIntosh, 2002), commonly on a scale larger than the individual elements such as nerve cells and synaptic connections that make up the networks involved.

Explicit approaches that reconstruct actual connections between individual elements are more difficult in general, but where feasible they offer exact, fine-grained results, for instance by establishing the presence or absence of individual synapses. These more detailed approaches have attracted serious attention recently, triggered by the availability of large high-resolution data sets and the computational power to handle them. Noteworthy from a theoretical point of view is a method of stochastic optimization (Makarov et al., 2005) that fits a network model of spiking leaky integrate-and-fire (LIF) neurons to an oncoming data stream and thus avoids estimating parameters beforehand. The performance of this reconstruction method has been addressed for simulated, as well as real, extracelluar multiunit array (MUA) recordings. The computational requirements seem however somewhat prohibitive as the largest network which Makarov et al. report having reconstructed consists of $N = 5$ neurons.

Other studies have focused on networks of smoothly coupled units. One of the earlier results in this regard is a synchronization method presented by Yu et al. (2006). Assuming pre-knowledge of all model parameters, a second network of the same units is set up as a model, and its interaction topology is varied via continual error minimization until it synchronizes with the original system. Via a thresholding process, the resulting topology is identified as that of the original network. This approach has been shown to perform well for networks of up to $N = 16$ nodes. Yu & Parlitz (2008) combine the basic idea of error minimization with an intricate mechanism for driving the network to a steady state, at which point the connectivity can be recovered from the individual node dynamics. Since this technique dispenses with the need to explicitly model the connectivity of the network they are able to achieve reconstructions of significantly larger networks ($N = 50$).

An alternative approach for smoothly coupled systems close to steady states (Timme, 2007) uses measurements of the collective response dynamics under different external driving conditions. As the collective dynamics depends on both the (known) driving signals and the (unknown) interaction topology, an increasing number of driving-response experiments yields more and more restrictions onto the topology. Worked out explicitely for phase-locking oscillators, the method is insensitive to details of the units' dynamics and their coupling function and a linear approximation is often sufficient to infer the topology. Hence, the experiments yield a system of linear equations that restrict the topology consistent with all experiments that is then solved to obtain the incoming connections for each unit. This approach has been successfully applied to Kuramoto oscillators, where networks of size N up to about 10^3 units were reconstructed. Assuming sparsity of the network connectivity, such reconstructions may be supplemented by an optimization step (Timme, 2007) that reduces the number of experiments needed to reconstruct up to a desired accuracy (cf. also Yeung et al., 2002; Napoletani & Sauer, 2008). Since all experimental approaches in practice have to deal with a systematic undersampling with regard to finite measurement times and the number of neurons that can be reliably recorded, such optimization methods are likely a necessary step in any potentially useful algorithm.

Here, we present a method to explicitly reconstruct both topology and weights of networks of pulse-coupled, spiking units. It extends the driving-response approach for smoothly coupled phase-oscillators given by Timme (2007) to pulse-coupled leaky integrate and fire neurons. In particular, it uses the information provided by spike times only and does not require explicit access to the actual state variables (membrane potentials) over time. Moreover, the method developed here does not require steady states as in Timme (2007) or Yu & Parlitz (2008), but works as well for complex spatio-temporal activity, such as irregular or even chaotic spiking, given that certain technical conditions are met. The robust optimization technique (Yeung et

al., 2002; Timme, 2007; Napoletani & Sauer, 2008) can be applied here as well, to reconstruct networks of several hundred nodes from a comparably small number of experiments with only a small loss of accuracy.

Biological neuronal networks naturally present much more challenging problems for reconstruction than moderately sized networks of model neurons. The technique presented here should hence not be considered as a substitute to current approaches that work with spiking data from experiments, but as a theoretical complement. This notwithstanding, it has the particular feature to require only spike times and no access to the actual membrane potential. Since in practice intracellular recordings from a large group of neurons are still beyond technical feasibility, in general any method that can work given spike times only has strong merits. In this sense it presents a necessary first step towards a robust explicit method that could be used in tandem with larger-scale statistical methods.

2. Neuron model and reconstruction theory

2.1. Dynamics and collective solution between subsequent spikes

Leaky integrate-and-fire neurons with current-based synapses incorporate two key features of biological neurons: they integrate the input currents of their presynaptic neurons and send spikes whenever this accumulated and filtered presynaptic input sufficiently depolarizes the membrane potential $V(t)$. Though this class of LIF neurons lacks a lot of biophysical detail such as the spike dynamics itself, voltage-dependent integration properties, or neuronal morphology, their study can help to understand biologically relevant collective dynamic states of neuronal networks, such as population rate oscillations (Brunel & Hakim, 1999; Brunel, 2000), propagation of synchronous activity volleys (Diesmann et al., 1999), or asynchronous irregular activity states (Brunel, 2000; Jahnke et al., 2009). Moreover, they are simple enough to be amenable to analytical methods and thus reveal principle insights regarding relations between network interactions and dynamics. In general, the collective dynamics of even simple networks of LIF neurons can be highly complex, depending not only on interaction topology but also on synaptic coupling strengths, local neuron parameters, and initial conditions (van Vreeswijk & Sompolinsky, 1996; Timme et al., 2002; Denker et al., 2004; Zillmer et al., 2009).

The subthreshold dynamics of the membrane potential $V_i(t)$ of a LIF neuron i is given by the linear ordinary differential equation

$$\frac{dV_i(t)}{dt} = \gamma \left(RI_i(t) - V_i(t) \right) + S_i(t) \,, \tag{1}$$

where γ is the inverse of the membrane time constant, R is the membrane resistance, $I_i(t)$ is the incoming synaptic current, and $S_i(t)$ represents the interaction term. Whenever $V_i(t)$ crosses a threshold value V_T the neuron emits a spike and is reset to the potential $V_R < V_T$. Without recurrent interactions ($S_i(t) \equiv 0$), for constant I_i and $V_i(0) = V_R$, the solution of (1) is

$$V_i(t) = RI_i(1 - e^{-\gamma t}) + V_R \, e^{-\gamma t}. \tag{2}$$

When $RI_i > V_T$ (suprathreshold current), the solution becomes periodic, $V_i(t + T_i) = V_i(t)$ with period $T_i = \frac{1}{\gamma} \ln \left(\frac{RI_i - V_R}{RI_i - V_T} \right)$. The subthreshold dynamics of neuron i within a network of N coupled LIF neurons with a non-zero interaction S given by a weighted δ-pulse function is

described by

$$\frac{dV_i(t)}{dt} = \gamma\left(RI_i - V_i(t)\right) + \sum_{j=1}^{N}\sum_{k\in\mathbb{N}} a_{ij}\,\delta\left(t - t_{j,k} - \tau\right)\,, \tag{3}$$

where $t_{j,k}$ is the k-th spike time of neuron j, τ is the synaptic transmission delay, and $a_{ij} < (V_T - V_R)$ is the synaptic coupling strength of the connection $j \to i$. We remark that there is no prior restriction on whether the synaptic coupling strengths a_{ij} are zero (no synapse present), positive (excitatory), or negative (inhibitory). The general solution of the non-autonomous inhomogeneous linear differential equation (3) for times $t > t_0$ with initial condition $V_i(t_0)$ is given by (cf. Arnol'd, 1992)

$$V_i(t) = RI_i\left(1 - e^{-\gamma(t-t_0)}\right) + V_i(t_0)\,e^{-\gamma(t-t_0)} + \sum_{j=1}^{N}\sum_{\substack{k\in\mathbb{N},\\ t_0<t_{j,k}+\tau\le t}} a_{ij}\,e^{-\gamma\left(t-t_{j,k}-\tau\right)}\,. \tag{4}$$

This solution, which holds for all times $t > t_0$ until neuron i next experiences a threshold crossing, provides the basis for the reconstruction method described below.

2.2. General reconstruction methodology

How can we obtain the interaction strengths a_{ij} for a particular neuron i and thus the network topology? The aim is to assemble multiple linearly independent instances of (4) into a linear system of rank N and solve for the unknown a_{ij} values. If the dynamical quantities t_0, t, $V_i(t_0)$ and $V_i(t)$ are specified, equation (4) provides a restriction on the N input coupling strengths a_{ij} for a given neuron i. Choosing sufficiently many different values of the pairs t and t_0 so as to provide linearly independent instances of (4) will therefore give N restrictions that fully determine all coupling strengths a_{ij}, $j \in \{1,\ldots,N\}$. Repeated for each i this yields the entire network topology.

Acceptable inter-spike intervals. We can use subsequent spike times $t_0 = t_{i,\ell-1}$ and $t = t_{i,\ell}$ to fix the dynamical quantities if they are chosen in an appropriate way. We first note that we can safely use any spike time for an initial condition, since post-spike the LIF neuron always immediately resets to voltage V_R. Hence our criterion for choosing an interval depends only on the way threshold crossings are induced, either by an incoming excitatory spike (*spike-induced* spiking) or by passive propagation due to sufficiently large driving I_i (*current-induced* spiking). For current-induced spiking the membrane potential $V_i(t_{i,\ell}^-)$ of neuron i is known to be exactly at the threshold value V_T, while spike-induced spiking generally supplies excess depolarization such that the free membrane potential would be at some unknown value $V_i(t_{i,\ell}^-) > V_T$. If we restrict our choice of intervals $(t_{i,\ell-1}, t_{i,\ell})$ to those terminating with current induced spikes the following substitutions are valid in (4):

$$\begin{aligned} V_i(t_0) &:= V_i(t_{i,\ell-1}^+) = V_R \\ V_i(t) &:= V_i(t_{i,\ell}^-) = V_T\,. \end{aligned} \tag{5}$$

In order to find these intervals we must compare the spike times $t_{i,\ell}$ of neuron i with all possible arrival times $t_{j,k} + \tau$ of all neurons j (including i itself). To exclude intervals with potentially spike-induced spikes, we restrict the analysis to intervals where no $t_{j,k} + \tau$ coincides with the respective $t_{i,l}$ and hence define the inter-spike interval $(t_{i,\ell-1}, t_{i,\ell})$ as *acceptable* for neuron i if

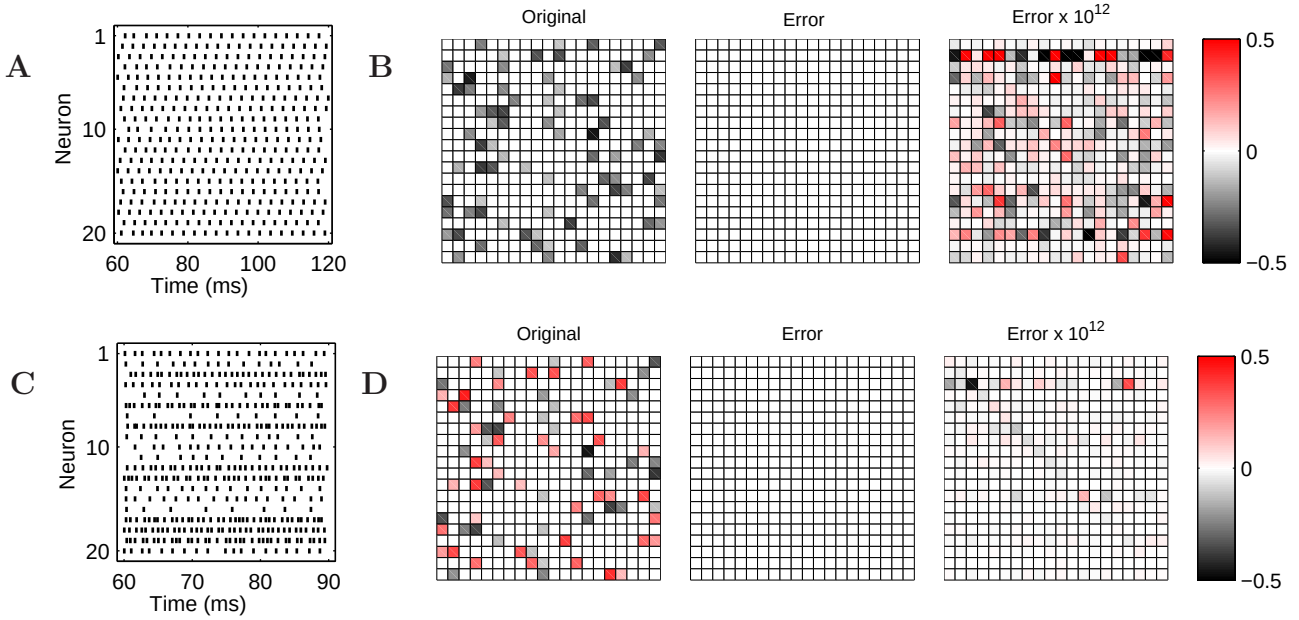


Figure 1: **Topology Reconstruction from simple periodic and complex aperiodic spike patterns (A)** Simple periodic spike pattern of an inhibitory network. **(B)** Connection weights of the same network, with full-rank reconstruction error $a_{ij} - \hat{a}_{ij}$ and its magnification; strengths and errors are color coded on same scale. Error is on the level of numerical accuracy **(C)** Complex spatio-temporal spike pattern of a heterogeneous network with both inhibitory and excitatory connections. **(D)** Connection weights and error for the heterogeneous network. Both networks share the same underlying adjacency structure; the heterogeneous network was obtained from the inhibitory network by randomly changing the signs of connections with probability $p = 0.5$. Simulation parameters for both systems: driving $\gamma RI = 1\,\mathrm{mV/ms}$, $\Delta I = 0.05$ (5% uniform random variation), $\gamma = 0.6321/\mathrm{ms}$, $V_R = 0\,\mathrm{mV}$, $V_T = 1\,\mathrm{mV}$, $\tau = 0.25\,\mathrm{ms}$.

$$t_{j,k} + \tau \neq t_{i,\ell} \quad \text{for all} \quad j \in \{1, \ldots, N\}, k \in \mathbb{N}\,. \tag{6}$$

It is important to note two things: first, that we often have at our disposal many more than N acceptable intervals; and second, that not every set of N acceptable intervals will provide N linearly independent instances of (4). In particular, consecutive acceptable intervals might contain very similar arrival patterns, resulting in badly-conditioned or even low-rank systems. Of course, there is no constraint to select only N intervals; if we use $M > N$ intervals then the system will be formally overdetermined but solvable, though $M \gg N$ is not favorable for efficiency reasons. Usually, selecting a subset of N or more intervals distributed across the respective incoming spike trains will be sufficient if the spiking is asynchronous and irregular (meaning that inter-spike interval correlations decrease quickly with time), the mean firing rate for all neurons is > 0, and the spike recording is long enough to find sufficiently many uncorrelated acceptable inter-spike intervals to derive N linearly independent instances of (4). Since we do not know a priori how long "long enough" is, an alternative approach that for all practical purposes guarantees linear independence (as long as all other conditions hold) is to gather data from multiple, relatively short spike trains obtained under varied driving conditions $I_{i,m}$, $m \in \{1, \ldots, N\}$ (cf. Timme, 2007) which additionally satisfy $\forall_{i,j}\ i \neq j \Rightarrow I_{i,m} \neq I_{j,m}$.

In section 3.1 we will discuss various strategies for selecting subsets of acceptable intervals, in section 3.2 how much spike data is commonly needed to obtain full accuracy, and in section 3.4 how singular value decomposition can be applied to solve underdetermined systems i.e. using $M < N$ intervals.

Assuming there are $M \geq N$ acceptable intervals that will ensure linear independence, all presynaptic connections for neuron i, $A_i = (a_{i1}, \ldots, a_{iN})$ are obtained in the following way: let $D_{i,m} = (t_{i,\ell_m-1}, t_{i,\ell_m})$ be the m-th member of our subset of acceptable inter-spike intervals, and $T_{i,m} = t_{i,\ell_m} - t_{i,\ell_m-1}$ be the respective inter-spike interval of neuron i. We then define the $M \times N$ matrix $\Theta^{(i)}$ and the column vector $B^{(i)}$ by

$$\Theta^{(i)}_{mj} = \sum_{\substack{k\in\mathbb{N},\\ t_{j,k}+\tau \in D_{i,m}}} \exp\left(-\gamma\left[t_{i,\ell_m} - t_{j,k} - \tau\right]\right) \quad \text{and}$$

$$B^{(i)}_m = V_T - RI_{i,m}\left(1 - e^{-\gamma\, T_{i,m}}\right) - V_R e^{-\gamma\, T_{i,m}}\,. \tag{7}$$

The incoming connectivity for neuron i is obtained by solving the linear system

$$\Theta^{(i)} A_i^\top = B^{(i)} \tag{8}$$

for A_i. Repeating the procedure for each neuron $i \in \{1, \ldots, N\}$ separately yields all N rows of the coupling matrix A, and thus the entire synaptic connectivity of the network.

2.3. Reconstruction in the presence of simple periodic spiking

The method described above can be applied to a broad range of spiking patterns and does not rely on network synchronization, phase locking, or other steady state requirements; however, systems that do exhibit phase locked dynamics are well-suited for systematic tests of the method. For this purpose we applied it to networks that produce *simple periodic patterns*, i.e. all neurons fire exactly once before the same spike pattern repeats.

If the neurons fire in a simple periodic pattern such that for any given interspike interval $(t_{i,\ell-1}, t_{i,\ell})$ each neuron $j \neq i$ spikes exactly once within $(t_{i,\ell-1} - \tau, t_{i,\ell} - \tau)$, all intervals are acceptable and (4) simplifies to

$$V_T = RI_i\left(1 - e^{-\gamma(t_{i,\ell} - t_{i,\ell-1})}\right) + V_R\, e^{-\gamma(t_{i,\ell} - t_{i,\ell-1})} + \sum_{j=1}^{N} a_{ij} e^{-\gamma\left(t_{i,\ell} - t_j - \tau\right)} \tag{9}$$

for all $i \in \{1, ..., N\}$. Such networks are in particular phase-locked. If a network is able to achieve such periodic simple spike patterns under N independent driving conditions $I_{i,m}$, $m \in \{1, ..., N\}$, all intervals are hence acceptable and for convenience we can always choose the last intervals recorded in the simulation to solve the N linear systems (4). This means that although this type of network dynamics makes it necessary to repetitively drive the system with independent sets of currents (otherwise repeated spike patterns within the inter-spike intervals will lead to linear dependent rows in the derived Θ matrix (7)), it spares us the necessity of scanning the spike data for acceptable intervals and hence allows for fast reconstruction. We will refer to this case as the *specialized method* and will use a sub-class of networks that can reliably produce simple periodic spiking, *homogeneous inhibitory* LIF networks, as a testbed for it.

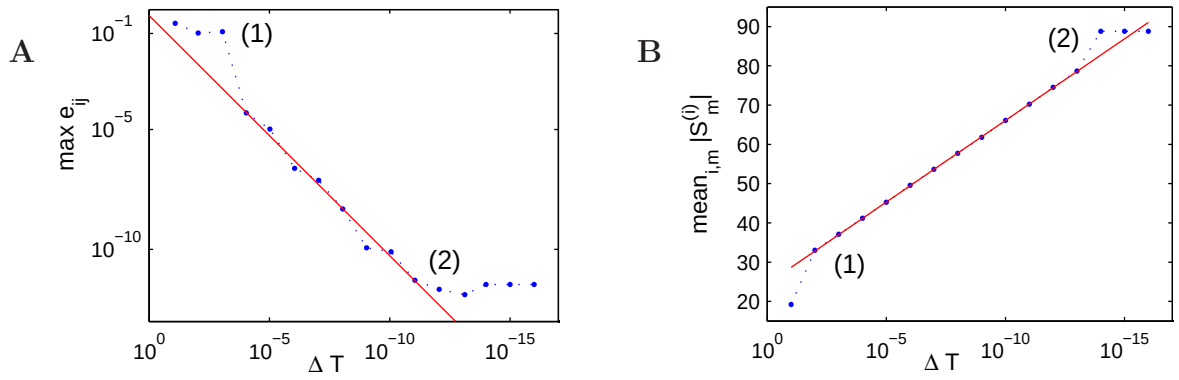


Figure 2: **Reconstruction accuracy for phase-locking networks as affected by periodicity. (A)** Maximum reconstruction error $e_{ij} = |a_{ij} - \widehat{a}_{ij}|$ vs. phase locking as measured by the maximum difference between final period lengths of different neurons (ΔT). **(B)** Length of simulation runs S_m (given as number of spikes emitted by neuron i, $|S_m^{(i)}|$) needed to achieve a specified ΔT (exit tolerance). Annotations: (1) indicates non-full-rank solutions due to insufficient synchronization; (2) indicates numerical saturation. Data is from sample network used for figures 1(A) and (B).

Characterization of the homogeneous inhibitory subclass: when stimulated with a homogeneous current $I_i = I$ (where I is constant), inhibitory networks where $a_i \equiv \sum_j a_{ij} < 0$ and $0 < (\max_i a_i - \min_i a_i) < \tau$ will generate stable phase-locked states with common period T^{net} (Denker et al., 2004). However, heterogeneous currents I_i, e.g. randomly drawn from some distribution $\mathbb{P}(I)$ with small but finite variance, will additionally broaden the temporal spread of the pattern of spikes and might lead to loss of stability. Hence, to better control stability, we assume that $a_i = a < 0$ for each i, such that the spread of the spike pattern is solely caused by the spread $\Delta I_{i,m} \equiv \max_i I_{i,m} - \min_i I_{i,m}$ of the driving currents per experiment $m \in \{1, ..., N\}$.

3. Implementation and performance

3.1. Acceptable interval searches

For LIF networks other than the homogeneous inhibitory networks described in section 2.3, spike patterns will almost always be asynchronous rather than simple periodic; hence we must actively search through spike trains for acceptable intervals.

In principle, if a full-rank system *is* embedded in the data then only N intervals are required, some of which may contain zero or more than one representative $t_{j,k}+\tau$ value. One alternative to trying to identify such a minimum set of intervals would be to select $M > N$ intervals according to some criterion which either guarantees or at least maximizes the chance that we have a full rank subset; we could, for example, select *all* available acceptable intervals. Similarly, if we have $M > N$ intervals we have the choice of solving the overdetermined $M \times N$ by e.g. a least-squares method, or aggregating instances of (4) to patch together an $N \times N$ system.

In our implementation we used independent partial scans of spike train segments to collect $M \geq N$ intervals; the scans use the heuristic described below to construct a full-rank system if possible while reducing search time and keeping M reasonably close to N. Multiple intervals found in close proximity are combined to create exactly solvable $N \times N$ systems, since they

are preferrable to overdetermined systems in terms of both time requirements and numerical stability.

Within segments, our search strategy is based on a minimum-scan-length policy. Given data segments S_m, $m \in \{1, \ldots, N\}$, then for each neuron i each segment S_m is scanned only as far as needed to obtain representative $t_{j,k} + \tau$ values for all neurons j, and intervals are only chosen if they contributed a new member to the representative set. Instances of (4) for the intervals are then summed to become the m-th row of the solution matrix $\Theta^{(i)}$ and m-th element of the vector $B^{(i)}$. When particular representatives are not present in particular segments then corresponding $\Theta^{(i)}_{j,m}$ elements are set to zero. Balancing representatives by requiring a full set for each row and excluding redundancies is more than we technically require, but in practice this results in better conditioned $\Theta^{(i)}$ matrices and hence more accurate results.

3.2. Accuracy

The reconstruction error is negligible for networks with phase-locked spike patterns as well as those with irregular spiking, as shown for example in figure 1. The numerical noise that does appear in the reconstructions is the result of small truncation errors in the derived data being propagated through global operations such as matrix inversion. When the specialized version of the method is used on phase-locking networks, this error level is as well directly and linearly dependent on the degree of synchronization experienced by the network (see figure 2). As the exit tolerance is decreased this tunable error meets a hard limit where the truncation is no longer caused by early termination of the simulation, but instead to the fixed precision numbers used by the machine the calculations are performed on. It should be noted that (apart from their influence on the network's ability to synchronize) the magnitude of simulation parameters does not affect the accuracy of the results in any noticable way.

Since the general reconstruction method does not depend on synchronization, for data characterized by sustained irregular or chaotic spiking the entire error can be attributed to fixed-precision effects. A necessary precondition for achieving this level of accuracy is of course that enough data be made available. As figure 3(A) shows, the minimum number of intervals required to achieve full accuracy is not large; roughly 7 or 8 per segment (≈ 160 total) for our 20 neuron sample network, for example. Whether data is obtained from N independent drivings or from a single run does not as well seem to have a large effect; sampling out of a very long run does reduce the minimum requirement by an interval or two, but the amount of data that must be generated in the first place to achieve this effect is appreciably greater. Compare also the specialized method featured in figure 2, where the data requirement is exactly one interval per segment, but to achieve e.g. 10^{-10} accuracy each segment needs to be ≈ 60 intervals long.

Independent of the driving conditions applied, there is no prior constraint on network configurations solvable by the methodology (subject to obvious provisos e.g. there should not be individual suprathreshold connections); there is likewise no prior guarantee that any particular driving will give good results. In practice, randomly varied driving which satisfies $RI_i > V_T$ almost always gave us fully accurate reconstructions. There are two situations can cause problems for the general method when otherwise long-enough data streams are made available: i) quiescent neurons (no recoverable incoming or outgoing connections), and ii) neurons for which all spiking throughout entire segments is spike-induced (outgoing connections recoverable, incoming connections not). Only (i) arose in our testing on random networks with random drivings, and never with networks subject any kind of balance conditions e.g. $\left|\sum_{j,\text{inhib}} a_{ij}\right| \simeq \left|\sum_{j,\text{excit}} a_{ij}\right|$.

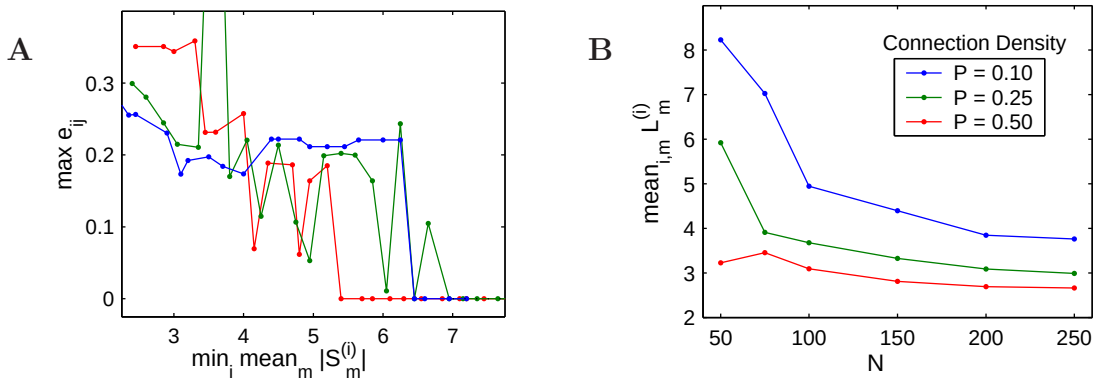


Figure 3: **Reconstruction accuracy for complex spatio-temporal patterns as affected by search requirements.** **(A)** Maximum reconstruction error $e_{ij} = |a_{ij} - \widehat{a}_{ij}|$ vs. data availability as measured by segment length (in spikes) for slowest neuron. Different curves compare effect of data source: (blue) segments obtained under N different drivings, (green) contiguous segments taken from one run of moderate length where driving was not modified, or (red) same conditions, but segments taken at equally spaced intervals from a single long run (roughly 1000 spikes per neuron). Runs use the same sample network and parameters as shown in figures 1(C) and (D). **(B)** Interval search length L using min-scan-length strategy vs. size of network N for different connection densities P. Note: all reconstructions here achieved the numerical limit of accuracy. Simulation parameters: balanced networks ($\sum_j a_{ij} = 0$) with no other restriction on pre- or postsynaptic connectivity, data obtained under N uniformly randomized driving conditions, base driving $\gamma RI = 1.5\,\mathrm{mV/ms}$, $\Delta I = 0.01$; $\tau = 2\,\mathrm{ms}$, $\gamma = 0.05/\mathrm{ms}$, $V_R = 0\,\mathrm{mV}$, $V_T = 20\,\mathrm{mV}$.

It should be noted that with respect to both situation (i) and situation (ii) the accuracy of reconstruction of the remainder of the network is not affected, and if need be both situations can be prevented by adjusting the driving to the relevant neurons.

3.3. Efficiency

To check the speed of the methods we reconstructed networks of $N = 50$ to $N = 500$ neurons. As the range of N here implies, moderately sized networks (several hundred neurons) are still quite feasible for both the specialized and the general method; using the specialized method reconstruction of 250 neuron networks took about one minute on a single node of a 4-CPU 2 GHz machine, while general reconstruction of the same network took about 20 minutes.

As these time ranges imply, the general method's reliance on data acquired from directly searching spike trains makes it noticably slower than the specialized method in the size range tested here. This is in part due to the length of searches, but fixed costs for setting up the searches and computational platform were also factors (implementations were done in MATLAB, which is optimized for matrix operations but not generic search routines).

The worst case running time for the search phase of the general method is $\mathcal{O}(S_{\max} N^3)$, where $S_{\max}$ is the length of the longest spike train. In practice most searches are not worst case; for the network of figure 1(D), for example, the mean spike train length was 92 spikes, while the mean search length was 12 intervals and the mean number of intervals selected was around 3. As figure 3(B) shows, the search length does not seem to increase as N grows but rather converges

to a constant value; a similar effect is seen with the connection density (and hence the number of connections). We note that if either situation (i) or (ii) mentioned in section 3.2 occurs, the search time will in fact revert to the theoretical worst case.

3.4. Exploiting sparsity - solutions for underdetermined systems

Recent studies (Yeung et al., 2002; Timme, 2007) used singular value decomposition $USV^{\mathrm{T}} = \Theta^{(i)}$ and optimization for sparsity to reconstruct with $M \ll N$ distinct driving conditions. In general, this yields the underdetermined system $A_i = V\widetilde{S}^{-1}U^{\mathrm{T}}B^{(i)} + Vx$ where $\widetilde{S}^{-1}$ is the pseudoinverse of S and x is any vector. Setting $x = \vec{0}$ gives us the usual least squares solution, but on the naturalistic assumption that neuronal network connectivity matrices A are very sparse we can choose x so as to maximize the number of zeros in each A_i. To do this we use the heuristic of solving the overdetermined system $Vx = -V\widetilde{S}^{-1}U^{\mathrm{T}}B^{(i)}$ for x so as to minimize the L_1-norm of the difference; our implementation uses the fast L_1-norm solver of Barrowdale & Roberts (1974).

Of primary interest is the lowest $M < N$ we require to obtain sufficiently many reconstructed couplings $\widehat{a}_{ij}$ close enough to the corresponding elements of the original connectivity matrix a_{ij}, so that $\widehat{A}$ (after appropriate rounding) can be used as a plausible estimate for A in any hypothetical analysis. Hence along with the usual error measurements we used a custom quality-of-fit metric, $Q_\alpha^{(w)}(M)$, that essentially yields the proportion of sufficiently close element-to-element matches for a given M:

$$\begin{aligned} Q_\alpha^{(w)}(M) &= w\,\frac{1}{|\,\mathrm{nz}(A)|}\sum_{a_{ij}\neq 0} H\left((1-\alpha) - \frac{|\widehat{a}_{ij} - a_{ij}|}{\max|A|}\right) \\ &+ (1-w)\frac{1}{|\,\mathrm{z}(A)|}\sum_{a_{ij}=0} H\left((1-\alpha) - \frac{|\widehat{a}_{ij} - a_{ij}|}{\max|A|}\right) \end{aligned} \tag{10}$$

where $\mathrm{nz}(A)$ and $\mathrm{z}(A)$ are the sets of all non-zero and all zero elements of A, respectively, and H is the Heaviside step function. The weighting is incorporated to prevent false high scores when M is low and the target A is very sparse (cf. the original version of the quality measure in Timme, 2007). We exclusively used $Q_\alpha^{(w)}$ with $w = 0.5$, which in the following we simply denote by Q_α. Figure 4.a shows a sampling of Q_α values versus M.

For networks with simple phase-locked spike pattern dynamics, comparison of interpolated values of Q_α across different values of N, M, the number of connections E, and other simulation parameters confirm that the quality of reconstruction for $M < N$ depends almost entirely on M/N and the edge density $P = E/N^2$. To determine the minimum M necessary to achieve a desired quality of fit we define

$$M_{q,\alpha} = \min_{1\leq M\leq N}\{M : Q_\alpha(M) \geq q\}\,. \tag{11}$$

The Q_α profiles in figure 4.a show that there will be values of q for which Q_α does not exist or is not unique. When both $q, \alpha \geq 0.75$, however, $M_{q,\alpha}$ is well-defined and we consistently obtain relations of the form

$$M_{q,\alpha} \approx E^\beta + c \tag{12}$$

where $\beta = \frac{1}{2} + \epsilon$, with $|\epsilon| \ll 1/2$ and $c \ll E$. A simple log-log fit of the data from 4(B), for example, gives a scaling exponent of 0.504, with c below 1. The overall result implies that we can expect $M_{q,\alpha}$ to grow linearly with N when P is fixed, and roughly as $\sqrt{P}$ when N is fixed.

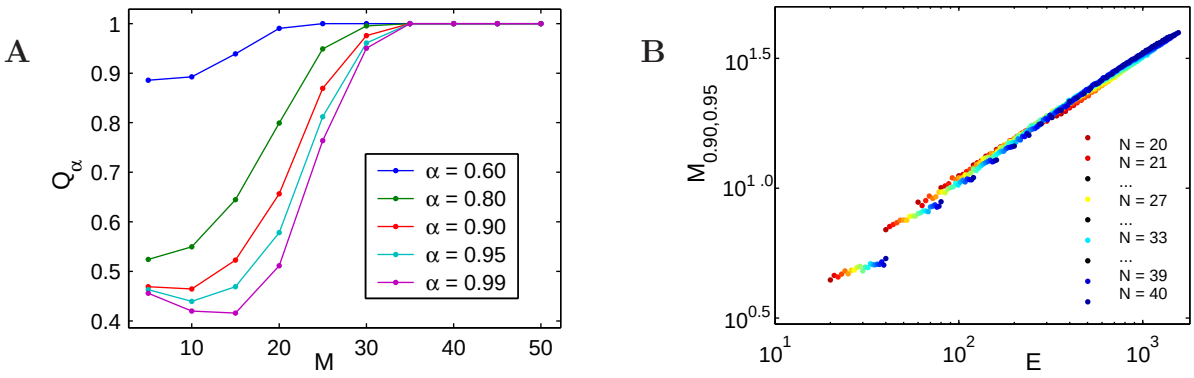


Figure 4: **Quality of rank-deficient reconstruction.** **(A)** Various Q_α values vs. M ($N = 50$, mean number of presynaptic connections $K = 12$, $\tau = 2\,\mathrm{ms}$, $\gamma = 0.05/\mathrm{ms}$, $V_R = 0\,\mathrm{mV}$, $V_T = 20\,\mathrm{mV}$, base driving $\gamma RI = 1.5\,\mathrm{mV/ms}$, $\Delta I = 0.01$). **(B)** Log-log plot of $M_{0.90,0.95}$ vs. E for networks with $N = 20$ to 40 neurons and average presynaptic connections $K = 1, 2, \ldots, N-1$ (over 600 points in total). Q_α values for (B) were obtained from averages of 20 trials for each N, K combination over each M from 1 to N; simulation parameters were $\tau = 0.25\,\mathrm{ms}$, $\gamma = 0.6321/\mathrm{ms}$, $V_R = 0\,\mathrm{mV}$, $V_T = 1\,\mathrm{mV}$, base driving $\gamma RI = 1\,\mathrm{mV/ms}$, $\Delta I = 0.001$. All networks: random homogeneous inhibitory with normally distributed random connection weights.

For general spatio-temporal patterns we obtain similar results, though they are not as pronounced or consistent, and there seems to be a greater dependence on factors other than M, N, and E, such as actual network parameters and specific spike pattern. The particular interval search strategy used has an influence as well. These intermingled technical issues require further theoretical and computational studies.

4. Discussion

Understanding the relationship between structure and function of neuronal circuits lies at the very core of neuroscience. Modern recording devices, such as multi-unit arrays (MUA) that record the extracellular spike activity of tens to hundreds of neurons, provide time series whose interpretation can aid the understanding of the relevant functional connectivity and its spatial structure (cf. Aertsen & Gerstein, 1985; Aertsen et al., 1989; Berger et al., 2007, 2010, and others). Yet, to reconstruct the detailed connectivity of a neuronal network from recorded spike data is beyond recent data analysis techniques due to the complexity of both the individual neuron dynamics as well as the intricate mesh of somata and neurites encountered.

Here we have developed an explicit method for the reconstruction of network connectivity of leaky integrate-and-fire (LIF) neurons from spike-time information that is fast, accurate, and robust. The method works natively on networks exhibiting nonstationary complex spatio-temporal dynamics (i.e. "plain" irregular spiking, Sec. 2.2); to solve networks with strong synchronizing tendencies it can, for example, make use of multiple randomized external driving conditions. A specialization of the method allows it to very efficiently reconstruct networks exhibiting simple spike patterns (Sec. 2.3), a dynamics that is reliably generated by a large subset of uniformly inhibitory LIF networks. This class is often used as a stand-in for inhibition dominated cortical networks (cf. Brunel & Hakim, 1999; Denker et al., 2004; Jahnke et al., 2009).

The general method is capable of reconstructing networks with several hundred neurons and ar-

bitrary connectivity in a quite reasonable time given standard computational resources. In this regard the methodology compares favorably with reconstruction methods previously put forward for neuron networks (Makarov et al., 2005) and oscillator networks (Yu et al., 2006). Furthermore, the method is amenable to an optimization used earlier with reconstruction techniques for networks with fixed point dynamics (Yeung et al., 2002) and periodic dynamics (Timme (2007), cf. also Napoletani & Sauer (2008)) that substantially reduces the amount of data required.

The method introduced here is one of the first to achieve synapse-level reconstructions working from spike-based data, and it extends recoverability of pulse-based networks far beyond previously accessible sizes. However, as it currently stands, it does have its limitations that need to be addressed by future research. To begin with, it is model dependent, and requires fixed known parameter values, even though these values need not be the same for every neuron in the network. Neuron models with more realistic spike generation mechanisms are worth further analysis within the response-dynamics idea that underlies our methodology. Possible extensions include: the *spike-response* model, a modifications of LIF neuron that incorporates refractoriness; the two variable *adaptive exponential integrate-and-fire* neuron, which is capable of a much wider range of individual neural behaviour (bursting, fast and slow mode, etc.); and further non-linear models such as the quadratic integrate-and-fire neuron. Ultimately, the general aim is that a single algorithm contains the specifications for a repetoire of different models, which can be adapted to particular self-contained networks or also parts of networks, given an input model that mimicks the residual large-scale network or upstream input areas. Since all available recording techniques have to deal with the general problem of undersampling, both with regard to the number of neurons as well as with regard to the finite time a stable recording is possible, minimizing the data required for desired accuracy is an important feature of any potentially useful reconstruction algorithm.

