dc_2046_22

Computational complexity of counting and sampling problems in bioinformatics

István Miklós

Rényi Institute

Preface

I have been having a special interest in computational methods since my childhood. I got my first personal computer, a Commodore +4 in my age of 13, and I started writing computer programs. I reinvented pseudo-random numbers and implemented methods generating fancy maps for arcade games based on pseudo-random numbers.

I also loved biology. I spent my childhood in a small village and my regular activity was to play on the fields. The fields were covered with different types of plants, and it was easy to recognize the analogy between the patches of vegetation and the objects appearing on my arcade game maps. Obviously, there had to be some rules behind the patterns, although finding the rules by looking at the patterns was extremely hard. I have been enchanted by the miracle of many possible combinations of the patterns, be these patches of vegetation on a field or objects on a (pseudo)randomly generated map in a computer game, and they heavily influenced my research interest.

Combinatorics and computation – they came hand by hand. At the first plant taxonomy seminar, our teacher gave an overview on classification methods. My immediate question was how to do efficiently a divisive clustering. Obviously, there are $2^{n-1}-1$ ways to split n elements into two parts, and trying all possibilities would be time-consuming. I am still preferring the question "How to do?" to questions like "How does it look like?" or "What are its properties?".

I wrote my PhD on statistical alignment, so I learned dynamic programming, some introduction to probability theory, but the stochastics became my love only when I was a postdoctoral researcher in Oxford and we learned Jun S. Liu's book on Monte Carlo Strategies in Scientific Computing in the book reading club. It was amazing that stochastics might be the answer to much more "How to do?" questions than I could imagine before starting reading that book.

Between 2003 and 2005, I worked on Markov chain Monte Carlo methods, but only at engineering level. It is sufficient to fulfill some easy conditions to guarantee that a Markov chain converge to the distribution of interest, and I loved this game, not caring with the speed of convergence. Bence Mélykúti become my MSc student in 2005, and I offered the speed of convergence of Markov chains as possible research topic for his MSc thesis. We learned together the background theory, and I realized that this is a research topic I want to devote my life to. Indeed, it is a well-balanced mixture of combinatorics, computation and stochastics, my beloved topics in mathematics.

I moved to the Rényi Institute in 2006 September, and I felt that I had to pick up the "definition-theorem-proof" style of pure mathematics, even if I have been mainly working on applied mathematics. Since 2006, I have got deeper and diaper into the research topic that could be described with the following keywords: stochastic approximate sampling and counting (FPRAS and FPAUS), speed of convergence of Markov chains, #P-complete counting problems.

I attended to a Dagstuhl seminar in 2016 February, co-organized by Michael Albert, Miklós Bóna, Einar Steingrimsson and myself. Miklós Bóna emphasised that a large fraction of enumerative combinatorists know very little about computational complexity and argued for the need of a textbook on computational complexity of counting and sampling. I very much agreed with him, and suggested several experts to ask them to write such a book. After all of them declined due to miscellaneous reasons, I accepted to write a monograph about that topic published in the Discrete Mathematics and Its Applications series by CRC Press. I wanted to give a full spectrum of computational complexity in that book. That is, I also gave a detailed overview about easy counting and sampling problems that is typically missing in other books on computational complexity of counting and sampling.

In this thesis, I also give a full spectrum of computational complexity. I introduce an algebraic approach to dynamic programming and I show how two of my early works fit into this framework. Two other easy counting problems also solvable by dynamic programming are also presented. Then rapidly mixing Markov chains are presented that can be used to almost uniformly sample combinatorial objects. The combinatorial objects for which rapidly mixing Markov chains are given are realizations of degree sequences, realizations of joint degree matrices and most parsimonious genome rearrangement sorting scenarios under a certain rearrangement model called Double Cut and Join. Negative results are also presented in this thesis. There is a wide

spectrum of possible negative results. One possible negative result is that a widely used Markov chain is torpidly mixing and thus cannot be used for almost uniform sampling. A proof of torpid mixing of such a Markov chain is given. Another possible negative result is a proof that a counting problem is in #P-complete, which is at least as hard as NP-hard. There are combinatorial objects that are not only in #P-complete but they cannot be sampled even nearly uniformly neither the number of them can be reasonably approximated under standard complexity assumptions (assuming that RP $\neq$ NP). One-one example of such counting problems are introduced in this thesis. At the end of the thesis, some partial results are presented. That is, for some combinatorial objects for which the corresponding counting problems have unknown complexity, Markov chains are given that have small diameter (and thus, they are irreducible), and have certain properties that suggest that they might be rapidly mixing on the combinatorial objects in question.

For such a diverse topic, it is inevitable that the thesis should have a detailed introduction. Both the computational complexity and the mathematical models defining the combinatorial structures are described in sufficient detail. Another aspects of this thesis is that it covers a large amount of publications. These publications are given at the end of the thesis in a separate reference list. The majority of these publications are written by co-authors. At the beginning of each chapter, the theorems, lemmas and concepts given by myself is clearly indicated.

June 2022

István Miklós
Rényi Insitute
Eötvös Loránd Research Network

Contents

Chapter 1

Preliminaries

In applied mathematics, different aspects of real world objects are modeled with mathematical objects in order to understand the properties of the real world objects and answer certain questions emerging. There is a natural trade-off between the simplicity and the accuracy of the model. Usually, computations are faster in a simpler model. The computational burden is striking especially in case of discrete mathematical, combinatorial models, where the combinatorial explosion might cause that naïve computations are not feasible even for moderate size of input data.