The method presented here is not aimed to replace large-scale correlation-based methods, as functional connectivity continues to be the main approach to reconstruction problems in neuroscience. However, to determine the details within the regions of interest that functional connectivity studies identify will require methods that can provide data on the scale of individual synaptic connections. If explicit methods such as the one presented here can be developed to work on large-scale networks with appropriate biophysical neuron models, they may become invaluable in these studies. Thus we see our methodology as an eventual complement to techniques currently in use.

Acknowledgments: We thank Sara Solla and Fred Wolf for helpful comments. We also thank Christian Henrich for his contributions in the early stages of this work, and Zeina S. Khan for her careful reading of the manuscript. We acknowledge funding by the Federal Ministry of Education and Research (BMBF) Germany for partial support under Grant No. 01GQ0430.

References

Aertsen, A., and Gerstein, G.L. (1985). Evaluation of neuronal connectivity: sensitivity of cross-correlation. *Brain Res.* 340, 341–354.

Aertsen, A., Gerstein, G.L., Habib, M.K., and Palm, G. (1989). Dynamics of neuronal firing correlation: modulation of "effective connectivity". *J. Neurophysiol.* 61(5), 900–917.

Arnol'd, V.I. (1992). *Ordinary Differential Equations*, R. Cooke, trans. (Berlin: Springer-Verlag).

Barrowdale, I., and Roberts, F.D.K. (1974). Algorithm 478: solution of an overdetermined system of equations in the l_1 norm [F4] *Commun. ACM* 17, 319-320.

Berger, D., Warren, D., Normann, R., Arieli, A., and Grün, S. (2007). Spatially organized spike correlation in cat visual cortex. *Neurocomputing* 70, 2112–2116.

Berger, D., Borgelt, C., Louis, S., Morrison, A., and Grün, S. (2010). Efficient identification of assembly neurons within massively parallel spike trains. *Comput. Intell. Neurosci.* 2010, 439648.

Brunel, N., and Hakim, V. (1999). Fast global oscillations in networks of integrate-and-fire neurons with low firing rates. *Neural Comput.* 11, 1621-1671.

Brunel, N. (2000). Dynamics of sparsely connected networks of excitatory and inhibitory spiking neurons. *J. Comput. Neurosci.* 8, 183-208.

Cadenasso, M.L., Pickett, S.T.A., and Grove, J.M. (2006). Dimensions of ecosystem complexity: heterogeneity, connectivity, and history. *Ecological Complexity* 3, 1-12.

Denker, M., Timme, M., Diesmann, M., Wolf, F., and Geisel, T. (2004). Breaking synchrony by heterogeneity in complex networks. *Phys. Rev. Lett.* 92, 074103.

Diesmann, M., Gewaltig, M.-O., and Aertsen, A. (1999). Stable propagation of synchronous spiking in cortical neural networks. *Nature* 402, 529-533.

Jahnke, S., Memmesheimer, R.-M., and Timme, M. (2009). How chaotic is the balanced state?. *Front. Comput. Neurosci.* 3:13, doi:10.3389/neuro.10.013.

Jirsa, V.K., and McIntosh, A.R., Eds. (2007). *Handbook of Brain Connectivity* (Berlin: Springer).

Makarov, V.A., Panetsos, F., and de Feo, O. (2005). A method for determining neural connectivity and inferring the underlying network dynamics using extracellular spike recordings. *J. Neurosci. Meth.* 144, 265-279.

Memmesheimer, R.-M., and Timme, M. (2006). Designing complex networks. *Physica D* 224, 182–201.

Napoletani, D., and Sauer, T. (2008). Reconstructing the topology of sparsely connected dynamical networks. *Phys. Rev. E* 77, 026103

Timme, M., Wolf, F., and Geisel, T. (2002). Coexistence of regular and irregular dynamics in complex networks of pulse-coupled oscillators. *Phys. Rev. Lett.* 89, 258701.

Timme, M. (2007). Revealing network connectivity from response dynamics. *Phys. Rev. Lett.* 98, 224101.

van Vreeswijk, C., and Sompolinsky, H. (1996). Chaos in neuronal networks with balanced excitatory and inhibitory activity. *Science* 274, 1724-1726.

Yeung, M.K.S., Tegnér, J., and Collins, J.J. (2002). Reverse engineering gene networks using singular value decomposition and robust regression. *Proc. Natl. Acad. Sci. USA* 99, 6163-6168.

Yu, D., and Parlitz, U. (2008). Driving a network to steady states reveals its cooperative architecture. *Europhys. Lett.* 81, 48007.

Yu, D., Righero, M., and Kocarev, L. (2006). Estimating topology of networks. *Phys. Rev. Lett.* 97, 188701.

Zillmer, R., Brunel, N., and Hansel, D. (2009) Very long transients, irregular firing, and chaotic dynamics in networks of randomly connected inhibitory integrate-and-fire neurons. *Phys. Rev. E* 79, 031909.

Conflict of interest statement: The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

Chapter 3

Article: *Counting Complex Disordered States by Efficient Pattern Matching: Chromatic Polynomials and Potts Partition Functions*

Marc Timme, Frank Van Bussel, Denny Fliegner, and Sebastian Stolzenberg
New Journal of Physics **11** 023001 (2009)
Submitted: September 6, 2008; published: February 4, 2009

New Journal of Physics
The open-access journal for physics

Counting complex disordered states by efficient pattern matching: chromatic polynomials and Potts partition functions

Marc Timme[1,3], **Frank van Bussel**[1], **Denny Fliegner**[1] **and Sebastian Stolzenberg**[2]

[1] Max Planck Institute for Dynamics and Self-Organization, Bunsenstrasse 10, 37073 Göttingen, Germany
[2] Department of Physics, Cornell University, 109 Clark Hall, Ithaca, NY 14853-2501, USA
E-mail: timme@nld.ds.mpg.de

New Journal of Physics **11** (2009) 023001 (10pp)
Received 6 September 2008
Published 4 February 2009
Online at http://www.njp.org/
doi:10.1088/1367-2630/11/2/023001

Abstract. Counting problems, determining the number of possible states of a large system under certain constraints, play an important role in many areas of science. They naturally arise for complex disordered systems in physics and chemistry, in mathematical graph theory, and in computer science. Counting problems, however, are among the hardest problems to access computationally. Here, we suggest a novel method to access a benchmark counting problem, finding chromatic polynomials of graphs. We develop a vertex-oriented symbolic pattern matching algorithm that exploits the equivalence between the chromatic polynomial and the zero-temperature partition function of the Potts antiferromagnet on the same graph. Implementing this bottom-up algorithm using appropriate computer algebra, the new method outperforms standard top-down methods by several orders of magnitude, already for moderately sized graphs. As a first application, we compute chromatic polynomials of samples of the simple cubic lattice, for the first time computationally accessing three-dimensional lattices of physical relevance. The method offers straightforward generalizations to several other counting problems.

[3] Author to whom any correspondence should be addressed.

Given a set of different colors, in how many ways can one color the vertices of a graph such that no two adjacent vertices have the same color? The answer to this question is provided by the chromatic polynomial of a graph [1, 2], which gives the number of possible colorings as a function of the number q of colors available. It is a polynomial in q of degree N, the number of vertices of the graph. The chromatic polynomial is closely related to other graph invariants e.g. to the reliability and flow polynomials of a network or graph (functions that characterize its communication capabilities) and to the Tutte polynomial. These are of widespread interest in graph theory and computer science and pose similar hard counting problems.

The chromatic polynomial is also of direct relevance to statistical physics, as it is equivalent to the zero-temperature partition function of the Potts antiferromagnet [3, 4]: the Potts model [3] constitutes a paradigmatic characterization of systems of interacting electromagnetic moments or spins, where each spin can be in one out of $q \geqslant 2$ states; it thus generalizes the Ising model where $q = 2$. For antiferromagnetic interactions, neighboring spins tend to disalign such that at zero-temperature the partition function of the Potts antiferromagnet counts the number of ground states of a spin system on a graph just as the chromatic polynomial counts the number of proper colorings of the same graph. For sufficiently large q there are *many* system configurations in which *all* pairwise interaction energies are minimized at zero-temperature. Indeed, these systems exhibit a large number of disordered ground states that is exponentially increasing with system size. Thus, the Potts model exhibits positive ground state entropy, an exception to the third law of thermodynamics. Experimentally, complex disordered ground states and related residual entropy at low temperatures have been observed in various systems [5]–[10].

Although there are several analytical approaches to find chromatic polynomials for families of graphs and to bound their values [1]–[4], [11]–[17], there is no closed-form solution to this counting problem for general graphs. Algorithmically it is hard to compute the chromatic polynomial, because the computation time, in general, increases exponentially with the number of edges in the graph [18]. It also strongly depends on the structure of the graph and rapidly increases with the graph's size, and the degrees of its vertices, cf [19]–[21]. Therefore, most studies on chromatic polynomials up to date have focused on small graphs and families of graphs of simple structure and low vertex degrees, e.g. two-dimensional lattice graphs [15]–[17] (an interesting recent attempt to analytically study simple cubic lattices considered strips with reduced degrees [11]). In fact, it is not at all straightforward to computationally access larger graphs with more involved structure, including physically relevant three-dimensional lattice graphs. Finding the chromatic polynomial of a graph thus constitutes a challenging, computationally hard problem of statistical physics, graph theory and computer science (cf [18, 19]).

Below we present a novel, efficient method to compute chromatic polynomials of larger structured graphs. Representing a chromatic polynomial as a zero-temperature partition function of the Potts antiferromagnet, we transform the computation into a local and vertex-oriented, ordered pattern matching problem, which we then implement using appropriate computer algebra. In contrast to conventional top-down methods that represent and process the entire graph (and many modified copies thereof) from the very beginning, the new method presented here works through the graph bottom-up and thus processes comparatively small local parts of the graph only.

Consider a graph G that is defined by a set of N vertices $i \in V = \{1, \ldots, N\}$ and a set of $M := |E|$ edges $\{i, j\} \in E$, each edge joining two vertices i and j which are then called adjacent or neighboring. This graph is said to be (properly) q-colored if every vertex is given one out of

q colors $\{1, 2, \ldots, q\}$ such that every two adjacent vertices have different colors. The number of q-colorings of a graph G is expressed by its chromatic polynomial $P(G, q)$, a polynomial in q of order N [2].

The deletion–contraction theorem of graph theory [2] suggests a simple algorithm to compute the chromatic polynomial of a given graph recursively. In principle, this algorithm works for arbitrary graphs and is therefore, with certain improvements, implemented in general-purpose computer algebra systems such as Mathematica [22]–[24] and Maple [25] (cf also [26, 27]). However, applying the theorem recursively the chromatic polynomial of exponentially many graphs must be found, the (weighted) sum of which yields the chromatic polynomial of the original graph. This reflects how hard the problem is algorithmically and severely restricts the applicability of computational methods, in particular if they employ standard top-down processing.

We now describe our novel algorithm. It is based on the antiferromagnetic ($J < 0$) Potts model [3, 4] with Hamiltonian

$$H(\boldsymbol{\sigma}) = -J \sum_{\{i,j\}\in E} \delta_{\sigma_i\sigma_j} \tag{1}$$

giving the total energy of the system in state $\boldsymbol{\sigma} = (\sigma_1, \ldots, \sigma_N)$. Here individual spins σ_i can assume q different values $\sigma_i \in \{1, \ldots, q\}$, generalizing the Ising model ($q = 2$) [28]. Two spins σ_i and σ_j on the graph G interact if and only if they are neighboring, $\{i, j\} \in E$, and in the same state, $\sigma_i = \sigma_j$, i.e. the Kronecker-delta is $\delta_{\sigma_i\sigma_j} = 1$ (otherwise, for any pair $\sigma_i \neq \sigma_j$, it is $\delta_{\sigma_i\sigma_j} = 0$). Thus, the total interaction energy is minimized if all pairs of neighboring spins are in different states.

The partition function $Z(G, q, T) = \sum_{\boldsymbol{\sigma}} \exp(-\beta H(\boldsymbol{\sigma}))$ at positive temperature $T = (k_\mathrm{B}\beta)^{-1}$, where k_B is the Boltzmann constant, can be represented as

$$Z(G, q, T) = \sum_{\boldsymbol{\sigma}} \prod_{\{i,j\}\in E} (1 + v\delta_{\sigma_i\sigma_j}), \tag{2}$$

where $v = \exp(\beta J) - 1 \in (-1, 0]$. In the limit $T \to 0$ (implying $\beta J \to -\infty$ and thus $v \to -1$) this partition function counts the number of ways of arranging the spins σ such that no two adjacent spins are in the same state. Thus, the zero-temperature partition function (2) exactly equals [4] the chromatic polynomial

$$P(G, q) = \lim_{T\to 0} Z(G, q, T) \tag{3}$$

on the same graph G leading to the representation [4, 12]

$$P(G, q) = \sum_{\sigma_1=1}^{q} \cdots \sum_{\sigma_N=1}^{q} \prod_{\{i,j\}\in E} (1 - \delta_{\sigma_i\sigma_j}), \tag{4}$$

of the chromatic polynomial in terms of sums over products of Kronecker-deltas.

The algorithm exploits this representation by expanding the products in (4) and symbolically evaluating the right-hand side vertex by vertex (cf figure 1), considering each individual sum $\sum_{\sigma_k}$ as an operator. This operator interpretation relies on a recently studied algebraic structure of expressions containing Kronecker-deltas [29]. Here such an operator has the simple actions

$$\sum_{\sigma_k} 1 = q, \tag{5}$$

$$\sum_{\sigma_k} \delta_{\sigma_k \sigma_{j_1}} = 1, \tag{6}$$

$$\sum_{\sigma_k} \delta_{\sigma_k \sigma_{j_1}} \delta_{\sigma_k \sigma_{j_2}} = \delta_{\sigma_{j_1} \sigma_{j_2}}, \tag{7}$$

and for an arbitrary number $r \in \mathbb{N}_0$ of factors,

$$\sum_{\sigma_k} \delta_{\sigma_k \sigma_{j_1}} \cdots \delta_{\sigma_k \sigma_{j_r}} = \delta_{\sigma_{j_1} \sigma_{j_2}} \cdots \delta_{\sigma_{j_{r-1}} \sigma_{j_r}}, \tag{8}$$

if the j_ρ, $\rho \in \{1, \ldots, r\}$, are pairwise distinct and all $j_\rho \neq k$.

For illustration consider the chromatic polynomial

$$P(G, q) = \sum_{\sigma_3=1}^{q} \sum_{\sigma_2=1}^{q} \sum_{\sigma_1=1}^{q} (1 - \delta_{\sigma_1 \sigma_2})(1 - \delta_{\sigma_1 \sigma_3})(1 - \delta_{\sigma_2 \sigma_3}) \tag{9}$$

of a triangular (complete) graph comprised of $N = 3$ vertices and $M = 3$ edges. We start at vertex $i = 1$ by expanding the relevant product

$$p_1 = (1 - \delta_{\sigma_1 \sigma_2})(1 - \delta_{\sigma_1 \sigma_3}) \tag{10}$$

$$= 1 - \delta_{\sigma_1 \sigma_2} - \delta_{\sigma_1 \sigma_3} + \delta_{\sigma_1 \sigma_2} \delta_{\sigma_1 \sigma_3} \tag{11}$$

that is comprised of all factors that contain σ_1. (We note that Birkhoff [1] in 1912 already used closely related expansions to theoretically derive an alternative representation of chromatic polynomials.) Symbolically applying the above replacement rules (8) yields a 'partial partition function'

$$z_1 = \sum_{\sigma_1=1}^{q} p_1 \tag{12}$$

$$= q - 2 + \delta_{\sigma_2 \sigma_3}, \tag{13}$$

and thus $P(G, q) = \sum_{\sigma_3=1}^{q} \sum_{\sigma_2=1}^{q} z_1 (1 - \delta_{\sigma_2 \sigma_3})$. Proceeding with the vertices $i = 2$ and $i = 3$ in a similar fashion, we obtain $p_2 = (q - 2 + \delta_{\sigma_2 \sigma_3})(1 - \delta_{\sigma_2 \sigma_3})$, $z_2 = \sum_{\sigma_2=1}^{q} p_2 = (q-1)(q-2)$, $p_3 = z_2$, and reduce the chromatic polynomial to the final result

$$P(G, q) = z_3 = \sum_{\sigma_3=1}^{q} p_3 = q(q-1)(q-2), \tag{14}$$

successively.

For a general graph G on N vertices, the algorithm is analogous to the example. First define $z_0 = 1$. Then, passing through the vertices from $i = 1$ sequentially up to $i = N$,

1. construct and expand $p_i = z_{i-1} \prod_{\{i,j\} \in E} (1 - \delta_{\sigma_i \sigma_j})$ where the product is over all edges incident to i that have not been considered before, i.e. $j > i$;
2. *symbolically* evaluate the sum $z_i = \sum_{\sigma_i} p_i$ applying the simple rules (5)–(8) given above.

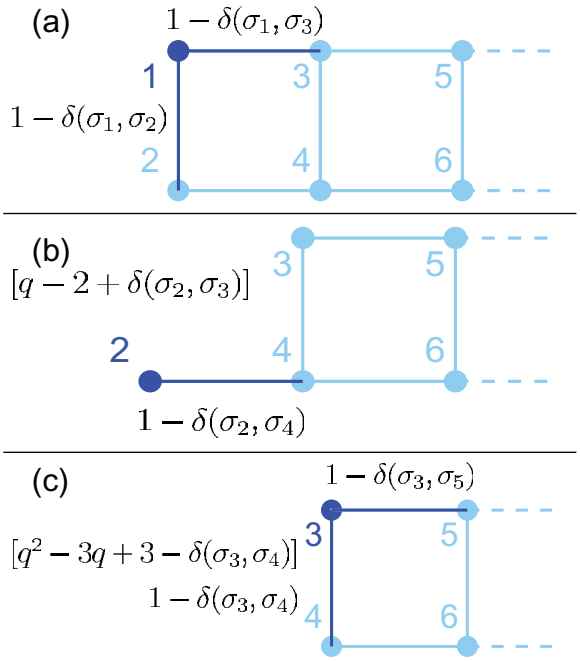


Figure 1. The bottom-up, local and vertex-oriented nature of the algorithm. Sequential processing of edges (dark) adjacent to vertices (a) $i = 1$, (b) $i = 2$, and (c) $i = 3$. The remainder graph (light) is not affected when processing vertex i. Partial partition functions z_{i-1} computed so far (before each vertex step i) are shown in square brackets.

These operations are local and vertex oriented in the sense that they jointly consider all edges $\{i, j\}$ incident to an individual vertex i at any one time. A major advantage of this bottom-up algorithm is that all edges that are not currently processed are kept outside the computations until they are needed, quite in contrast to standard top-down deletion–contraction algorithms. If a graph G has a layered structure,

$$G = \bigcup_{\nu} H_\nu \tag{15}$$

with layers H_ν, constituting samples of periodic lattices or aperiodic graphs, the vertices i are selected (i.e. numbered) layer by layer such that the operations only affect a particularly small portion of the graph at once (figure 1). These graphs have bounded tree widths, cf [30, 31]. More generally, vertices are numbered appropriately beforehand, for instance, using minimal bandwidth of the graph as a heuristic criterion [32].

The computation of the chromatic polynomial has been reduced to a process of alternating expansion of expressions and symbolically replacing terms in an appropriate order. In the language of computer science, these operations are represented as the expanding, matching, and sorting of patterns, making the algorithm suitable for computer algebra programs optimized for pattern matching.

To exploit fully the capabilities of this algorithm, we implemented it using the language Form [33, 34] which is specialized to large-scale symbolic manipulation problems and as such a successful standard tool for, e.g., Feynman diagram evaluation in precision high-energy physics [35, 36].

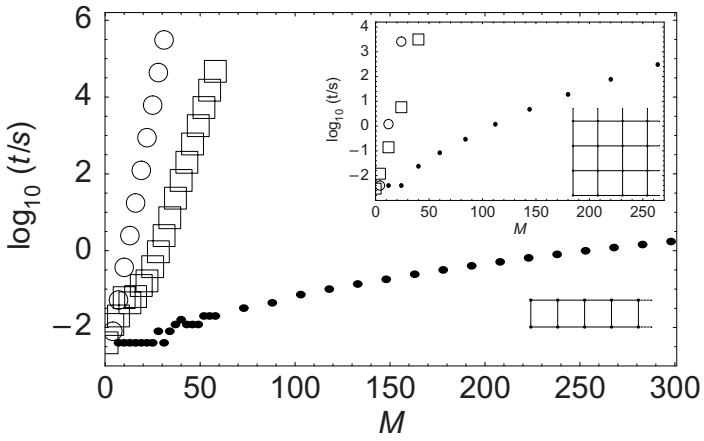


Figure 2. Computation time t (in s) for square lattice samples of sizes $2 \times n$ (main panel) and $n \times n$ (inset) increases with the number of edges M of the graph. The new method (●) drastically outperforms standard methods used in Mathematica (○) and Maple (□), both with respect to the scaling of the algorithm (quantified by the local slope) and the absolute effectiveness (quantified by the absolute times needed), even for moderately sized graphs.

A practically relevant measure for the speed of our method is the total CPU time t it needs for a specific calculation. For hard counting problems one generally expects an exponential increase $t \approx A \exp(\alpha m)$ with the size $m \gg 1$ of the problem, here defined as the number $m = M$ of edges for chromatic polynomials of general graphs. For graphs with bounded tree widths [30, 31], the solution time of the counting problem typically only grows exponentially with the width of the graph, i.e. in the square of the number of vertices in the subgraphs H_ν. The factor α in the exponent determines the scaling of the computation time with problem size and measures the efficiency of the algorithm, whereas the prefactor A fixes the absolute time needed and depends, among other things, on the software environment and hardware used.

To compare our method to existing ones, we first computed chromatic polynomials of samples of the two-dimensional square lattice with free boundary conditions ($2 \times n$ strips that have $M = 3n - 2$ edges and $n \times n$ patches that have $M = 2n(n-1)$ edges). The total computation times t have been measured as a function of M for the new method as well as for the standard methods used in Mathematica [22, 23] and Maple [25], respectively. Figure 2 shows that the scaling α of the algorithm (given by the local slope of the data points in the logarithmic plot) of our new method is markedly better than the ones found for the standard deletion–contraction methods. This implies that the new method outperforms these standard computational methods in the absolute computation time by several orders of magnitude already for moderately sized graphs. With increasing graph size, the advantages of our method become more pronounced. For example, for the 2×100 strip of the square lattice ($M = 298$ edges), the pattern matching method needs a computation time of the order of $t \approx 2$ s whereas extrapolation of the data shown in figure 2 indicates that the same problem is not computationally accessible using the standard deletion–contraction methods.

Moreover, in contrast to transfer matrix or other, analytical recurrence methods [12, 13, 15], our bottom up method also works in a simple way for graphs with non-identical subgraphs H_ν, such as randomly diluted lattices. The same comparison as above for randomly diluted

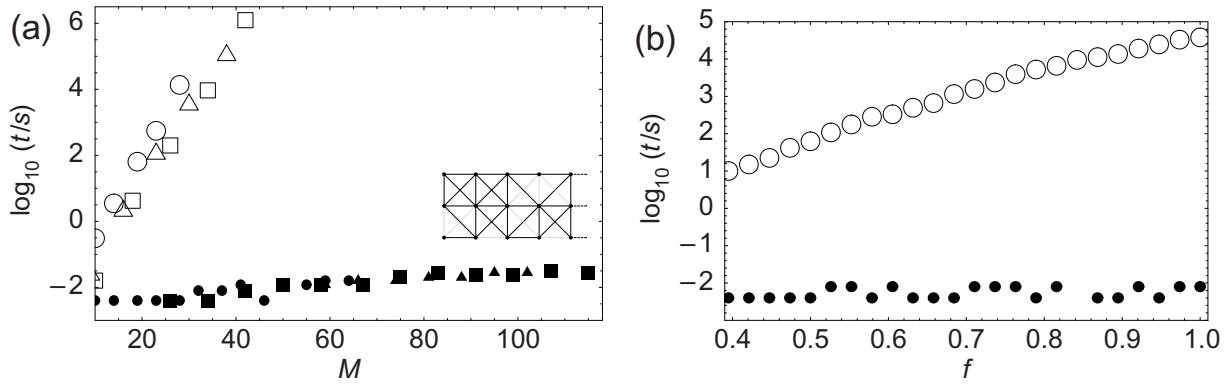


Figure 3. Computation time t (in s) for randomly edge-diluted square lattice samples with next-nearest neighbour interactions (displayed as a cartoon inset in panel (a)). The new method (filled symbols) strongly outperforms standard deletion-contraction method (Mathematica, open symbols). (a) Computation time versus the actual total number M of edges for samples of $3 \times n$ vertices where each original edge has been deleted independently with probability $p = 0.1$ (circles) $p = 0.2$ (triangles) and $p = 0.5$ (squares). (b) Computation time versus the fraction f of edges present. Edges are sequentially randomly removed independently, starting from an undiluted 3×5 square lattice graph ($f = 1$) with next-nearest neighbour interactions; the graph is diluted until before it becomes disconnected below $f \approx 0.4$. Whereas the computation times using the new method are still at fluctuation level ($t < 10^{-2}$ s), standard methods take at factor of 10^3–10^7 longer.

$3 \times n$ square lattice samples with next-nearest neighbor interactions (figure 3) confirms the pronounced outperformance and moreover illustrates the general applicability of our method, also in comparison with recursive analytical methods.

As a first application to an open hard counting problem in physics, we now turn to three-dimensional lattices of direct physical relevance. First, we consider $n \times n \times n$ samples of the simple cubic lattice with free boundary conditions, which have $N = n^3$ vertices and $M = 3n^2(n-1)$ edges. We found chromatic polynomials up to $n = 4$ ($N = 64$, $M = 144$). A representation of the chromatic polynomial $P(G, q)$ in terms of its N complex zeroes $q_1, \ldots, q_N$ [37] is shown in figure 4(a) for $n = 3$ and 4. We further consider simple cubic lattice strips that extend in the *diagonal (111)* direction with periodic boundaries in the two other (transverse) directions. This keeps the number of vertices within one layer low, at the same time allowing for a large number N_c of vertices with the same degree (equal to six) as vertices in the infinite lattice, a fact that is heuristically known to be essential for a rapid convergence towards the thermodynamic limit ($N, M \to \infty$). The largest three-dimensional sample graphs shown in figure 4(b) have $N = 384$ vertices, $M = 1128$ edges and a fraction $N_c/N = 368/384 \approx 0.96$ of vertices with correct degree (as compared to $N_c/N = 8/64 \approx 0.13$ for the $4 \times 4 \times 4$ sample extending along the Cartesian axes and to $N_c = 0$ for previous attempts to address three-dimensional lattices). The computation time was approximately 11 h on a single Linux machine with an Intel Pentium 4, 2.8 GHz-32 bit processor.

In summary, we have presented a novel method to calculate chromatic polynomials of graphs. Using the partition function representation, it proceeds vertex by vertex employing an

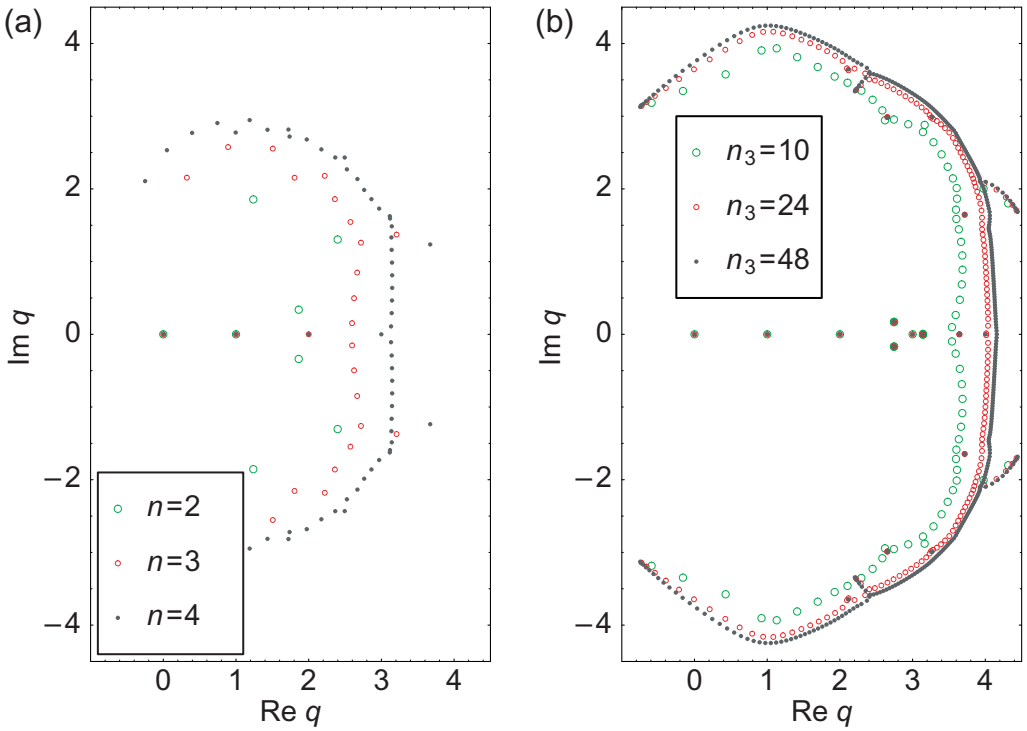


Figure 4. Complex zeroes representing chromatic polynomials of samples of the simple cubic lattice: (a) $n \times n \times n$ Cartesian samples with free boundary conditions and (b) $2 \times 4 \times n_3$ diagonal samples with periodic transverse boundary conditions, with up to $M = 1128$ edges.

a priori reduction to local operations only and is thus particularly suited for graphs exhibiting a layered structure. The method combines a symbolic bottom-up algorithm, which is based on systematic term-wise expansion and pattern matching, with an appropriate computer algebra program [33, 34]. Our method is applicable to general types of graphs, including graphs with bounded and unbounded tree widths as well as randomized graphs. We demonstrated by several sets of examples that it drastically outperforms existing standard methods for all these types of graphs. As a practical application, we computed chromatic polynomials for samples of the simple cubic lattice, for the first time computationally accessing three-dimensional lattices of physical relevance.

Since the main ideas underlying our method are simple to apply, they may be generalized in a straightforward way and also be transferred to other challenging counting problems. Among others, one may compute quantitative measures relevant in computer science that give information about the communication capabilities of a network, such as (i) the flow polynomial and (ii) the reliability polynomial [41, 42]. It is equally possible to determine (iii) ferromagnetic and (iv) positive temperature partition functions of statistical physics [38]–[40] and, equivalently, (v) the Tutte polynomial (of two variables q and v, cf equation (2)), valuations of which directly result in the number of spanning subgraphs, the number of spanning trees, and other invariants of a graph [12, 13]. Of course, applications in graph theory may include studies of families of graphs where computational results seemed impossible so far, because the computational effort is substantially reduced. As the new method

yields exact results not only for the final solution (in our examples, the chromatic polynomial) but also in the intermediate steps (the partial partition functions above), it may moreover be combined with analytical tools [11], [15]–[17], [38, 40, 42] to obtain unprecedented results for various classes of graphs. Finally, the method can easily be implemented in parallel computations. Taken together, the novel bottom-up pattern matching algorithm combined with specialized computer algebra presented here constitutes a promising starting point to access a number of challenging, computationally hard counting problems from statistical physics, graph theory and computer science.

Acknowledgments

MT thanks R Shrock for introducing him to the subject. We thank C Ehrlich, T Geisel, L Hufnagel, D Kozen, H Schanz, R Shrock, A Sokal, J Vermaseren and M Weigt for helpful comments. This work was supported by the Max Planck Society via a grant to MT, by the German Ministry for Education and Research (BMBF), grant number 01GQ0430, and by the Max Planck Advisory Committee for Electronic Data Processing (BAR).

References

[1] Birkhoff G D A 1912 Determinant formula for the number of ways of coloring a map *Ann. Math.* **14** 42
[2] Read R C 1988 *Selected Topics in Graph Theory* vol 3 *Chromatic Polynomials* ed L W Beineke and R J Wilson (London: Academic)
[3] Potts R B 1952 Some generalized order-disorder transformations *Proc. Camb. Phil. Soc.* **48** 106
[4] Wu F Y 1982 The Potts model *Rev. Mod. Phys.* **54** 235
Wu F Y 1983 The Potts model *Rev. Mod. Phys.* **55** 315 (erratum)
[5] Pauling L 1960 *The Nature of the Chemical Bond* (Ithaca, NY: Cornell University Press)
[6] Parsonage N G and Staveley L A K 1978 *Disorder in Crystals* (Oxford: Oxford University Press)
[7] Ramirez A P, Espinosa G P and Cooper A S 1990 Strong frustration and dilution-enhanced order in a quasi-2D spin glass *Phys. Rev. Lett.* **64** 2070
[8] Broholm C, Aeppli G, Espinosa G P and Cooper A S 1990 Antiferromagnetic fluctuations and short-range order in a Kagomé lattice *Phys. Rev. Lett.* **65** 3173
[9] Wills A S, Harrison A, Mentink S A M, Mason T E and Tun Z 1998 Magnetic correlations in deuteronium jarosite, a model $S = 5/2$ Kagomé antiferromagnet *Europhys. Lett.* **42** 325
[10] Fennell T, Bramwell S T, McMorrow D F, Manuel P and Wildes A R 2007 Pinch points and Kasteleyn transitions in kagome ice *Nat. Phys.* **3** 566
[11] Salas J and Shrock R 2001 Exact $T = 0$ partition functions for Potts antiferromagnets on sections of the simple cubic lattice *Phys. Rev.* E **64** 011111
[12] Sokal A D 2000 Chromatic polynomials, Potts models and all that *Physica* A **279** 324
[13] Sokal A D 2001 Bounds on the complex zeros of (di)chromatic polynomials and Potts-model partition functions *Comb. Probab. Comput.* **10** 41
[14] Biggs N 2001 A matrix method for chromatic polynomials *J. Combin. Theory* B **82** 19
[15] Salas J and Sokal A D 2001 Transfer matrices and partition-function zeros for antiferromagnetic Potts models. I. General theory and square-lattice chromatic polynomials *J. Stat. Phys.* **104** 609
[16] Jacobsen J L and Salas J 2001 Transfer matrices and partition-function zeros for antiferromagnetic Potts models. II. Extended results for square-lattice chromatic polynomial *J. Stat. Phys.* **104** 701
[17] Rocek M, Shrock R and Tsai S-H 1998 Chromatic polynomials for families of strip graphs and their asymptotic limits *Physica* A **252** 505
[18] Papadimitriou C 1994 *Computational Complexity* (Reading, MA: Addison-Wesley)

[19] Hartmann A K and Weigt M 2005 *Phase Transitions in Combinatorial Optimization Problems* (New York: Wiley-VCH)
[20] Mezard M, Parisi G and Zecchina R 2002 Analytic and algorithmic solution of random satisfiability problems *Science* **297** 812
[21] Weigt M and Hartmann A K 2000 Number of guards needed by a museum: a phase transition in vertex covering of random graphs *Phys. Rev. Lett.* **84** 6118–21
[22] 2007 MATHEMATICA Release 6 (Champaign, IL: Wolfram Research Inc.)
[23] Skiena S S 1990 *Implementing Discrete Mathematics* (Reading, MA: Addison-Wesley)
[24] Pemmaraju S V and Skiena S S 2003 *Computational Discrete Mathematics* (Cambridge, UK: Cambridge University Press)
[25] 2003 MAPLE, Release 9 (Waterloo, ON: Waterloo Maple Inc.)
[26] Nijenhuis A and Wilf H S 1975 *Combinatorial Analysis* (New York: Academic)
[27] Read R C 1987 An improved method for computing chromatic polynomials of sparse graphs *Research Report* CORR 87-20 (University of Waterloo)
[28] Ising E 1925 *Z. Phys.* **31** 253
[29] Kozen D and Timme M 2008 Indefinite summation and the Kronecker delta *Technical Report* (Cornell University) http://hdl.handle.net/1813/8352
[30] Noble S D 1998 Evaluating the Tutte polynomial for graphs of bounded tree-width *Comb. Probab. Comput.* **7** 307
[31] Andrzejak A 1998 An algorithm for the Tutte polynomials of graphs of bounded treewidth. *Discrete Math.* **190** 39
[32] West D B 2000 *Introduction to Graph Theory* (New York: Prentice Hall)
[33] 2002 FORM, *Version 3.1* Online at http://www.nikhef.nl/ ~form
[34] Vermaseren J 2000 New features of FORM arXiv:math-ph/0010025
[35] Laporta S and Remiddi E 1996 The analytical value of the electron $(g-2)$ at order α^3 in QED *Phys. Lett.* B **379** 283
[36] Misiak M and Steinhauser M 2004 Three-loop matching of the dipole operators for $b \to s\gamma$ and $b \to sg$ *Nucl. Phys.* B **683** 277–305
[37] Bini D A and Fiorentino G 2000 *Numer. Algorithms* **23** 127
1998 Numerical computation of polynomial roots: MPSOLVE 2.0 *Frisco Report of the Numerical Algorithms Group Ltd* Online at http://www.nag.co.uk/projects/frisco/frisco/node8.htm
[38] Chang S-C and Shrock R 2001 Exact $T = 0$ partition functions for Potts antiferromagnets on sections of the simple cubic lattice *Phys. Rev.* E **64** 066116
[39] Häggkvist R *et al* 2004 Computation of the Ising partition function for two-dimensional square grids *Phys. Rev.* E **69** 046104
[40] Chen C-N, Hu C-K and Wu F Y 1996 Partition function zeros of the square lattice Potts model *Phys. Rev. Lett.* **76** 169
[41] Jackson B 2003 Zeros of chromatic and flow polynomials of graphs *J. Geom.* **76** 95
[42] Chang S-C and Shrock R 2003 Reliability polynomials and their asymptotic limits for families of graphs *J. Stat. Phys.* **112** 1019

Chapter 4

Article: *Chromatic Polynomials of Random Graphs*

Frank Van Bussel, Christoph Ehrlich, Denny Fliegner, Sebastian Stolzenberg, and Marc Timme

Journal of Physics A: Mathematical and Theoretical **43** 175002 (2010)

Submitted: December 16, 2009; published: April 14, 2010

J. Phys. A: Math. Theor. **43** (2010) 175002 (12pp) doi:10.1088/1751-8113/43/17/175002

Chromatic polynomials of random graphs

Frank Van Bussel[1,2,3], **Christoph Ehrlich**[4], **Denny Fliegner**[1], **Sebastian Stolzenberg**[5] **and Marc Timme**[1,2,3]

[1] Max Planck Institute for Dynamics and Self-Organization (MPIDS), Göttingen, Germany
[2] Georg August University School of Science (GAUSS) and Faculty of Physics, University of Göttingen, Göttingen, Germany
[3] Bernstein Center for Computational Neuroscience (BCCN), Göttingen, Germany
[4] Department of Physics, Technical University of Dresden, Dresden, Germany
[5] Department of Physics, Ithaca & Weill Cornell Medical College, New York City, Cornell University, NY, USA

E-mail: fvb@nld.ds.mpg.de and timme@nld.ds.mpg.de

Received 16 December 2009, in final form 21 December 2009
Published 14 April 2010
Online at stacks.iop.org/JPhysA/43/175002

Abstract
Chromatic polynomials and related graph invariants are central objects in both graph theory and statistical physics. Computational difficulties, however, have so far restricted studies of such polynomials to graphs that were either very small, very sparse or highly structured. Recent algorithmic advances (Timme *et al* 2009 *New J. Phys.* **11** 023001) now make it possible to compute chromatic polynomials for moderately sized graphs of arbitrary structure and number of edges. Here we present chromatic polynomials of ensembles of random graphs with up to 30 vertices, over the entire range of edge density. We specifically focus on the locations of the zeros of the polynomial in the complex plane. The results indicate that the chromatic zeros of random graphs have a very consistent layout. In particular, the *crossing point*, the point at which the chromatic zeros with non-zero imaginary part approach the real axis, scales linearly with the average degree over most of the density range. While the scaling laws obtained are purely empirical, if they continue to hold in general there are significant implications: the crossing points of chromatic zeros in the thermodynamic limit separate systems with zero ground state entropy from systems with positive ground state entropy, the latter an exception to the third law of thermodynamics.