Discrete mathematical models naturally emerge in biology. Charles Darwin already depicted the evolution of species with a tree graph [24]. Sturtevant and Tan [86] as well as Sturtevant and Novitski [85] studied genome rearrangements in Drosophyla species. They clearly stated a problem that today is considered as a computational biology problem. The problem was to find for each genomic sequence the "minimum number of successive inversions required to reduce it to the ordinal sequence chosen as 'standard.'". They observed that "The mathematical properties of series of letters subjected to the operation of successive inversions do not appear to have been worked out, so that we are so far unable to present a detailed analysis." The conclusion was that " For numbers of loci[1] above nine the determination of this minimum number proved too laborious, and too uncertain, to be carried out. " It is worth mentioning that the demand for the solution of the stated computational problem was given before building the first electronic comput-

[1] A *locus* (in plural: loci) is a specific position in the chromosome where a particular gene or genetic marker is located. In the time the cited scientific paper was written, loci were determined by dyeing the chromosome thus obtaining a barcode-like structure.

ers and before the discovery of the DNA structure! In their seminal paper, Zuckerkandl and Pauling considered biological macromolecules as evolutionary fringerprints [101]. Biological macromolecules can naturally be modeled as sequences over a finite alphabet, and their mutations can be modeled as string operations.

The study of the above-mentioned discrete mathematical models initiated research on algorithmics of discrete mathematical objects and resulted in the emergence of a discipline known as computational biology or bioinformatics. Early research in bioinformatics focused on optimization problems following the principle of parsimony. That is, the most acceptable explanation of a phenomenon, biological observation is the simplest, involving the fewest changes. Clearly, Sturtevant and Novitski followed the principle of parsimony when they studied Drosophyla species. They stated the problem in 1941 and their algorithmic problem was solved first in 1995, and was published only in 1999 [45]. The first efficient algorithm computing the minimum number of insertion, deletion and substitution operations to transform a (biological) sequence into another one was developed in 1970 and published by two biologists [68]. More rigorous description were published shortly in mathematical journals [76, 79, 94].

Since the most probable conformation of a biological macromolecule is the one with the minimal free energy in a Boltzmann distribution, it is also natural to ask which conformation has the minimal (free)energy. In very simple models of RNA structures, this is the RNA structure with the maximum number of base-pairs [69]. More involved models were developed based on experimental measurements [91] for which also efficient algorithms exists to find the structure with minimal free energy [77, 62].

Starting with the seminal paper of Felsenstein [33] probabilistic methods become widespread in bioinformatics. This involves, among others, maximum likelihood methods [33, 90], Bayesian statistics [99, 50] or computing the partition function of the Boltzmann distribution of RNA structures [65]. In a few, very simple models, exact computations are possible [33, 90, 65], however, in most of the cases, approximate computations are used [50]. One of the most commonly used approximate computations is the Markov chain Monte Carlo method [66, 44]. The Metropolis-Hastings algorithm can transform any Markov chain satisfying some mild conditions into another Markov chain that converges to a prescribed distribution. One of the crucial point is the speed of convergence, that is, how many steps are required in the Metropolis-Hastings algorithm to get close to the required distribution. In

most of the cases, this is an extremely hard question. When theoretical results are not possible, heuristic methods are used to estimate the speed of convergence. Such methods can estimate speed of convergence only for each particular case, and a few positive results are no guarantee that the Markov chain mixes rapidly for all possible input data.

A possibly simpler problem is to prove rapid mixing of a Markov chain that converges to the uniform distribution of the most parsimonious cases/solutions. It is already a quite challenging task to sample (almost) uniformly most parsimonious genome rearrangement scenarios. Generating random graphs with prescribed degree sequences and/or other prescribed properties is important in hypothesis testing.

In practice, approximate computations are sufficient for several reasons. First, the mathematical models are only models of the real world. Any method whose approximation factor is negligible compared to how much the mathematical model deviates from the reality can be considered as a perfect solution. If samples are generated to construct a background distribution for hypothesis testing, then some systematic error might be introduced during that process. This systematic error is tolerated if it is negligible compared to the sampling error of the data subject to hypothesis testing.

For a large class of computational problems, approximate sampling and approximate counting have the same computational complexity [54]. That is, either both of them or none of them have efficient algorithmic solutions. Theoretical computer scientists developed the computational complexity of counting and sampling and defined computational classes. A nice bridge between pure and applied mathematics is the complexity classification of mathematical models. While significant progress has been achieved on classifying decision and optimization problems, we know much less about the computational complexity of counting and sampling problems appearing in applied mathematics. This thesis gives an overview of an almost twenty year work of the author on classifying counting and sampling problems appearing in bioinformatics and other disciplines of life sciences. In the Introduction, we first give a brief overview of computational complexity focusing on counting and sampling, then an overview of the mathematical models studied. We give a brief summary of the results presented in this thesis at the end of this chapter.

1.1 Computational complexity

In computational complexity theory, we distinguish decision, optimization, counting and sampling problems. Although this thesis is about the computational complexity of counting and sampling, counting and sampling problems are related to decision and optimization problems. Counting problems are always at least as hard as their decision counterparts. Indeed, if we can tell, say, the number of perfect matchings in a graph G, then naturally we can tell if there exists a perfect matching in G: G contains a perfect matching if and only if the number of perfect matchings in G is at least 1.

Optimization problems are also related to counting and sampling. For example, it is hard to count the cycles in a directed graph as well as sampling them since it is hard to find the longest cycle in a graph [54]. This might be surprising in the light that finding a cycle in a graph is an easy problem. There are numerous other cases when the counting version of an easy decision problem is hard since finding an optimal solution is hard in spite of the fact that finding one (arbitrary) solution is easy.