PACS numbers: 05.50+q, 02.10.Ox, 89.75.Hc

(Some figures in this article are in colour only in the electronic version)

1. Background

The chromatic polynomial $P(G, q)$ counts the number of ways one can colour the vertices of a graph G with q colours such that every two adjacent vertices are coloured differently. It was introduced by Birkhoff in 1912 [3] as a way to bring complex analysis to bear on what was then still the 4-colour conjecture. The function $P(G, q)$ is closely related to the Tutte polynomial and valuations of these polynomials provide information about many important graph invariants [18]. In recent decades chromatic polynomials, and in particular their zero sets (an equivalent representation), have become the focus of attention from physicists due to their connection to the Potts model in statistical physics [22]. For specific lattices, this has given us very detailed information about chromatic zeros [5, 14, 15]; however, there has been little progress towards understanding their location for general graphs. This is in large part due to limited computational accessibility; for all but the smallest, sparsest or most highly structured graphs, calculating the chromatic polynomial is extremely difficult because generally the computation time increases exponentially in the number of edges [21]. Since we lack instances of chromatic polynomials for 'ordinary' graphs from which to build intuition, we currently have little idea what to expect for moderately sized graphs that are not extremely sparse or highly structured. Hence, we possess no basic background for comparison of the few non-minimal instances known so far, cf e.g. [13, 14].

Random graphs constitute a natural first candidate towards building such an intuition. Since random graphs serve as initial approximations or null models of systems with unknown structure, they are ubiquitous in scientific and mathematical research and commonly used as testbeds for theories, algorithms and analytic tools. Random graph theory starts with the work of Erdös and Rényi in the late 1950s, and the standard random graph (where each edge is present independently with the same fixed probability) is therefore usually named an Erdös–Rényi (ER) random graph. ER random graphs have been shown to have some remarkably robust properties and precise invariants [9]. This applies to several problems related to chromatic polynomials, such as the value of the chromatic number of a random graph,

$$\left(\frac{1}{2} + o(1)\right) \log\left(\frac{1}{1-p}\right) \frac{N}{\log N}, \tag{1}$$

where $N \gg 1$ is the number of vertices and p is the edge probability [4]. The chromatic number is the lowest number q^* of colours required to achieve a (proper) colouring, $P(G, q^*) > 0$. However, to the best of our knowledge, the chromatic polynomial itself has not to date been subject of such investigation. Given the overlapping interests of random graph theory and statistical physics this might seem somewhat surprising, but computational accessibility has so far been highly limited.

In this paper, we start to fill this gap using a promising vertex-based, symbolic pattern matching method developed recently [17] to compute chromatic polynomials of moderately sized graphs. Like other techniques developed for physics applications (cf [7]), it is capable of fast computations on graphs consisting of periodically repeating subgraphs; however, one of the new method's important features is that it also works on arbitrary graphs, for example on samples of various random-bond-diluted lattices [17]. It performs significantly better than standard general purpose methods, allowing access to larger graphs with an arbitrary number of edges than have previously been considered, such as three-dimensional cubic lattices.

Here we compute and analyse chromatic polynomials for moderately sized ER random graphs across the entire range of edge densities for selected N. Our aim is to obtain general estimates for chromatic zero locations with respect to both mean value and variability, which can then be compared to what is known for specific lattices and other graphs. Since for decent statistics chromatic polynomials of many graphs for each given set of parameters need

to be computed, the maximum value we consider here is $N = 30$; chromatic polynomials for individual graphs with higher N are possible to compute, but resource constraints make computing a large number of realizations problematic.

As indicated in the following, the chromatic zeros of random graphs are laid out in a very regular way and systematically depend on edge density. While unanticipated, this feature is in fact entirely consistent with previous random graph results; until now, however, not enough was known for informed speculation on the subject. Based on our data, we provide a scaling law for the point at which the complex zeros approach the real line, as well as estimates for various other quantities of interest.

2. Chromatic polynomials, partition functions and their zeros

We begin with some notation: given a graph G with the vertex set $V = \{v_1, v_2, \ldots, v_N\}$ and edge set $E = \{e_1, e_2, \ldots, e_M\}$ and a set of colours $C = \{1, 2, \ldots, q\}$, we say that a *proper q colouring* of G is an assignment of values from C to the $v_i \in V$ such that no two vertices connected by an edge share the same value. G is *q-colourable* if there is a proper colouring of G using q or fewer colours; the *chromatic number* of G is then the minimum q such that G is q-colourable. The *chromatic polynomial* $P(G, q)$ is the associated counting function for proper q-colourings of G; that is, it tells us how many ways we can colour G with at most q colours.

The representation

$$P(G, q) = \sum_{\sigma_N=1}^{q} \cdots \sum_{\sigma_1=1}^{q} \prod_{(i,j)\in E} \left(1 - \delta_{\sigma_i\sigma_j}\right) \tag{2}$$

of the chromatic polynomial in terms of sums over polynomials in Kronecker deltas $\delta_{\sigma_i\sigma_j}$ seems particularly suited for computations [17] and also directly shows a link to Potts partition functions in statistical physics (see below). Here every $\boldsymbol{\sigma} = (\sigma_1, \ldots, \sigma_N)$ is an assignment of values (colours) from $\{1, 2, \ldots, q\}$ to the N vertices of G and the product runs over all edges $e = (i, j)$ of G. For a given colouring $\boldsymbol{\sigma}$, the product equals one if no two adjacent vertices have the same colour, and zero otherwise; it functions as an indicator that the assignment is a proper colouring of G. Equivalently, $\boldsymbol{\sigma}$ can be regarded as a global microscopic state of an antiferromagnetic Potts model with the individual σ_i's being local states or spin values. From this viewpoint, (2) counts the number of energy minimizing global states.

The connection to statistical physics arises from the fact that the chromatic polynomial equals the antiferromagnetic Potts partition function $Z(G, q, T)$ in the zero temperature limit since $Z(G, q, T)$ counts the number of spin configurations where all neighbouring spins disalign [22]. Let us be more specific. The standard q-state Potts model was introduced in 1952 as a generalization of Ising's 2-state model for interactions on a crystal lattice [1, 11, 12, 22]. It describes systems in which site variables (magnetic moments, spins or other kinds of local states) can take one of q different values, and interactions occur only between neighbouring sites that are in the same state. The total energy in the global state $\boldsymbol{\sigma}$ is given by the Hamiltonian

$$H(\boldsymbol{\sigma}) = -J \sum_{(i,j)\in E} \delta_{\sigma_i\sigma_j} \tag{3}$$

where J is the interaction strength and E is the edge set of the underlying graph G. The partition function for this system at positive temperature $T = (k_B\beta)^{-1}$ is

$$Z(G, q, T) = \sum_{\boldsymbol{\sigma}} \mathrm{e}^{-\beta H(\boldsymbol{\sigma})} = \sum_{\boldsymbol{\sigma}} \prod_{(i,j)\in E} \left(1 + (\mathrm{e}^{\beta J} - 1)\delta_{\sigma_i\sigma_j}\right) \tag{4}$$

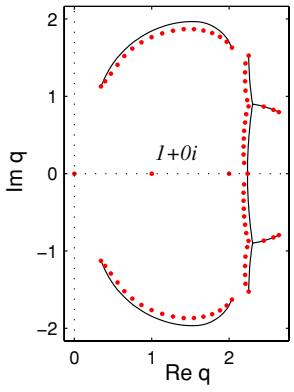


Figure 1. Chromatic roots for strip lattices. Small disks: roots for a 4×18 square lattice, free boundary conditions. Solid lines: limiting curve for a $4 \times \infty$ lattice (again free b.c.).

where k_B is the Boltzmann constant. If $J < 0$ (antiferromagnet), the $T \to 0$ limit $e^{\beta J} \to 0$ yields a partition function that equals (2).

Because of this equivalence statistical physicists have shown increasing interest in the chromatic polynomial, and in particular its zeros in the complex plane. The original Lee–Yang theorem [10] established the conditions on features of ferromagnetic systems that ensure that all zeros of the partition function have non-zero imaginary part, thereby bounding certain critical values [14]. This kind of reasoning has been extended to antiferromagnetic systems by what some authors refer to as *Lee–Yang theorems*, plural [16]. If in the $N \to \infty$ limit the complex zeros of the partition function converge to pinch the real axis, this indicates the existence of singularities in the system at these real values and so possible phase transitions or critical points. For instance, the q-state Potts antiferromagnet may exhibit a critical real $q_c > 0$ above which the zero-temperature partition function is analytic in the thermodynamic limit. Thus, for $q < q_c$, the zero-temperature phase is ordered, whereas the high-temperature phase ($T \to \infty$) is generically disordered, indicating a phase transition in temperature T between ordered and disordered states. For $q > q_c$, the system exhibits ground state disorder at $T \to 0$, an exception to the third law of thermodynamics [8].

For computational reasons, the bulk of statistical physics research on the chromatic polynomial has been conducted on strip lattices (see, e.g. [5, 13–15]): these are both sparse and have a repeating structure, and so are the most computationally tractable graphs also for moderate sizes. This structure lends itself to the use of specialized techniques, so lattices are to date the largest graphs chromatic polynomials have been computed for; in fact, for such strip graphs it is sometimes possible to calculate $N \to \infty$ limit sets of chromatic zeros analytically [5, 14]. Such lattice data represent probably our most detailed knowledge of how the entire zero sets of chromatic polynomials are laid out, and there are in fact certain qualitative consistencies in these findings; figure 1 gives an example of how the chromatic zero sets of strip lattices are laid out. Most noticeable is that, for many lattice strips, zero sets invariably trace out a somewhat elongated backwards 'C' curling around the point $1 + 0i$. Gaps and small horns are common features, showing up in characteristic locations.

3. Exploring the chromatic zeros of random graphs

To gain first insights towards chromatic zeros of random graphs, we consider the two models of the ER random graph [9]:

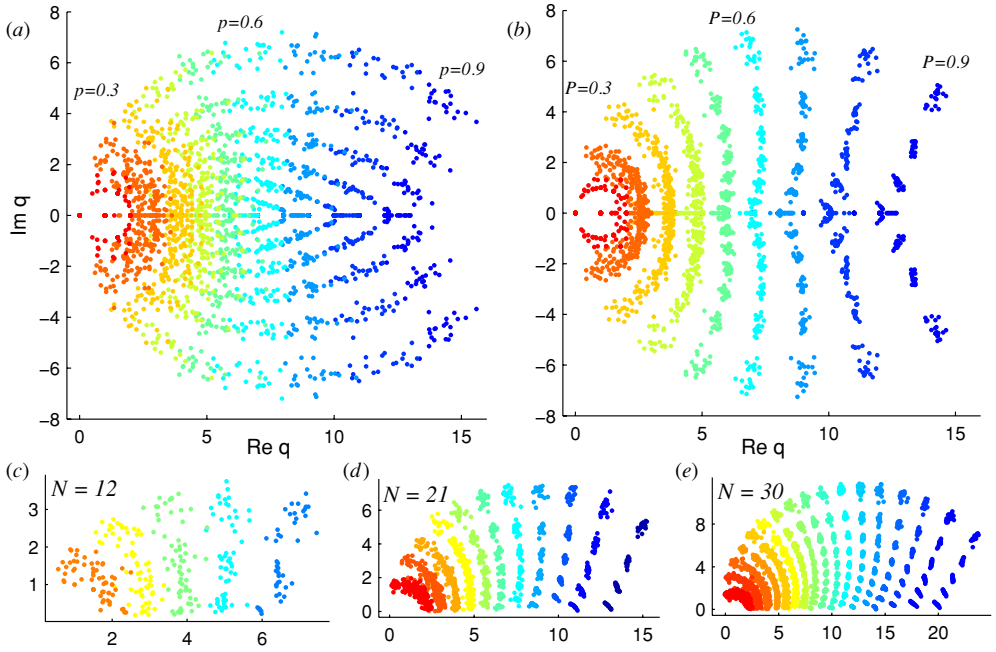


Figure 2. Complex zeros of chromatic polynomials of random graphs. (*a*) Zeros for $G(20, p)$ with selection probability $p \in \{0.1, 0.2, \ldots, 0.9\}$; (*b*) zeros for $G(20, M)$ graphs with M determined by fixed edge density $P \in \{0.1, 0.2, \ldots, 0.9\}$; (*c*), (*d*) and (*e*) zeros with positive real part for $G(N, M)$ graphs for selected N, $M = [KN/2]$ determined by the average degree $K \in \{3, 4.5, 6, \ldots\}$. In all figures, chromatic zeros for 20 sample graphs per parameter set are plotted together. Density is indicated by colour, running from red (low density) to blue (high density).

- $G(N,p)$: the graph has N vertices and between every pair of vertices an edge exists with uniform probability p. The edge density $P = |E|/\binom{N}{2}$ of any individual instance will then vary normally around p.
- $G(N,M)$: the graph is chosen uniformly from all graphs with N vertices and M edges. The density P then has the fixed value $M\big/\binom{N}{2}$.

In the $N \to \infty$ limit, the two models are largely equivalent; for finite N they can give different results, but the main reasons to use one or the other are practical. From the outset, we had no clear idea of what to expect, so we began with a preliminary investigation of $G(N,p)$ graphs restricted to $N = 20$. For each of the parameter values $p \in \{0.1, 0.2, \ldots, 0.9\}$, we generated 20 sample graphs; we then calculated the chromatic polynomials with a FORM [19] implementation of the new algorithm [17] and solved for the zeros of each polynomial numerically using MPSOLVE [2]. Complex zeros of all graphs plotted together are shown in figure 2(*a*). When first viewing these results, we were struck by the distinct banding in both the horizontal and vertical directions, which implied that the complex zero sets could have a very predictable layout depending on the basic graph parameters. After confirming that similar patterns did not arise from zero sets of random polynomials with similar coefficients, we proceeded to the second phase of our investigation, which was designed to measure the simultaneous effect of both edge-density and the number of vertices on the zero locations in a more systematic way.

3.1. Parameters and constraints

In these more extensive investigations, we consider the $G(N,M)$ instead of the $G(N,p)$ model. A cursory examination of the $G(20, p)$ data showed that the positions of chromatic zeros with non-zero imaginary part are very sensitive to the exact value of P. For example, within each p group we found the correlation between the mean real value of these complex zeros, and the exact P value (as measured from counting the edges in the generated graphs) was between 0.982 and 0.994. Since we were interested in the effect of density on zero location we then decided to directly control for P. Figure 2(*b*) shows zero sets for fixed-density $G(N,M)$ graphs.

Also, we parameterized density by average degree $K = 2M/N$ rather than directly by P, or by M. For a single value of N, it would not matter which of these was used since they can be simply converted one to another, but over different N the choice of metric affects the amount of data generated and the ways one can aggregate it. A deciding factor was that K arises naturally in the study of physical systems which feature a bounded number of interactions.

Finally, we added the constraint that graphs were to be 2-connected: that is, every pair of vertices in the graph belong to at least one cycle. ER random graphs experience a sharp connectivity transition as density is increased [9]. Though the theoretical result is in the large N limit, the effect is manifest even for N in the range of the random graphs we used; none of our $G(20, 0.1)$ graphs were even 1-connected, while all of the $G(20, p)$ graphs for $p \geqslant 0.3$ were at least 2-connected. Since the chromatic polynomial of an unconnected graph can be factorized into the chromatic polynomials of its connected components, aggregating data from graphs without controlling connectivity would have had the annoying effect of confounding the low-density data for runs with different values of N, without otherwise telling us anything we do not already know.

We used $N \in \{12, 15, 18, 21, 24, 27, 30\}$ and $K \in \{3, 4.5, 6, 7.5, \ldots, d\}$, where d is highest value divisible by 1.5 that is less than $N - 2$. For each of the 77 (N, K) pairs, 20 instances of $G(N,M)$ graphs were generated using the integer M that was closest to $KN/2$. As above, chromatic polynomials were then calculated with the new algorithm and zeros extracted using MPSolve. Figures 2(*c*)–(*e*) show all chromatic zeros in the upper half complex plane for some N. Note that the upper value and spacing of the N values chosen were primarily determined by computational concerns; for low N chromatic polynomials for all graphs could be calculated in a matter of seconds, but in the high range, a single graph could require two or more days. In total 1540 random graphs were generated, resulting in 35 700 individual data points (chromatic zeros).

3.2. Analysis of specific features of chromatic zeros

From the combined data, we extracted various graph invariants directly, such as the chromatic number, number of real and integer zeros, zero multiplicities, the maximum modulus, maximum real zero and maximum real and imaginary parts of zeros. Of particular interest to us were the shapes traced out by the complex chromatic zero set (i.e. zeros with non-zero imaginary part) in the complex plane and the point at which this set approaches closest to the real axis. The second of these we will denote as the *crossing point* $X_{N,r}$ for the parameter r (which can be average degree K or density P as required).

The crossing point is a finite system analogue to the critical value q_c discussed by previous authors [5, 13, 15], which properly speaking can arise only in the $N \to \infty$ limit when critical phenomena arise, and the zero-set becomes continuous such that its intersection with the real axis can be solved explicitly. Since our graphs (and hence their chromatic zero sets) are finite, the crossing points $X_{N,r}$ were instead estimated in the following way.

- For each parameter pair (N, K), we collected the chromatic zeros with positive imaginary value for all 20 graphs into one set (zeros with negative imaginary value are their complex conjugates, and can thus be neglected).
- We fit an arc segment to each collected zero set in the complex plane; moreover, we did a linear fit to determine orientation (left or right leaning). All fits used a least-squares Euclidean distance from each zero to the nearest point on the arc (or line) as a metric. It should be noted here that centres and radii of arcs were not constrained in any way (for example, to lie on the real axis).
- The crossing point $X_{N,K}$ is then given by the positive intersection of the arc and the real axis. Also, the curvatures of the zero sets are obtained from the arc radii, and their orientations from the slopes of the linear fits.

4. Scaling of chromatic zeros

Preliminary visual inspection of the complex chromatic zeros as shown in figure 2 revealed several consistent features. For all N tested, the zero sets at low densities are close to the origin and laid out almost circularly around the point $1 + 0i$; as the density is increased, they gradually move away from the origin and uncurl until they are almost vertical, after which the half-sets in the lower and upper complex plane remained relatively straight, but become shorter and angled, now away from the origin.

Perhaps most surprising is the way that zero sets for graphs with the same average degree K almost always fell into roughly the same location regardless of the N value. Figure 3(a) shows complex zeros for $K = 9$. We note that the curvatures of these zero sets do vary, and that those of the smallest (hence densest) graphs pull to the right somewhat. Thus, it would seem that while location and therefore crossing point is largely determined by average degree, features like the shape and height of the sets will depend on P and N in a more complex way.

The zero locations (see figure 3(b)) suggest that for random graphs with edge-density below a certain cutoff P_c the crossing point $X_{N,K}$ scales linearly with K but is otherwise insensitive to changes of N. As we will show below, it also turns out that $X_{N,K}$ has a predictable location for densities above P_c; in this range, however, it is no longer a linear function of K but a still relatively simple function of N and P. We remark that for low K, there is somewhat more variability in the data than elsewhere.

4.1. Determining cutoff density

By most measures the cutoff density P_c is somewhere between 0.6 and 0.7. Though by no means conclusive, the best empirical metrics we have point to a value of $P_c \approx 0.65$. The most dispositive evidence involves the shape and orientation of the zero set. As we have noted, for a given N, as K (or P) increases the shape of the zero set consistently goes from an arc curving around the origin to a pair of roughly straight lines leaning away from the origin. Our working assumption is thus that the cutoff should occur at the value of K (or P) where the zero set is most vertical, simultaneously consistent with either a very large arc curling left or a straight line leaning imperceptibly right. This can be made more precise by considering the slopes of linear fits to the complex chromatic zero sets. For our data the slopes are negative for all zero sets with $P \leqslant 0.648$, and positive for all zero sets with $P \geqslant 0.652$ (no zero sets were associated with a density between these values).

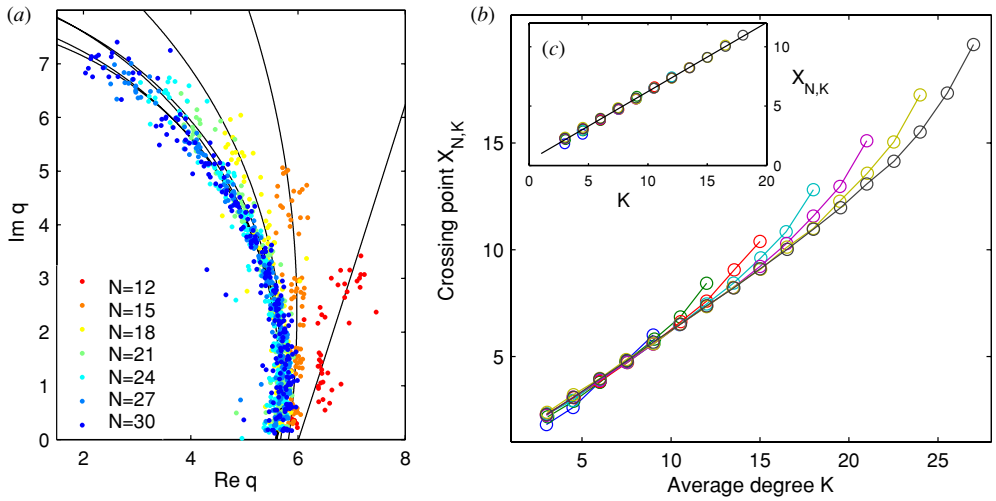


Figure 3. Crossing points for chromatic zero sets. (*a*) Complex chromatic zeros in upper half plane plus arc fits for average degree $K = 9$, various N. (*b*) Crossing point $X_{N,K}$ versus K for each N, K. (*c*) (Inset) Crossing points versus average degree restricted to graphs with density $P < 0.65$ with fit line $X_{N,K} = 0.58K + 0.41$.

4.2. Crossing point for $P < P_c$

Crossing points for $P < P_c$ exhibit a close to linear scaling with the average degree, cf figure 3(*c*). Fitting $X_{N,K}$ against K below the cutoff gives the relation as

$$X_{N,K} = 0.582K + 0.406. \tag{5}$$

Goodness of fit tests confirm that regardless of possible higher order terms the data points are aligned very closely to the fit line. The reduced chi-squared χ^2_r for the fit is a quite low 0.295 (χ^2_r is the standard χ^2 normalized by the degrees of freedom; a value of one or less indicates that the model accounts for the data adequately or better, while values significantly greater than one mean the fit is of poor quality).

We note that the variability in measured data is greater for low K values than for high. This is in part due to the fact that we have more low K data points than high ones, and that the variability of zero location is definitely greater at low densities (noticeable from simple visual inspection of zero plots). But there seems to be another factor at work as well. For individual N values, the differences between the calculated $X_{N,K}$ values and the fit are not scattered about the line, but instead are larger (and consistently positive) towards both ends of the applicable density range. Also, the crossing point for $K = 2$ (a cycle) should be at $\mathrm{Re}(q) = 2$, while the fit line predicts a too-low value of 1.57. This may imply that actual curves are either *very* gently rounded, or that there is a separate low density regime where a different slope applies. We do not currently have any good criterion for distinguishing a second cutoff, however. Moreover, the magnitude of these differences is small, and does not seem to increase with larger N.

4.3. Crossing point for $P > P_c$

The crossing points above the cutoff density do not scale directly with K; in fact, it is not clear from visual inspection that they conform to any simple function of K or P. Hence, it

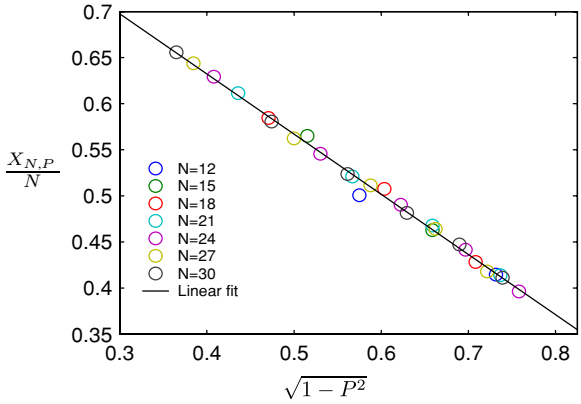


Figure 4. Rescaled crossing points $X_{N,P}/N$ versus $\sqrt{1-P^2}$, $P > P_c$.

is something of a surprise to find out that for $P > P_c$, the crossing points are rather well described by

$$X_{N,P} = N\big(a - b\sqrt{1-P^2}\big), \tag{6}$$

where $a \approx 0.9$ and $b \approx 0.65 \approx P_c$, as shown in figure 4. This relation was obtained by considering a rescaling of the crossing point values to correct for the fact that the distance from the cutoff crossing point X_{N,P_c} to N increases as roughly $0.4N$ with N. It turned out that when rescaled this way, the $X_{N,P}$ values for different N fell close to the same arc segment when plotted against P and hence the $\sqrt{1-P^2}$ term in the relation above. The exact linear fit has a χ^2_r of 0.2868.

4.4. Further quantities of interest

As mentioned above, we have considered a large number of other quantities of interest. Each of these quantities requires a thorough scaling analysis on its own, such that we here give only a brief summary of the most important results and refer the reader to an extensive presentation in a future publication. Both the maximum zero modulus $|z_{\max}|$ and the maximum imaginary value $\max(\mathrm{Im}(z))$ also turn out to have simpler scalings than were anticipated. The maximum modulus is well approximated across all densities by the linear relation

$$|z_{\max}| = 0.91K - 0.79, \tag{7}$$

though the exact relation again is probably nonlinear at least in the high and low density range; while all points are close to the fit line, the errors are not scattered normally but definitely pull up at both ends (cf figure 5). The maximum imaginary value does not grow linearly with K, but a linear relation can be obtained by rescaling:

$$\max(\mathrm{Im}(z)) = 1.15K\sqrt{1-P} - 1.67. \tag{8}$$

A χ^2_r value of 0.296 again confirms a high quality fit. Since $K = P(N-1)$, for fixed P this relation does give a linear scaling, this time in N. We note that $P\sqrt{1-P}$ achieves its maximum at $P = \frac{2}{3}$; hence, this result dovetails with our finding of the density cutoff for the crossing point scaling at $P_c \approx 0.65$, based on the most vertical layout of the chromatic zero set.

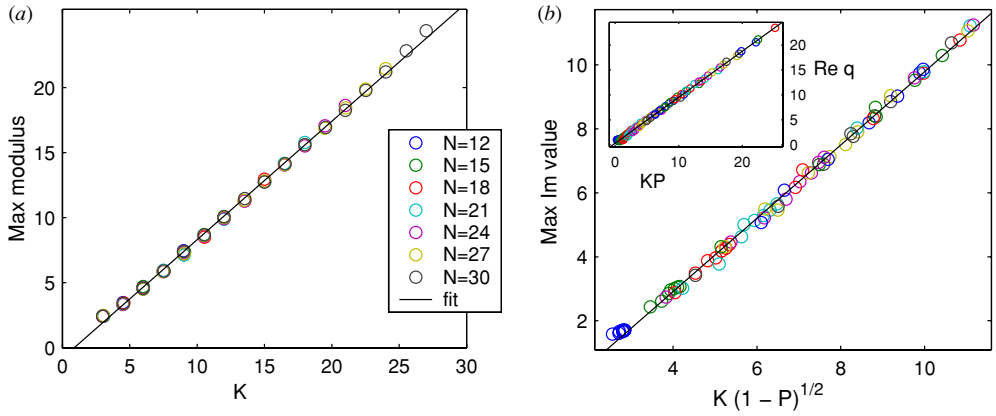


Figure 5. (*a*) Maximum modulus $|z_{\max}|$ (mean over each N, K parameter set) versus average degree K (cf. (7)). (*b*) Maximum imaginary value $\max(\mathrm{Im}(z))$ (mean over each N, K parameter set) versus $K\sqrt{1-P}$ (cf. (8)). Inset shows associated real values versus KP.

5. Conclusions

Applying the recently suggested vertex-based computational method [17] yielded some cornerstone insights into the features of chromatic polynomials of random graphs: the complex zeros of chromatic polynomials of moderately sized random graphs systematically change with the number of vertices N, the density P and the average degree K. For fixed N, the shapes of the complex zero sets evolve as P increases, from tracing an arc around the point $1 + 0i$ at low P, to being almost vertically aligned at the cutoff density $P_c \approx 0.65$, to sitting along pairs of straight lines leaning away from the origin at higher P. The crossing point $X_{N,K}$ of the complex zeros scales linearly with the average degree K for $P < P_c$; above this cutoff the crossing point appears to be a simple function of N and $\sqrt{1-P^2}$. If these results continue to hold for larger N, the implications are straightforward. Fixing the average degree K, we have $P \to 0$ as $N \to \infty$; this suggests that for an infinite random graph with bounded degree, the chromatic zero set converges to an arc around $1 + 0i$ that crosses the real axis at a finite position. On the other hand, if P is fixed or converges to a positive value, then $K \to \infty$, and the crossing point and in fact all chromatic zeros with non-zero imaginary value diverge to (complex) infinity. The former is most relevant for statistical physics: if G is an infinite disordered system with bounded connectivity, then in the zero-temperature limit the zeros of the antiferromagnetic Potts partition function meet the real axis at a finite, predictable location that depends on the system's coordination number.

Besides extending our studies to related polynomials such as Tutte or reliability polynomials [6, 18], the chromatic polynomial itself requires several future studies. In particular, there is of course much that still needs to be done with respect to Erdös–Rényi random graphs. Ideally we would like to obtain results for larger N for additional verification. However, while the new method [17] does achieve a significant speedup, the problem of computing the chromatic polynomial remains difficult and so there is a limit to feasible N, P values.

Nevertheless, even for accessible classes of graphs, several features of their chromatic polynomials are not yet settled. While our results indicate that for Erdös–Rényi (ER) random graphs, the location of chromatic zeros may be quite predictable, there are many related graphs

of interest with as yet unknown features. Preliminary studies on two other classes of graphs, regular random graphs and random bipartite graphs, suggest how far and in what direction our current results might extend. As one might expect, with respect to regular random graphs, the main effect is to reduce the variation in zero location across instances. The layout of the chromatic zero sets for random bipartite graphs, on the other hand, is markedly different than what we see for general ER random graphs, particularly at higher densities. Since many lattices are also bipartite, this bears a closer study.

Two further 'standard' classes of networks and graphs include the following.

- *Small-world graphs.* There is a possibility that our scalings may depend on properties that will almost always hold for small enough random graphs yet almost never hold for large enough ones. One such property relates to the presence of triangles and other small cycles. None of our graphs with average degree more than three were triangle free, though for densities $P < 1$, large enough random graphs will usually be triangle free, and infinite random graphs are locally indistinguishable from trees. Small-world graphs [20], which maintain their neighbourhood properties as they grow larger, are hence natural candidates for study.
- *Scale-free networks.* Some previous work on chromatic polynomials has brought up the possibility that the presence of high degree vertices may have a large influence on the location of some of the graph's chromatic zeros [16]. We have not detected a pronounced effect coming from the maximum degree itself but this might be due to the strongly central, unimodal distribution of the degrees for the random graphs we considered. Hence, it is entirely possible that the non-effect we noticed is due to our maximum degrees not being large enough compared to the average degrees to exert an independent influence. To test this supposition, a natural option would be to compute chromatic zeros of graphs with scale-free degree distributions. The vertex-based symbolic method we used [17] would need to be adapted to compute reasonable size instances.

Acknowledgments

We thank R Ruppelt for a key hint. This work was supported by the Max Planck Society via a grant to MT, by the German Ministry for Education and Research (BMBF), grant no 01GQ0430, and by the Max Planck Advisory Committee for Electronic Data Processing (BAR).

References

[1] Baxter R 1982 *Exactly Solved Models in Statistical Mechanics* (London: Academic)
[2] Bini D and Fiorentino G 2000 *Numer. Algorithms* **23** 127–73
[3] Birkhoff G 1912 *Ann. Math.* **14** 42–6
[4] Bollobás B 1988 *Combinatorica* **8** 49–55
[5] Chang S-C and Shrock R 2001 *Physica* A **290** 402–30
[6] Chang S-C and Shrock R 2003 *J. Stat. Phys.* **112** 1019–77
[7] Hartmann A and Weigt M 2005 *Phase Transitions in Combinatorial Optimization Problems* (New York: Wiley-VCH)
[8] Huang K 1987 *Statistical Mechanics* (New York: Wiley)
[9] Janson S, Łuczak T and Rucinski A 2000 *Random Graphs* (New York: Wiley)
[10] Lee T and Yang C 1952 *Phys. Rev.* **87** 410–19
[11] Onsager L 1944 *Phys. Rev.* **65** 117–49
[12] Potts R 1952 *Proc. Camb. Phil. Soc.* **48** 106
[13] Roček M, Shrock R and Tsai S-H 1998 *Physica* A **252** 505–46
[14] Salas J and Sokal A 2001 *J. Stat. Phys.* **104** 609–99

[15] Shrock R and Tsai S-H 1997 *Phys. Rev.* E **55** 5165–78
[16] Sokal A 2000 *Physica* A **279** 324–32
[17] Timme M, Van Bussel F, Fliegner D and Stolzenberg S 2009 *New J. Phys.* **11** 023001
[18] Tutte W 1954 *Can. J. Math.* **6** 80–91
[19] Vermaseren J 2000 arXiv:math-ph/0010025v2
[20] Watts D and Strogatz S 1998 *Nature* **393** 440–2
[21] Wilf H 1986 *Algorithms and Complexity* (Englewood Cliffs, NJ: Prentice-Hall)
[22] Wu F 1982 *Rev. Mod. Phys.* **54** 235–68

Chapter 5

Ongoing and Future Work

5.1 Network Reconstruction

In chapter 2 a method is proposed to recover the complete topology and connection strengths of a leaky integrate-and-fire neuron network from spike-time data. The method is robust with respect to irregular spiking dynamics and hence will work with a broad range of network topologies and mixtures of excitatory and inhibitory connections. Furthermore, it is quite fast, accurate, and does not require arbitrary amounts of raw data to work.

The main significance of the method as it now stands is: explicit reconstruction techniques for neuronal networks cannot now be ruled out due to inherent algebraic or computational intractability of the problem. We know now that reconstruction of networks of model neurons with discrete-pulse interactions on a moderately large scale is feasible, not highly vulnerable to numerical instabilities, and within the computational complexity class **P**, that is, problems with a polynomial running time.

To extend our methodology to more naturalistic situations would not involve any great conceptual leaps. While the method as given is model dependent, and requires exact, known parameter values and exact spike time data to achieve full accuracy, as we shall see below the underlying framework is quite flexible, and many of the prior conditions can be relaxed considerably.

To begin with, the article itself mentions the possibility of using different neuron models. This is an obvious next step which we do intend to follow up on, though time constraints have not permitted us to do any concrete work in this direction as of yet. I personally think that, as long as the dynamics are monotonic for individual neurons and satisfy some simple additivity properties under interaction, the method can be modified to function by way of an interpolation scheme with empirically derived numerical phase response curves as well.

Due to space constraints the article had to omit other possible extensions which would allow the method to work with more realistic data or larger networks. In the following subsections I will outline some of these that are already in the preliminary implementation stage:

- Relaxing desired precision in order to dramatically reduce data requirements;
- Dealing with noise and uncertainty in spike time data;
- Taking advantage of parallel and distributed computation possibilities.

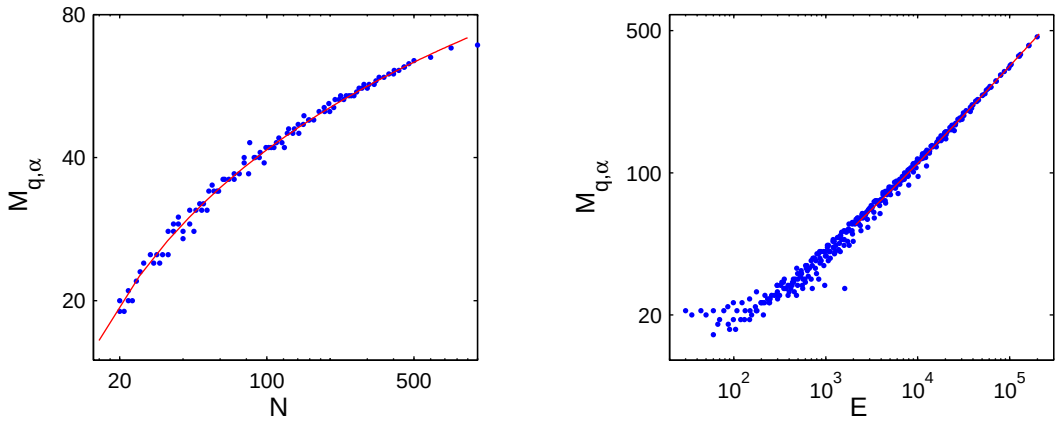


Figure 5.1: $M_{q,\alpha}$ scalings for Kuramoto networks. Left: $M_{0.98,0.95}$ vs. N (log-log) for networks with in-degree fixed at 10 (i.e. edge-density decreasing). Right: $M_{0.98,0.95}$ vs. number of connections $E \sim N^2$ (log-log) for networks with edge-density greater than 5%. Blue dots are measured $M_{q,\alpha}$ values, red lines show fits to $\log N$ and $\sqrt{E}$ respectively.

In subsection 5.1.4 I will then discuss how these various ideas can be put together to build a comprehensive computational system for the problem.

5.1.1 Logarithmic versus square-Root scaling of $M_{q,\alpha}$

A feature of both [113] and the chapter 2 is the use of the singular-value-decomposition with an L_1-norm optimization step to significantly reduce the number of different driving conditions required for an adequate reconstruction. In both articles the metric $M_{q,\alpha}$ is used to estimate the minimum number of drivings M that is needed to insure a desired accuracy, and in both a scaling is given for this quantity. An alert reader familiar with [113] may have noticed that there $M_{q,\alpha}$ grows logarithmically in the network size N (shown here in figure 5.1(left)), while in chapter 2 it scales roughly as the square-root of the number of connections E, hence linearly with N. It should be noted that for a similar application of SVD with L_1-norm minimization by Yeung et al [132] the scaling was found to be logarithmic as in [113].

What can explain these different scalings? While there are minor differences in how $M_{q,\alpha}$ is calculated in the respective papers, these are incorporated mostly to prevent false high scores at the margins of usability i.e. relatively low desired q and α; when $q, \alpha \geq 0.90$ these differences should not affect the order of the scaling. One may then conclude that the divergence in scalings results from the diverse dynamics of the continuously coupled Kuramoto system used as the testbed in [113] and the pulse-coupled leaky integrate-and-fire neurons of chapter 2. Attractive as such a notion might be, as one can see from figure 5.1(right) this is also not the case. For Kuramoto oscillator networks in the density range $P \in (0.05, 1)$ the value of $M_{q,\alpha}$ grows as $\sqrt{E}$ as well.

The problem is that, as figure 5.1(left) shows, there are too many data points supporting a logarithmic scaling with respect to at least *some* Kuramoto systems for us to just dismiss the possibility. My personal speculation was that the logarithmic scaling might apply to a low or decreasing density regime, and when serious testing was begun on the LIF reconstruction method I hoped that the new $M_{q,\alpha}$ results would confirm this.

No evidence whatsoever of a low-density regime came up in the LIF test results. Over 300,000 separate $M < N$ reconstructions were done on homogeneous inhibitory LIF networks in order to obtain $M_{q,\alpha}$ for over 600 size-density parameter pairs. Densities ranged from 0.025 on up; while not as low as the 0.01 densities tested with some of the Kuramoto networks, this should have been low enough to see at least some effect if it existed. The absence of two regimes in the new data led me to consider whether the different testing protocols used for the two projects might have had an influence on the results.

For the LIF networks the connections were given random real weights according to a Gaussian distribution; these weights were subject to some rescaling, but this went to insuring that the mini-distribution of incoming weights for each target neuron was sufficiently broad (among other things). For the Kuramoto networks the testing divided into two main sets, one on random networks with fixed in-degree 10 to generate data for [113], and a second on general random networks with various connection densities and no fixed degree. The connection weight was not random but fixed at one, then normalized so that for each node the total incoming weight was the same. When in-degree is also fixed such a scheme will result in a uniform weighting, but when in-degree is allowed to vary randomly (and the average degree is large enough) then the connection weights across the entire network will also vary in a random-like way. For additional comparison, the genetic networks that Yeung et al studied in [132] were $\{0, 1\}$ matrices i.e. connections uniformly had a weight of 1.

My revised hypothesis is that the explanation is information theoretic, and the divergent scalings seen with Kuramoto networks results from the connection weighting scheme used; the ultimate implication is that for either system if we only wanted indicators for the presence or absence of connections rather than accurate strengths the minimum M required would indeed scale logarithmically or at least less than square-root with N. There are various ways this could be tested. Since indicator reconstructions are likely to be useful when the target networks do not have fixed connection weights, we would want to test the hypothesis on randomly weighted networks, by rounding reconstructed connection weights to $\{-, 0, +\}$ i.e. detect presence or absence of arbitrarily weighted connections. This would be accompanied by a control run using fixed weight networks for comparison.

5.1.2 Reconstruction of noisy LIF networks

The reconstruction methodology as described in chapter 2 does not ostensibly deal with noisiness or variability in the data. Some preliminary work on incorporating random variation and uncertainty into our methods has already been done, with promising results. Modified versions of both the general and fast specialized method have been used to reconstruct small networks from simulation data where quantities such as pulse strength and delay time were allowed to fluctuate randomly as well as deterministic simulation data with externally added noise (spike time jitter). As one would expect, reconstructions of noisy systems never obtain the numerical limit of accuracy; but with bounded noise levels we do obtain predictably bounded error levels and so reasonably trustworthy estimates of network connectivity. Figure 5.2 shows $M < N$ reconstructions with random added noise for various Q_α values. The use of $M < N$ reconstruction for these plots is essentially diagnostic (simple checkerboards would reveal little of interest), but they also show that the SVD plus L_1-norm $M < N$ optimization can act like a numerical filter to improve accuracy (cf. [78]).

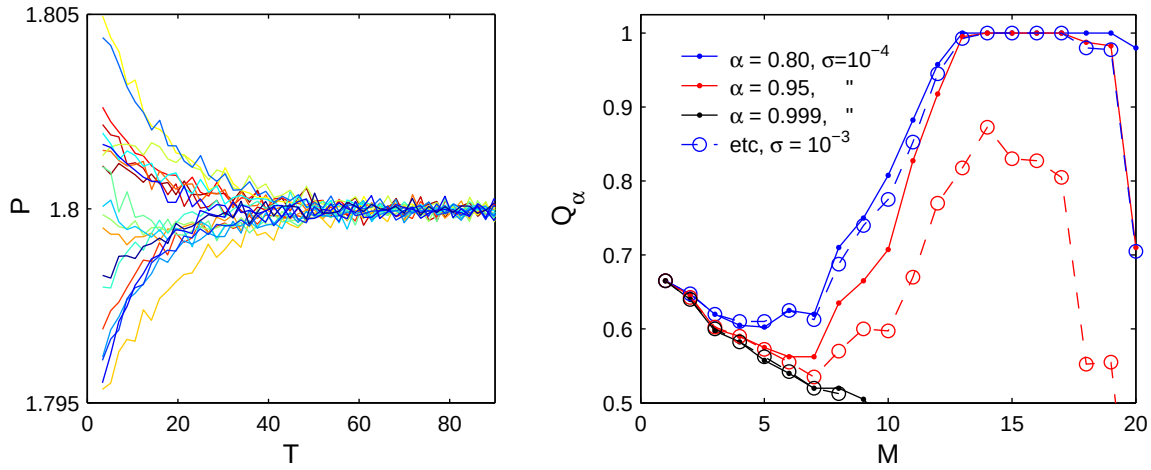


Figure 5.2: Quality of reconstruction for data from homogeneous inhibitory network (phase-locking) with externally added noise. Network: 20 neurons, random regular (6 presynaptic connections per neuron), Gaussian connection weights with mean strength -1/30 V. System: $\gamma = 0.6321\,/\mathrm{s}$, delay = 0.25s, threshold $V_T = 1\,\mathrm{V}$, reset $V_R = 0\,\mathrm{V}$, base driving $\widetilde{I} = 1.0\,\mathrm{V/s}$, driving spread $\Delta I = 1/250$. Left: time series data (period P vs. time T) for one run (all neurons), added Gaussian noise has standard deviation $\sigma = \frac{1}{10000}$ of mean period length. Right: Quality of reconstruction Q_α for various values of α and $\sigma = 10^{-4}\langle P\rangle$ and $10^{-3}\langle P\rangle$.

The general method and the fast specialized method for phase-locked simple spike patterns extend to noisy data in different ways. For the fast method it is simply a matter of extrapolating a representative spike pattern and set of period lengths from the given data, which requires removing transients from the time series and gathering appropriate statistics. The fast method with noise works well on spike trains with externally added noise, as well as data from networks that are able to remain phase-locked when there is variation in e.g. delay times or pulse strengths within the simulation. In the latter case, however, it is possible for this internal noise to perturb the network out of synchronization (often into two or more competing groups of internally phase-locked neurons), which then makes specialized reconstruction impossible.

While we have done some successful reconstructions of noisy LIF networks with an extended general method, this is still a work in progress. Incorporating noise here has issues that do not arise for the specialized method. To begin with, since spike arrivals can happen at any point in an interval, we cannot aggregate the data directly but must first incorporate it into rows of the solution system; this leads to biases in the solution, and we have yet to work out a formula that will correct for it entirely. More problematic is how spike time jitter / delay time uncertainty affects the acceptable interval searches. When exact delay/spike times are not known the criteria for an acceptable interval $(t_{i,\ell-1}, t_{i,\ell})$ must include:

$$\forall j,k: \quad t_{j,k} + \tau \neq t_{i,\ell} \pm \epsilon \tag{5.1}$$

where 2ϵ is the window of uncertainty. The larger ϵ is the more false negatives (acceptable intervals flagged as unacceptable) are generated; simple probability dictates that for a fixed ϵ increasing the network size will have the same effect. This both reduces the data available to construct a solution and makes the searches longer. As well, when $\epsilon > 0$ we now have to look out for possible arrivals near the beginning of intervals:

$$\exists j,k: \quad t_{j,k} + \tau \in (t_{i,\ell-1} - \epsilon, t_{i,\ell-1} + \epsilon)\,, \tag{5.2}$$

since of course if this is detected we do not know if the arrival occurred in the interval $(t_{i,\ell-2}, t_{i,\ell-1})$ or $(t_{i,\ell-1}, t_{i,\ell})$. This roughly doubles the number of rejections.

That said, the general method for noisy systems does achieve ballpark solutions. It is quite possible that in conjunction with a stringent rounding regimen such as described in the previous subsection it will be able to provide quite trustworthy results; this has yet to be studied.

5.1.3 Parallelization and subnetwork reconstruction

Another aspect of the general method that is not touched upon in chapter 2 are the possibilities of partitioning computations in order to make more use of available resources as well as use less time overall.

Since computations (including the search phase) of presynaptic connections for each neuron are independent the underlying algorithm is naturally parallelizable; all that would be required is generic distribution code to parcel out computations to processors and collect the results when each is done. Since spike trains are never modified these can be kept in shared memory while all searches proceed simultaneously. For an N neuron network each process would also require its own $\mathcal{O}(N^2)$ memory space in order to construct the respective $\Theta^{(i)}$ matrices and $B^{(i)}$ vectors; since these are no longer needed after the connectivity for neuron i is established, a single cluster node can process several neurons in sequence without requiring additional space.

Something that could work for us on a more fundamental level would be the ability to do reconstructions independently on subnetworks embedded into a larger network. Applicability here is not that we could compute different subnetworks simultaneously, but rather that we can avoid computing some connectivities altogether (i.e. ones we are not interested in or that we expect to be null). If a 500 neuron region of interest is embedded in a 3×10^4 assemblage we prefer not to have to do calculations for all possible connections in the greater network (almost a billion) in order to determine the status of the quarter million or so that we care about. Hence it would pay to take advantage of whatever modularity the problem possesses.

When I was working on Kuramoto system I already had begun to study the feasibility of subnetwork reconstruction using the general response-dynamics framework, though there (and with the specialized LIF method) global synchronization of the network was an issue. With the general LIF method we do not have strong constraints on how the entire network behaves, so subnetwork reconstruction easily falls into two aspects:

i) If we have reason to believe that there is no presynaptic connection from a neuron j to a neuron i then regardless of neuron j's activity it can be excluded from all searches and calculations relating to i's incoming connectivity, thereby reducing search/comparison time and the size of the linear systems constructed. More generally, if R is a large network and $Q \subset S \subset R$ with the property that no neuron in $R \backslash S$ has a potential connection to any neuron in Q, and we know all spike times for S, we can reconstruct all incoming connections for Q (internal to Q and from the boundary $S \backslash Q$ to Q) regardless of any larger dynamics playing out in R.

ii) Even if neuron j is a presynaptic neighbour of neuron i, if there are sufficiently many acceptable interspike intervals for i that do not contain any representative arrival times $t_{j,k} + \tau$ for j we can correctly solve for all the other possible presynaptic neighbours that are represented while excluding j. More generally, if $S \subset R$ and spiking in $R \backslash S$ is not

present in a large enough subset of interspike intervals we can reconstruct all connections internal to S regardless of what connections it actually has from the rest of R.

The idea then is to use both aspects to patch together connectivity graphs of large networks from full computations on smaller portions. I will note here that this is already the practice with respect to full and partial reconstructions on the individual neuron scale based on purely experimental techniques [108]. Aspect (i) is of particular relevance when areas of interest are well defined in terms of clustering metrics, with sparser connections to the rest of the network. In order to determine these areas and where the interfaces between them lie we could possibly use an intermediate scale covariance approach based on functional connectivity techniques currently in use. Aspect (ii) would come into play in conjunction with either pre-established external driving strategies (since a non-driven section of a neuronal network can be expected to spike much more infrequently than a driven section) or by explicitly suppressing output in certain areas by e.g. pharmaceutical means.

5.1.4 A comprehensive overview

The above is not an exhaustive list of possible extensions: I have not mentioned, for example,

- how assumptions about neuron specialization with respect to inhibitory or excitatory post-synaptic connections can be leveraged to reduce the solution space,
- how constraint satisfaction methods from optimization theory can be used to obtain bounds on connection strength when quiescence or consistent spike-induced spiking is present, or
- how external driving strategies could be utilized to promote neural behaviour best suited for reconstruction.

Our methodology going forward. It is important to note that all the above extensions (from the previous subsections as well as the previous paragraph) are fully compatible with each other. Furthermore, all the involved issues (requirements in terms of raw data points per recovered connection, network size and partitionability, noise, spiking regimes, etc.) would have to be addressed by any practical method for reconstructing biological neuron networks. What I ultimately envision for the reconstruction methodology then is that it will combine all the above extensions into a *general augmented method*:

i) A framework that supports a repertoire of theoretical and/or numerical models by switching in/out, rather than one that is model independent or uses a single model general enough to apply to all spiking regimes and all structural/functional classes of neurons.

ii) An algebraic kernel that uses rounding/bounding to obtain desired basic information about presence/absence/sign of connections, so that substantially less raw data is required and processed data discarded.

iii) Search, aggregation, and preprocessing routines designed to deal with noise, uncertainty, and variability in the spike time data.

This would be combined with targeted driving strategies during experiments, and a prior analysis stage undertaken to determine neuron models / parameter values, and to isolate the sections of the network likely to contain a significant number of connections.

The neuronal reconstruction problem. Nothing in the general augmented method outlined above depends on any specific developments in experimentation, only on sufficiently reliable spike time data being available in sufficient quantities. Hence if the augmented method can be brought to completion it could easily become the standard technique for detailed neuronal network reconstruction, considering the current state of the art with respect to this and the general oscillator network problem [70, 113, 133, 134].

The applications of detailed neuronal connectivity tables are many [26]. The one that I am personally most interested in is the search for neuronal "gadgets", as mentioned in section 1.2.1 of the introduction. The eventual aim of course is to determine the mechanisms for such things as memory lookup or formal rule-following [18]. The first step in this direction would be to search for small motifs that reappear significantly more often then would be implied by random or lattice-like (i.e. neighbourhood) connectivity, since the presence of small repeating substructures might signify some specific neural function e.g. activate or deactivate a neighbouring cluster. Such motif research is already underway [26, 107, 108], utilizing connectivity data laboriously obtained over the last two decades via whole-cell recordings and similar techniques; the size of the networks they have at their disposal for analysis range from less than 100 neurons for mammalian data to 197 neurons for the well-studied *c. elegans*. For computational reasons motif sizes considered so far are small (up to 5 neurons), but results already indicate that some do show up more often than expected [108].

Work in this vein is only just starting, but if a motif was found that was a plausible candidate for e.g. a neuronal router, then the obvious next step would be to determine whether there was any stereotypical structural/topological relation between separate instances of such a motif occurring in the same network. If a defined repeating pattern of motifs were found then this would require analysis as a possible mechanism for a higher order function, and so on [87]. Any such research would be greatly aided by the presence of more and larger detailed neuronal connectivity tables.

5.2 Chromatic Polynomials

In chapter 3 a new algorithm for computing chromatic polynomials of graphs based on the antiferromagnetic Potts partition function is presented. The algorithm makes use of symbolic pattern matching to evaluate algebraic operators, so is well suited to highly optimized computer algebra systems such as J. Vermaseren's FORM [121]. While designed to calculate strip lattices commonly studied in statistical physics, it is capable of computing chromatic polynomials for arbitrary graphs and is much faster than generic chromatic polynomials solvers such as those provided with MATHEMATICA or MAPLE. Chromatic root sets for various lattices of physical interest are shown, including the unprecedented $4 \times 4 \times 4$ cubic lattice. In chapter 4 the new algorithm is used to compute chromatic polynomials of Erdös-Rényi random graphs, which have not to this point received any significant attention in this regard. It is found that the root sets of edge-density parameterized ER random graphs are laid out in a very regular and predictable fashion, something that could not be anticipated from the prior literature on chromatic polynomials but is consistent with previous random graph results for other invariants such as chromatic number or clique size. While empirical, the results imply that in the infinite

size limit chromatic root sets of random graphs with fixed average degree K will cross the real axis at a consistent (finite) location: $\approx 0.6K + 0.4$.

As with the neuronal network research there are future directions for which some preliminary work or planning has already been done. In the following subsections I will discuss:

- Ongoing lattice calculations, including those for crystals other than the simple cubic lattice and a limiting curve technique developed within our research group for infinite lattice strips;
- Chromatic polynomials of random graphs generated under various structural constraints (different degree distributions, bipartness, etc.);
- Extension of the new algorithm to methods for related functions such as the Tutte polynomial and the Reliability polynomial.

5.2.1 Lattice calculations

By design the new algorithm presented in chapter 3 is natively optimized for chromatic polynomial calculations on lattice strips similar to those studied by Shrock [28] or Sokal [96]. On square lattices with fixed width the running time is linear in the length of the strip (though running time will grow exponentially with the width if it is allowed to vary). This allows long, thin strips to be calculated quite quickly.

The feature that sets this algorithm apart from competing techniques, such as transfer matrices, is that it does not depend on repeating substructures at all, but only on the width (for a discussion of graph theoretical generalizations of the width metric see appendix C.5.2). For this reason not only are ordinary square lattices feasible, but also e.g. bond-diluted lattices or other kinds of graphs where layers are not identical; as well, the algorithm can be run "as is" on any kind of repeating structure without extensive preprocessing, as long as the vertices are ordered from one end of the strip to the other. Other fast specialized techniques, such as Shrock's method of generating functions, require extensive (by hand) recalculations of bases when a new kind of lattice is attempted.

One thing this has made possible is work on 3-dimensional lattices. The signature calculation of chapter 3 was the first ever computation of the chromatic polynomial for the $4 \times 4 \times 4$ simple cubic lattice (free boundary conditions). For my own part I have continued calculations on the $4 \times 4 \times 4$ simple cubic lattices with periodic boundary conditions. As figure 5.3(left) shows, so far the computations have made it to 2 periodic boundaries, with each new periodic boundary having a non-trivial effect on the real axis crossing point of the root set. As of this writing a calculation for the cubic lattice with all 3 boundaries periodic is ongoing, but the addition of each periodic boundary condition has resulted in an order-of-magnitude increase in running time, so it is quite likely that this will remain unfinished by the time of my defense.

Work on 3-D lattices is an ongoing project by multiple members of our group. In particular, we have been working towards a systematic investigation of AF Potts partition zeros in the complex plane for various other crystal lattices (face-centered cubic, body centered cubic, diamonds, hexagonal etc.). This has included calculations for finite length strips, and analytic solutions for the limiting curves of the zero set in the infinite length limit.

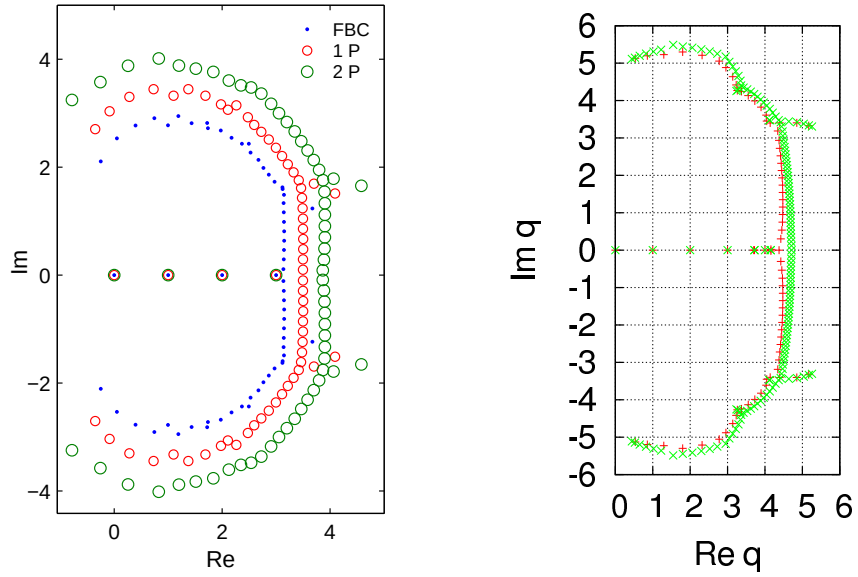


Figure 5.3: Chromatic roots for finite lattices. Left: $4\times4\times4$ simple cubic lattices with $\{0,1,2\}$ periodic boundaries. Right: $2\times2\times n$ body-centered cubic lattices for $n=20$ (red $+$) and 40 (green $\times$) obtained from computations on diagonal slices, periodic in first 2 dimensions. Second plot courtesy of Niels Kurz [65].

Figure 5.3(right) shows a plot of roots for finite 3-dimensional *diagonal slices* of the body-centered cubic lattice. The strips of 3-D lattices we work on come in two flavours: the familiar $j\times k\times n$ straight sections, and $j\times k\times n$ diagonal slices taken through infinite lattices (j and k here indicate the maximum width of individual layers of the slice with respect to the first two dimensions). Under free boundary conditions diagonal slices have the advantage that they have a larger number of vertices with a full set of neighbours (i.e. that achieve the infinite system's coordination number) than a comparable straight section; furthermore, under periodic boundary conditions they can avoid some issues that come up with thin 3-D sections (e.g. for a $2\times4\times n$ straight section applying a periodic boundary to the first dimension is redundant since the affected vertices already have a connection, but this will not happen with a similarly-sized diagonal strip taken at the appropriate angle).

Limiting curves for root sets. As mentioned in the introduction, some researchers have obtained limiting curves for the chromatic root sets of infinite length strips by analytic means [28, 96]. Notable in this regard is the *method of generating functions* used by Shrock and collaborators [28, 29, 92, 103]. This technique uses the Deletion-Contraction formula, given here in section 1.3 as (1.8), to build a recurrence relation for a sequence P_n where the individual terms are the chromatic polynomials of a given repeating strip of length n. A generating function of a sequence $a_0, a_1, a_2, \ldots$ is a formal power series for which a_k is the coefficient of the x^k term. One common use of these is to obtain information about a sequence defined by a recurrence relation; it is, for example, possible to solve the recurrence entirely if the generating function can be reduced to known functions by way of sum, product, derivative, etc. operations.

The interested reader is referred to [92] for more details on the method. Suffice it to say that for strip lattices the recurrence relation itself is not solved, but rather it is shown that any generating function constructed from the Deletion-Contraction based recurrences Shrock uses is the ratio of two functions $\mathcal{N}/\mathcal{D}$, with all the information required to numerically calculate the limiting curves contained in the relatively tractable divisor function $\mathcal{D}$.

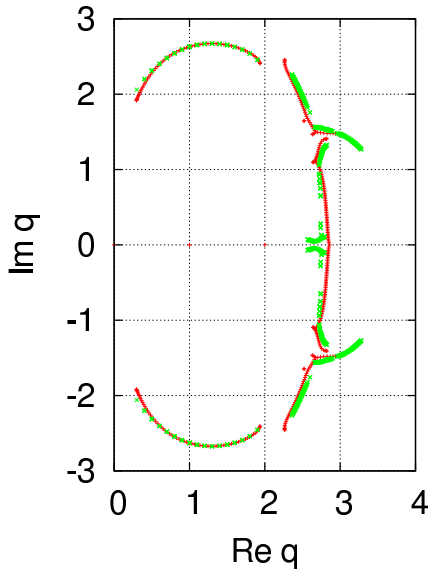


Figure 5.4: Limiting curve (in green) for chromatic root set of $2 \times 3 \times \infty$ simple cubic lattice obtained by the generating function method (free boundary conditions). Plot courtesy of Niels Kurz [65].

While the approach is ingenious, it does have the limitation that the base cases for each recurrence relation have to be determined by manual application of the Deletion-Contraction formula, so it cannot just be set to work upon any lattice of interest without a fair amount of human effort beforehand. As the size or complexity of the layers of the strip increase determining the best order for applying the Deletion-Contraction operations so as to minimize the number of base cases becomes a mathematically hard problem in its own right. Hence applying the Shrock method as it stands to lattices in higher dimensions, especially crystal lattices, becomes problematic in practice (though conceptually the problem is unchanged).

Our hypothesis was that the entire procedure could be automated using the new algorithm in its FORM implementation. This has been verified by a Diplome student, Niels Kurz, working partly under my supervision; he has developed a set of complementary FORM, MATHEMATICA, and numeric programs that when given a generic specification for a repeating strip graph will derive a recurrence relation, extract the divisor function $\mathcal{D}$, and then numerically determine the limiting curve for the chromatic zeros. Figure 5.4 gives an example of the results we have obtained. It should be noted that while the generating function and hence numerator and divisor functions for a given strip graph calculated by our automated procedure are necessarily equal to those determined by Shrock's manual method, the form of the recurrences is radically different. Shrock's method uses multiple base cases (chromatic polynomials of small sublattices) to define a single sequence, but our method finds a set of mutual recurrences for separate sequences $A_n, B_n, C_n, \ldots$, each having a single basis; the number of sequences involved depends on the size and complexity of an individual layer of the strip.

Ironically, Mr. Kurz's success with regards to the recurrences has revealed a second computational bottleneck. For both the original Shrock method [92] and our automated procedure [65] deriving the divisor function $\mathcal{D}$ from the recurrence(s) involves symbolically solving the determinant of an $n \times n$ matrix, where n here is the number of base cases [92] or sequences [65]. Solving symbolic determinants is an *NP*-complete problem, so e.g. 100 by 100 matrices are

out of the question, at least in MATHEMATICA. Furthermore, the number of sequences in our mutual recurrences grows exponentially with the number of vertices in an individual layer of the strip, for the same reason that the running time of the new algorithm in general grows exponentially with the width of a finite strip graph. The current limit for the size of arbitrarily connected layers the automated procedure can handle is about 10 vertices (sparsely connected layers can be a bit larger). It should be noted that with the Shrock recurrences the number of base cases needed will as well grow exponentially in the width, so their group has possibly encountered the same problem.

Mr. Kurz and I have discussed the possibility of speeding up the determinant calculation by moving the operation from MATHEMATICA into FORM. Since FORM does not have a symbolic determinant operation of its own while MATHEMATICA does, this would involve a new round of coding from scratch, though it also means we could take computational advantage of any special structure our matrices have. The *NP*-completeness of the determinant problem means that size of the layers would still be limited, but even an extra 4 or 5 vertices greatly increases the structural range of strips we can access. As well, FORM's superior handling of memory and disk access might make it possible to push a little further, say to 4×4 layers of an infinite straight section of a simple cubic lattice.

5.2.2 Random graphs

As chapter 4 reveals, there is a characteristic layout for the chromatic roots of a generic arbitrary graph (as manifested by a random graph) that depends primarily on the size N and edge-density P, and only secondarily on any other (variable) structural characteristics. When these layouts are compared to chromatic root sets and limiting curves for strip lattices there seems to be a) much more disparity in the layouts for the lattice strips than the random graphs, and b) among those lattice strips having roots sets in roughly the same inverted "C" shape as random graphs of similar density, some noticable differences (e.g. features like horns and gaps, as well as general stretching in the imaginary direction). The question now is, what accounts for the difference and diversity. Possibilities are:

Large scale structure. Since lattices are highly structured, they tend to have global properties not shared by random graphs. For example, a square lattice (FBC) is close-to-regular, bipartite, planar etc.

Small scale structure. The fact that lattices are defined by simple repeating subgraphs may mean their root sets inherit eccentricities from the subgraph that are emphasized by the repetition.

Sparsity. Lattices are relatively sparse, and sparse random graphs showed the most variation in chromatic root location across instances.

Width. For computational reasons lattice calculations are commonly done on long, thin sections.

While these characterizations are all graph theoretical, there is a physical implication. Crystals, for example, are (as graphs) regular, often bipartite, and sparse; but they are rarely planar or only a few atoms wide.

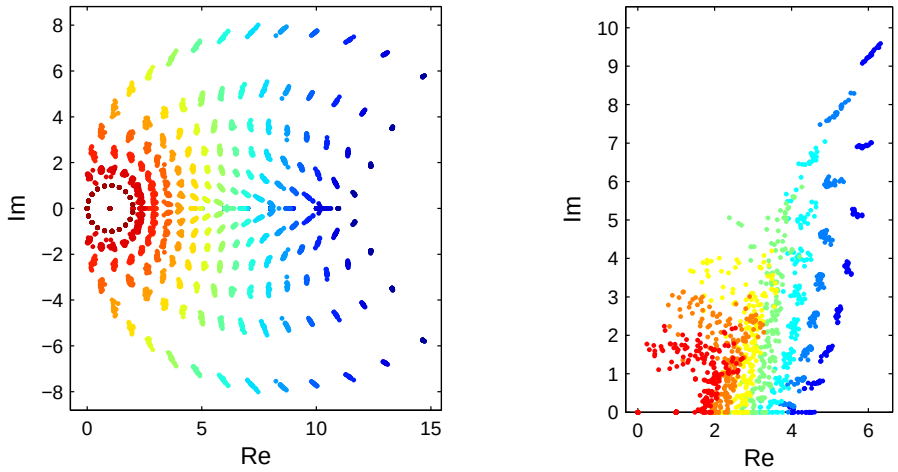


Figure 5.5: Left: chromatic roots for random regular graphs, $N = 20$, degree $K = 2, 3, ..., 17$ by color (red: 2 to blue: 17). Right: roots for random bipartite graphs in upper half plane, $N = 20$, average degree $K = 3, 4, ..., 9$ by color (red: 3 to blue: 9).

To begin with, we have done chromatic polynomial calculations for a selection of larger Erdös-Rényi random graphs, specifically for order $N = 33$ and 36 across a subset of the range of edge-densities P. Since both running times and space requirements increase exponentially far fewer trials were done than the 20 per parameter set featured in chapter 4. The results for these runs largely confirm the scalings and qualitative assertions given in the article, though the "end-of-range" effect noted there, where the variability/deviation in e.g. crossing point measurements increases for very low densities, is present as well and seems in fact a bit more pronounced.

While there is a great deal of variability in the running time, the large number of runs done in total means we have fairly trustworthy empirical estimates how the running time increases with N (and P) in general for arbitrary graphs (reminder: $P(1-P)$ has its maximum at $P = 0.5$):

$$t_{\text{alg}} \approx c_{\text{m}} \frac{\exp(\sqrt{P(1-P)}N)}{\exp(5\sqrt{1-P})} \quad = \mathcal{O}\left(e^{\frac{N}{2}}\right) \tag{5.3}$$

where c_{m} is a constant dependent on the machine we are running on. From this we can predict that sparse random graphs ($P \leq 0.10$) will still be quite doable for N up to 60 and maybe a bit more, while graphs in the density range $0.50 \leq P \leq 0.60$ will become problematic by the time N gets to 35 or so (a four week calculation on one of our institute's cluster nodes, by my calculation).

The above puts a limit on the size of graphs we can handle, and in particular means that some classes of graphs are beyond reach, if they are defined by statistical or other properties that only become evident when the number of vertices is large. Among such classes are scale free graphs, since we would need at least 100 vertices to get more than two decades in the degree distribution. Small world and other clustering-coefficient determined graphs of the size usually seen in physics papers are similarly infeasible. We must instead limit our tests to structural properties that are well defined at all sizes.

Two such properties are fixed vertex degree, and two-colorability. As mentioned in chapter 4, preliminary work has already been done on regular graphs and bipartite graphs; see figure 5.5

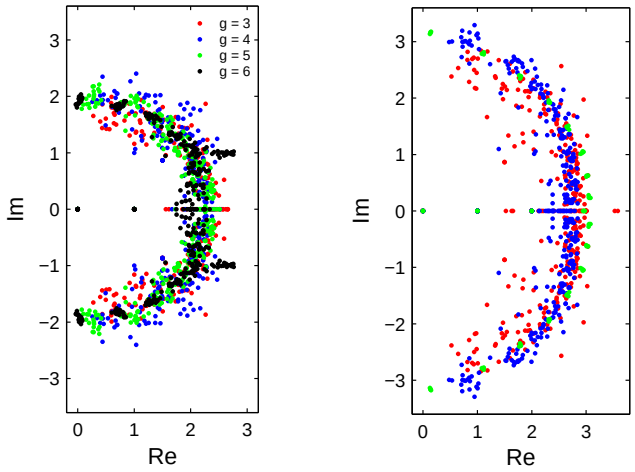


Figure 5.6: Chromatic roots for 20 vertex random graphs having girth $g = 3, 4, 5, 6$ (by color) with average degree $K = 3$ (left) and 4 (right).

for chromatic root layouts. The basic pattern we see with random regular graphs is quite similar to that for ER random graphs, for which the degrees have a Poisson distribution, though as one might expect the scatter in root location across different graphs is much less. With random bipartite graphs we see a significantly different layout, especially as the density P approaches the maximum limit of a little over one-half. At lower densities the location and shape of the chromatic root sets are not that different than those for ER random graphs, though somewhat more angular. As $P \to 0.50$ the shape morphs into something like the a left curly bracket "{"; the effect across the entire range of densities is like nothing so much as a Persian recurve bow being drawn back. This certainly begs for a more in-depth mathematical analysis.

As mentioned in the introduction, several researchers in both physics and graph theory have postulated that maximum degree (or possibly second highest degree) has a determinative effect on chromatic root location [94, 104], though I am personally skeptical that this holds for any but special cases. For random graphs maximum degree is correlated with density P and size N, which (together) are correlated with root location as determined by the crossing point $X_{N,P}$, maximum modulus $|z_{\max}|$, and other metrics; so across all graphs we do see a correlation between maximum degree and location. However, when controlled for average degree (i.e. parameter set) all correlation in our random graph data disappears. Since the degree distribution of ER random graphs has a strong central tendency this might not be dispositive, so for additional confirmation tests need to be done where one or two vertices have very high degree while the average degree is kept low (the poor man's scale-free graph). I predict that if a significant effect due to maximum degree is seen it will only be in the location of a few outlier roots ($|z_{\max}|$) rather than in the main body of the root set as reflected in $X_{N,P}$ or the mean real value.

As well, there is an issue that should be addressed relating to the neighbourhood structure of, respectively, degree- and density-parameterized random graphs. A common property of density-parameterized ER random graphs, small worlds, and various lattices is that as size N is increased certain local statistics are maintained: the number of triangles and other small cycles an individual vertex is part of. With respect to small worlds and lattices these statistics pertain to spatially-defined localities; for lattices in particular they continue to hold as $N \to \infty$

even though $P \to 0$. With ER random graphs, however, for local statistics to be maintained while N is increased the density P must be held constant. If instead the average degree K is fixed then (as with a lattice) $P \to 0$ but (unlike a lattice) all fixed-size neighbourhoods grow sparser. In the $N \to \infty$ limit almost all finite connected subgraphs of a random graph with fixed or bounded vertex degree are in fact trees.

Since a main conjecture in chapter 4 is based on extrapolating our empirical results for fixed average degree random graphs to infinite size N, resolving the effect of neighbourhood structure is a priority; ideally we would like evidence that it does not have a large effect on layout or location of chromatic roots either way. Some preliminary work has already been done with high *girth* graphs (the girth of a graph is its minimum cycle length), which can stand in for the general situation where larger N implies sparser neighbourhoods; girth 4 implies no triangles, girth 5 no triangles or 4 cycles, etc. Figure 5.6 shows chromatic roots for graphs of various girth plotted against each other, including girth = 3 (i.e. ordinary random graphs). The overall location seems little changed (in particular, the crossing points coincide), though for the graphs with $K = 3$ and girth = 6 there are definitely features one usually sees in lattice plots.

The more physically relevant situation where local statistics are maintained as network size grows still requires exploration. This of course is exactly what lattices are supposed to model, so a randomized lattice topology leaps to mind. Note that what is described here is not the kind of bond- or site-diluted lattice that arises in percolation theory as much as a tiling of a spatially extended vertex set with distinct small ER random graphs satisfying a given set of parameters. For the same reason that we cannot undertake calculations on lattices of arbitrary width, testing would probably have to be restricted to random layered graphs; similarly, the kind of long jumps seen in small worlds are ruled out.

One thing that our recent lattice work described in section 5.2.1 makes possible is calculation of chromatic zero set limiting curves for infinite length strips made up of repetitions of the same random layer. While such a graph would not properly be random (more like a randomly distorted strip lattice), it would be of interest to compare such curves to both the arc fits to chromatic roots of multiple finite random graphs given in chapter 4 and the limiting curves for regular lattices obtained by ourselves and other researchers; in particular, if the curves consistently looked more like the arc fits it would mean that particular structural aspects of regular lattices are responsible for their chromatic variability and quirks (gaps, horns, etc.), while if the same kind of quirks and variability showed up as for the lattice curves we might need to look more directly at the effect of small width itself on root location.

5.2.3 Other graph polynomials

Finally, we have a longer term project involving the extension of the basic formula underlying the new algorithm to related polynomials and functions of interest. These of course include the Potts partition function $Z(G, q, T)$ at greater than zero temperatures and the Tutte polynomial $T(G, x, y)$ at arbitrary x, y values, which are both discussed in the introduction.