When we are talking about easy and hard problems, we use the convention of computational complexity that a problem is defined as an easy computational problem if there is a polynomial running time algorithm to solve it. Very rarely we can unconditionally prove that a polynomial running time algorithm does not exist for a computational problem. However, we can prove that no polynomial running time algorithm exists for certain counting problems if no polynomial running time algorithm exists for certain hard decision problems. This fact also underlines why discussing decision problems is inevitable in a thesis about computational complexity of counting and sampling.

When exact counting is hard, approximate counting might be easy or hard. Surprisingly, hard counting problems might be easy to approximate stochastically, however, there are counting problems that we cannot approximate well. We conjecture that they are hard to approximate, and this is a point where stochastic approximations are also related to random approaches to decision problems. Particularly, if no random algorithm exists for certain hard decision problems that run in polynomial time and is any better than random guessing, then there is no efficient good approximation for certain counting problems.

Here, we give a brief introduction to computational complexity and show how computational complexity of counting and sampling is related to com-

putational complexity of decision and optimization problems.

1.1.1 General overview of computational problems

A *computational problem* is a mathematical object representing a collection of questions that computers might be able to solve. The questions belonging to a computational problem are also called *problem instances*. An example of a *decision problem* is the triangle problem which asks if there is triangle in a finite graph. The problem instances are the finite graphs and the answer for any problem instance is "yes" or "no" depending on whether or not there is a triangle in the graph. In this computational problem, a triangle in a graph is called a *witness* or *solution*. In general, the witnesses of a problem instance are the mathematical objects that certify that the answer for the decision problem is "yes". An example for an *optimization problem* is the clique problem which asks what the largest clique (complete subgraph) is in a finite graph. The problem instances are again the finite graphs and the solutions are the largest cliques in the graphs.

Any decision or optimization problem has its natural counting counterpart problem asking the number of witnesses or solutions. For example, we can ask how many triangles a graph has, as well as how many largest cliques a graph has.

Computational problems might be solved with algorithms. We can classify algorithms based on their properties. Algorithms might be exact or approximate, might be deterministic or random, and probably their most important feature is if they are feasible or infeasible. To define feasibility, we have to define how to measure it. Larger problem instances might need more computational steps, also called running time. Therefore, it is natural to measure the complexity of an algorithm with the necessary computational steps as a function of the input (problem instance) size. The size of the problem instance is defined as the number of bits necessary to describe it. A computational problem is defined as *tractable* if its running time grows with a polynomial function of the size of the input, and *intractable* if its running time grows exponentially or even more with the size of the input. This definition ignores constant factors, the order of the polynomial and typical input sizes. This means that theoretically tractable problems might be infeasible in practice, and *vice versa*, theoretically intractable problems might be feasible in practice if the typical input sizes are small. In practice, most of the tractable algorithms run in at most cubic time, and their constant factor is

less than 10. These algorithms are not only theoretically tractable but also feasible in practice. The given definitions of tractable and intractable problems do not cover all algorithms as there are functions that grow faster than any polynomial function but slower than any exponential function. Such functions are called *superpolynomial* and *subexponential*. Although there are remarkable computational problems, most notably the graph isomorphism problem [6, 7], which is conjectured to have superpolynomial and subexponential running time algorithms in the best case, such problems are relatively rare, and not discussed in detail in this thesis.

Observe that both the size of the problem instance and the number of computational steps are not precisely defined. Indeed, a graph, for example, might be encoded by its adjacency matrix or by the list of edges in it. These encodings might have different numbers of bits. Similarly, on many computers, different operations might have different running times: the time necessary to multiply two numbers might be much more than the time needed to add two numbers. To get rigorous mathematical definitions, theoretical computer science introduced mathematical models of computations; the best known are the Turing machines. In this thesis, we avoid these formal descriptions of computations. The reason for this is that we are interested in only the *order* of the running time, and constant factors are hidden in the O (big O, ordo) notation. Even if sizes are defined in different ways, different definitions almost never have exponential (or more precisely, superpolynomial) gaps. For example, if a graph has n vertices, then it might have $\Omega(n^2)$ edges. However, it does not make a theoretical difference if an algorithm on graphs runs in $O(n^3)$ time or $O(m^{1.5})$ time, where n is the number of vertices and m is the number of edges: both functions are polynomials. An example for the difference when there is an exponential gap between the two concepts of input sizes is when we distinguish the value of the number and the number of bits necessary to describe a number. When we would like to emphasize that the input size is the value of the number, we will say that the input numbers are given *in unary*. A typical example is the *subset sum* problem, where we ask if we can select a subset of integers whose sum is a prescribed value W. There is a dynamic programming algorithm to solve this problem whose running time is polynomial with the value of W. However, it is a hard decision problem if W is not given in unary [56].

1.1.2 Deterministic decision problems: P, NP, NP-complete

Definition 1. *In computational complexity theory,* P *is the class that contains the decision problems solvable in polynomial time.*

For example, the perfect matching problem that asks if there is a perfect matching in a simple graph [29] and the primality testing that asks if a natural number is a prime [2] are both non-trivial examples of computational problems that are in P. It is worth mentioning that deciding if a number is a prime can be decided in time that is upper bounded by a polynomial of the number of digits of the input number. That is, the input is also not encoded unary for this problem.

We are going to define the NP class. Formally, the complexity class NP contains the problems that can be solved in polynomial time with non-deterministic Turing machines. The name NP stands for "non-deterministic polynomial". Since we do not introduce Turing machines in this thesis, an alternative, equivalent definition is given here.

Definition 2. *The complexity class* NP *contains the decision problems for which solutions can be verified in polynomial time.*

This definition is more intuitive than the formal definition using Turing machines. The k-clique problem that asks if there is a clique of size k in a graph and the feasibility problem that asks if there is a list of integer numbers satisfying a set of inequalities are both non-trivial examples of computational problems that are in NP.

One of the most important and unsolved questions in theoretical computer science is whether or not P is equal to NP.

In many cases, finding a solution seems to be harder than verifying a solution. There are problems in NP for which no polynomial running time algorithm is known. We cannot prove that such an algorithm does not exist, however, we can prove that these hard computational problems are as hard as any problems in NP. To state this precisely, we first need the following definitions.

Definition 3. *Let A and B be two computational problems. We say that A has a* polynomial reduction *to B, if a polynomial running time algorithm exists that solves any problem instance $x \in A$ by generating problem instances*

$y_1, y_2, \ldots, y_k$ all in B and solves x using the solutions for $y_1, y_2, \ldots y_k$. The computational time generating problem instances $y_1, y_2, \ldots y_k$ counts in the running time of the algorithm, but the computational time spent in solving these problem instances is not considered in the overall running time. We also say that A is polynomial-time reducible *to B.*

An example for this might be the following. An independent set in a graph is a subset of the vertices such that no two vertices in it are adjacent. The k-independent set problem asks if there is an independent set of size k in a graph. The k-independent set problem is polynomial-time reducible to the k-clique problem. Indeed, a graph contains an independent set of size k if and only if its complement contains a clique of size k. Taking the complement of a graph can be done in polynomial time. Similarly, the k-clique problem is also polynomial-time reducible to the k-independent set problem.

Polynomial reduction is an important concept in computational complexity. If a computational problem A is polynomial-time reducible to B and B can be solved in polynomial time, then A also can be solved in polynomial time. Similarly, if B is polynomial-time reducible to A, and A can be solved in polynomial time, then B can be solved in polynomial time, as well. Therefore, if A and B are mutually polynomial-time reducible to each other, then either both of them or none of them can be solved in polynomial time. These thoughts lead to the following definitions.

Definition 4. *A computational problem is in the complexity class* NP-hard *if every problem in NP is polynomial-time reducible to it. The* NP-complete *problems are the intersection of NP and NP-hard.*

What follows from the definition is that P is equal to NP if and only if there is a polynomial running time algorithm that solves an NP-complete problem. It is widely believed that P is not equal to NP, and thus, there are no polynomial running time algorithms for NP-complete problems.

The SAT problem asks if there is a satisfying assignment of a disjunctive normal form. Stephen Cook proved in 1971 that SAT is NP-complete [21], and Richard Karp demonstrated in 1972 that many natural computational problems are NP-complete by reducing SAT to them [56]. These famous Karp's 21 NP-complete problems drove attention to NP-completeness and initiated the study of the P versus NP problem. The question whether or not P equals NP has become the most famous unsolved problem in computational complexity theory. In 2000, the Clay Institute offered $1 million for a proof or disproof that P equals NP [20].

1.1.3 Deterministic counting: FP, #P, #P-complete

Definition 5. *The complexity class* #P *contains the counting problems that ask for the number of witnesses of the decision problems in NP. If A denotes a problem in NP, then #A denotes its counting counterpart.*

For example, #SAT denotes the counting problem that asks for the number of satisfying assignments of conjunctive normal forms. Since the decision versions of #P problems are in NP, there is a witness that can be verified in polynomial time. This does not automatically imply that all witnesses can be verified in polynomial time, although it naturally holds in many cases. When it is questionable that all solutions can be verified in polynomial time, a polynomial upper bound must be given, and only those witnesses are considered that can be verified in that time.

Some counting problems are tractable. Formally, they belong to the class of tractable function problems.

Definition 6. *A* function problem *is a computational problem where the output is more complex than a simple "yes" or "no" answer. The complexity class* FP *(Function Polynomial-Time) is the class of function problems that can be solved in polynomial time with an algorithm.*

We can define the #P-hard and #P-complete classes analogously to the NP-hard and NP-complete classes.

Definition 7. *A computational problem is in* #P-hard *if any problem in #P is polynomial-time reducible to it. The* #P-complete *class is the intersection of #P and #P-hard.*

As one can naturally guess, #SAT is a #P-complete problem. Indeed, the following theorem holds.

Theorem 8. *For every problem #A in #P, and every problem instance x in #A, there exists a conjunctive normal form Φ such that the number of satisfying assignments of Φ is the answer for x. Furthermore, such a Φ can be constructed in polynomial time of the size of the problem instance x. Since #SAT is in #P, this means that #SAT is a #P-complete problem.*

It is clear that $\#P \subseteq FP$ if and only if there exists a polynomial running time algorithm for a #P-complete problem. It is also trivial to see that $\#P \subseteq FP$ implies $P = NP$. However, we do not know if the reverse is true,

namely, whether or not P = NP implies that counting the witnesses of any #P-complete problem is easy.