The new algorithm in its bare form (sans optimizations) can be converted into a Potts partition solution for both ferromagnetic and antiferromagnetic models at any temperature T simply by changing $(1 - \delta_{\sigma_i\sigma_j})$ in (1.12) to $(1 + (e^{\beta J} - 1)\delta_{\sigma_i\sigma_j})$ as in (1.11), where $T = (k_B\beta)^{-1}$, k_B is

the Boltzmann constant, and $J = \{-1, 1\}$ depending on whether the interaction is repulsive or attractive. For notational simplicity we let $v = (e^{\beta J} - 1)$, giving us the representation

$$Z_2(G, q, v) = \sum_{\sigma_1=1}^{q} \cdots \sum_{\sigma_N=1}^{q} \prod_{(i,j)\in E} (1 + v\delta_{\sigma_i\sigma_j}) \,, \tag{5.4}$$

where E is the edge-set of G. This can be solved using the same algebraic operators as the chromatic polynomial representation, hence the same pattern matching rules. However, if v is left unspecified the result is a bivariate polynomial in v and q, which means the running time is substantially longer. We have done some testing of this version of the Potts partition function in a FORM implementation using both symbolic and numeric v. Specifying a value for v beforehand gives us a faster running time, but it still never comes close to the speed of the chromatic polynomial solver since there is practically no cancellation of terms resulting from the fact that for Kronecker deltas

$$\delta_{\sigma_i\sigma_j} \times (1 - \delta_{\sigma_i\sigma_j}) = 0 \,. \tag{5.5}$$

An important optimization in the current implementation of the chromatic polynomial solver is based on pre-clearing terms that will eventually be cancelled out this way (see appendix B.1 for details).

The Tutte polynomial $T(G, x, y)$ is by design a generalization of the chromatic polynomial $P(G, q)$ from one to two variables, so solutions for P will directly give us the values of T along the x-axis:

$$T(G, x, 0) = (-1)^N \frac{P(G, 1 - x)}{(x - 1)^{\kappa(G)}} \,. \tag{5.6}$$

where N (as usual) is the size of G and $\kappa(G)$ counts the number of its connected components. Outside of this domain we can make use of the fact that evaluations of the Tutte polynomial along selected hyperbolas in the (x, y) plane give us the Potts partition function, so that

$$T(G, x, y) = \frac{Z_2(G, (x - 1)(y - 1), y - 1)}{(x - 1)^{\kappa(G)}(y - 1)^N} \,. \tag{5.7}$$

Note that we are using the simplified version of the general Potts partition function (5.4) here i.e. $y - 1 = v$. Since the partition function is only defined in the limit for zero states or zero temperature special handling would be required for Tutte values when x or y equals one. While it would not be difficult to redefine our summation operators to evaluate this directly as e.g. a FORM program, there is no immediate computational advantage in doing so (for example, coefficients of the delta terms become bivariate rational fractions, which is unhelpful), though possible optimizations have yet to be investigated.

As well as these there are a number of related univariate graph theoretical polynomials such as the Reliability polynomial [29] and the Flow polynomial [126]. The Jones polynomial from topology as well is derivable from the generalized Potts partition function for a subclass of knots (which are prime, reduced, and alternating; see [37]).

Of these we have made a preliminary foray into calculations of the Reliability polynomial. The *all terminal reliability* $R(G, p)$ gives the probability that there is a working pathway between every pair of nodes in G when individual links have probability p of being valid and $1 - p$ of

failing. While it is much studied by theorists [126] computational hardness limits its applicability in communications engineering. The basic definition gives us the following expression in terms of connected spanning subgraphs of G induced by subsets E' of the edge set E:

$$R(G,p) = \sum_{\kappa(E')=1}^{E' \subseteq E} p^{|E'|}(1-p)^{|E-E'|}. \tag{5.8}$$

If $\kappa(G) > 1$ then $R(G,p)$ is uniformly 0, otherwise it is greater than 0 for all p in the range $(0,1]$. Since $R(G,p)$ satisfies a Deletion-Contraction formula of its own it is reducible to the Tutte polynomial via:

$$R(G,p) = p^{N-1}(1-p)^{M-N+1} T\left(G, 1, \frac{1}{1-p}\right), \tag{5.9}$$

where M is the number of edges $|E|$. Note that $x = 1$ here means that solution by simple evaluation of (5.7) is problematic. However, if we now feed the changes of variables through both (5.7) and (5.9) and bring the prefactors into the body of the expression given in (5.4) then R can be solved by:

$$R(G,p) = \mathrm{S}_1\,\mathrm{S}_2 \cdots \mathrm{S}_{N-1} \prod_{(i,j)\in E} (1 - p + p\,\delta_{i,j}), \tag{5.10}$$

where (since we are no longer summing over q states) S is now simply a formal operator which evaluates as

$$\mathrm{S}_k\, 1 \to 0 \qquad \mathrm{S}_k\, \delta_{k,j_1} \to 1 \qquad \mathrm{S}_k\, \delta_{k,j_1}\delta_{k,j_2} \to \delta_{j_1,j_2} \qquad \text{etc.} \tag{5.11}$$

Similarly, since we are no longer assigning different colors to vertices $\delta_{i,j}$ is now an indicator that an edge exists between vertices i and j. This means that the entire product in (5.10) is technically equal to 1 from the start; deferring evaluation of the delta-terms until they reach the appropriate operator is necessary to determine how much retention $(p\,\delta_{i,j})$ or deletion $(1-p)$ of given edges affects the probability that the whole graph is still connected.

Expression (5.10) effectively traces its way edge-by-edge through all simple paths between vertex 1 and all other vertices 2 through N, which is why the S operator is only applied $N-1$ times. As with the basic chromatic polynomial algorithm (which traces through all q-colorings) an important aspect of this is that not every distinct possibility is explicitly instantiated. Instead, at the k-th iteration the coefficient of each delta-term contains information in polynomial form about all partial solutions (respectively edge selections or vertex colorings) compatible with any single route going forward i.e. computational branches are merged when possible. Similarly, every delta-term is consistent with many possible forward routes, but branching does not occur until necessary.

As such, (5.10) as implemented in FORM achieves running times that are comparable to the chromatic polynomial algorithm without any of the subsequent optimizations applied to it. This is certainly nice but it will not get us a $4 \times 4 \times 4$ cubic lattice. Since in (5.10) there is nothing to give rise to significant cancellations along the lines of (5.5), we are currently looking at alternative ways of achieving a similar speedup.

5.3 General Prospects and Conclusions

The astute reader may have figured out by now that I do not plan on invoking any overarching theory to connect the two scientific topics of this thesis. Commonly, when a thesis or monograph deals at length with several seemingly unrelated phenomena, there is a single underlying mathematical or formal principle involved e.g.

- fireflies, Josephson junctions, bridges falling down ⇒ synchronization;
- earthquakes, financial markets, baby names ⇒ scaling laws;

and so forth. In our case what is common to both topics is network structure, hence graph theory and combinatorial optimization are needed to deal with them. This alone, however, does not mean that any particular computational or analytic technique is sufficient to our purposes, but rather the opposite.

The first of our topics was the inverse problem for leaky integrate-and-fire neuron networks. It turns out that the base problem is in fact tractable i.e. it belongs to the computational class **P**, problems that have a polynomial running time in terms of the size of the input. While we have yet to apply the method to other models, I believe it is likely that the general problem of abstract neuronal network reconstruction (that is, simple model neurons without uncertainty displaying irregular but not pathological spiking) will similarly be in **P**. Going forward the issue will not be one of **NP**-completeness, but instead reliability and practical applicability when more realistic models and data are brought in. We should note that these are the same issues that arise in development of first order numerical or statistical techniques, which are as well (usually) unproblematic in algorithmic terms.

The second of our topics was evaluation of the antiferromagnetic Potts partition function in the zero temperature limit a.k.a. the chromatic polynomial. In this case the base problem is computationally hard, harder in fact than **NP**-complete, since it is a counting problem rather than a simple decision problem. Here a fair amount of effort was put into algorithmic tweaking to improve performance in terms of running time, since this is the primary bar to dealing with realistic physical systems. While these improvements are a real advance on the state of the art and do make significantly more systems accessible, almost all moderately large graphs (e.g. more than 100 vertices) are still well out of reach, and are likely to remain so forever. Hence we have no choice but to focus on still relatively small but feasible systems in hope that results will be representative of the more general situation; such systems include the lattice strips and random graphs featured in the articles.

Solutions to the main problems and the many subproblems involved required application of a range of mathematical and optimization techniques. Individually many of these techniques are quite basic and well-known, such as the algebra used in the reconstruction method for full rank systems; other techniques were designed from scratch to deal with particular issues, such as the vertex reordering used for the chromatic polynomial work (see appendix B.1).

As work in complex networks becomes more ambitious, this kind of wide band approach will probably be necessary for at least those problems with a combinatorial or graph theoretic aspect. Computational complexity theory does not give us a magic bullet for dealing with **NP**-complete problems embedded in natural systems, but it does give us a road map for navigating our way around them. This will no doubt become an essential aspect of future research on large scale complex systems.

Bibliography

[1] M. Abeles, H. Bergman, E. Margalit, and E. Vaadia, Spatiotemporal Firing Patterns in the Frontal Cortex of Behaving Monkeys, *Journal of Neurophysiology* **70** 1629–1638 (1993)

[2] A. Arenas, A. Díaz-Guilera, and C.J. Pérez-Vicente, Synchronization Reveals Topological Scales in Complex Networks, *Phys. Rev. Lett.* **96** 114102 (2006)

[3] N. Alon and J.H. Spencer, *The Probabilistic Method*. Wiley (2005)

[4] V.I. Arnol'd (Roger Cooke, trans.), *Ordinary Differential Equations*. Springer-Verlag (1992)

[5] A-L. Barabási and R. Albert, Emergence of Scaling in Random Networks, *Science* **286** 509–512 (1999)

[6] I. Barrowdale and F.D.K. Roberts, Algorithm 478: Solution of an Overdetermined System of Equations in the l_1 Norm [F4], *Communications of the ACM* **17** 319–320 (1974)

[7] R. Baxter, *Exactly Solved Models in Statistical Mechanics*. Academic Press Inc. (1982)

[8] S. Beraha, J. Kahane, and N.J. Weiss, Limits of Zeroes of Recursively Defined Polynomials, *Proc. Natl. Acad. Sci. USA* **72** 4209 (1975)

[9] P. Berthomé, S. Lebresene, and K. Nguyễn, Computation of Chromatic Polynomials Using Triangulations and Clique Trees, in *Graph-Theoretic Concepts in Computer Science*, Springer Lecture Notes in Computer Science 3787, 362–373 (2005)

[10] H. Bielak, Roots of Chromatic Polynomials, *Discrete Mathematics* **231** 97–102 (2001)

[11] N.L. Biggs, Chip-Firing and the Critical Group of a Graph, *Journal of Algebraic Combinatorics* **9** 25–45 (1999)

[12] D. Bini and G. Fiorentino, Design, Analysis, and Implementation of a Multiprecision Polynomial Rootfinder, *Numerical Algorithms* **23** 127–173 (2000)

[13] D. Bini and G. Fiorentino, MPSolve, Version 2.2 (2001)

[14] G. Birkhoff, A Determinant Formula for the Number of Ways of Coloring a Map, *Ann. Math.* **14** 42–46 (1912)

[15] A. Björklund, T. Husfeldt, P. Kaski, and M. Koivisto, Computing the Tutte Polynomial in Vertex-Exponential Time, *Proceedings of the 2008 49th Annual IEEE Symposium on Foundations of Computer Science*, 677-686 (2008)

[16] A. Björner, L. Lovász, and P.W. Shor, Chip-Firing Games on Graphs, *European J. Combin.* **12** 283–291 (1991)

[17] M. Blank and L. Bunimovich, Long Range Action in Networks of Chaotic Elements, *Nonlinearity* **19** 329–344 (2006)

[18] M.A. Boden, *Mind as Machine: A History of Cognitive Science*. Clarendon Press (2006)

[19] B. Bollobás, The Chromatic Number of Random Graphs, *Combinatorica* **8** 49–55 (1988)

[20] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge University Press (2004)

[21] J. Brown, On the Roots of Chromatic Polynomials, *Journal of Combinatorial Theory B* **72** 251–256 (1998)

[22] N. Brunel, Dynamics of Sparsely Connected Networks of Excitatory and Inhibitory Spiking Neurons, *Journal of Computational Neuroscience* **8** 183–208 (2000)

[23] N. Brunel and V. Hakim, Fast Global Oscillations in Networks of Integrate-and-Fire Neurons with Low Firing Rates, *Neural Computation* **11** 1621–1671 (1999)

[24] S-L. Bu and I-M. Jiang, Estimating the Degree Distribution in Coupled Chaotic Oscillator Networks, *Europhys. Lett.* **82** 69001 (2008)

[25] J. Buchli, L. Righetti, and A.J. Ijspeert, Frequency Analysis with Coupled Nonlinear Oscillators, *Physica D* **237** 1705–1718 (2008)

[26] E. Bullmore and O. Sporns, Complex Brain Networks: Graph Theoretical Analysis of Structural and Functional Systems, *Nature Reviews Neuroscience* **10** 186–198 (2009)

[27] M.L. Cadenasso, S.T.A. Pickett, and J.M. Grove, Dimensions of Ecosystem Complexity: Heterogeneity, Connectivity, and History, *Ecological Complexity* **3** 1–12 (2006)

[28] S-C. Chang and R. Shrock, Ground State Entropy of the Potts Antiferromagnet on Strips of the Square Lattice, *Physica A* **290** 402–430 (2001)

[29] S-C. Chang and R. Shrock, Reliability Polynomials and Their Asymptotic Limits for Families of Graphs, *J. Stat. Phys.* **112** 1019–1077 (2003)

[30] G. Chen and Z. Duan, Network Synchronizability Analysis: A Graph-Theoretic Approach, *Chaos* **18** 037102 (2008)

[31] T. Cormen, C. Leiserson, R. Rivest, and C. Stein, *Introduction to Algorithms*, MIT Press (2001)

[32] M. Diesmann, M-O. Gewaltig, and A. Aertsen, Stable Propagation of Synchronous Spiking in Cortical Neural Networks, *Nature* **402** 529–533 (1999)

[33] M. Denker, M. Timme, M. Diesmann, F. Wolf, and T. Geisel, Breaking Synchrony by Heterogeneity in Complex Networks, *Phys. Rev. Lett.* **92** 074103 (2004)

[34] F.M. Dong, K.L. Leo, K.M. Koh, and M.D. Hendy, Non-Chordal Graphs Having Integral-Root Chromatic Polynomials II, *Discrete Mathematics* **245** 247–253 (2002)

[35] S.N. Dorogovtsev, J.F.F. Mendes, and A.N. Samukhin, Giant Strongly Connected Component of Directed Networks, *Phys. Rev. E* **64** 025101 (2001)

[36] J. Edmonds, Paths, Trees, and Flowers, *Canad. J. Math.* **17** 449–467 (1965)

[37] P. Fendley and V. Krushkal, Tutte Chromatic Identities from the Temperley-Lieb Algebra, *Geometry & Topology* **13** 709-741 (2009)

[38] A. Fingelkurts, A. Fingelkurts, and S. Kähkönen, Functional Connectivity in the Brain–Is It an Elusive Concept?, *Neuroscience and Biobehavioural Reviews* **28** 827–836 (2005)

[39] C.M. Fortuin and P.W. Kasteleyn, On the Random-Cluster Model: I. Introduction and Relation to Other Models, *Physica* **57** 536–564 (1972)

[40] F. Gabbiani and C. Koch, Principles of Spike Train Analysis, *Methods in Neuronal Modeling* (C. Koch and I. Segev, eds.), The MIT Press (1989)

[41] T.S. Gardner, D. di Bernardo, D. Lorenz, and J.J. Collins, Inferring Genetic Networks and Identifying Compound Mode of Action via Expression Profiling, *Science* **301** 102–105 (2003)

[42] M-O. Gewaltig and M. Diesmann, NEST (NEural Simulation Tool), *Scholarpedia* **2** 1430 (2007)

[43] V. Gogate and R. Dechter, A Complete Anytime Algorithm for Treewidth, *ACM International Conference Proceeding Series: Proceedings of the 20th Conference on Uncertainty in Artificial Intelligence* **70** 201–208 (2004)

[44] D. Goodman, R. Brette, Brian: a Simulator for Spiking Neural Networks in Python, *Front. Neuroinform.* **2** `doi:10.3389/neuro.11.005.2008` (2008)

[45] B. Gutkin, D. Pinto, and B. Ermentrout, Mathematical Neuroscience: from Neurons to Circuits to Systems, *J. Physiology – Paris* **97** 209–219 (2003)

[46] A. Hartmann and M. Weigt, *Phase Transitions in Combinatorial Optimization Problems.* Wiley-VCH (2005)

[47] C. Henrich, *Inferring Network Connectivity from Dynamical Response Measurements.* Masters thesis, Faculty of Biology, Georg-August University, Göttingen (2007)

[48] H. Hong, M.Y. Choi, and B.J. Kim, Synchronization on Small-World Networks, *Phys. Rev. E* **65** 026139 (2002)

[49] J. Hopcroft and J. Ullman, *Introduction to Automata Theory, Languages, and Computation.* Addison-Wesley (1979)

[50] B. Horwitz, The Elusive Concept of Brain Connectivity, *NeuroImage* **19** 466–470 (2003)

[51] T. Hu and D.B. Chklovskii, Reconstruction of Sparse Neural Circuits Using Compressive Sensing, *preprint* (2009)

[52] J. Hu, A.H. MacDonald, and B.D. McKay, Correlations in Two-Dimensional Vortex Liquids, *Phys. Rev. B* **49** 15263-15270 (1994)

[53] K. Huang, *Statistical Mechanics*. John Wiley & Sons (1987)

[54] C.P. Hughes and A. Nikeghbali, The Zeros of Random Polynomials Cluster Uniformly near the Unit Circle, *Compos. Math.* **144** 734–746 (2008)

[55] Y. Ikegaya, G. Aaron, R. Cossart, D. Aronov, I. Lampl, D. Ferster, and R. Yuste, Synfire Chains and Cortical Songs: Temporal Modules of Cortical Activity, *Science* **304** 559–564 (2004)

[56] B. Jackson, Zeros of Chromatic and Flow Polynomials of Graphs, *J. Geom.* **76** (1-2), 95–109 (2003)

[57] S. Jahnke, R-M. Memmesheimer, and M. Timme, How Chaotic Is the Balanced State?, *Frontiers in Computational Neuroscience* **3** `doi:10.3389/neuro.10.013` (2009)

[58] S. Janson, D.E. Knuth, T. Łuczak, and B. Pittel, The Birth of the Giant Component, *Random Structures and Algorithms* **4** 233–358 (1993)

[59] S. Janson, T. Łuczak, and A. Rucinski, *Random Graphs*. John Wiley & Sons (2000)

[60] V.K. Jirsa, A.R. McIntosh (Eds.), *Handbook of Brain Connectivity*. Springer (2007)

[61] V.F.R. Jones, The Jones Polynomial, *Discrete Math.* **294** 275–277 (2005)

[62] H. Kori and A.S. Mikhailov, Entrainment of Randomly Couple Oscillator Networks by a Pacemaker, *Phys. Rev. Lett.* **93** 254101 (2004)

[63] D. Kozen and M. Timme, Indefinite Summation and the Kronecker Delta, *Computing and Information Science Technical Reports*, Cornell University (2007)

[64] Y. Kuramoto, *Chemical Oscillations, Waves and Turbulence*. Springer-Verlag (1984)

[65] N. Kurz, *Symbolic Algebra Calculations of Critical Points of Potts Antiferromagnets at* $T = 0$. Diploma thesis, Faculty of Physics, Georg-August University, Göttingen (2010)

[66] T. Lee and C. Yang, Statistical Theory of Equations of State and Phase Transitions. II. Lattice Gas and Ising Model, *Phys. Rev.* **87** 410–419 (1952)

[67] S. Łęski and D.K. Wójcik, Inferring Coupling Strength from Event-Related Dynamics, *Phys. Rev. E* **78** 041918 (2008)

[68] P.H. Lundow and K. Markström, Broken-Cycle-Free Subgraphs and the Log-Concavity Conjecture for Chromatic Polynomials, *Experimental Mathematics* **15** 343–353 (2006)

[69] H-W. Ma and A-P. Zeng, The Connectivity Structure, Giant Strong Component and Centrality of Metabolic Networks, *Bioinformatics* **19** 1423–1430 (2003)

[70] V.A. Makarov, F. Panetsos, and O. de Feo, A Method for Determining Neural Connectivity and Inferring the Underlying Network Dynamics Using Extracellular Spike Recordings, *Journal of Neuroscience Methods* **144** 265–279 (2005)

[71] The Mathworks Inc., MATLAB, Version 6.5 (2002)

[72] A. Mauroy and R. Sepulchre, Clustering Behaviors in Networks of Integrate-and-Fire Oscillators, *Chaos* **18** 037122 (2008)

[73] R-M. Memmesheimer and M. Timme, Designing the Dynamics of Spiking Neural Networks, *Phys. Rev. Lett.* **97** 188101 (2006)

[74] R-M. Memmesheimer and M. Timme, Designing Complex Networks, *Physica D* **224** 182–201 (2006)

[75] R. Mirollo and S. Strogatz, Synchronization of Pulse-Coupled Biological Oscillators, *SIAM Journal on Applied Mathematics* **50** 1645–1662 (1990)

[76] M. Molloy and B. Reed, The Size of the Giant Component of a Random Graph with a Given Degree Sequence, *Combinatorics, Probability and Computing* **7** 295–305 (1998)

[77] J. Nagler, A. Levina, and M. Timme, Discontinuous Phase Transitions in Random Network Percolation, *MPIDS preprint* (2010)

[78] D. Napoletani and T. Sauer, Reconstructing the Topology of Sparsely Connected Dynamical Networks, *Phys. Rev. E* **77** 026103 (2008)

[79] M.E.J. Newman, Assortative Mixing in Networks, *Phys. Rev. E* **89** 208701 (2002)

[80] M.E.J. Newman, Random Graphs with Clustering, *Phys. Rev. Lett.* **103** 058701 (2009)

[81] M.E.J. Newman, S. Strogatz, and D. Watts, Random Graphs with Arbitrary Degree Distributions and Their Applications, *Phys. Rev. E* **64** 026118 (2001)

[82] J.G. Nicholls, A.R. Martin, B.G. Wallace, and P.A. Fuchs, *From Neuron to Brain*. Sinauer Associates (2001)

[83] A. Nijenhuis and H.S. Wilf, *Combinatorial Analysis*. Academic Press (1975)

[84] L. Onsager, Crystal Statistics. I. A Two-Dimensional Model with an Order-Disorder Transition, *Phys. Rev.* **65** 117-149 (1944)

[85] C. Papadimitriou, *Computational Complexity*. Addison Wesley (1994)

[86] U. Parlitz, Estimating Model Parameters from Time Series by Autosynchronization, *Phys. Rev. Lett.* **76** 1232 (1996)

[87] G. Polya, *How to Solve It*. Princeton University Press (1945)

[88] R. Potts, Some Generalized Order-Disorder Transformations, *Proc. Camb. Phil. Soc.* **48** 106–109 (1952)

[89] F. Qi, Z. Hou, and H. Xin, Ordering Chaos by Random Shortcuts, *Phys. Rev. Lett.* **91** 064102-1 (2003)

[90] F. Radicchi and H. Meyer-Ortmanns, Entrainment of Coupled Oscillators on Regular Networks by Pacemakers, *Phys. Rev. E* **73** 036218 (2006)

[91] R.C. Read, An Improved Method for Computing Chromatic Polynomials of Sparse Graphs, *Research Report CORR 87-20*, Univ. of Waterloo (1987)

[92] M. Roček, R. Shrock, and S-H. Tsai, Chromatic Polynomials for Families of Strip Graphs and Their Asymptotic Limit, *Physica A* **252** 505–546 (1998)

[93] C. Rossant, D.F.M. Goodman, J. Platkiewicz, and R. Brette, Automatic Fitting of Spiking Neuron Models to Electrophysiological Recordings, *Frontiers in Neuroinformatics* **4** 2 (2010)

[94] G. Royle, Roots of Chromatic and Flow Polynomials of Graphs, *31st Australasian Conference in Combinatorial Mathematics and Combinatorial Computing*, invited talk (2006)

[95] G. Royle, Planar Triangulations with Real Chromatic Roots Arbitrarily Close to 4, *Annals of Combinatorics* **12** 195–210 (2008)

[96] J. Salas and A.D. Sokal, Transfer Matrices and Partition-Function Zeros for Antiferromagnetic Potts Models. I. General Theory and Square-Lattice Chromatic Polynomial, *J. Statistical Physics* **104** 609–699 (2001)

[97] T. Sauer, Reconstruction of Dynamical Systems from Interspike Intervals, *Phys. Rev. Lett.* **72** 3811 (1994)

[98] E. Schneidman, M.J. Berry II, R. Segev, and W. Bialek, Weak Pairwise Correlations Imply Strongly Correlated Network States in a Neural Population, *Nature* **440** 1007–1012 (2006)

[99] N. Schwartz, R. Cohen, D. ben-Avraham, A-L. Barabási, and S. Havlin, Percolation in Directed Scale-Free Networks, *Phys. Rev. E* **66** 015104 (2002)

[100] J. Sethna, *Statistical Mechanics: Entropy, Order Parameters, and Complexity.* Clarendon Press (2006)

[101] Y. Shen, Z. Hou, and H. Xin, Transition to Burst Synchronization in Couple Neuron Networks, *Phys. Rev. E* **77** 031920 (2008)

[102] Y. Shen, Z. Hou, and H. Xin, Revealing Degree Distribution of Bursting Neuron Networks, *Chaos* **20** 013110 (2009)

[103] R. Shrock and S-H. Tsai, Asymptotic Limits and Zeros of Chromatic Polynomials and Ground-State Entropy of Potts Antiferromagnets, *Phys. Rev. E* **55** 5165–5178 (1997)

[104] A.D. Sokal, Chromatic Polynomials, Potts Models and All That, *Physica A* **279** 324–332 (2000)

[105] A.D. Sokal, Chromatic Roots are Dense in the Whole Complex Plane, *Combin. Probab. Comput.* **13** 221–261 (2004)

[106] L. Sommerlade, M. Eichler, M. Jachn, K. Henschel, J. Timmer, and B. Schelter, Estimating Causal Dependencies in Networks of Nonlinear Stochastic Dynamical Systems, *Phys. Rev. E* **80** 051128 (2009)

[107] S. Song, P.J. Sjöstrom, M. Reigl, S. Nelson, D.B. Chklovskii, Highly Nonrandom Features of Synaptic Connectivity in Local Cortical Circuits, *PLoS Biol.* **3**(3): e68. (2005)

[108] O. Sporns and R. Kötter, Motifs in Brain Networks, *PLoS Biol.* **2**(11): e369. (2004)

[109] S. Strogatz, *Nonlinear Dynamics and Chaos.* Addison-Wesley (1994)

[110] M. Timme, *Studies on Ground State Entropy in Antiferromagnetic Potts Models*. M.A. Thesis, Physics Department, State University of New York at Stony Brook (1998)

[111] M. Timme, *Collective Dynamics in Networks of Pulse-Coupled Oscillators*. Ph.D thesis, Faculty of Mathematics and Natural Sciences, Georg-August University, Göttingen (2002)

[112] M. Timme, Does Dynamics Reflect Topology in Directed Networks?, *Europhys. Lett.* **76** 367–373 (2006)

[113] M. Timme, Revealing Network Connectivity from Response Dynamics, *Phys. Rev. Lett.* **98** 224101 (2007)

[114] M. Timme, F. Wolf, and T. Geisel, Coexistence of Regular and Irregular Dynamics in Complex Networks of Pulse-Coupled Oscillators, *Phys. Rev. Lett.* **89** 258701 (2002)

[115] B. Torben-Nielsen, M. Uusisaari, and K.M. Stiefel, A Comparison of Methods to Determine Neuronal Phase-Response Curves, (accepted by *Frontiers in Neuroinformatics*) `arXiv:1001.0647v3 [q-bio.QM]` (2010)

[116] Z. Toroczkai and K.E. Bassler, Network Dynamics: Jamming Is Limited in Scale-Free Systems, *Nature* **428** 716 (2004)

[117] W. Tutte, A Contribution to the Theory of Chromatic Polynomials, *Can. J. Math.* **6** 80–91 (1954)

[118] E. Ullner, A. Zaikin, E.I. Volkov, and J. García-Ojalvo, Multistability and Clustering in a Population of Synthetic Genetic Oscillators via Phase-Repulsive Cell-to-Cell Communication, *Phys. Rev. Lett.* **99** 148103 (2007)

[119] C. van Vreeswijk and H. Sompolinsky, Chaos in Neuronal Networks with Balanced Excitatory and Inhibitory Activity, *Science* **274** 1724–1726 (1996)

[120] J. Vermaseren, New features of FORM, `arXiv:math-ph/0010025`, (2000)

[121] J. Vermaseren, FORM, Version 3.1 (2007)

[122] K. Wagstaff, C. Cardie, S. Rogers, and S. Schroedl, Constrained K-means Clustering with Background Knowledge, *Proceedings of the Eighteenth International Conference on Machine Learning*, 577–584 (2001)

[123] Waterloo Maple Inc., MAPLE, Version 9.01 (2003)

[124] D. Watts and S. Strogatz, Collective Dynamics of 'Small-World' Networks, *Nature* **393** 440–442 (1998)

[125] E.W. Weisstein, Chromatic Polynomial, *MathWorld–A Wolfram Web Resource* `mathworld.wolfram.com/ChromaticPolynomial.html` (2010)

[126] D. Welsh, The Tutte Polynomial, *Random Structures and Algorithms* **15** 210–228 (1999)

[127] D. West, *Introduction to Graph Theory*, Prentice-Hall (1996)

[128] H. Whitney, The Coloring of Graphs, *Proc. Natl. Acad. Sci. USA* **17** 122-125 (1931)

[129] H.S. Wilf *Algorithms and Complexity*. Prentice-Hall (1986)

[130] Wolfram Research Inc., MATHEMATICA, Release 5.2 (2005)

[131] F.Y. Wu, The Potts model, *Rev. Mod. Phys.* **54** 235–268 (1982)

[132] M.K. Yeung, J. Tegnér, and J.J. Collins, Reverse Engineering Gene Networks Using Singular Value Decomposition and Robust Regression, *Proc. Natl. Acad. Sci. USA* **99** 6163–6168 (2002)

[133] D. Yu and U. Parlitz, Driving a Network to Steady States Reveals Its Cooperative Architecture, *Europhysics Letters* **81** 48007 (2008)

[134] D. Yu, M. Righero, and L. Kocarev, Estimating Topology of Networks, *Phys. Rev. Lett.* **97** 188701 (2006)

[135] D.H. Zanette, Propagation of Small Perturbations in Synchronized Oscillator Networks, *Europhys. Lett.* **68** 356–362 (2004)

Statement of Contributions to Articles

1. *Inferring Synaptic Connectivity from Spatio-Temporal Spike Patterns* (authors: Frank Van Bussel, Birgit Kriener, and Marc Timme)

   Candidate: I was responsible for the original solution to the model equations, development and implementation of the reconstruction methodology, and all testing, plus wrote the first draft of the article and major revisions.

   Other authors: Dr. Kriener was involved in the planning as well as neuroscientific and mathematical aspects of the research; she contributed sections to the manuscript and participated in editing and revision. Dr. Timme made the original proposal for the research project and oversaw its progress; he had a large role in the editing of the manuscript including one major revision.

2. *Counting Complex Disordered States by Efficient Pattern Matching: Chromatic Polynomials and Potts Partition Function* (authors: Marc Timme, Frank Van Bussel, Denny Fliegner, and Sebastian Stolzenberg)

   Other authors: Dr. Timme developed the new algorithm (theoretical framework plus original implementation); he wrote the first draft of the article and revisions. Dr. Fliegner and S. Stolzenberg did the FORM implementation and preliminary testing, plus helped in editing and revision of the manuscript.

   Candidate: I was responsible for re-implementation of the new algorithm with incorporation of major additional optimizations; as well, I conducted the major calculations presented in the article and benchmarked running time tests, plus contributed to editing and proofreading of final manuscript.

3. *Chromatic Polynomials of Random Graphs* (authors: Frank Van Bussel, Christoph Ehrlich, Denny Fliegner, Sebastian Stolzenberg, and Marc Timme)

   Candidate: I was responsible for planning, protocols, and implementation of computations used in the article, as well as major preliminary testing, all data-analysis, and working up of results, plus wrote the first draft and major revisions.

   Other authors: Dr. Timme had the original idea to investigate random graphs, managed the project through its various stages, and was heavily involved in planning the article as well as editing and revision of the manuscript. C. Ehrlich and S. Stolzenberg conducted the first small-scale preliminary testing, as well contributed to editing and proofreading. Dr. Fliegner advised and aided in managing the large computational runs, plus helped with editing and proofreading the manuscript.

Curriculum Vitae

Frank Van Bussel
Curriculum Vitae

Address: Leinestraße 12,
37073 Göttingen, Germany
Telephone: (0551) 4996 495
Email: fvb@nld.ds.mpg.de

EDUCATION:

Ph.D. (in progress), Theoretical and Computational Neuroscience program, Göttingen Graduate School for Neurosciences and Molecular Biosciences (2007–present)
Supervisor: Marc Timme
Area: Network Dynamics
Thesis: "Topological Optimization in Network Dynamical Systems"

M.Sc. in Computer Science, University of Toronto (1998–2000)
Supervisor: Mike Molloy
Area: Applied Discrete Mathematics
Thesis: "Towards the Graceful Tree Conjecture"

B.Sc., Computer Science Specialist with Math Major, University of Toronto (1995–1998)

PUBLICATIONS:

Frank Van Bussel, Christoph Ehrlich, Denny Fliegner, Sebastian Stolzenberg, and Marc Timme, Chromatic Polynomials of Random Graphs, *Journal of Physics A: Mathematical and Theoretical*, Volume 43, Number 17, 2010, 175002

Marc Timme, Frank Van Bussel, Denny Fliegner, and Sebastian Stolzenberg, Counting Complex Disordered States by Efficient Pattern Matching: Chromatic Polynomials and Potts Partition Functions, *New Journal of Physics*, Volume 11, February 2009, 023001

Joe Geraci and Frank Van Bussel, A Theorem on the Quantum Evaluation of Weight Enumerators for a Certain Class of Cyclic Codes with a Note on Cyclotomic Cosets, *CERN Document Server* cs.IT/0703129, 2007

Dimitris Achlioptas, Mike Molloy, Cristopher Moore, and Frank Van Bussel, Rapid Mixing for Lattice Colourings with Fewer Colours, *Journal of Statistical Mechanics*, Issue 10 (October 2005), P10012

Dimitris Achlioptas, Mike Molloy, Cristopher Moore, and Frank Van Bussel, Sampling Grid Colorings with Fewer Colors, *LATIN 2004: Theoretical Informatics*, Springer Lecture Notes in Computer Science 2976, 2004, pp. 80–89

Frank Van Bussel, 0-Centred and 0-Ubiquitously Graceful Trees, *Discrete Mathematics*, Volume 277, Issues 1-3, 28 February 2004, pp. 193–218

Frank Van Bussel, Relaxed Graceful Labellings of Trees, *Electronic Journal of Combinatorics*, Volume 9(1), 2002

CONFERENCES / WORKSHOPS / INTERNATIONAL MEETINGS:

(organizational help) Dynamics Days Europe 2009, Göttingen (September 2009)

(poster) "Reconstruction of Network Connectivity from Neural Response Dynamics", Göttingen Graduate School for Neurosciences and Molecular Biosciences (GGNB) Science Day 2009 (November 2009)

(talk) "Counting Complex Disordered States by Efficient Pattern Matching: Chromatic Polynomials & the Potts Partition Function", Dynamics and Statistical Physics Division, German Physical Society (DPG) Spring Meeting, Dresden (March 2009)

(poster) "Counting Complex Disordered States: When the 3rd Law of Thermodynamics Breaks Down", Scientific Symposium, GGNB Grand Opening (November 2008)

(poster) "Symbolic Computation of Chromatic Polynomials", Scientific Advisory Board Meeting 2008, Max Planck Institute for Dynamics and Self-Organization, Göttingen (February 2008)

(poster) "Which Network Connectivities Generate a Given Neural Network Dynamics II: Network Reconstruction", Biological Physics Division, DPG Spring Meeting, Berlin (February 2008)

(poster) "Reconstruction and Design of Networks: Which Network Connectivities Generate a Given Neural Network Dynamics?", 3rd Bernstein Symposium for Computational Neuroscience, Göttingen (September 2007)

(talk) "Sampling Grid Colourings with Fewer Colours", LATIN 2004: Theoretical Informatics, 6th Latin American Symposium, Buenos Aires, Argentina (April 2004)

(participant) SODA 2003: 14th Annual ACM-SIAM Symposium on Discrete Algorithms / ALICE03: 1st Workshop on Algorithms for Listing, Counting, and Enumeration, Baltimore, MD, USA (January 2003)

(talk) "0-Centred and 0-Ubiquitously Graceful Trees", 32nd Southeastern International Conference on Combinatorics, Graph Theory, and Computing, Louisiana State University, Baton Rouge (February 2001)

(visiting Member) Fields Institute for Research in Mathematical Sciences, Special Year on Graph Theory and Combinatorial Optimization (August 1999 - May 2000)

AWARDS:

OGSST: Ontario Graduate Scholarship in Science and Technology (1998 - 2000)

Dean's List scholar in the Faculty of Arts and Sciences, University of Toronto (1998)

TEACHING / ADVISING:
(Codes refer to courses offered by the Department of Computer Science and the Department of Electrical and Computer Engineering at the University of Toronto, St. George campus, except where noted)

As Advisor:

Assisted in supervision of Diplome Thesis (two semesters). Topic: Algebraic Computational Methods for Chromatic Polynomials; Student: Niels Kurz, Faculty of Physics, George-August University, Göttingen. (January 2007 to March 2010)

As Lecturer:

UniVZ 530488 "Networks for Dummies" (Introductory Graph Theory), Max Planck Institute for Dynamics and Self-Organization Seminar Course (Summer 2008)

CSC470: Computer Systems Modelling and Analysis (Fall 2005)

CSC366: Computational Complexity and Computability (Spring 2004)

CSC378: Data Structures and Algorithm Analysis (Summer 2003)

As Teaching Assistant:

CSC488/CSC467: Compilers and Interpreters (Fall 2000, 2001, 2002, 2003, 2004, 2005, as well as Spring 2003)

ECE345: Algorithms and Data Structures (Fall 2005)

CSC373: Algorithm Design & Analysis (Mississauga campus, Spring 2005)

ECE242: Algorithms and Data Structures (Fall 2003, Spring 2004)

ECE1762: Data Structures and Algorithm Analysis (ECE graduate course, Spring 2003)

CSC270: Fundamental Data Structures and Techniques (Summer 1999, Spring 2000, Summer 2002)

CSC364: Computational Complexity and Computability (Summer 2000, Spring 2001, Summer 2001)

CSC190/191: Computer Algorithms, Data Structures and Languages (Spring 2001)

CSC378: Data Structures and Algorithm Analysis (Fall 1999)

CSC238: Discrete Mathematics for Computer Science (Fall 1998, Spring 1999)

CSC148: Introduction to Computer Science (Spring 1998)

RELATED EXPERIENCE:

Research Assistant, Non-linear Dynamics Group, Max Planck Institute for Dynamics and Self-Organization, Göttingen, (September 2006 to September 2007)

Network Technician, Computer Disciplines Facility, University of Toronto. (Summer 1998)

Junior Programmer / Technical Writer, Desktop Innovations Inc., Toronto. (summer 1997)

Appendix A

Network Reconstruction: Additional Material

A.1 Overview of Network Reconstruction Procedures

In this appendix I will briefly discuss the general computational framework for the leaky integrate-and-fire network reconstruction project, as well as some aspects of prior work on the Kuramoto system. Listings for the main programs used in the LIF project are given below in appendix A.2.