By assuming that P is not equal to NP, we cannot expect a polynomial running time algorithm counting the witnesses of an NP-complete problem. Naturally, the counting versions of many NP-complete problems are #P-complete. However, we do not know if it is true that for any NP-complete problem A, its counting version $\#A$ is #P-complete. On the other hand, there are easy decision problems whose counting versions are #P-complete. For example, deciding if there is a perfect matching in a bipartite graph is in P [49], however, counting the perfect matchings in a bipartite graph is a #P-complete problem [92]. More surprisingly, The #LE (number of linear extensions) problem that asks how many total orderings are there which is an extension of a (finite) partially ordered set is a #P-complete counting problem [16]. It is surprising because any finite partial ordering can trivially be extended to a total ordering, and thus, deciding if there is a total ordering that agrees with a partial ordering is trivially in P.

1.1.4 Random decision algorithms: RP, BPP. Papadimitriou's theorem

In this section, we introduce the two basic complexity classes for random decision algorithms: the BPP and RP classes, and state a theorem that is an exercise in Papadimitriou's book on computational complexity [71].

Definition 9. *A decision problem is in the* BPP (Bounded-error Probabilistic Polynomial) *class if a random algorithm exists such that*

(a) it runs in polynomial time on all inputs,

(b) if the correct answer is "yes" it answers "yes" with probability at least 2/3*,*

(c) if the correct answer is "no" it answers "no" with probability at least 2/3*.*

Any such algorithm is also called a BPP algorithm.

The 2/3 in the definition of BPP is just a convention. The number 2/3 is the rational number p/q between 1/2 and 1 such that $p + q$ is minimal. Indeed, any fixed constant number between 1/2 and 1 would provide an

equivalent definition or even it would be enough if one of the probabilities was strictly 1/2 and the probability for the other answer would converge to 1/2 only polynomially fast. Similarly, if a BPP algorithm exists for a decision problem, then also a random algorithm exists that runs in polynomial time and gives the wrong answer with very small probability (say, with probability $\frac{1}{2^{100}}$).

Definition 10. *A decision problem is in* RP (Randomized Polynomial time) *if a random algorithm exists such that*

(a) it runs in polynomial time on all inputs

(b) if the correct answer is "yes" it answers "yes" with probability at least 1/2

(c) if the correct answer is "no" it answers "no" with probability 1.

Any such algorithm is also called RP algorithm.

Again, the 1/2 in the definition of RP is only a convention; it is the rational number p/q between 0 and 1 such that $p + q$ is minimal. Just like in the case of the BPP class, any fixed constant probability or a probability that converges only polynomially fast to 0 would suffice. Also, the existence of an RP algorithm means that the correct answer can be calculated with very high probability (say, with probability $1 - \frac{1}{2^{100}}$) in polynomial time.

We know that

$$\mathrm{P} \subseteq \mathrm{RP} \subseteq \mathrm{NP} \tag{1.1}$$

however, we do not know if any containment is proper. Surprisingly, we do not know if BPP $\subseteq$ NP or NP $\subseteq$ BPP. In this thesis, we will assume the following conjecture.

Conjecture 1. *For the decision classes* P, RP *and* NP, *the relation*

$$\mathrm{P} = \mathrm{RP} \subset \mathrm{NP} \tag{1.2}$$

holds.

Corollary 11. *The* P $\neq$ NP *assumption also means that*

$$\mathrm{NP}\text{-}\textit{complete} \cap \mathrm{P} = \emptyset \tag{1.3}$$

and

$$\#\mathrm{P}\text{-}\textit{complete} \cap \mathrm{FP} = \emptyset. \tag{1.4}$$

The conjecture that RP $\neq$ NP also implies that RP $\cap$ NP-complete $= \emptyset$. We can say even more.

Theorem 12. *(Papadimitriou's theorem) If the intersection of* NP*-complete and* BPP *is not empty, then* RP = NP*.*

The theorem says the following: if we can stochastically solve any NP-complete problem with anything better than random guessing, then we could solve any problem in the NP class with probability almost 1. We will use this theorem to prove that certain counting problems cannot be well approximated stochastically unless RP = NP.

1.1.5 Stochastic counting and sampling: FPRAS, FPAUS, self-reducible counting problems

In a random computation, we cannot expect that the answer be correct with probability 1, and we even cannot expect that the answer have a given approximation ratio with probability 1. However, we might require that the computation have small relative error with high probability. This leads to the definition of the following complexity class.

Definition 13. *A counting problem is in* FPRAS (Fully Polynomial Randomized Approximation Scheme) *if for any problem instance x and parameters $\epsilon, \delta > 0$ it has a randomized algorithm generating an approximation $\hat{f}$ for the true value f satisfying the inequality*

$$P\left(\frac{f}{1+\epsilon} \leq \hat{f} \leq f(1+\epsilon)\right) \geq 1-\delta \tag{1.5}$$

and the running time of the algorithm is polynomial in $|x|$, $\frac{1}{\epsilon}$ and $-\log(\delta)$.

An algorithm itself with these prescribed properties is also called FPRAS.

The following folklore theorem shows that we cannot expect a counting problem to be in FPRAS if its decision version is NP-complete.

Theorem 14. *If there exists an* NP*-complete decision problem A such that $\#A$ is in* FPRAS*, then* RP = NP*.*

Proof. By Theorem 12, it is sufficient to show that an FPRAS algorithm for the number of solutions provides a BPP algorithm for the decision problem.

Indeed, let x be a problem instance in A, then the answer for x is "no" if the number of solutions is 0 and the answer is "yes" if the number of solutions is a positive integer. Consider an FPRAS with input x, $\epsilon = 1/2$ and $\delta = 1/3$. Such an FPRAS runs in polynomial time with the size of x, and provides an answer larger than $1/2$ with probability at least $2/3$ if there is at least one solution for x, namely, if the correct answer for the decision problem is "yes". Furthermore, if the correct answer for the decision problem is "no", then the FPRAS returns 0 with probability at least $2/3$. Therefore, the algorithm that sends the input x, $\epsilon = 1/2$ and $\delta = 1/3$ to an FPRAS and answers "no" if FPRAS returns a value smaller that $1/2$ and answers "yes" if the FPRAS returns a value larger than or equal to $1/2$ is a BPP algorithm even if the running time of the FPRAS is included in the running time. □

There are also counting problems whose decision versions are easy (they are in P), however, they cannot be approximated unless RP = NP. For example, Jerrum, Valiant and Vazirani already observed in 1986 that it is hard to count the number of cycles in a directed graph [54].

The situation will remain the same if we could generate roughly uniformly distributed cycles from a directed graph. Below we state this precisely after defining the necessary ingredients of the statement.

Definition 15. *The* total variation distance *of two discrete distributions p and π over the same (countable) domain X is defined as*

$$d_{TV}(p, \pi) := \frac{1}{2} \sum_{x \in X} |p(x) - \pi(x)|. \tag{1.6}$$

It is easy to see that the total variation distance of two distributions is between 0 and 1, it is 0 if and only if the two distributions are pointwise the same (for all x, $p(x) = \pi(x)$), and it is 1 if and only if the two distributions have disjoint support (that is, $p(x) \neq 0$ implies that $\pi(x) = 0$ and vice versa). It is also easy to see that the total variation distance is indeed a metric.

Definition 16. *A sampling problem $\#X$ is in* FPAUS (Fully Polynomial Almost Uniform Sampler) *if $\#X$ is in $\#P$, and there is a sampling algorithm that for any problem instance in $\#X$ and $\epsilon > 0$, it generates a random witness following a distribution p satisfying the inequality*

$$d_{TV}(p, U) \leq \epsilon \tag{1.7}$$

where U is the uniform distribution of the witnesses. The algorithm must run in polynomial time both with the size of the problem instance and $-\log(\epsilon)$. This algorithm itself is also called FPAUS.

There is a strong relationship between the complexity classes FPRAS and FPAUS. For a large class of counting problems, called *self-reducible counting problems*, there is an FPRAS algorithm for a particular counting problem if and only if there is an FPAUS algorithm for it. To formally define self-reducible counting problems we can think of the #P problems as *p-relations*, defined below.

Definition 17. *A relation $R \subseteq \Sigma^* \times \Sigma^*$ is a* p-relation *iff*

- $\langle x, y\rangle \in R \implies |y| = O(\text{poly}(|x|))$.
- *the predicate $\langle x, y\rangle \in R$ can be tested in deterministic polynomial time.*

Here Σ is the common alphabet we use to describe problem instances and solutions, and $|x|$ denotes the length of sequence x.

Definition 18. *A p-relation is* self-reducible *iff*

- *there exists a polynomial time computable function $g : \Sigma^* \to \mathbb{N}$ such that*
$$\langle x, y\rangle \in R \implies |y| = g(x)$$
- *there exist polynomial time computable functions $\phi : \Sigma^* \times \Sigma^* \to \Sigma^*$ and $\sigma : \Sigma^* \to \mathbb{N}$ satisfying*
$$\sigma(x) = O(\log(|x|))$$
$$g(x) > 0 \implies \sigma(x) > 0 \ \forall x \in \Sigma^*$$
$$|\phi(x, w)| \leq |x| \ \forall x, w \in \Sigma^*$$
and for all $x \in \Sigma^*$ and $y_1 y_2 \ldots y_n \in \Sigma^*$
$$\langle x, y_1 y_2 \ldots y_n\rangle \in R \Leftrightarrow \langle \phi(x, y_1 y_2 \ldots y_{\sigma(x)}), y_{\sigma(x)+1} \ldots y_n\rangle \in R$$

The explanation of the definition is that the σ function tells the 'granulation' of the solutions of x in a sense that taking any $\sigma(x)$ long prefix w of a solution to x, there is another problem instance x', defined by the

function $\psi(x, w)$, such that the solutions of x' are exactly the words which gave solutions to x when concatenated to w. Although the definition of self-reducible counting problems is quite technical, there are many counting problems which are naturally self-reducible. For example, a partially ordered set can be defined by giving a list of all comparable pair of elements, showing their relationships, $a_i \preceq a_j$. Any total ordering might be the list of the elements $a_{i_1}, a_{i_2}, \ldots, a_{i_n}$. The granulation function must find the length of the string "a_{i_1},", and $\phi(x,"a_{i_1},")$ is the partially ordered set from which a_{i_1} is removed.

In their seminal paper, Jerrum, Valiant and Vazirani proved the following theorem.

Theorem 19. *(Jerrum-Valiant-Vazirani) Let #A be a self reducible counting problem. Then #A has an FPRAS approximation if and only if it has an FPAUS sampling.*

There are #P-complete problems that do have FPRAS algorithms. For example, Karzanov and Khaciyan gave a rapidly mixing Markov chain that can almost uniformly generate total orderings agreeing with a given partially ordered set [57]. Since #LE is a self-reducible counting problem, Karzanov and Khaciyan also proved that #LE is in FPRAS. Since #LE is also in #P-complete[16], the intersection of FPRAS and #P-complete is not empty.

One might ask if the fact that the intersection of #P-complete and FPRAS is not empty implies that RP = NP. Papadimitriou's theorem says that we can nicely handle probabilities in polynomial reductions. However, we might not be able to handle relative errors. Indeed, there are operations that do not keep the relative error, most notably, the subtraction and modulo prime number calculations. We will see that such operations appear in each #P-completeness proof of a counting problem that is also in FPRAS. Therefore the polynomial reductions used in such proofs cannot be used to propagate FPRAS approximations to other counting problems in #P: we will lose the small relative error. Thus, it does not follow that any counting problem in #P has an FPRAS.