Reconstruction research involves the following programs:

1. Simulation code;
2. Reconstruction programs:
   (a) Assemble raw simulation data into linear systems;
   (b) Solver for determined and under-determined linear systems;
3. Medium and large scale testing:
   (a) Network generation and weighting;
   (b) Scripts for multiple parameterized test runs;
   (c) Collect and analyze test run results.

1. Simulations: For both the preliminary Kuramoto network research and for the development of the LIF neuron network reconstruction methods I began by writing simulation programs in MATLAB. The main purpose was to have data for testing the various reconstruction methodologies readily available in a convenient format, since large scale testing and analysis was also done in Matlab. As well, for the LIF reconstruction problem writing the simulation code myself rather than depending on a publicly available system such as BRIAN or NEST [42, 44] definitely gave me an insight into how to solve the LIF inverse problem.

Kuramoto oscillator simulation `kuramoto.m`*:* this is a simple implementation of a numerical solution to the Kuramoto ODE system [64]. It takes the following parameters: the connectivity matrix J, the oscillators' natural frequencies ω, the external driving I (one value for each oscillator), optional initialization values so that runs could be stopped and then restarted, and several performance related parameters such as time-step, exit tolerance, and timeout. It

returned values for the oscillators' phases ϕ and derivatives $\dot{\phi}$ at the point of termination in two vectors, as well as optional diagnostic data.

LIF neuron simulation `spike.m`*:* this is an event-based simulation with somewhat more involved internal workings and interface than the `kuramoto` program described above. The system parameters are in the "working model" units as described in section 1.2.5. Two are mandatory: the connectivity matrix A, and the external driving $\widetilde{I}$, in V/s, one value per neuron. Optional parameters are given in (string,value) pairs; system relevant ones are the leakage γ, the threshold voltage V_T, the delay τ, as well as the reset voltage V_R and refractory period r (for the sake of simplicity we did not test non-zero values for these last two, though in principle the LIF reconstruction methods can be easily modified to work with them). Different values of γ, V_R, etc. can be specified for each neuron, and a different τ for each connection if desired; this has in fact been made use of in the preliminary testing of the noisy/variable reconstruction methods mentioned in section 5.1.2. Initial values for neuron voltages and in fact the entire state of an ongoing simulation run can as well be given as inputs, in order to allow stop/restart runs as with the Kuramoto networks, and a large number of performance parameters (various timeouts, tolerances, and such) are also available.

The `spike.m` simulator returns three data objects. The first is a structure which I will denote as R containing the following information for every neuron at the point of termination: last period length, last spike time, elapsed time from last spike, voltage level, number of spikes so far, difference between last and previous period lengths (used in default exit condition), and projected time until the next spike (if no further arrivals). The second object is the raw spike-time data S, as a set of vectors of times (one vector per neuron; vector element $S_k^{(i)}$ holds the time of k-th spike emitted by neuron i). The last object is the simulator's internal state representation, which is needed if the run is to be restarted from the point of termination.

With respect to its internal workings `spike` is a classic event-based simulation using e.g. a heap-based event queueing system and adaptive memory allocation (both are common optimizations in this context [31]). For the general reader the most relevant aspect is likely the simultaneous event handling, which can affect the network dynamics. To start, delays and refractory periods are always invoked even when the value is zero. The order of simultaneous events in the queue is: spike, arrivals by pulse strength (signed minimum to maximum), exit from refractory state. This guarantees that, first, arrivals coinciding with spikes are always discarded, and second, simultaneous arrivals always have the effect of a single arrival with a pulse strength equal to the sum of the components. The latter in particular means that arrivals will only induce a spike in response if the total strength is suprathreshold regardless of whether any of the individual values are.

2a. Linear systems: Construction of linear systems from the Kuramoto simulation data is a fairly simple process, so a "reconstruction program" separate from the general batch testing programs was never written. Assuming we have N oscillators with natural frequency ω and connectivity J, when `kuramoto` is run M times with driving $I_{i,m}$ on the i-th oscillator during the m-th run, we obtain values for $\phi_{i,m}$ and $\dot{\phi}_{i,m}$ (cf. (1.1) in section 1.2.3 of the introduction). If each run achieves a phase-locked condition then $\dot{\phi}_{i,m} = \Omega_m$, the network's collective frequency at the end of run m. Since in most testing the oscillators were given a uniform natural frequency this value would also be the undriven collective frequency $\Omega = \omega_i$, $i = 1, 2, \ldots, N$. To solve for the Laplacian matrix $\widehat{J}$ (cf. (1.3)) we require matrices D and Θ, where $D_{i,m}$ is $\Omega_m - \Omega - I_{i,m}$, and $\Theta_{j,m} = \phi_{j,m} - \min_i \phi_{i,m}$ (i.e. we normalize Θ to obtain nonnegative values relative to the

slowest oscillator on each run). This gives us the solvable system $D = \widehat{J}\Theta$. In testing the normalization etc. operations were generally done by the calling procedure on a per-vector basis (i.e. for each `kuramoto` run), after which the collected vectors were concatenated into the required matrices.

For neuron reconstruction things are a bit more complicated, most immediately by the fact that there are two major and several minor/preliminary versions of the reconstruction program, which assemble the data required for the linear systems in quite different ways. Chronologically the first major version was what is denoted as the *specialized method* in chapter 2 (code available in section A.2.1 as `spk2net_special`), which uses the per-run data from the R structure. The second major version was what is denoted as the *general method* (`spk2net` in section A.2.2).

The specialized method for LIF networks `spk2net_special`*:* as mentioned in chapter 2, one of the main advantages of using the specialized method when it is applicable is that it simplifies the business of gathering whatever raw data is needed. Since in our simulations the period and last spike times are included in the state information given in the R structure, `spk2net_special` does not in fact do any data gathering at all, but expects the necessary values in pre-packaged form as $M \times N$ matrices (i.e. each row corresponds to a different run, each column to a neuron). Some additional processing is required to turn these values into the desired elapsed times between possible arrivals $t_{j,m} + \tau$ and final spike times $t_{i,m}$. The only aspects of this that might be notable are a) arrivals for some neurons may need to be back-shifted by a period length or two in order to fall into last interspike intervals for other neurons, and b) no assumptions about the nature of the network with respect to phase-locking, inhibitory connections etc. are made use of. In the production version, if after the requisite time shifts there are either missing or multiple $t_{j,m} + \tau$ representatives the program will either terminate or log possible errors in the output (this is omitted from the code below for readability).

The general method for LIF networks `spk2net`*:* In chapter 2 it is mentioned that there are in fact many alternative ways we could go about selecting acceptable interspike interval data and assembling it into a (possibly overdetermined or underdetermined) linear system. Determining which of these alternatives is definitively the best is a subject for further study, so here I will limit myself to the current implementation using the minimum scan-length policy. The idea is that for the incoming connectivity for each neuron i we only scan each data segment long enough to get a representative arrival time from each prospective pre-synaptic neuron j; this does not itself guarantee linear independence, but does guarantee non-zero columns in the solution matrices if this is at all possible, which in practice seems to be all we need when the segments are obtained under different driving conditions or the spiking is sufficiently irregular, and there is not too much spike-induced spiking.

Another aspect of our data aggregation is that expressions derived from all selected intervals of a particular segment are summed to create one row of the solution matrix. This again is not necessary, but it is both more efficient computationally, and seems to very consistently give us much better conditioned linear systems and hence better accuracy. This mode of aggregation is most simply implemented when data from a single run S_0 is pre-divided into time-disjoint segments S_m, $m = 1, 2, \ldots, M$; an additional bonus is that the same code then works for spike times obtained under M different drivings. At some point I would certainly like to investigate whether some more global approach to sifting the intervals might reduce overall data requirements, but so far I have not had the time.

Code for the `spk2net` program is given below. Aside from issues mentioned above, one aspect of the search implementation that the reader might remark upon is that the scans are conducted in reverse, from the end to the beginning of the spike trains. The primary reason for this is that detection of spike-induced spiking happens slightly earlier, saving some computational effort. As well, when data is obtained under M different drivings, if the initial conditions are the same for each run (e.g. in our testing, uniformly zero voltages), then selecting data from the ends rather than beginnings helps insure linear independence in the solution matrices.

2b. System solver: After selection of data and construction of linear systems the Kuramoto network reconstruction and all versions of the LIF neuron network reconstruction revert to essentially the same procedure: if the linear system has N rows the Θ matrix is inverted and multiplied into the relevant RHS, if not singular value decomposition with the L_1-norm optimization is used. In our suite of programs the SVD with optimization is handled by `sparsolve`, which in turn calls `barfit` to do the L_1-norm minimization; code for both is given in appendix A.2 as well. The one major difference with respect to the network type is that Kuramoto networks have continuous interactions so only one Θ matrix is needed to recover the incoming connectivity for all nodes, while for the LIF neuron networks a separate $\Theta^{(i)}$ needs to be constructed for each post-synaptic neuron i (`sparsolve` is structured so as to work for either situation).

The basic idea for how the optimization is applied is given in both chapter 2 and [113], so I will not reiterate it here. It is important to note that at this point using $M < N$ driving conditions or data segments achieves no computational savings, since SVD will result in an $N \times N$ system regardless of the size of the original Θ matrices and the L_1-norm minimization is an extra step that must be executed for every set of incoming connections. Furthermore, while the `barfit` implementation is much faster than generic approaches to L_1-norm minimization available in MATLAB or MATHEMATICA, its running time depends (nonlinearly) not only on N and M but also the connection density P, and is as well somewhat variable across instances where all these parameters are fixed. Since this has not been a priority I do not have an exact formula worked out (though I would like to eventually), but based on the runs for homogeneous inhibitory LIF networks the average running time is $\mathcal{O}(f(M/N)g(P)N^2)$, where both f and g are roughly decreasing (f is not monotonic, while g seems to level out for $P \geq 0.5$).

3a. Network generation: The networks used for debugging, development testing, and large scale testing were almost always strongly connected random directed graphs generated in MATLAB. Appendix C.1 discusses the graph generation and manipulation programs (which were also used for the other line of research) in more detail. As mentioned in section 5.1.1 above, for large scale testing the weighting was done differently for the Kuramoto and LIF networks, with Kuramoto connection weights being fixed and normalized (so that every node had the same total weight of incoming connections), while the LIF connections weights were randomized according to a Gaussian distribution, and for some banks of runs normalized by total incoming weights, while for other banks only adjusted to keep them within preset bounds on minimum and maximum magnitude.

3b. Test runs: Large scale runs were dealt out onto our institute's clusters using compiled MATLAB controller scripts that go through the entire cycle of generating and weighting networks, generating random drivings, running simulations and collecting data, running the reconstruction programs for the (usually multiple) desired M values, and then collecting the

results into `.mat` files for further analysis. During the course of work on the Kuramoto networks the protocol went through some adjustments, until by the time we started testing the LIF reconstruction methods it had settled into the following:

- Fixed system parameters such as delay time τ and threshold V_T were held in a `.mat` file that was distributed along with the compiled script. These parameters were always given in units applicable to the original model (e.g. conductance in Farads, resistance in Ohms etc.; see section 1.2.5) and in the script translated into terms of the working model (γ, $\widetilde{I}$, etc.)
- Varied parameters were given in the command line. This always included N and the relevant connectivity measure (density P, number of connections $|E|$, or average degree K), as well as any parameter of interest for a particular bank of runs, such as the connection weight factor w or the desired driving spread ΔI.
- We usually had a target of 20 trials for each parameter set, which allowed us to gather statistics on the reliability of reconstruction etc. Due to time constraints this target was relaxed for larger N. For some runs these trials were collected by the compiled script, for others they were done separately and the aggregation was done in the next stage.

3c. Data analysis: On completion the compiled MATLAB scripts would deposit final results in `.mat` files. Saved data would not include simulation data or reconstructed networks, since storage requirements would be excessive given the large number of runs done. In all cases sufficient setup data (such as target networks in sparse format and drivings used) would be kept in case we wanted to rerun simulations from scratch, but so far we have not needed to do this since no problems with the implementation of the method itself (i.e. requiring further debugging) have arisen. Saved data included various quality of reconstruction metrics, such as Q_α as described in the article, and also mean and maximum absolute and relative errors (according to the standard definition), and the correlation coefficient of values in the original and reconstructed matrices. Some simulation statistics were saved (though not the entire contents of each generated R structure) as well as extensive running time and reconstruction diagnostic information.

Before we go on I will briefly note here that differing notations are used for directed networks in different disciplines. While this is largely a cosmetic issue easily resolved by taking matrix transposes at appropriate times, there is the potential for confusion so it should be addressed. In the general usage among graph theorists and computer scientists, as well as implementations in general purpose mathematics software such as MATHEMATICA, a directed edge $v_i \mapsto v_j$ is indicated by a non-zero $\{i, j\}$ element in the graph's adjacency matrix A; and when directed graphs are used to model things such as traffic, flow, or causal networks, direction follows the flow or causality (e.g. $A_{i,j} > 0$ and $A_{j,i} = 0$ means that there is a one-way flow from location i to location j). Basically, outputs are counted along rows and inputs along columns.

Among the papers in the references above most of those that involve neuron and oscillator networks reverse this order, so that $A_{i,j}$ is the strength of the $v_j \mapsto v_i$ connection. Not all: at least one physics paper [90] uses the graph theory / MATHEMATICA ordering, and a number of others avoid direct reference to the indexed connectivity matrix entirely. As a graph theorist I myself prefer the graph theoretical convention, and this is how the graph generation programs are implemented, as well as the LIF simulator and various versions of the

LIF neuron reconstruction program from the development, debugging, and large scale testing phases. However, in the description from chapter 2 as well as in the simplified reconstruction code given below the reverse ordering $v_j \mapsto v_i$ more prevalent in the neuroscience community is adopted.

A.2 Selected Code

A.2.1 Specialized (fast) reconstruction method for LIF networks

(MATLAB code)

```
function A = spk2net_special(T, L, II, gamma, tau, vT)
%
% Reconstruction of synchronized N-node inhibitory LIF network.
%   T : MxN, periods (when stable); T(m,j) => run m, neuron j
%   L : MxN, last spike times of each node for each run
%   II : MxN, input currents to each node for each run
%   gamma : leakage
%   tau : delay time
%   vT : threshold potential
%
% Returns reconstructed network in matrix A.

% Note: this code has been simplified from the test version, which contains
%   error checking, diagnostics, timing, and various Matlab optimizations.

[M, N] = size(T);
A = zeros(N, N);
B = vT - II / gamma .* (1 - exp(-gamma * T));
% Arrival times are taken relative to node i's last spike from each run.
% Note: if j last spiked after i did, we need to use its 2nd (or, depending
% on delay times, 3rd) last spike time. Since system is stable this can be
% obtained by subtracting the affected nodes' periods.
for i = 1:N
    tmpD = L(:,i)*ones(1,N) - (L + tau);
    ij = tmpD < 0;
    tmpD(ij) = tmpD(ij) + ceil(abs(tmpD(ij))./T(ij)) .* T(ij);
    Theta = exp(-gamma * tmpD);
    if M == N
        A(i,:) = (inv(Theta) * B(:,i))';
    else
        A(i,:) = sparsolve(Theta, B(:,i))';
    end
end

return;
```

A.2.2 General reconstruction method for LIF networks

(MATLAB code)

```
function A = spk2net(S, II, gamma, tau, vT)
%
% General reconstruction method (heterogeneous connections)
%   S : MxN cell array, S{m,j} contains spiketimes for node j during run m
%   II : MxN, input currents for each run for each node
% Remaining inputs (gamma, tau, vT) have same meaning as for specialized
% spk2net reconstruction program. Returns reconstructed network in
% matrix A.

[M N] = size(S);
Theta = repmat({zeros(M, N)}, 1, N);    % cell array of system matrices
B = zeros(M, N);                        % RHS's of the systems
lenS = cellfun('length', S);            % number of spikes of all trains

% Basic procedure for pulling out data used for reconstruction: scan from end
% for acceptable intervals. A particular acceptable interval is used if it
% has arrivals from nodes that we still need information for; otherwise it is
% discarded. Scan continues until we have data from all possible connections,
% or we run out of spikes. In the latter case entries for the some connections
% are left at 0.

for m = 1:M    % for every run
  for i = 1:N    % for every receiving node
        idx = lenS(m,:);    % indices into spike times for sending nodes
        thetami_tot = zeros(1, N);
        bmi_tot = 0;
        for k = length(S{m,i}):-1:2   % for every interspike interval
            ti = S{m,i}(k); t0 = S{m,i}(k-1);  % current interval is [t0,ti]
            thetamik = zeros(1, N);
            failed = false; needed = false;
            for j = 1:N    % for every possible sending node
                while (idx(j) > 0) && (S{m,j}(idx(j))+tau > (ti + eps*ti))
                    idx(j) = idx(j) - 1;
                end
                if idx(j) == 0
                    continue;    % end of data reached for this connection
                elseif S{m,j}(idx(j))+tau >= (ti - eps*ti)
                    failed = true;    % unacceptable interval, don't use
                    break;
                end
```

(continued on next page)

(continued from previous page)

```
                while (idx(j) > 0) && (S{m,j}(idx(j))+tau > (t0 + eps*t0))
                    if thetami_tot(j) == 0
                        needed = true;
                    end
                    thetamik(j) = thetamik(j) + ...
                                  exp(-gamma * (ti - (S{m,j}(idx(j)) + tau)));
                    idx(j) = idx(j) - 1;
                end
            end        % end loop over potential neighbours j = 1:N
            if needed && ~failed
                bmi_tot = bmi_tot + (vT - (II(m,i)/gamma) * ...
                                                (1-exp(-gamma*(ti - t0))));
                thetami_tot = thetami_tot + thetamik;
                if all(thetami_tot ~= 0)
                    break;    % have data from every possible
                end           % incoming connection, we're done
            end
        end            % end loop over intervals k = length(S{m,i}):-1:2
        B(m,i) = bmi_tot;
        Theta{i}(m,:) = thetami_tot;
    end                % end loop over receiving nodes i = 1:N
end                    % end loop over runs m = 1:M

% Now solve the systems ...
for i = 1:N
    if M == N
        A(i,:) = (inv(Theta{i}) * B(:,i))';
    else
        A(i,:) = sparsolve(Theta{i}, B(:,i))';
    end
end

return;
```

A.2.3 Sparse system solver

(MATLAB code)

```
function x = sparsolve(A, b, s_tol)
% x = sparsolve(A, b, s_tol)
%
% Solve the underdetermined system A*x = b for x, where A is M by N,
% B is M by K, and x is N by K, so as to hopefully maximize the number
% of 0 and near-0 elements in x. This is done using singular value
% decomposition and the barfit L-1 norm minimizer. Will also work
% if M >= N (system is exact or overdetermined); in this case method
% reduces to using Matlab's pinv to obtain the usual solution of Ax = b.
%
% Optional 3rd argument s_tol sets tolerance for singular values. Default
% is value used by Matlab for the pinv function.

[m n] = size(A);
[m2 k] = size(b);
if m ~= m2
    error('Matrices A and b must have compatible dimensions.');
end

[U S V] = svd(A);
if nargin < 3
    s_tol = max(n,m) * S(1,1) * eps;
end
St = S';
ii = find(St);
jj = St(ii) < s_tol;
St(ii(jj)) = 0;
St(ii(~jj)) = 1./St(ii(~jj));

X1 = V * St * U'* b;    % = least squares (pinv) solution to system

if m >= n      % least squares = exact or closest solution
    x = X1;
else           % o.w. V*(y + St*U'*b) are also solutions.
    x = zeros(size(X1));
    for i = 1:k  % want V*y = -V*St*U'*b in as many elements as possible.
        y = barfit(V(:,(m+1):n), -X1(:,i));
        x(:,i) = V(:,(m+1):n)*y + X1(:,i);
    end
end

return;
```

A.2.4 Barrowdale-Roberts L_1-norm minimizer

(MATLAB code)

```
function x = barfit(A, b)
%  x = barfit (A, b)
%
% Finds x which minimizes |Ax - b|_1 (ie. 1-norm of the residuals) using
% the Barrowdale/Roberts algorithm (Comm.of the ACM June 1974)

% Last row and col of tableau store indices so we can extract sol'n when done
% m = number of equations,  n = number of unknowns (m >= n)
[m n] = size(A);
AA = [A b n+[1:m]'; [zeros(1,n); [1:n]] zeros(2,2)];
tmp = find(b<0);
AA(tmp,:) = -AA(tmp,:);  % normalize so that all vals in b col are +ve
AA(m+1,1:n) = sum(AA(1:m,1:n));  % marginal costs

stage = 1;  % If A is not full rank useless columns are swapped to the
start = 1;  %   front and start variable is incremented
indep = 0;  % number of independent basis vectors
x = zeros(n,1);  % solution vector

while stage < 3
    % determine entering vector. Stage 1: if last row element is > n col is
    % slack which has already been swapped out of the basis. Of available
    % cols choose the one with the largest marginal cost (abs because cost
    % goes both ways when you are trying to minimize L1 norm). In stage 2 we
    % are no longer swapping vectors into the basis etc, so we don't need
    % to pre-filter this way.
    if stage == 1
        t1 = find(abs(AA(m+2,start:n)) <= n);
        [mx id] = max(abs(AA(m+1,t1)));
        p_in = t1(id) + start - 1;
    else % stage == 2
        [mx xid] = max(AA(m+1,start:n));
        [mn nid] = min(AA(m+1,start:n));
        if start > n
            stage = 3;
            break;
        elseif mx >= -mn - 2 && mx > eps
            p_in = xid + start - 1;
        elseif -mn - 2 > eps
            p_in = nid + start - 1;
        else % no decent marginal costs left...
            stage = 3;
            break;
        end
    end
```

(continued on next page)

(continued from previous page)

```
if AA(m+1,p_in) < 0
    AA(:,p_in) = -AA(:,p_in);
    if stage == 2
        AA(m+1,p_in) = AA(m+1,p_in) - 2;
        % Note: this -2 fiddling is how B&R avoid the necessity of
        % having 2 constraints for every residual...
    end
end

% determine leaving vector
t1 = find(AA(indep+1:m,p_in) > eps) + indep;
chk = AA(t1,n+1) ./ AA(t1, p_in);    % ratio of bound to potential pivots
while ~isempty(chk)
    [mn id] = min(chk);
    if mn == 0
    % If there are 0's in b col then there is possibility of cycling
    % because of degeneracy; idea here is to use magnitude of AA vals as
    % tie breakers to avoid that (seems to work well in practise, but I
    % have no rigorous proof etc). When mn is not 0 we can depend on
    % numerical errors to perturb the system enough to nudge it out of
    % cyclic behaviour (don't freak, this is what the pros do).
        id2 = find(chk == 0);
        [mn id3] = max(abs(AA(t1(id2),p_in)));
        id = id2(id3);
    end

    p_out = t1(id);
    pivot = AA(p_out, p_in);
    if AA(m+1,p_in) - 2*pivot <= eps;
        break    % values close enough to pivot, so do it...
    else
    % o.w. we reduce marginal costs across the board and check another
    % pivot. This while loop should only do O(m) iterations the first
    % couple of times, after which the marg.costs should be in spitting
    % distance of the A matrix values...
        AA(m+1,start:n+1) = AA(m+1,start:n+1) - 2.*AA(p_out,start:n+1);
        AA(p_out,start:n+2) = -AA(p_out,start:n+2);
        chk(id) = [];
        t1(id) = [];
    end
end
```

(continued on next page)

(continued from previous page)

```
    if isempty(chk)      % most likely because nothing was found in the 1st
        if stage == 1    % place. column is useless, but we still have others
                         % to try, so swap it out of the working space
            AA(:, [start, p_in]) = AA(:, [p_in, start]);
            start = start + 1;
            if indep + start > n
                stage = 2;
            end
            continue;
        else % stage = 2
            disp('CALCULATION TERMINATED PREMATURELY DUE TO ROUNDING ERRORS.');
            x = [];
            return;
        end
    end

    % pivot on p_in, p_out
    t_out = AA(p_out,start:n+1) ./ pivot;
    t_out(p_in - start + 1) = 1 + 1/pivot;
    t_in = AA(1:m+1, p_in);
    t_in(p_out) = pivot - 1;
    AA(1:m+1,start:n+1) = AA(1:m+1,start:n+1) - t_in * t_out;
% this should have the effect of setting
%    pivot -> 1/pivot,
%    a in pivot row -> a/pivot
%    a in pivot col -> -a/pivot
%    a_i,j = a_i,j - a_i,pivrow * a_pivcol,j / pivot
    t = AA(p_out, n+2);    % swap index vals in last row,col of pivot
    AA(p_out, n+2) = AA(m+2, p_in);
    AA(m+2, p_in) = t;
    if stage == 1    % move new basis vector to top of the tableau
        indep = indep + 1;
        AA([indep p_out],start:n+2) = AA([p_out indep],start:n+2);
        if indep + start > n
            stage = 2;
        end
    end

end

% extract solution from tableau (in indep rows of b col, not necessarily in
% any kind of order...)
t = AA(1:indep,n+2);
x(abs(t)) = AA(1:indep,n+1) .* sign(t);

return;
```

Appendix B

Chromatic Polynomials: Additional Material

B.1 Overview of Chromatic Polynomial Procedures

I will outline here the general procedure used for calculating the chromatic polynomials involved in the random graph work as well as some of the original and continuing lattice work that I myself am conducting. Note: the computational framework used by Niels Kurz for the crystal lattice and limiting curve work is quite distinct, incorporating among other things, extensive MATHEMATICA code (for generating FORM-compatible crystal specifications, doing calculations on FORM output, etc.); see [65].

For general graph calculations the procedure is multi-stage:

1. Graph generation or specification (usually in MATLAB);
2. (Optional) calculate an optimized vertex ordering and (if necessary) export reordered graph to text format;
3. Generate FORM code for given graph;
4. Run FORM with generated code to calculate the chromatic polynomial;
5. Calculate roots of polynomial obtained using MPSOLVE numerical root finder;
6. Import root files (text) back into MATLAB for further analysis.

1. Graph generation: Since this is a common feature to both my lines of research technical details are dealt with separately in appendix C.1. For the random graph research banks of test graphs for each parameter set were generated in groups using a common MATLAB script.

2. Vertex reordering: For a general discussion of graph-width metrics and vertex reorderings see appendix C.5.2 below. Currently for serious calculations I use what I call a minimum *termwidth* ordering, which is specially-designed to address computational issues for our worst case graphs.

The termwidth metric: the idea is that we want to reduce the number of delta-terms within the intermediate expressions as much as possible, since the new algorithm spends most of its time searching through them and sorting them. Due to the sorting as well as the possibility of running out of storage space our focus is on reducing the maximum size the expression reaches

over an entire run. In the i-th iteration the number of delta-terms can be affected by several things; what are most under our control via ordering are n_i, the number of forward neighbours of already processed vertices $v_1, v_2, \ldots, v_i$, and m_i, the number of edges between these forward neighbours. While of course we want n_i as low as possible, for a given n_i we would also like m_i as *high* as possible in order to take advantage of cancellation as given in (5.5) of section 5.2.3; ideally, the ordering we obtain would in fact minimize the number of independent sets within this set of forward neighbours. Since counting independent sets is a **#P**-complete problem in its own right, the metric we use is an approximation based on the assumption that the distribution of edges is uniformly random:

$$tmw_i = (n_i + 1) + \sum_{k=2}^{n_i} \binom{n_i}{k} p^{\binom{k}{2}} \qquad \text{where } p = 1 - \frac{2m_i}{n_i(n_i - 1)}\,. \tag{B.1}$$

The $(n_i + 1)$ term here stands for the minimum possible number of delta-terms (one for each forward neighbour as well one delta-free term). For a given N vertex graph G and permutation $\phi = (j_1, j_2, \ldots, j_N)$, the *termwidth* can now be defined for the vertex reordering $\phi(G)$ as $\max_{1 \le i \le N} tmw_i(\phi(G))$.

Close-to-optimal termwidth orderings: Finding a vertex ordering that minimizes termwidth is an **NP**-complete problem (as is, in fact, finding an ordering that simply minimizes the maximum n_i value over all iterations). To find decent orderings I have adopted a hybrid approach that uses branch-and-bound search with a timeout parameter along with multiple restarts from random initial orderings. Branch-and-bound search will always find an optimal solution if given arbitrary time, though it is commonly the case that such an optimal solution is found early in the search and that the bulk of the running time is spent merely confirming this. Since we are not concerned with provably optimal solutions here but are happy with any ordering that is significantly better than the alternatives we can set a timeout value t that kicks in if the period between finding two consecutive solutions that improve the bound exceeds t. Randomized restarts prevent the search from being adversely affected by a bad initial ordering.

For graphs in the 20 vertex range optimal solutions are always found in a single pass with a timeout of $t \ge 10$ ms CPU time; for graphs in the 30 vertex range doing 5 or so restarts with a timeout of 10 to 20 seconds gives good results, but for a set of 20 or so graphs this should be done over lunch or overnight since the timeouts do have a tendency to add up. In my own setup the entire operation (multiple randomized searches) is implemented in the MATLAB program `mintermw`. See figure B.1 for example of a minimum termwidth ordering applied to a small graph.

3. Code generation: This is done using the standalone program `mkfrm`. It accepts a text file in one of various generic graph formats (adjacency list, edge list, or adjacency matrix), converts to an internal graph format, and then outputs FORM code to calculate the chromatic polynomial (examples of desired input formats are included in figure B.1). Since `mkfrm` is coded in C and does mostly text-processing a program listing would be of little interest so is not included here. I will discuss tweaks and optimizations of the FORM code itself below; first we will briefly look at the second major optimization, which is the instituting of *forward checks* to remove terms that will eventually cancel by way of (5.5) in section 5.2.3. Aside from simplifying the process of making FORM code for specific graphs, the calculation of the forward check sets and implementation of the checks is the main "value-added" contribution of `mkfrm`.

The implementation of the forward checking is based on some properties of Kronecker deltas and the way they are made use of by the new algorithm. The first of these is given in (5.5), namely that since all positive powers of δ_x are equal then $\delta_x(1-\delta_x)=0$. At the beginning of the i-th iteration of any calculation the current expression can be written as

$$P(G,q)=\sum_{\sigma_i=1}^{q}\cdots\sum_{\sigma_N=1}^{q}P_i\prod_{\{k,j\}\in E,k\geq i}(1-\delta_{\sigma_k,\sigma_j})\,, \tag{B.2}$$

where P_i is the result of previous application of the algebraic operators, so is made up of complex delta terms having coefficients which are polynomials in q. Note that if any of these terms contains a δ_x that is equivalent to $\delta_{\sigma_k,\sigma_j}$ for some $\{k,j\}$ in the product $\prod$ then that term can be cancelled immediately. However, for reasons of efficiency P_i is the only part of the above expression that can be resident in memory, and if $k\gg i$ then the redundant term will survive and maybe even proliferate for many iterations before it and its descendants are finally removed when the particular $(1-\delta_{\sigma_k,\sigma_j})$ that cancels it is brought into the expression.

An additional complicating but ultimately useful aspect of this is that there are multiple ways that the product of Kronecker deltas can be written that are equivalent under the algebraic operators:

$$\sum_{\sigma_k}\delta_{\sigma_k,\sigma_{j_1}}\delta_{\sigma_k,\sigma_{j_2}}\cdots\delta_{\sigma_k,\sigma_{j_r}}\equiv\sum_{\sigma_k}\delta_{\sigma_k,\sigma_{j_1}}\delta_{\sigma_{j_1},\sigma_{j_2}}\cdots\delta_{\sigma_{j_{r-1}},\sigma_{j_r}}\to\delta_{\sigma_{j_1},\sigma_{j_2}}\cdots\delta_{\sigma_{j_{r-1}},\sigma_{j_r}}\,. \tag{B.3}$$

In the implementation we therefore adopt a normalized higher arity form of the Kronecker delta which collects all sigmas that can be chained as above into a single factor of the term. This augmented delta works as follows:

$$\delta_{\sigma_{i_1},\sigma_{i_2},\ldots,\sigma_k,\ldots,\sigma_{i_{s-1}},\sigma_{i_s}}\times\delta_{\sigma_{j_1},\sigma_{j_2},\ldots,\sigma_k,\ldots,\sigma_{j_{r-1}},\sigma_{j_r}}=\delta_{\sigma_{i_1},\ldots,\sigma_{j_1},\ldots,\sigma_k,\ldots,\sigma_{i_s},\ldots,\sigma_{j_r}} \tag{B.4}$$

i.e. we can merge the product of deltas that contain any common index into a single delta,

$$\delta_{\sigma_{i_1},\sigma_{i_2},\ldots,\sigma_k,\ldots,\sigma_{i_j},\ldots,\sigma_k,\ldots,\sigma_{i_{s-1}},\sigma_{i_s}}=\delta_{\sigma_{i_1},\sigma_{i_2},\ldots,\sigma_k,\ldots,\sigma_{i_j},\ldots,\sigma_{i_{s-1}},\sigma_{i_s}} \tag{B.5}$$

i.e. any redundant indices fall out, and

$$\delta_{\sigma_{i_1},\sigma_{i_2},\ldots,\sigma_k,\ldots,\sigma_j,\ldots,\sigma_{i_{s-1}},\sigma_{i_s}}\times\left(1-\delta_{\sigma_k,\sigma_j}\right)=0\,. \tag{B.6}$$

i.e. any pair of indices matching an edge in the remainder of the graph allows us to remove the term.

Hence the forward check step consists of scanning these augmented deltas for index pairs that belong to edges in the graph. While it results in much simpler looking code if all edges in the graph are scanned every time, the actually relevant edges are necessarily between forward neighbours during a particular iteration, and a great deal of work is saved if the checks are restricted to these. The original motivation for implementing a code generator as opposed to, say, writing a single generic FORM program that read in graph specifications itself, was that this allowed a distinct set of scans to be set up each time.

Additionally, FORM has a quite useful feature that multiple pair-wise scans of an index set can be set up simultaneously, as long as both halves of the target index set are monotonically increasing. Hence an additional task for `mkfrm` is to partition the edges of the forward neighbour

sets into as small a number as possible of conforming target index sets (in the second FORM program in section B.2 these target sets are explicitly given as the sets `k1a1`, `k1b1`, `k2a1`, and so on). While different methods of varying complexity have been tried for this, a simple greedy selection process has given results that are as good as any other method, so that is what is currently used.

4. The FORM code: As can be seen from the programs given in section B.2, the layout and ordering of operations in the generated code have evolved somewhat over time (all variations given here, and some others, are available from `mkfrm` with different command line options).

The first program shown here gives the new algorithm as described in chapter 3 in the most direct way possible in FORM. The reader is referred to [120, 121] for more information on the notation etc. The second program is essentially the first with the two main optimizations implemented: the vertex reordering (visible in the program only in the `L F1 =...` statements giving the forward connections of every vertex), and the forward checking for cancellation described above (as multiple `set` declarations and a one-line subloop in the main body of the program).

Visually the third program (not shown in its completeness) seems to be radically different from the other two; however, the change from the first to the second is actually more substantial. The modifications to the third program would not noticeably affect performance except for worst case graphs e.g. more than 30 vertices, not dense but not extremely sparse, and having a relatively high width by any metric (treewidth, bandwidth, or our own termwidth); they have been implemented largely to help us squeeze by performance bottlenecks posed by memory and disk constraints as well as certain limits arbitrarily put in place by FORM.

Loop unrolling and explicit invocation of check sets: since in FORM loops are implemented textually in the preprocessing stage our own unrolling of the loop does not actually constitute a real change. The reason for doing this is that FORM has an internal limit on the number of variables that can be declared across an entire program, which prevents us from declaring an arbitrarily large number of forward check sets at the head of the file. This limit is high enough that it did not cause problems for random graph runs ($N \leq 36$), but it did come into play with the $4 \times 4 \times 4$ lattices with two or three periodic boundary conditions.

Introducing the $(1 - \delta_{\sigma_i,\sigma_j})$ *factors for each* $\{i, j\}$ *separately:* that is, instead of bringing in all factors involving σ_i simultaneously in the i-th iteration after pre-multiplying them together. The idea is that when all factors for i are brought in at once the current expression P_i temporarily swells to many times its previous size (or subsequent size, after collection and pre-cancellation of terms). When memory resources are sufficiently large this fluctuation is not a serious problem, and can even be dealt with more efficiently sometimes by doing an additional forward check on each subproduct before it is incorporated into the main expression. However, for graphs at the margin of feasibility it really does help that a) the temporary size of the expression never increases to more than twice its previous/subsequent size, and b) a forward check can be instituted each time to keep the ongoing size as low as possible.

The remaining changes are put in place to take advantage of the more predictable nature of the result when only one factor at a time is multiplied in. For example, terms that are the same as in the previous iteration since they come from $1 \times P_i$ instead of $-\delta_{\sigma_i,\sigma_j} \times P_i$ do not need to be checked at all, nor do terms already containing an index for j. Similarly, when the term is

new/modified it saves a bit of time to first check for neighbours of j and only afterwards do the delta merges and general forward checking.

5. Chromatic root sets: Upon completion of the FORM calculations roots for each polynomial are determined using the `unisolve` root-finder which is part of the MPSOLVE package [13]. Among the advantages of MPSOLVE is that like FORM it natively works with arbitrarily large integers. Coefficients for chromatic polynomials of large sparse graphs such as 3D diagonal slices could have so many digits they ran out FORM's line length limit of 256 chars (long lines are wrapped, so no information is lost, but catching and fixing this when it occurred was indeed an annoyance). For the random graph runs the `unisolve` performance parameters were set to give results to 40 decimal places of precision (since complex roots could be very close to the real line at times), and output was saved in GNUPLOT compatible two-column text files.

It should be noted that for almost calculations the FORM statistical output is also saved. The running time data and maximum storage requirements for the random graph research was used extensively in the development of the major optimizations, and in particular determining the reordering metrics which were best for different density graphs.