On the other hand, if a #P-completeness proof for a problem #A preserves the relative error, it also proves that there is no FPRAS for #A unless RP = NP.

1.1.6 Markov chains

Discrete time, discrete space Markov chains

The Karzanov-Khaciyan Markov chain highlights the importance of Markov chains in computational complexity. Below we define the discrete time, discrete state Markov chains.

Definition 20. *A* discrete time, finite Markov chain $X = (X_0, X_1, \ldots)$ *is a sequence of random variables taking values on a finite state space S, such that X_n depends on only X_{n-1}. That is, for all $m \geq 0$ and all possible sequence $i_0, i_1, \ldots, i_{m+1} \in S^{m+2}$,*

$$P(X_{m+1} = i_{m+1} \mid X_0 = i_0, \ldots, X_m = i_m) = P(X_{m+1} = i_{m+1} \mid X_m = i_m).$$

For the sake of simplicity, we will write $P(i_{m+1}|i_m)$ instead of $P(X_{m+1} = i_{m+1} \mid X_m = i_m)$. We will denote by $\mathbf{P}$ the transition matrix containing the transition probabilities, that is, if $\mathbf{P} = \{p_{i,j}\}$, then $p_{i,j} = P(i|j)$. In this thesis, we talk about only discrete time, finite Markov chains that we will call Markov chains for short.

An irreducible *Markov chain is a Markov chain with only one communicating class, i.e. for any two states $i, j \in S$, the probability of getting from i to j and from j to i (in multiple steps) are both positive.*

A Markov chain is aperiodic *if all its states $i \in S$ satisfy that*

$$d_i := \gcd\, \{t \geq 1 |\ p_{ii}^{(t)} = P(\textit{getting from } i \textit{ to itself via } t \textit{ steps}) > 0\} = 1,$$

where gcd *stands for greatest common divisor.*

A standard technique to make a Markov chain aperiodic is to have $P(i|i) \neq 0$ for all states i. Frequently, $P(i|i)$ is set at least $\frac{1}{2}$. This can be obtained, for example, by defining a new Markov chain whose transition matrix is the average of the transition matrix and the identity matrix. Such a Markov chain is called a *Lazy Markov chain* [52, 115]. The spectral analysis of Lazy Markov chains is technically simpler, see also Theorem 26 about the average of a transition matrix and the identity matrix and also its Corollary 27.

Below we give definitions regarding the convergence of Markov chains.

Definition 21. *A* stationary distribution π *of a Markov chain with a transition matrix $\mathbf{P}$ is an eigenvector of $\mathbf{P}$ with eigenvalue* 1.

A Markov chain converges *to distribution π from a starting state $x_0 = i$ if*

$$\lim_{t\to\infty} d_{TV}(\mathbf{P}^t\mathbf{1}_i, \pi) = 0 \tag{1.8}$$

where $\mathbf{1}_i$ is the vector that contains all 0 except at coordinate i where it is 1, and d_{TV} denotes the total variation distance (Definition 15).

The relaxation time *is defined as*

$$\tau_i(\varepsilon) := \min\{n_0 \in \mathbb{N} : d_{TV}(\mathbf{P}^n\mathbf{1}_i, \pi) \leq \varepsilon \ \forall n \geq n_0\}, \tag{1.9}$$

and the convergence time is the maximum of the relaxation time, that is

$$\tau(\varepsilon) := \max_i \tau_i(\varepsilon). \tag{1.10}$$

We say that a Markov chain satisfies the detailed balance condition *with respect to a distribution π if for all i, j it holds that*

$$\pi(i)P(j|i) = \pi(j)P(i|j). \tag{1.11}$$

The followings are well-known theorems on convergence of Markov chains, see for example, [15] for reference.

Theorem 22. *Any Markov chain has a stationary distribution, that is, for any Markov chain with transition matrix $\mathbf{P}$, $\mathbf{P}$ has an eigenvector π with eigenvalue 1.*

This stationary distribution might not be unique. However, if the Markov chain is irreducible, then this stationary distribution is unique.

Furthermore, if the Markov chain is aperiodic and irreducible, then the Markov chain converges to its unique stationary distribution from any starting point.

If a Markov chain is irreducible, aperiodic and reversible with respect to distribution π, then π is its unique stationary distribution.

Markov chain Monte Carlo

The *Markov chain Monte Carlo* (MCMC) method is a class of algorithms that achieves the goal of sampling from a probability distribution. One typical example is the Metropolis-Hastings algorithm [44, 66], described below.

Definition 23. *The* Metropolis-Hastings algorithm *takes a Markov chain on a state space X and transforms it into another one.*

For a given function $g : X \to \mathbb{R}^{>0}$, and an irreducible, aperiodic Markov chain with transition probabilities such that for all pair of states $x, y \in X$ it holds that $P(y|x) \neq 0 \Leftrightarrow P(x|y) \neq 0$, a new Markov chain is constructed on the same state space. The state x_i is defined based on x_{i-1} in the following steps.