6. Data analysis: Once the roots are calculated they are reimported into MATLAB for further analysis. For the random graphs root data was processed in several stages: i) automated reading of root files in batches corresponding to parameter sets, saved into primary data structures as is and also divided among integer, real non-integer, and complex roots; ii) extraction of unique complex roots in upper half-plane from the primary structure, which are saved after an initial sorting by distance from the real-axis into a secondary data structure along with some basic statistics such as mean real value etc.; iii) more in-depth analysis of the root data contained in the secondary structure, including arc fitting and extremal value calculations.

B.2 Selected Code

As mentioned above in section B.1, no single FORM program is used, instead programs for specific graphs are automatically generated by `mkfrm`. Here is code for a sample graph using 3 different output options (bare, with basic optimizations, and heavily optimized). Since the code is machine generated it does not contain comments, so I will here give a brief description of what is happening in it. For detailed information on specific keywords, program initialization and data types, and other language constructs see FORM documentation [120, 121].

To start, figure B.1 below shows sample input used to generate the programs in `mkfrm`; the first is a 10 vertex random graph as originally created in MATLAB (used for the bare program), and the second is the same graph subject to a minimal termwidth reordering (used for the other two programs).

In all programs, `ki` where `i` is either a constant or a form preprocessor variable e.g. `'i'` is an index to a vertex of the graph, and `d(ki,kj,...)` is a generalized higher-arity delta term $\delta_{\sigma_i,\sigma_j,\ldots}$ as described in the previous appendix, part 3. The expression `F` holds the eventual result. To enhance efficiency, the polynomial coefficients of each delta term are contained in an accumulator `acc()`, and operations involving the polynomial variable q are done in terms of the program variable `[q-1]` $= q - 1$.

Section B.2.1. The bare program begins by setting up expressions `Fi` which are equal to $\prod_i(1-\delta_{\sigma_i,\sigma_j})$ for individual i values. Each such expression starts as the product of `e(ki,kj)` factors (one for each edge in the graph); before actual processing `(1 - d)` factors are substituted for the `e` factors and expanded, with the products of 2-ary delta terms then collected into generalized deltas. The `.sort` statement has the effect of sorting all unhidden expressions (by term), but it is also necessary to invoke it whenever on wants to move on to a new block of code. After this is done the FORM `Hide` statement is used to protect the `Fi` expressions from further modification.

In the main loop, the statement `Muliply F‘i’` has the effect of multiplying the i-th product of delta terms into the current expression. Upon doing so, products of delta terms that share an index are merged (as described in the previous appendix part 3), indices are sorted via `Symmetrize`, and redundant indices are removed.

Match and replace rules are implemented via `id,ifmatch->1` statements; the first of these corresponds to equation (6) in chapter 3 and the second to (7). The `ifmatch->1` part of the rule sends subsequent processing to the line `Label 1;` i.e. skips the next match-and-replace (this is one way to do “if-then-else” in FORM). The last match and replace, corresponding to (5) in chapter 3, is implemented by simply multiplying `[q-1]+1` $= q$ into any terms that did not match the previously invoked rules.

After the loop is finished the `F` expression consists of a single accumulator containing the result polynomial. The final operations are mostly cosmetic, including rewriting the expression in terms of q rather than $(q-1)$ and specifying the order that terms (of the polynomial) are displayed.

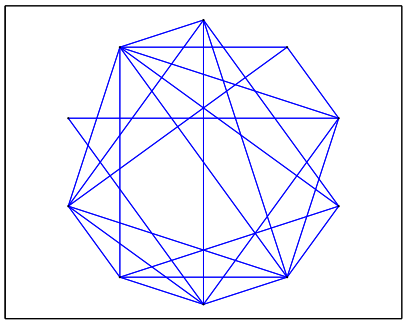
Section B.2.2. The program with basic optimizations works in exactly the same way, except we are now using the second graph in figure B.1 (optimized vertex ordering) and forward checking is in place. The addition of forward checking means that the first part of the code looks radically different, but that is all merely putting definitions in place to use during the main program loop.

The first set of definitions are C-style `#define` preprocessor instructions at the head of the code to set the number of forward checks required during each iteration (recall in the previous appendix part 3 that the forward checks have to be partitioned so as to be compatible with FORM’s fast vectorized multiple index search). This number is given as text (e.g. `"1"` etc.) since the preprocessor is a text processor, so will not take numerical or variable values at the onset (it does the text-to-numerical and -variable conversion as it proceeds).

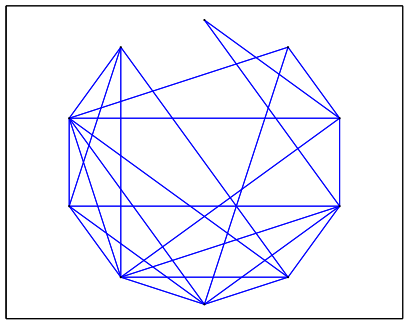
The second, longer set of definitions are the forward check sets themselves. The sets are indices corresponding to vertices in the graph. Each set comes in a pair `kiaj` and `kibj` for the j-th check during the i-th iteration, with `a` and `b` corresponding to endpoints of the edge involved.

Checking is invoked within a subloop after the `Symmetrize` statement in the main loop. If the reader requires further explanation of the mechanism involved in this one line of code it is really recommended that they consult the FORM documentation [121].

Section B.2.3. Here we show a section of the highly optimized production code corresponding to one iteration of the previous program. Note: the term “production code” may not be that well known outside of programming circles, but it refers to finished code that is ready for final application i.e. not still in the development or debugging stage. I do not know of any word



```
1: 4, 5, 6, 8, 10;
2: 3, 8, 10;
3: 2, 5, 6, 9, 10;
4: 1, 5, 7, 10;
5: 1, 3, 4, 6, 7, 8, 10;
6: 1, 3, 5, 7, 8, 9;
7: 4, 5, 6, 8, 10;
8: 1, 2, 5, 6, 7, 10;
9: 3, 6;
10: 1, 2, 3, 4, 5, 7, 8;
```



```
1: 3, 4;
2: 3, 6, 9;
3: 1, 2, 4, 7, 9;
4: 1, 3, 5, 6, 7, 8;
5: 4, 6, 7, 9, 10;
6: 2, 4, 5, 7, 8, 9;
7: 3, 4, 5, 6, 8, 9, 10;
8: 4, 6, 7, 9, 10;
9: 2, 3, 5, 6, 7, 8, 10;
10: 5, 7, 8, 9;
```

Figure B.1: Graph used for sample FORM programs. Left: Original random graph. Right: Same graph, subject to `mintermw` vertex reordering. Below each figure is adjacency list for the graph with respective orderings (`mkfrm` input).

that means the same thing with any wider currency, so I will use it here to denote the version of the program that would be used for any serious calculations.

As explained in part 4 of the previous appendix, the changes here involve unrolling the main loop, unpacking the $\prod_i$ expressions by multiplying in each $(1 - \delta_{\sigma_i,\sigma_j})$ factor individually, instantiating as constants or indices many previously defined variables and sets, and threading the application of the forward checks to avoid checking delta terms we know beforehand do not contain any matches. The match-and-replace rules are invoked as in the previous versions of the program, though only after all relevant $(1 - \delta_{\sigma_i,\sigma_j})$ factors for each i have been multiplied in.

The evaluation of the section is conditional on preprocessor variables `'startv'` and `'endv'` since one aspect of the production code is that it supports a limited version of *call-with-current-continuation* i.e. the ability to temporarily stop the process and then restart it at a different time and even location using previously saved state information. In this case the stopping points are fixed at beginnings of $\prod_i$ evaluations and have to be specified by the user when beginning a run.

Otherwise, this version of the code has the same basic structure as the others, though of course predefinitions of e.g. forward check sets and subsidiary expressions `Fi` for each $\prod_i$ are now gone since they are not needed.

B.2.1 FORM code without optimizations

```
#-

CFunctions acc;
Tensors e,d;
Symbols x,q,[q-1];
Indices k,k0,...,k10;

Off Statistics;
Format nospaces;
Format 80;

L  F1 = e(k1,k4)*e(k1,k5)*e(k1,k6)*e(k1,k8)*e(k1,k10);
L  F2 = e(k2,k3)*e(k2,k8)*e(k2,k10);
L  F3 = e(k3,k5)*e(k3,k6)*e(k3,k9)*e(k3,k10);
L  F4 = e(k4,k5)*e(k4,k7)*e(k4,k10);
L  F5 = e(k5,k6)*e(k5,k7)*e(k5,k8)*e(k5,k10);
L  F6 = e(k6,k7)*e(k6,k8)*e(k6,k9);
L  F7 = e(k7,k8)*e(k7,k10);
L  F8 = e(k8,k10);
L  F9 = 1;
L  F10 = 1;
id e(k?,k0?) = 1-d(k,k0);
repeat id d(k?,?a)*d(k?,?b) = d(k,?a,?b);
.sort

PolyFun acc;
On Statistics;
Hide;
L  F = 1;

#do i = 1,10
Multiply F`i';
repeat id d(?a,k?,?b)*d(?c,k?,?d) = d(?a,?c,k,?b,?d);
Symmetrize d;
repeat id d(?a,k?,k?,?b) = d(?a,k,?b);
id,ifmatch->1,d(k`i',k?) = 1;
id,ifmatch->1,d(k`i',?b) = d(?b);
Multiply acc([q-1]+1);
Label 1;
id  d = 1;
.sort:`i';
#enddo

PolyFun;
On highfirst;
id acc(x?) = x;
id [q-1] = q-1;
Print +f +s;
.end
```

B.2.2 FORM code with basic optimizations

```
#-

#define chk1 "1"
#define chk2 "2"
#define chk3 "2"
#define chk4 "4"
#define chk5 "3"
#define chk6 "3"
#define chk7 "2"
#define chk8 "1"
#define chk9 "0"
#define chk10 "0"

CFunctions acc;
Tensors e,d;
Symbols x,q,[q-1];
Indices k,k0,...,k10;

set k1a1:k3;
set k1b1:k4;
set k2a1:k3,k4,k6;
set k2b1:k4,k6,k9;
set k2a2:k3;
set k2b2:k9;
set k3a1:k4,k6,k7;
set k3b1:k6,k7,k9;
set k3a2:k4,k6;
set k3b2:k7,k9;
set k4a1:k5,k6,k7,k8;
set k4b1:k6,k7,k8,k9;
set k4a2:k5,k6,k7;
set k4b2:k7,k8,k9;
set k4a3:k6;
set k4b3:k9;
set k4a4:k5;
set k4b4:k9;
set k5a1:k6,k7,k8,k9;
set k5b1:k7,k8,k9,k10;
set k5a2:k6,k7,k8;
set k5b2:k8,k9,k10;
set k5a3:k6,k7;
set k5b3:k9,k10;
set k6a1:k7,k8,k9;
set k6b1:k8,k9,k10;
set k6a2:k7,k8;
set k6b2:k9,k10;
set k6a3:k7;
set k6b3:k10;
set k7a1:k8,k9;
```

```
set k7b1:k9,k10;
set k7a2:k8;
set k7b2:k10;
set k8a1:k9;
set k8b1:k10;
Off Statistics;
Format nospaces;
Format 80;

L  F1 = e(k1,k3)*e(k1,k4);
L  F2 = e(k2,k3)*e(k2,k6)*e(k2,k9);
L  F3 = e(k3,k4)*e(k3,k7)*e(k3,k9);
L  F4 = e(k4,k5)*e(k4,k6)*e(k4,k7)*e(k4,k8);
L  F5 = e(k5,k6)*e(k5,k7)*e(k5,k9)*e(k5,k10);
L  F6 = e(k6,k7)*e(k6,k8)*e(k6,k9);
L  F7 = e(k7,k8)*e(k7,k9)*e(k7,k10);
L  F8 = e(k8,k9)*e(k8,k10);
L  F9 = e(k9,k10);
L  F10 = 1;
id e(k?,k0?) = 1-d(k,k0);
repeat id d(k?,?a)*d(k?,?b) = d(k,?a,?b);
.sort

PolyFun acc;
On Statistics;
Hide;
L  F = 1;

#do i = 1,10
Multiply F‘i’;
repeat id d(?a,k?,?b)*d(?c,k?,?d) = d(?a,?c,k,?b,?d);
Symmetrize d;
#do j = 1,‘chk‘i’’
id  d(k‘i’,?a,k1?k‘i’a‘j’[x],?b,k2?k‘i’b‘j’[x],?c) = 0;
#enddo
repeat id d(?a,k?,k?,?b) = d(?a,k,?b);
id,ifmatch->1,d(k‘i’,k?) = 1;
id,ifmatch->1,d(k‘i’,?b) = d(?b);
Multiply acc([q-1]+1);
Label 1;
id  d = 1;
.sort:‘i’;
#enddo

PolyFun;
On highfirst;
id acc(x?) = x;
id [q-1] = q-1;
Print +f +s;
.end
```

B.2.3 FORM production code

(excerpt: lines 351–397 of 474)

```
#if ( ( `startv' <= 7 ) && ( `endv' >= 7 ) )
multiply left 1-d(k7,k8);
id,ifmatch->E7a0,d(k7,k8)*d(k7,?a) = d(k7,k8,?a);
id d(k7,k8)*d(?a,k8,?b) = d(k7,?a,k8,?b);
Goto E7b0;
Label E7a0;
id d(k7,k8,?a,k?{k9,k10},?b) = 0;
id d(k7,k8,?b)*d(?a,k8,?c) = d(k7,?a,k8,?b,?c);
Symmetrize d;
id d(k7,?a,k9,?b,k10,?c) = 0;
repeat id d(?a,k?,k?,?b) = d(?a,k,?b);
Label E7b0;
.sort:v=7(8);

multiply left 1-d(k7,k9);
id,ifmatch->E7a1,d(k7,k9)*d(k7,?a) = d(k7,k9,?a);
id d(k7,k9)*d(?a,k9,?b) = d(k7,?a,k9,?b);
Goto E7b1;
Label E7a1;
id d(k7,k9,?a,k?{k8,k10},?b) = 0;
id d(k7,k9,?b)*d(?a,k9,?c) = d(k7,?a,k9,?b,?c);
Symmetrize d;
id d(k7,?a,k8,?b,k10,?c) = 0;
repeat id d(?a,k?,k?,?b) = d(?a,k,?b);
Label E7b1;
.sort:v=7(9);

multiply left 1-d(k7,k10);
id,ifmatch->E7a2,d(k7,k10)*d(k7,?a) = d(k7,k10,?a);
id d(k7,k10)*d(?a,k10,?b) = d(k7,?a,k10,?b);
Goto E7b2;
Label E7a2;
id d(k7,k10,?a,k?{k8,k9},?b) = 0;
id d(k7,k10,?b)*d(?a,k10,?c) = d(k7,?a,k10,?b,?c);
Symmetrize d;
id d(k7,?a,k8,?b,k9,?c) = 0;
repeat id d(?a,k?,k?,?b) = d(?a,k,?b);
Label E7b2;
.sort:v=7(10);

id,ifmatch->E7,d(k7,k?) = 1;
id,ifmatch->E7,d(k7,?b) = d(?b);
Multiply acc([q-1]+1);
Label E7;
id  d = 1;
.sort:7;
#endif
```

Appendix C

Some Remaining Issues

In this final appendix I will tie up a few loose ends, mostly relating to technical and coding issues which apply equally to both projects; while the main programs and analysis for the projects were quite distinct, many of the same utilities etc. were used for both. Most of these are of no interest theoretically (e.g. routines for plotting multiple data-sets, handling variable-order argument lists, etc.); here I will discuss those where implementation could have a bearing on results (e.g. with respect to random sampling statistics and such).

C.1 Graph Generation and Manipulation Programs

One aspect that was common to both projects was that they both required many specific graphs for both testing during the code development and producing data for the articles. In particular, random graphs of various types were extensively used; hence I put together a suite of simple programs in MATLAB for generating both random and structured graphs as well as performing basic operations and calculations on existing graphs.

All graphs are instantiated as $\{0, 1\}$ adjacency matrices (weighting of connections, when applied, was a separate step). MATLAB sparse matrix format was used for graphs with more than 20 vertices. Programs `gtofile.m` and `filetog.m` were used to export/import graphs from/to MATLAB to/from common text formats (adjacency list, edge list, adjacency matrix, and MATHEMATICA compatible).

Random graph generators: these all had a common interface:

- Number of vertices N, or an existing graph G (as adjacency matrix) that we want to randomly augment.
- Density parameter: edge selection probability p ($0 < p < 1$), a fixed integer number of edges M, or degree K as required by the type of graph.
- Any additional structural parameters as needed.
- (optional) text flag to specify whether desired graph is to be simple (default) or directed. None of the generation programs supported graphs with multiple edges or self-loops; these are easily obtained by from random integer matrices anyways.
- (optional) connectivity specification. By default there is no constraint on connectivity of the graphs generated, but in most applications we preferred graphs to be strongly

connected (every pair of vertices are on a cycle) or at least weakly-connected (every pair of vertices have a set of links between them). When this was requested the generation program would retry a fixed number of times (1000 by default) until the generated graph satisfied the connectivity test; if none was found the empty matrix `[]` would be returned. This is not as efficient as e.g. laying down a random tree or cycle as a connected bed for the rest of the graph, and it could in borderline cases give false negative results, but it insures that the generated graphs are selected uniformly from the set of all weakly-/ strongly-connected labelled graphs sharing the same parameters.

The current suite of random graph programs include the following:

`randgraph.m` : Erdös-Rényi (ER) random graph generator for both $G(N, p)$ and $G(N, M)$ models (depending on whether density parameter is given as real between 0 and 1 or integer between 1 and the maximum possible number of edges). Potential edges are assigned a uniform random number between 0 and 1 and selected if that value is less than or equal to the relevant baseline; for $G(N, p)$ this is of course p, while for $G(N, M)$ it is the random value associated with the M-th order statistic.

`randkdeg.m` : The main use of this program is generation of random K-regular graphs, though the interface allows generation of random graphs with arbitrary degree sequences by giving an N-element degree sequence vector instead of a single integer K as the density parameter. For directed graphs a $2 \times N$ matrix of out- and in-degrees can be given, with ∞ used to flag that the particular out-/in-degree is unconstrained. Graphs are generated according to the commonly used *configuration model*: vertices are assigned stubs according to desired degree, and these are randomly fused to create edges. It is possible for the fuse operation to paint itself into a corner; when this happens the entire generation is restarted from scratch (there is a fixed number of retries before the procedure gives up). The configuration model does not sample uniformly from the space of all labelled graphs with a given degree sequence, but no method currently exists that works as well and as quickly.

`randbpart.m` : Random bipartite graph generator. The size of the parts $\{m_1, m_2\}$ is specified as a third parameter, if omitted they are equally sized or differ by one. The part sets will consist of consecutive vertices in the generated graph; this samples uniformly from the space of all graphs with this kind of partitioning, but note that randomly selecting an integer value for m_1 and then randomly permuting the vertex order of the resulting graph does not uniformly sample from the space of all labelled bipartite graphs. An older version of the program did attempt to do this kind of global sampling, but the part sizes were then necessarily not fixed. For our uses it was more important to be able to set part sizes ahead of time, so I reverted to the simpler format used now, though at some future point I would like to merge to two versions so as to support both fixed and variable part sizing.

`randtree.m` : Generate random trees from scratch or random spanning trees from a given graph G. The second parameter is optional and sets the maximum degree of the tree. Spanning tree generation is via a randomized version of Kruskal's minimum spanning tree algorithm. Note that a low maximum degree may make it difficult or impossible to generate a spanning tree from a particular graph; for example, when maximum degree $= 2$ it is equivalent to the **NP**-complete problem of finding a Hamiltonian path.

`randorient.m` : Generate a random orientation of a given simple graph G (an orientation is a directed graph that is obtained from an undirected graph by giving every edge a direction). As well, random tournaments (orientations of the complete graph on N vertices) can be generated from scratch. Optional arguments allow the user to specify that the graph is to be regular or acyclic (but not both at the same time).

`rewire.m`, `randwire.m` : Rewiring for e.g. creation of Strogatz small worlds from lattices and other structured graphs. The first works by simple random deletion and addition of edges under no other constraints, the second does rewiring by changing endpoints of non-adjacent edges so as to preserve the degree sequence of the graph. As well, I have done some preliminary work on random graphs with small-world or lattice like neighbourhood properties created by randomly connecting points embedded in $\mathbb{R}^n$ using probabilities based on distances (`engraph.m`, i.e. Euclidean nearest-neighbour graph). Creation of "naturalistic" small-worlds e.g. graphs with pronounced clustering that otherwise look random at the neighbourhood level is a topic of interest among network researchers [80]. Since my own work in the end did not heavily depend on small-world or related properties these initial forays were set aside as I progressed with the main projects.

Aside from this I made regular though not continual use of certain simple structured graphs; a single program, `makeadj.m`, was used to generate these. Its output includes paths, cycles, lattices, complete graphs, circulant graphs (the basis for Strogatz's small-world construction), hypercubes, etc.; where relevant these are available in either the directed or undirected version. More involved constructions are available by applying graph operations such as union, Cartesian product, or join to the results (for example, a toroidal lattice via the Cartesian product of two cycles).

C.2 Graph Diagnostic and Analysis Programs

Since for the most part my research involved generating graphs to structural requirements rather than doing calculations on existing / "real world" graphs, there was no real need for an extensive library of routines for graph theoretical analysis. In MATLAB the programs used most were various graph plotters (e.g. with and without arrowheads) and a straightforward implementation of the Floyd-Warshall all-pairs shortest path-lengths algorithm, `floydw.m`; the output (given in a matrix of distances) allows one to determine characteristic path length, maximum shortest path length i.e. diameter, and minimum cycle length i.e. girth.

Additionally some programs were programmed in the C language since they involved computations that MATLAB is not well suited to. The most important of these is of course the `mkfrm` program for generating FORM programs to calculate chromatic polynomials, discussed above in section B.1.

One other standalone graph program of note is `mintw`, a graph treewidth calculator. It takes the same kind of input as `mkfrm` i.e. a text file holding a graph specification as an adjacency list, and returns the treewidth, and optionally the tree decomposition of the graph or an ordering of vertices I will denote as a "general elimination ordering", which is the order the vertices first appear as one traces through the decomposition. For certain classes of graphs such as chordal graphs this ordering is the *perfect elimination ordering*, which has important algorithmic properties; for example, a chordal graph in a perfect elimination ordering will be optimally colored by the polynomial-time greedy coloring heuristic, which otherwise does not perform particularly well.

Calculating the treewidth of a graph is an **NP**-complete problem in its own right. My implementation uses a simplified version of Gogate and Dechter's branch-and-bound treewidth algorithm [43]. In addition to using an upper bound based on the best result found so far (like any branch-and-bound algorithm would) it makes clever uses of lower bounds and pruning rules based on the theory of graph minors; they call it an "anytime" algorithm because if it is terminated prematurely the decomposition returned is usually very good if not known-to-be-optimal.

As mentioned in appendix C.5.2, treewidth is an important diagnostic concept in graph theory with many applications in the study of **NP**-complete problems (the perfect elimination ordering being just one). However, we found that the treewidth calculator was of limited usefulness in the work on chromatic polynomials. To begin with, for lattices there are more direct and meaningful measures of width (the transverse dimension lengths), so as a diagnostic treewidth is somewhat redundant; similarly for dense random graphs we know the treewidth will always be large, so knowing the exact value is a bit like having insult added to injury.

In particular, doing calculations of chromatic polynomials using the general elimination ordering did not result in any speedup, except for the chordal graphs mentioned above (these graphs have little in common with lattices and are very unlikely to be randomly generated, so they were not a focus of any serious work we did). It should be mentioned that the ordering I first used for the non-lattice chromatic polynomial calculations was that given by MATLAB's bandwidth minimizer `symrcm`, which implements a reverse Cuthill-McKee heuristic. For arbitrary graphs `symrcm` outperforms the g.e.o. by a large margin, while our homemade ordering `mintermw` correspondingly outperforms `symrcm`. That said, a treewidth calculator still has potential uses, outside of the context of chromatic polynomials.

C.3 Subsidiary Non-Graph Techniques

In this section I will outline a couple of techniques that I developed for certain aspects of data analysis done for the two projects. While the particular analyses these were used for have not yet appeared in any articles, I believe the techniques themselves might be more generally useful. Note: as far as I know from discussion with people at our institute and internet searches no algorithms or code were available for these specific problems, though I have not done any kind of thorough survey of the areas of mathematics involved.

Transient removal from time series data. Concerning the work on reconstructing LIF networks from noisy data in section 5.1.2 it was mentioned the specialized method could be made to work with noisy data, by calculating the mean period over the entire spike train and then estimating a relative spike time value by linear fitting. For this to be at all reliable initial transients had to be removed from the time series; with both noisy and noiseless homogeneous inhibitory networks under variegated driving, the firing rate for different neurons is widely distributed at first and only eventually converges to the collective frequency.

A traditional fast, "hands-off" method for removing transients from stationary time series is a quite simple technique called *mean-crossing*, used widely in experimental science and performance evaluation of man-made systems such as computer networks. Given a time series $X = X_1, X_2, \ldots, X_r$, the mean crossing point t is the index where the mean value of the initial subsequences $\langle X_1, X_2, \ldots, X_s \rangle$, $s = 1, 2, \ldots, t$ first crosses the global series mean $\langle X \rangle$. One then discards the initial subsequence, retaining $X_t, X_{t+1}, \ldots, X_r$.

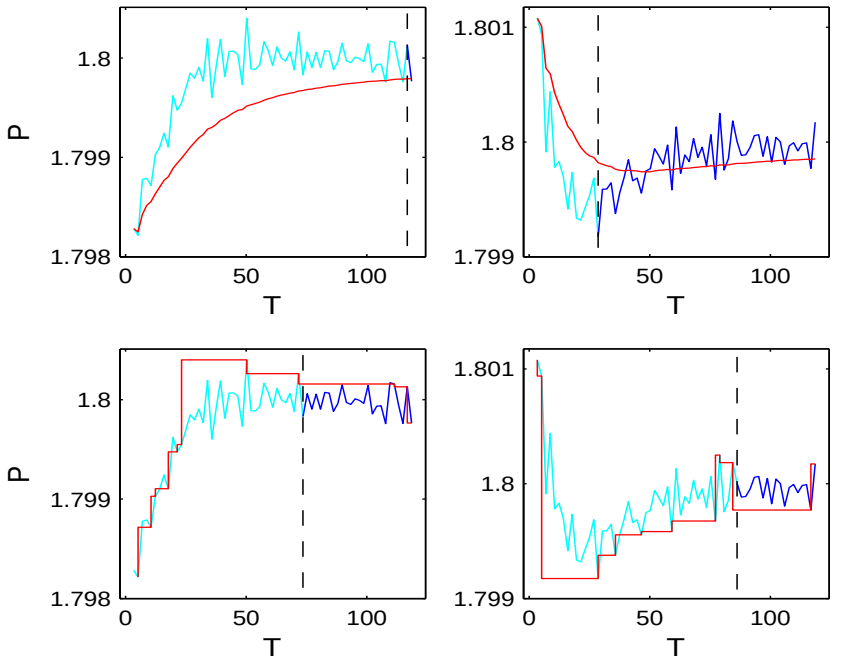


Figure C.1: Comparative results for transient removal methods using two sample time series taken from noisy spike trains that generated figure 5.2 (period P vs. time T). All plots: blue lines indicate data that is kept, cyan data that is discarded. Dashed black line is cutoff point. Top row: transient removal using mean crossing method. Solid red line indicates mean of initial subsequences. Bottom row: transient removal by the reverse cumulative maximum method for the same data. Solid red line here is values that maximally diverge from the mean of the remainder of the series.

This will work well enough if the time series is nice and long relative to the transient, but as can be seen from the top row of figure C.1 for short time series there are several different problems that can arise. One problem, shown on the top left, occurs when the subsequence means trace out a roughly monotonic function; this causes the truncation to happen too far into the series, leaving little to work with. The second problem, shown on the top right, occurs when there is overshooting as the series tries to become stable. A related problem (not shown) occurs when the transient is marked by large fluctuations we want to remove, but not steady progression in one or another direction. In both of these cases the series mean value is crossed too early, and members of the series are retained that could adversely affect further calculations.

After discussion with members of our institute who have worked extensively with time series data (Jan Nagler and Annette Witt) it seemed to me that mean crossing was essentially the only "sight-unseen" transient removal method in common use i.e. the alternative is to remove transients by eye or reasoning about the source of the particular data set. Since this was not an option with respect to my transients I decided to try putting together something of my own. The criteria were that the technique had to run in linear time (like mean crossing), not require preset parameters which had optimal values that depended on the particular time series (such as maximum allowable variance), and avoid metrics that prioritize either as-long-as-possible or as-short-as-possible transients.

I will denote the following heuristic as the *reverse cumulative maximum* method. It is designed to retain a representative section of substantial length from a short time series, though not to minimize transient length, so may be of limited use for long time series (where mean crossing should do an acceptable job anyhow). The idea is that for any sequence of random numbers the probability that the next number is the maximum of what we have seen so far is monotonically nonincreasing; hence the distance between numbers that increase the value of the cumulative maximum is a metric of sorts for stationarity. In particular, if we start to see more rather than less increases in the value of the cumulative maximum divergence from the mean it is because the magnitude of the fluctuations is becoming larger or the sequence is beginning to trend away from the previous mean. For a time series that eventually becomes stationary we can utilize this by working in reverse i.e. from the end to beginning.

Formally: let $X = X_1, X_2, \ldots, X_r$ be our time series, and let u_i be the absolute difference between X_i and the mean of the *remainder subsequence* starting at i and going to the end:

$$u_i = |X_i - \langle X_i, X_{i+1}, \ldots, X_r \rangle| \,. \tag{C.1}$$

Note: all values for u_i are easily calculated together in linear time by taking a cumulative sum of X, going from X_r to X_1. The associated series v_i evaluates our distance function:

$$v_i = \arg\max_j \{u_j : i \leq j \leq r\} - i \,. \tag{C.2}$$

Again, all v_i values together can be determined in linear time by working from the end of the u_i sequence. When $v_i = 0$ then X_i is the maximum or minimum of its remainder subsequence; taken together the indices of all v_i that equal 0 point to increases in the reverse cumulative maximum of the u_i sequence. Between any consecutive pair of these the movement in X is bounded by a fluctuation that occurs later in the time series, so a long sequence of non-zero v values is a reasonable indicator that the series is somewhat stable in that region. Hence we index the beginning of the retained section of the series according to the beginning of the longest sequence of non-zero v_i values:

$$i_{\text{tr}} = \arg\max_i \{v_i : i = 1, 2, \ldots, r\} \,. \tag{C.3}$$

The transient $X_1, X_2, \ldots, X_{i_{\text{tr}}-1}$ is discarded and $X_{i_{\text{tr}}}, X_{i_{\text{tr}}+1}, \ldots, X_r$ is kept.

As can be seen from the bottom row of figure C.1, this certainly gives better results than mean crossing for the data at my disposal. It could be argued that the transients removed are not minimal, though as stated above the concern was only to make sure that enough remained for decent statistics; this is a heuristic for removing transients from relatively large sets of time series when one cannot visually or otherwise verify the results, so it is designed to err on the side of caution in both directions. I have yet to subject this method to any kind of rigorous analysis, for example, look more closely into the behaviour of the cumulative maximum function for various probability distributions. Note: this transient removal technique is implemented within the code for the noisy version of specialized reconstruction method, but as yet I have not coded an independent version (in MATLAB or elsewhere) for general use.

Cluster detection with transitive negative constraints. A great deal had to be left out of chapter 4 due to simple space constraints. Our criteria for what was retained was simply common sense: we presented the results that were most striking in terms of their implications, how simple the fit functions were, and how low the error was. Left out were some things that

I believe are very intriguing, though admittedly not as developed or pointing as clearly to an easily stated conclusion. In particular, the layout of the chromatic root sets suggests that a formula exists that specifies the location of all complex roots, though as of the writing we had come up with nothing as simple as the formulas for the crossing point, the maximum modulus, and the maximum imaginary value.

As part of an initial foray into determining if such a global formula existed part of the data analysis for the project involved partitioning the chromatic roots for sets of polynomials into clusters, so that mean values and variances could be taken to give us an initial estimate for the generic location of each root. For graphs with density $P \approx 0.5$ this was simply a matter of sorting by imaginary value, since their complex chromatic roots are laid out almost vertically in the complex plane. For lower density graphs this was made difficult by the fact that i) the layout was not vertical but arclike, ii) the locations of the roots had more scatter across different graphs, and iii) some graphs had fewer complex roots than others.

Since the arc fits were done independently using data from all 20 sample graphs at once, one option when the associated polynomial had a full set of complex roots was to simply project the roots onto the fitted arc and then sort the roots according to their projection points. This, however, would be problematic if we wanted to compare the centroid locations to the arc placement, which is what we intended. A more scrupulous approach dictated that the clusters be determined independently i.e. we would try to find the natural clustering dictated by the data. This would also avoid problems arising from polynomials that did not have the full set of complex roots (a minority, but generally one or two instances in every sample set).

An immediate problem was that the clustering was under the constraint that no two roots from the same polynomial could be in the same cluster. While MATLAB has two different clustering programs available (`kmeans` for the fast K-means clustering heuristic, and `clusterdata` using the more general hierarchical approach), neither of these support constraints on the clusterings they return. After scanning through the journal databases I found a few previous attempts at constrained clustering; of these, the only one that strictly maintained negative constraints was the COP K-means algorithm of Wagstaff et all [122].

Before I can discuss COP K-means or my own approach any further I should outline the basic K-means algorithm:

1. Given: a set of points $X = x_1, x_2, \ldots$, an integer K, and C, a set of K initial centroid locations.
2. Each iteration do:
   (a) Initially all clusters S_j are emptied.
   (b) Calculate the distance of each point x_i to each centroid C_j.
   (c) Put each x_i into the cluster S_j that minimizes $\text{dist}(x_i, C_j)$, the distance to the current centroid for S_j.
   (d) Recalculate all centroid locations C_j based on the points now contained in each cluster S_j.
3. Continue iterating until the allocation of points X to clusters S does not change.

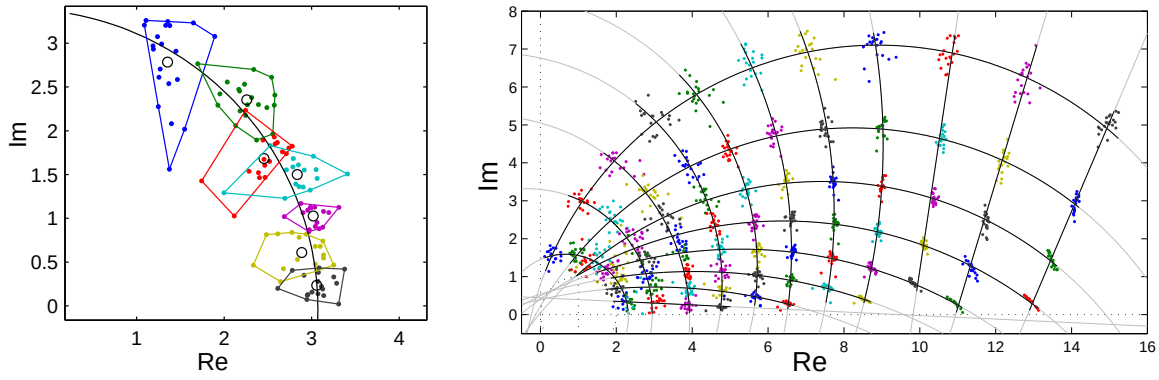


Figure C.2: Clustering of complex root data from chromatic polynomial calculations on random graphs obtained using `rckmeans`. Left: clustering for $N = 18$, average degree $K = 4.5$, with arc fit for same data. Black circles indicate cluster centroids. Right: all clusters (indicated by color) with vertical and horizontal arc fits for $N = 21$.

This heuristic is not guaranteed to find the clustering S that minimizes the point-to-centroid distance across all possible partitions of X into K sets, but its performance is usually quite good with respect to both running time and point placement. Most implementations do not require an initial centroid set (which can be generated out of X easily enough), and the algorithm is not restricted to any particular distance metric. The main limitation noted with it is that it requires the user specify the number of clusters beforehand rather than have it arise naturally out of the data, but for us that is a feature (we do not want more clusters than a full set of complex chromatic roots implies, and we can not have less).

COP K-means is probably the simplest possible extension of the basic K-means algorithm to handle constraints (which is not necessarily a bad thing). Positive constraints are necessarily transitive i.e. if x_i has to be clustered with x_j, and x_j with x_k, then necessarily x_i will be clustered with x_k. Hence we can collect all points with positive constraints into equivalence classes E_ℓ, remove the points from the original X, and then add the centroid of each E_ℓ as a (single) representative value. Negative constraints (which are not necessarily transitive) are handled differently: during step 2c, clusters S_j are checked in increasing order of centroid distance to x_i, which is allocated to the first one which does not already contain a proscribed point i.e. not necessarily the closest. As Wagstaff et al note, the allocation of points during each iteration is then highly dependent on the order they are processed.

For my data the COP K-means algorithm gave very sub-optimal results even for data sets taken from the mid-density range (which as mentioned above do not require a clustering algorithm in the first place). After playing around with a few point reordering ideas, none of which gave great results, I decided that some dynamic reclustering was required. Furthermore, I could utilize the fact that our negative constraints were in fact transitive as well to streamline the extra computations required. I call this variant `rckmeans`, for "COP K-means with resettlement".

We use the same framework as the basic and COP K-means, with an added depth parameter D. Step 2c is modified as follows: for each point x_i let k be its negative constraint class

1. Find the cluster S_{j_1} that does not contain a member of class k and is closest to x_i.
2. For every cluster S_{j_2} that is closer to x_i i.e. $c = \text{dist}(x_i, C_{j_1}) - \text{dist}(x_i, C_{j_2}) > 0$, recursively determine the cost of resettling the current occupant of S_{j_2} that belongs to class k.

3. If any other cluster has a total resettlement cost lower than c, place x_i in the S_{j_3} that minimizes total distances to centroids and resettle the chain of displaced points into the appropriate clusters down the line. Otherwise x_i goes into cluster S_{j_1}.

The parameter D governs the depth of the recursion. If it is set to 0 then the algorithm reverts to COP K-means i.e. no displacement occurs, if it is set to 1 then the current occupant from class k can only be displaced to a k-empty cluster, and so on. If $D = \infty$ then the recursion goes as far as needed to insure that the total distance to centroids over all points is minimum possible for the given centroid locations, which in the worst case will require resettlement calculations that are exponential in the number of clusters K. For the data used in the article a value of $D = 2$ gave quite good results in an acceptable amount of time (K-means itself is after all a heuristic with no global optimality guarantee to begin with).

Figure C.2 (left) shows an example of a clustering obtained using `rckmeans` with $D = 2$ on chromatic root data from a relatively sparse graph. The technique was subsequently applied to the random graph data to obtain generic estimates of all individual chromatic root locations. Ranking of clusters according to centroid displacement along the previously obtained vertical arc fits allows us to associate clusters for different parameter pairs, making it possible to do horizontal arc fits i.e. across density parameter values. Figure C.2 (right) shows these horizontal fits for $N = 21$ along with the vertical arc fits we already had.

C.4 Specialized Chromatic Polynomial Techniques

Here I will outline a couple of results/methods for chromatic polynomials which do not utilize the new algorithm presented in chapter 3, and are otherwise somewhat tangential to the main project.

Closed formula for $3 \times n$ square lattice. While both Shrock and Sokal have results for the limiting curve of the root set of the FBC $3 \times n$ square lattice in the $n \to \infty$ limit, to my knowledge nobody had worked out a *closed* formula i.e. something in the vein of what we have for the $2 \times n$ square lattice:

$$P(L_{2\times n}, q) = (q-1)\, q\, (q^2 - 3q + 3)^{n-1}\,, \tag{C.4}$$

or the $3 \times n$ square lattice with a periodic transverse boundary, which in graph theory terminology would be called an iterated 3-prism:

$$P(Y_{3\times n}, q) = (q-2)(q-1)\, q\, (q^3 - 6q^2 + 14q - 13)^{n-1}\,. \tag{C.5}$$

The simplicity of these formulas arises from the fact that the layers of the strips in question are complete graphs (K_2 and K_3 respectively), allowing us to build from a simple base graph (the 4-cycle C_4 for $L_{2\times n}$ and the ordinary 3-prism Y_3 for $Y_{3\times n}$) using the principle that if the graph can be disconnected by the removal of a complete subgraph then its chromatic polynomial can be easily factorized [127].

Since the $3 \times n$ square lattice with free boundary conditions is not so easily detachable, the closed formula is nowhere as simple as the above. I came upon it during my first year at the MPI (i.e. as a research assistant) while experimenting with various approaches to speeding up the new chromatic polynomial algorithm's calculations. One idea that was checked was whether a specialized version for strip lattices which bundled operations over an entire layer might be

more efficient. My ultimate conclusion was that the new algorithm itself was not improved by doing this. This work, while a non-result in one sense, did have an influence on future directions: with respect to the later versions of the new algorithm (which could be considered going into the opposite direction of breaking down the operations into more atomic parts, see code in appendices B.2.2 and B.2.3); and as well the implementation of the recurrence-finding phase of Niels Kurz's automated limiting curve technique (the new algorithm is used *as is* to determine a set of mutual recurrences i.e. a per-layer bundling, see section 5.2.1 of the thesis).

Before I came to my ultimate conclusion I took a short detour through some alternate approaches for specialized graphs that did not use the new algorithm. One idea I had was to track the operations of a state machine doing a proper q-coloring of a $3 \times n$ lattice row by row. There are two kinds of row colorings: one using 2 colors ("r b r") and one using 3 colors ("r b g"); in each case, we can use some of the colors from the previous row and some new colors (except in the special case of the first row). The two kinds of colorings correspond to states of the machine, and the number of ways we incorporate colors from the previous row (the current state) determines how many ways there are to there are to go to the next state, while the number of colors used in the current row quantifies the number of colors available. The following table gives an example, the simplest transition (from a 2-colored row to a 2-colored row):

r	b	r	$q(q-1)$	used
b	r	b	1	counts
b	α	b	$(q-2)$	
α	r	α	$(q-2)$	
α	γ	α	$(q-2)(q-3)$	.

Here α and γ are colors that are not used in the current row, with respectively $q-2$ and $q-3$ choices. Summing over the counts here gives the total number of ways to go from one row to another. Doing this for the other 3 possible transitions as well gives us the following transition table:

Transition	Label	Count
$(2) \to (2)$	$A =$	$q^2 - 3q + 3$
$(3) \to (2)$	$B =$	$q^2 - 4q + 5$
$(2) \to (3)$	$C =$	$q^3 - 6q^2 + 13q - 10$
$(3) \to (3)$	$D =$	$q^3 - 6q^2 + 14q - 13$

These transitions give us a recursive expression for the number of q-colorings of $L_{3\times n}$. Our state machine has to end in one of its two states i.e. the last row has to be colored with either 2 or 3 colorings. If we let P_n and Q_n be the numbers of ways to color $L_{3\times n}$ so that the last row has respectively 2 or 3 colors, we then have

$$\begin{aligned} P(L_{3\times n}, q) &= P_n + Q_n \\ P_i &= AP_{i-1} + BQ_{i-1} \qquad \text{(C.6)} \\ Q_i &= CP_{i-1} + DQ_{i-1} \qquad \text{(C.7)} \\ P_1 &= q(q-1) \\ Q_1 &= q(q-1)(q-2) \end{aligned}$$

It should be noted that this mutual recurrence is not the same as what would be obtained by the automated recurrence calculator of Niels Kurz, which for the $3 \times n$ FBC lattice involved recurrences for 4 sequences rather than the 2 here.

While the recurrences in (C.6) and (C.7) do allow us to calculate $P(L_{3\times n}, q)$ for particular values of n, in terms of efficiency it is no improvement on the new algorithm. However, with some manipulation the two recurrences above can be rewritten as the following Fibonacci type recurrences:

$$P_i = (A + D)P_{i-1} + (BC - AD)P_{i-2} \tag{C.8}$$
$$Q_i = (A + D)Q_{i-1} + (BC - AD)Q_{i-2} \tag{C.9}$$

Fibonacci type recurrences have an easy solution since the characteristic equation (a quadratic polynomial) can be solved symbolically,

$$\begin{aligned} F_n &= \alpha F_{n-1} + \beta F_{n-2} \\ &= \frac{(\alpha - \phi)F_1 - F_2}{\alpha - 2\phi}\,\phi^{n-1} \;-\; \frac{\phi F_1 - F_2}{\alpha - 2\phi}\,(\alpha - \phi)^{n-1} \\ \text{where } \phi &= \frac{\alpha + \sqrt{\alpha^2 + 4\beta}}{2}\,. \end{aligned}$$

From this point it is merely a matter of substituting in $(A+D)$ for α, q^2-3q+3 for A, and so on, and then add the resulting expressions for P_n and Q_n. Since the forms of (C.8) and (C.9) are so similar the final result turns out to have a fair amount of symmetry:

$$\begin{aligned} H_1 &= (q-2)(q^2-3q+5) \\ H_2 &= (q-1)(q-2)(q^2-3q+3)+2 \\ H &= (q^2-5q+7)\left((q^2-2q+2)(q^2-3q+3)+2\right) \\ P(L_{3\times n}, q) &= \frac{q(q-1)}{2^n}\left(\left(q-1+\frac{H_2}{\sqrt{H}}\right)(H_1+\sqrt{H})^{n-1} + \left(q-1-\frac{H_2}{\sqrt{H}}\right)(H_1-\sqrt{H})^{n-1}\right) \end{aligned}$$

Using the state machine approach I was able to also get mutual recurrences for $4 \times n$, $5 \times n$, and other FBC lattices, though since the number of states in the machine grows exponentially with the width of the lattice these did not turn out to be solvable in the above manner. The approach could be automated for square lattice strips but did not lend itself to a more general application in the same way the automated recurrence generator of Niels Kurz does. That said, this does stand as a precursor to his use of mutual recurrences, and as well gives the most succinct solution to the problem that I know of.

Very fast recursive formula for complete bipartite graphs. The complete bipartite graph $K_{n,m}$ is a mainstay of graph theory. Since it is commonly used to model global resource allocation problems, in terms of applications it shows up in industrial, engineering, and economics contexts much more frequently than in the natural sciences. Complete bipartite graphs are extremal with respect to edge-density of bipartite graphs, and bipartite graphs in general have chromatic roots that stretch further from the real line than ER random graphs (as we can see from figure 5.5 in section 5.2.2). Hence the chromatic root locations for $K_{n,m}$ may have relevance for questions about bounds on chromatic root layout / location, in particular, with respect to the maximum imaginary value $\max Im(z)$ and the maximum root modulus $\max |z|$ (see section 1.3.2 in the introduction).

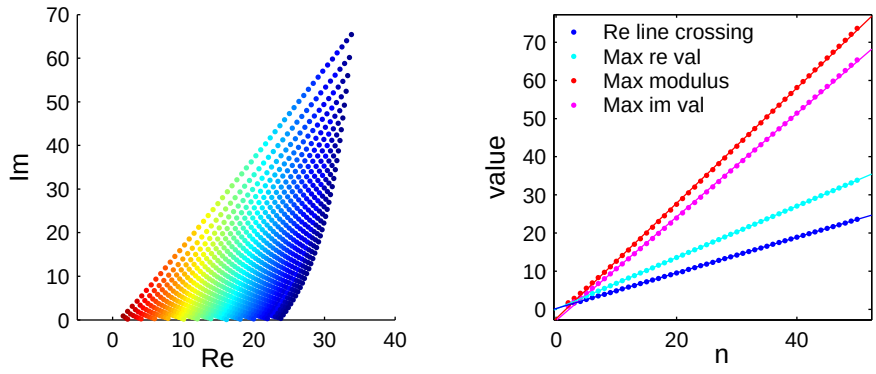


Figure C.3: Left: complex chromatic roots in upper half plane for $K_{n,n}$, $n = 2, 3, \ldots, 50$. Color by n (red: 2 to blue: 50). Right: some per graph root location metrics against n (dots are measured values, lines are fits).

One problem with $K_{n,m}$ when n is close to m (and in particular $K_{n,n}$, the densest bipartite graph on $N = 2n$ vertices) is that these are precisely the worst case graphs for the new algorithm. The edge-density of $K_{n,n}$ is close to one half (as $n \to \infty$ the density $P \to \frac{1}{2}$ from above), and bipartite graphs in general have the property that the neighbourhood of every vertex is an independent set (meaning we do not get as much speedup from the forward check that pre-clears delta terms from the expression). Hence during the preliminary phase of the random graph project, when I was looking at the effect of various structural properties on chromatic root location, I also looked at a specialized chromatic polynomial solver for $K_{n,m}$. The idea is that actually coloring these graphs is quite easy: the sets of colors used for the two partitions have to be disjoint, but within each partition the coloring can be arbitrary.

This gives rise to the following summation formula for the chromatic polynomial of $K_{n,m}$:

$$P(K_{n,m}, q) = \sum_{k=1}^{n} \left\{ {n \atop k} \right\} \frac{q\,!}{(q-k)!} (q-k)^m \tag{C.10}$$

Where $\left\{ {n \atop k} \right\}$ is the Sterling number of the 2nd kind for n and k, which counts the number of ways to partition a set of n elements into k non-empty subsets. Each term within the summation counts the number of ways to color the graph using k of the q colors in the first partition (up to the maximum possible n) and the remaining colors in the other partition. This in fact can be extended to the general complete ℓ-partite graph $K_{n_1,n_2,\ldots,n_\ell}$ by nesting summations, though we will stick with the bipartite formula for the sake of simplicity.

While there is no closed form solution for Sterling numbers in n and k they have a nice recurrence relation,

$$\left\{ {n \atop k} \right\} = \left\{ {n-1 \atop k-1} \right\} + k \left\{ {n-1 \atop k} \right\}, \qquad \left\{ {n \atop 1} \right\} = 1, \qquad \left\{ {n \atop n} \right\} = 1\,. \tag{C.11}$$

The full solution of (C.10) is then fairly easily computed in FORM by precalculating the needed Stirling numbers in a triangular table and then evaluating the summation as given. This was the basis for my original complete ℓ-partite graph chromatic polynomial solver. It achieved a significant speedup over the new algorithm (enough to accomplish my specific aims at the time), but it would still take a while on even bipartite graphs if n was large.

The real speedup occurs when instead of calculating Stirling numbers separately the recurrence is incorporated directly into the summation formula. While neither of the two summations that result is itself a chromatic polynomial, they do give us a per term relation between the summations for $P(K_{n-1,m}, q)$ and $P(K_{n,m}, q)$ which can be exploited. Let $T^{(k)}_{n,m}$ be the k-th term of (C.10) for a particular n and m. Then it simultaneously satisfies the two recurrences

$$T^{(k)}_{n,m} = (q-k)\, T^{(k)}_{n,m-1}, \qquad = \frac{(q-k)^m}{(q-k+1)^{m-1}}\, T^{(k-1)}_{n-1,m} \;+\; k\, T^{(k)}_{n-1,m} \tag{C.12}$$

where of course $T = 0$ if k is outside of the range $1 \le k \le n$. To reach a particular $P(K_{n,m}, q)$ we start with $K_{1,m-n+1}$, which is a tree, and then alternately increment n and m. The efficiency of the implementation depends on how one handles all the $(q-k)^m$ factors when m is sizable; if expansion of these products is deferred as long as possible then the time savings turn out to be very great.

Computations using this recurrence implemented in FORM are extremely fast. Figure C.3 shows chromatic roots for all $K_{n,n}$ for $n = 2$ to 50; $K_{50,50}$ has 100 vertices and edge-density $P \approx 0.5$, but the calculation required only 3 seconds on our cluster machine. I have done computations of the $K_{n,n}$ chromatic polynomials for n up to 100 (200 vertices, 1 minute and 13 seconds), which most likely sets a size record for graphs in that density range (I only plotted the roots for the first 50 because, ironically, FORM would wrap the extremely large coefficients of the later polynomials into multiple lines, requiring time-consuming manual labor to put into a format acceptable to the root finder).

I have not had time to do any kind of in-depth analysis of the results obtained so far, but as figure C.3(right) shows, for $K_{n,n}$ the growth in various metrics of interest is quite linear in n (it should be kept in mind that for these graphs n is not only the partition size, but also the average degree K and maximum degree Δ). The measured scaling for the maximum modulus $\max|z|$ is 1.513Δ, and for the maximum imaginary value $\max Im(z)$ is $0.679N$. These speak to, respectively, Sokal's conjectured upper bound of 1.6Δ for $\max|z|$, and Brown's theoretical result that for every N there is a graph with a chromatic root z such that $Im(z) > \frac{\sqrt{N}}{4}$ (both are mentioned in the introduction, section 1.3.2). As well, by doing ellipse fits to the root sets (arcs do not work as well here as with the ER random graphs) a scaling for the crossing point $X_{N,K}$ of $0.467K$ was obtained. This is less than for the random graphs ($0.582K$) but not orders-of-magnitude less. It should be noted that $X_{N,K}$ is substantially greater than the value of the chromatic number, which is fixed at 2 for all these graphs. In fact, all the larger $K_{n,n}$ graphs had chromatic roots very close to each integer from 2 to $X_{N,K}$, so close that the root finder MPSOLVE was not able to distinguish them even using 40 digits of numerical precision.

C.5 Computational Complexity

C.5.1 Complexity classes

Here I will very briefly review computational complexity theory, or at least the part of it that is most relevant to our implementation issues. All the material here is in any standard undergraduate algorithmics textbook such as Cormen, Leiserson, and Rivest [31]; for readers who really want to sink their teeth into the topic Papadimitriou [85] or Hopcroft and Ullman [49] are the recommended texts for complexity theory and general computability theory respectively.

To start, while we are always concerned with the actual running time of an algorithm on a particular problem instance, to talk about this with any kind of generality requires that we consider things in terms of how running times respond to changes in the size of the input. Since actual running time can vary a great deal we will restrict ourselves to the worst-case running time over all inputs of the same size; we can then say for an algorithm A that its running time is a function $T(n)$ of the input size n. Encoding of inputs will be discussed below, for now we will just assume that the variables n, m, etc. will refer to common sense things like the number of nodes or connections in a graph, clauses or variables in a logic problem, rows or elements of a matrix, etc. Things like constant factors or lower order terms are usually machine or platform dependent, so we will almost exclusively use asymptotic ("big Oh") notation:

$$\mathcal{O}(f(n)) = \{g(n) \ : \ \exists\, c, n_0 > 0 \ \text{ s.t. } \forall\, n \geq n_0, \ \ 0 \leq g(n) \leq cf(n)\}\,. \tag{C.13}$$

Note that this is technically set-theoretical notation: $\mathcal{O}(f(n))$ is the set of all functions that grow no faster than $f(n)$; but by a common abuse of notation we would say, if e.g. $T(n) = 5n^2 + 3n + 1$, that $T(n)$ is $\mathcal{O}(n^2)$ or $T(n) = \mathcal{O}(n^2)$ (I have never met anyone who insisted on the more correct form $T(n) \in \mathcal{O}(n^2)$). But it should be kept in mind that this is an upper bound, and that $3n + 1$, $\sqrt{n}$, $\log n$, and the like are all as well in $\mathcal{O}(n^2)$.

The computational complexity class **P**. An algorithm A with running time $T(n)$ is said to run in *polynomial time* if $T(n) = \mathcal{O}(n^k)$ for some constant k. In such a case one may also say that A is a *polytime* algorithm. Note that this definition includes algorithms with running times described by e.g. $T(n) = \mathcal{O}(n^{100})$ or $T(n) = 10^{20}n$, neither of which would be considered fast; cases like these, however, do not commonly come up in practice, or mathematically affect the classification.

By extension, a problem R is said to be *polynomial-time solvable* if there exists an algorithm A that solves R in polynomial time. The computational complexity class **P** is then simply the set of all polynomial-time solvable problems (to put this into a rigorous formulation requires some formal language theory [31], but the outcome is the same).

Problems in **P** are regarded as "the tractable problems" for partly historical/psychological and partly set-theoretical reasons. Historically, many important algorithms have running times described by low-order polynomials, for example, Quicksort ($\mathcal{O}(n^2)$, where n is the length of the list to be sorted), or Gaussian Elimination ($\mathcal{O}(n^3)$ where n is the number of equations). When running implementations of these or similar algorithms in e.g. MATLAB we do not worry about increasing in the size of the problem instance from n to $n+1$ or $n+2$; though obviously if one were to e.g. double the size of the problem the change in running time would be noticeable.

More importantly, in set-theoretical terms the class **P** is closed under composition. A polytime algorithm that invokes a polytime subroutine will remain polytime; hence requiring polytime transforms and similar operations on input or output do not affect the a problem's membership in the class. This is relevant not only for the definitions of **NP** and **NP**-completeness below, but also because:

– **P** is the same under several important abstract models of computation: the universal Turing machine, Church's *lambda calculus*, the register machine (an idealization of the ordinary computers we actually use), etc. Such models are denoted as *Turing-complete* since they represent the limit of what is known to be achievable by an effective-procedure. While they do differ in native efficiency, it has been shown that all Turing-complete

models proposed so far are able to simulate the operations of all other TCM's with only a polynomial amount of overhead.

– **P** is the same under commonly-used input encodings. How the input is encoded will affect the size of problem instances and therefore the specific behaviour of $T(n)$; however, if T is polynomial in any k-ary encoding for $k \geq 2$ it will be polynomial in all such encodings. This extends to common encodings involving integers in a fixed-word binary format and fixed-precision real numbers, as long as all atomic operations on individual words etc. are polynomial in the number of bits (which is the case for the operations done by CPU's and math coprocessors in ordinary computers as well as basic functions in e.g. the C math library).

Note that the second point here justifies common sense generically optimized encodings only. In particular, many algorithms that would be super-polynomial in the usual fixed-word size binary encoding are polynomial in terms of a *unary* encoding, simply because the size of the input expands exponentially. Relatedly, problems with a succinct description such as a single number or index have to be considered in terms of the binary representation of that number. Hence prime-factorization of a number n is (as of this writing) super-polynomial, even though it can be done in time $\mathcal{O}(n^2)$, since the actual *length* of n on one's computer would be $\log n$ (a unary encoding of any number used for e.g. RSA encryption could not be contained in an ordinary computer memory anyhow).

The computational complexity class **NP**. By definition, if a problem R is known to be in **P** then there must exist a polynomial time algorithm for it. The class **NP** can be regarded as the epistemological limit of **P**, that is, all problems that we can *not* say are *not* in **P**. The original definition of **NP** has it as the set of all problems that do have a polynomial time solution, if implemented on a nondeterministic Turing machine (hence the name, "**N**onderministic **P**olynomial").

Here we will define **NP** in a more down-to-earth way by way of *polynomial time verifications*. To do so we will need to restrict ourselves to the set of *decision problems*, i.e. those requiring a 1-bit {"yes", "no"} answer. We will see shortly that related optimization and other kinds of problems can also be justly described as being in **NP**, though technically (in the complexity literature) **NP**, as well as **P** and the set of **NP**-complete problems are all formal languages consisting of encodings of positive instances of decision problems.

Let R be a decision problem. We first define R^+ as the set of all positive problem instances x of R i.e. those with a "yes" answer. An algorithm $A_R(x, y)$ is defined as a *verification* algorithm for R^+ if

$$\begin{aligned} &\forall\, x \in R^+ \;\; \exists\, y \;\text{ s.t. }\; A_R(x, y) = \text{“yes”} \\ &\forall\, x \notin R^+ \;\; \forall\, y \;\; A_R(x, y) \neq \text{“yes”} \end{aligned} \tag{C.14}$$

The additional input, y, is known as the *certificate* for the problem instance x; it will be discussed below. Note that A is not required to terminate for any x that is not a positive problem instance of R (though presumably if it does it answers "no"); the complement of R^+ does not necessarily contain only the negative instances of R, but also possibly any kind of heterogeneous set-theoretical garbage.

The nature of the certificate in the proofs and definitions is purposely left open, but in practice for a particular decision problem the set of certificates for positive instances are solutions to

the non-decision version of the problem. For an example, consider the decision problem *Graph Coloring*:

$COL(G,k)$: is there a proper coloring C of G using k or fewer colors?

For any positive instance of $COL(G,k)$ the certificate is simply a proper k-coloring of G.

Technical definition of **NP**: the language (i.e. set of positive problem instances) R^+ belongs to **NP** if and only if there exists a polynomial time verification algorithm A_R for R^+ and constant c such that

$$x \in R^+ \;\Leftrightarrow\; \exists\, y \text{ with } |y| = \mathcal{O}(|x|^c) \text{ s.t. } A_R(x,y) = \text{“yes”} \tag{C.15}$$

i.e. A_R verifies R^+ in polynomial time.

Extensions to ordinary usage: to begin with, if the instances x of a decision problem R partition into a set of positive instances defining a language $R^+ \in \mathbf{NP}$ and a set of negative instances which do not it is proper to say that the abstract problem R itself is a problem in **NP**. In this sense, **NP** consists of all decision problems for which a solution can be verified in polynomial time. Note that this verification remains asymmetrical, since (as of this writing) in general there is no negative certificate with polynomially bounded size, i.e. the class **NP** is not known or expected to be closed under complement.

To extend further to various non-decision versions of problems we start by noting that **NP**, like **P**, is closed under composition. More particularly, if $R \in \mathbf{NP}$, and an algorithm A_1 can solve a problem S by making a polynomial number of calls with polynomially bounded input sizes to a subroutine A_2 that solves R, then S is in **NP** as well. This kind of reduction of the problem S to R is known as an *oracle reduction* (though the bounds on calls and input belong to this context; oracle reductions are widely used in complexity and computability theory for classes far outside of **P** or **NP**).

When needed the decision problem oracle is standard method to show that an associated non-decision problem belongs in the extended sense to **NP**. Note that this does not apply to every possible associated problem, since some, such as counting and enumeration problems, are very likely not in **NP** regardless of whether $\mathbf{P} = \mathbf{NP}$ or not. Here we will look at three variations: the optimization problem, the optimal value problem (what value does the solution of an optimization problem give the relevant metric), and what I will call simply the certificate problem (i.e. the most directly related non-decision problem). $COL(G,k)$ will be used as the example problem:

Certificate. $COL_y(G,k)$: provide a proper k-coloring of G (or `NULL` if no such coloring exists)

Optimal value. $\chi(G)$: what is the chromatic number of G i.e. the fewest number of colors needed for a proper coloring?

Optimization. $COL_{\text{opt}}(G)$: provide an proper coloring of G using as few colors as possible.

First we note that all of these are at least as hard as COL, since they all can be used to solve the decision problem (e.g. compare the return value of $\chi(G)$ to k).

The reduction of $\chi(G)$ to $COL(G,k)$ is quite simple: call $COL(G,k)$ for each $k = 1, 2, \ldots$, in order, stopping at the first “yes” answer we receive. Since G must be at most N-colorable, where N is the number of vertices in G, this is a polynomial number of calls. For problems where the metric can range much higher than the input size this idea can still be used: instead of iterating we use binary search to find where the metric changes from “no” to “yes”.

Reductions of certificate problems to decision problems commonly involve progressively filling in a solution by testing slightly modified versions of the input on the oracle. For example, to solve $COL_y(G, k)$ one possible procedure would be:

1. Let $G' \leftarrow G$
2. For $j = 1, 2, \ldots, k$ select any uncolored vertex v from G and assign it color j
   (a) For each uncolored vertex $u \in G$ s.t. uv is not an edge, let G'' be the result of merging the vertices u and v in G', with the merged vertex retaining the label v.
      i. If $COL(G'', k) =$ "yes" then assign u color j (in G) and let $G' \leftarrow G''$.
      ii. Else leave u uncolored and G' as is ...
3. End of loop.

The idea is that a coloring of the graph obtained by merging the vertices u and v is equivalent to a coloring where u and v get the same color. The graph needs to be shrunk progressively in order to make sure that incompatible non-neighbours of each v are not included in the same color class. When the outer loop is done G' should be a k vertex graph (or smaller) and G should be properly k-colored; the coloring defines a homomorphism between G and G', and if the coloring is actually optimal then G' should be the complete graph K_k.

To get from the decision problem to the optimization problem we just do both of the above reductions, the second one using the optimal k value found by the first. It should be noted that if the decision problem is in **P** then reductions of this sort should keep associated optimization and optimal value problems in **P** as well. Hence decision problems in **P** do in fact have polynomial time verifiers for negative problem instances (e.g. check the optimal value against the query value), which means the decision problem restricted version of **P** is known to be closed under complement.

Anyhow, at this point we can ask: **P** = **NP**? The next step is to show why the answer is "Probably not".

NP-*completeness.* In popular intuition, the **NP**-complete problems are the "hard" computational problems, in opposition to the "easy" problems in **P**. The intuition is essentially correct, the complicating factor is that nothing in the definition of **NP** given above warrants further subdivision into "easy" and "hard" pieces. The only real evidence of this division is empirical: the "easy" problems currently have polytime algorithms, while none have been devised for any of the "hard" problems yet, despite a great deal of effort. In principle the status of unresolved mathematical questions does not change even in the face of massive amounts of empirical data (cf. the Riemann hypothesis, the Graceful Tree conjecture, and the like). The beauty of **NP**-completeness theory is that it puts the above-mentioned empirical division into a mathematical framework that has force in even the most rigorous contexts.

To begin, we need a technical definition of "hard" that will give us an entry to the problem. This precludes the obvious choices such as *hard* = *exponential* or *hard* = *superpolynomial* since this just leaves us where we are except with a redundant new term (for the same reason, no one has bothered to technically define *easy* = *polynomial* either). Instead, we will base our definition of hardness on mathematical reducibility (note: not restricted to the oracles mentioned above)

with an algorithmic constraint. The idea is that if problem S can be "easily" reduced to problem R then we can say that S is "no harder" than R.

We will have to return briefly to the positive instance decision problem restricted versions of our sets (i.e. the formal language framework). Let R^+ and S^+ be the sets of positive instances for the problems the decision problems R and S respectively, and let Σ^* be the universe of encodings i.e. including not only the members of R^+ and S^+ but also all instances (positive and negative) for all decision problems plus arbitrary nonsense encodings. S^+ is *polynomial-time reducible* to R^+ (notation: $S^+ \leq_{\mathrm{P}} R^+$) if there exists a polynomial-time computable *reduction function* $f : \Sigma^* \to \Sigma^*$ such that

$$\forall\, x \in \Sigma^* \;:\; x \in S^+ \text{ iff } f(x) \in R^+. \tag{C.16}$$

An algorithm A_f that computes a reduction function f is known as a *reduction algorithm*. Note that outside of the time constraint that these are ordinary/arbitrary mathematical problem reductions; in particular, these reductions are transitive (by polynomial closure under composition).

Now we can define (a type of) hardness: a problem (i.e. a language composed of positive problem instances) R^+ is **NP**-*hard* if

$$\forall\;\; S^+ \in \mathbf{NP} \;:\; S^+ \leq_{\mathrm{P}} R^+. \tag{C.17}$$

Once the technical matters are out of the way this definition extends readily to ordinary usage as with the definition of **NP** itself. The set of **NP**-hard problems contains problems outside of **NP**, so we need one more definition to bring it all home.

Definition: A problem R is **NP**-*complete* if

1. $R \in \mathbf{NP}$,
2. R is **NP**-hard.

In this sense we can rigorously say that the **NP**-complete problems are the *hard* ones in **NP**.

We still need to populate the set of **NP**-complete problems. One thing that made this easier than one might anticipate is that, by transitivity of the $\leq_{\mathrm{P}}$ relation,

$$R \text{ is } \mathbf{NP}\text{-hard and } R \leq_{\mathrm{P}} S \;\Rightarrow\; S \text{ is } \mathbf{NP}\text{-hard}, \tag{C.18}$$

that is, to show any problem $S \in \mathbf{NP}$ is **NP**-complete one merely has to show that any problem R currently known to be **NP**-complete can be reduced to it. Much of the early work (1970's) in complexity theory was devoted to these kinds of reductions, by which the list of known **NP**-complete problems has grown into the thousands; textbooks that deal with **NP**-completeness such as CLR [31] will usually feature many fully worked-out examples. As well, sketch reductions are still a useful tool for algorithm designers such as myself to quickly check if a new problem they encounter may pose tractability issues.

To make use of the above we need at least one problem that can be proven to be **NP**-complete by itself e.g. by showing every other problem in **NP** can be explicitly reduced to it. This is given by:

Theorem (Cook, 1971): The problem *Boolean Satisfiability* ($SAT(V, F)$) is **NP**-complete.

The $SAT(V, F)$ problem is that of determining where there is an assignment of truth values to variables $V = \{x_1, x_2, \ldots, x_n\}$ such that the logical formula F made up of variables in V plus connectives AND, OR, and NOT, is true. The proof is a classic of the genre, involving a non-deterministic Turing machine that solves SAT problems being run on logical expressions that are in essence simulations of other non-deterministic Turing machines solving other problems in **NP**, the basic idea being that these logical expressions always have polynomially bounded length.

A few more brief points. This is the basic outline of the theory of **NP**-completeness, but a few things should noted before leaving the subject.

1. To resolve the famous **P** = **NP** question in the positive direction requires merely providing a polynomial time algorithm for any **NP**-complete problem. No such algorithm has been found so far, which has lead most interested parties to believe that **P** $\neq$ **NP**. The search for a proof in the negative direction has of course also not yielded a resolution to the problem, but in this case it was never expected to be easy in the sense of a simple algorithmic construction.

2. Reminder: **NP**-complete problems are far from the hardest computational problems in existence. **P** and **NP** actually sit at the bottom two rungs of what complexity theorists call the *Polynomial Hierarchy*; if **P** = **NP** this collapses, but the Polynomial Hierarchy itself resides within an even bigger hierarchy that includes monsters such as **EXPSPACE**, i.e. the set of all problems that require at most an exponential amount of memory resources. There are actually normal sounding problems that belong in these higher realms: for example, the problem of deciding whether two regular expressions are equivalent is known to have an exponential lower bound on its worst case running time.

3. Plus note: **NP** does not divide neatly into **P** and **NP**-complete problems, but has regions consisting of problems that are (so far) neither. These problems include *Graph Isomorphism* (decide whether the labelled graphs G_1 and G_2 are equivalent) and *Prime Factorization* (factor a given integer into its prime components).

4. Lastly, there are complexity classes for types of computations we have not been able to touch on here. These include classes for approximation algorithms (example: **FPTAS**, *fully polynomial-time approximation schemes*) and randomized algorithms (example: **RS**, *randomized polynomial time*), and in fact combinations (example: **PRAS**, *polynomial-time randomized approximation schemes*). While many **NP**-complete problems support quite decent approximation schemes, it been has shown that unless **P** = **NP** some are not approximable to even a constant factor (*Graph Coloring* is one of them).

C.5.2 Graph width metrics

Here I will discuss and define some graph width metrics as well as associated vertex orderings, since these have had an impact on my own research as well as being an important part of graph theory.

To start, the word "width" itself has no specific technical definition in graph theory, being reserved for common sense usage, such as to refer to the transverse length of a 2-dimensional lattice. What this native sense of the word shares with various defined graph widths is certain algorithms have difficulty pulling a *wide* graph through a computational bottleneck.

The affected algorithms are those for which the unprocessed neighbours of processed vertices (what I will call the *frontier*) have a direct influence on the computation. This can of course vary between algorithms. For example, Dijkstra's shortest path algorithm [31] requires maintenance of a priority queue containing all the frontier vertices. In any decent implementation individual operations on the queue are logarithmic in its length, and there can never be more than N elements in the queue for an N vertex graph, so this is livable. On the other hand, for the new chromatic polynomial algorithm presented in chapter 3 the number of delta terms in the current expression can become exponential in the size of the frontier, if the frontier vertices constitute an independent set. Avoiding this situation was the main reason I devised my own width metric, *termwidth* and its associated ordering (see part 2 of appendix B.1).

Bandwidth. Aside from the width of a lattice, the graph width metric scientists will be most familiar with is bandwidth, since it is applied to matrices as well as graphs. Like many graph widths, including termwidth, it applies natively to labelled graphs, and in a derived sense to unlabelled graphs according to a minimax formula. For the labelled graph $G_\ell = (V_\ell, E_\ell)$ with ordered vertex set $V_\ell = v_1, v_2, \ldots, v_N$, the bandwith is

$$\mathrm{bw}(G_\ell) = \max_{v_i v_j \in E_\ell} |i - j| \,, \tag{C.19}$$

which is exactly the bandwidth of the adjacency matrix of G_ℓ. Note that any graph except one with no edges whatsoever can be given an ordering that achieves a bandwidth of $N - 1$. The bandwidth for the unlabelled graph G is obtained from the above by

$$\mathrm{bw}(G) = \min\{\mathrm{bw}(G_\ell) \ : \ \ell \text{ is any ordering of } G\}. \tag{C.20}$$

Calculating the unlabelled graph bandwidth is an **NP**-complete problem. A fast heuristic for bandwidth optimal orderings that gives decent results is the *Cuthill-McKee* algorithm for matrices. Apparently for some algebraic applications the reverse of this ordering is preferable (the metric is unchanged), so MATLAB for example supplies a function `symrcm` to calculate this reverse Cuthill-McKee ordering. During the first phase of my own chromatic polynomial research I used this ordering, but re-reversed since the original Cuthill-McKee works better for graphs with density below ≈ 0.5.

Minimizing bandwidth is always good for graph computation since then you can treat your frontier as if it was a bounded-size moving window. However, for some computations we do not really need to worry about the entire frontier at all times, but only those parts of the frontier that could potentially interact (in a computational way) with the vertex currently being processed. This, for example, is the case with the greedy graph coloring algorithm. There are some vertex orderings for which the greedy coloring is the optimal coloring; these are not orderings that minimize bandwidth, but instead minimize the possibility of the algorithm surprising itself as it e.g. colors around a cycle.

Treewidth. Within the graph theory community treewidth is regarded as the most important width metric. The idea of treewidth is that computations on trees are usually very fast since computations on each branch are independent (i.e. tree computations never surprise themselves). The metric is based on a grouping of the vertices called a *tree decomposition*, which associates vertices that have potential interactions while separating out groups that can be dealt with independently. This does not itself define a fixed ordering since, as with trees, an algorithm can start anywhere, as long as it goes down one branch at a time.

Definition of a tree decomposition: let $G = (V, E)$ be a graph on N vertices, $X = X_1, X_2, \ldots, X_t$ be a family of subsets of V, and T_X be a tree on t vertices such that each vertex of T_X is

associated with an $X_i \in X$ (so we can simply call the members of X the vertices of T_X). The tree T_X constitutes a tree decomposition of G if:

1. $V = \bigcup_{k=1}^{t} X_k$
2. $\forall\, v_i v_j \in E \;\; \exists\, X_k$ s.t. $v_i \in X_k$ and $v_j \in X_k$ (C.21)
3. $v_i \in X_{k_1}$ and $v_i \in X_{k_2} \;\Rightarrow\; v_i \in X_k$ for all X_k on the path between X_{k_1} and X_{k_2} in T.

Reminder: every path in a tree is unique. Condition 1 insures that every vertex v is covered, while condition 2 basically states that we are concerned with edge-interaction. Condition 3 is the definitive one: those X_k containing a particular vertex v make up a (connected) subtree of T. We can consider this subtree as putting a tractable topology on v's (algorithmic) interaction with other vertices.

Tree decompositions are not unique, and there is a trivial decomposition where the entire graph is contained in a single tree node that does not tell us anything about the graph. The width of a tree decomposition is defined as the size of the largest X_k, minus 1 (this is in order that trees themselves have an optimal decomposition width of 1). The treewidth of a graph G is then defined as

$$\mathrm{tw}(G) = \min \left\{ \max_{X_k \in X} (|X_k| - 1) \;:\; T_X \text{ is a tree decomposition of } G \right\}. \tag{C.22}$$

Computing the treewidth of an arbitrary graph is of course **NP**-complete. Treewidth itself has a reserved place in computational complexity theory: many **NP**-complete problems can be computed in polynomial time on treewidth-bounded graphs (graph coloring is one of them). Before we get too excited, it should be kept in mind what this really means: it is easy to do computations on very thin graphs. It is true that the **NP**-complete results extend to arbitrary bounds on treewidth, but as this bound grows linearly the embedded constant in the running time grows exponentially. In section 5.2.1 of the thesis we note the same thing happening with the running time of the new chromatic polynomial algorithm on fixed width lattice strips.

Treewidth vs. bandwidth. For any graph, $\mathrm{tw}(G) \leq \mathrm{bw}(G)$, since for any labelled graph G_ℓ we can obtain a valid tree decomposition by collecting every set of consecutive vertices in a moving window of size $\mathrm{bw}(G_\ell) + 1$. The difference is often large, since bandwidth does not respect any kind of branching structure (e.g. trees can have arbitrary bandwidth). *Pathwidth* is a related width metric that uses essentially the same construction as treewidth, but with T_X constrained to be a path rather than an arbitrary tree. As one would expect, the pathwidth $\mathrm{pw}(G)$ of a graph G satisfies $\mathrm{tw}(G) \leq \mathrm{pw}(G) \leq \mathrm{bw}(G)$, and it too is not bounded for trees.

For the $m \times n$ square lattice (FBC), treewidth, pathwidth, and bandwidth are all equal to $\min(m, n)$. While it is easy to find graphs on $N = mn$ vertices with a width by any of these metrics greater than $\sqrt{N}$, these tend to be very dense, such as the complete graph K_N. When density is taken into account lattices are in a sense among the worst case graphs for width metrics of all sorts.