1. *Propose a candidate y for the next sample $y \sim P(\cdot|x_{i-1})$, where $\sim$ stands for "following distribution".*

2. *Generate a random number $u \sim U(0,1)$, where $U(0,1)$ is the uniform distribution on the $[0,1]$ interval.*

3. *The proposed state y is* accepted*, that is, $x_i = y$ if $u \leq \frac{g(y)P(x_{i-1}|y)}{g(x_{i-1})P(y|x_{i-1})}$. Otherwise the proposed state is* rejected*, that is, $x_i = x_{i-1}$.*

Observe that the probability of accepting a proposed state y is

$$\min\left\{1, \frac{g(y)P(x|y)}{g(x)P(y|x)}\right\}.$$

The fraction $\frac{g(y)P(x|y)}{g(x)P(y|x)}$ is called the *Metropolis-Hastings ratio*, and the probability is called the *acceptance probability*. Also observe that the Metropolis-Hastings ratio has the following property. If $M(x,y)$ denotes the Metropolis-Hastings ratio when proposing y from a state x, then

$$M(x,y) = \frac{1}{M(y,x)}.$$

The inverse of a lower bound on the Metropolis-Hastings ratio gives an upper bound on the expected number of proposals before the first acceptance at an arbitrary state. Due to the above mentioned property, the inverse of a lower bound of the Metropolis-Hastings ratio is an upper bound of the Metropolis-Hastings ratio and *vice versa*.

It is easy to prove that a Markov chain defined by the Metropolis-Hastings algorithm is irreducible, aperiodic, and reversible with respect to π, the probability distribution obtained after g is normalized, and thus it converges to distribution π starting in an arbitrary state [52, 115]. In this thesis, we will consider only Markov chains obtained by the Metropolis-Hastings algorithm

and its variants (see below), so from now we will consider only irreducible, aperiodic and reversible Markov chains, and we will call them Markov chains for short.

A variant of the Metropolis-Hastings algorithm is when for each pair of states x and y such that $P(y|x) \neq 0$ (and thus $P(x|y) \neq 0$), there are several ways to transform x to y: $W = \{w_1, w_2, \ldots, w_t\}$, and there is a one-to-one mapping from W to the set $W' = \{w'_1, w'_2, \ldots, w'_t\}$, the possible ways to transform y back to x. The sets W and W' depend on x and y, but for the sake of simplicity, we do not indicate this dependency in the notation. Also, for all x, y, w_i and w'_i we require that

$$P(y, w_i|x) \neq 0 \Leftrightarrow P(x, w'_i|y) \neq 0.$$

Then the Metropolis-Hastings ratio can be changed to

$$\frac{g(y)P(x, w'_i|y)}{g(x)P(y, w_i|x)} \tag{1.12}$$

when y is proposed from x via the way w_i. Here $P(y, w_i|x)$ is the probability that y is proposed from x via way w_i. It can be proved that the so-obtained Markov chain still converges to π, the probability distribution obtained after g is normalized [113]. We will use this version of the Metropolis-Hastings algorithm twice in this thesis, in Chapter 7 and Chapter 12.

Speed of convergence of Markov chains

The eigenvalues of a Markov chain are the eigenvalues of its transition matrix. For reversible Markov chains, these eigenvalues are all real, denoted by $1 = \lambda_1, \lambda_2, \ldots, \lambda_r$.

Definition 24. *The* second-largest eigenvalue modulus *of a Markov chain is defined as*

$$\max\{\lambda_2, |\lambda_r|\} \tag{1.13}$$

and is denoted by ρ. It is also abbreviated as SLEM.

The following theorem is known about the second largest eigenvalue modulus and the relaxation time.

Theorem 25. *For a Markov chain, it holds that*

$$\tau_i(\varepsilon) \leq \frac{1}{1-\rho}\left(\log\left(\frac{1}{\pi(x_i)}\right) + \log\left(\frac{1}{\varepsilon}\right)\right) \tag{1.14}$$

and

$$\max_i\{\tau_i(\varepsilon)\} \geq \frac{\rho}{2(1-\rho)}\log\left(\frac{1}{2\varepsilon}\right). \tag{1.15}$$

The proof of the first inequality can be found in [26], while the proof of the second inequality can be found in [5]. Theorem 25 says that the relaxation time is proportional to the inverse of the difference between the largest eigenvalue (that is, 1) and the SLEM. The following theorem says that it is sufficient to consider the second-largest eigenvalue of a Markov chain.

Theorem 26. *Let M be a quadratic matrix with eigenvalues $\lambda_1, \ldots, \lambda_r$ and eigenvectors $v_1, \ldots, v_n$. Then the matrix*

$$\frac{M+I}{2} \tag{1.16}$$

has eigenvalues $\frac{\lambda_1+1}{2}, \ldots, \frac{\lambda_r+1}{2}$ and eigenvectors $v_1, \ldots, v_n$, where I is the identity matrix.

Proof. It trivially comes from the basic properties of linear algebraic operations. Indeed,

$$\frac{M+I}{2}v_i = \frac{1}{2}(M+I)v_i = \frac{1}{2}(Mv_i + Iv_i) = \frac{1}{2}(\lambda_i v_i + v_i) = \frac{\lambda_i+1}{2}v_i. \tag{1.17}$$

□

Corollary 27. *If M is a Markov chain, then the random process defined by the following algorithm (so-called* lazy version *of M) is also a Markov chain whose SLEM is $\frac{\lambda_2+1}{2}$ and converges to the same, globally stable stationary distribution.*

1. *Draw a random number u uniformly from the $[0,1]$ interval.*
2. *If $u \leq \frac{1}{2}$ then do nothing; the next state of the Markov chain is the current state. Otherwise, the next state is drawn following the Markov chain M.*

Indeed, this process is a Markov chain, since the series of random states satisfies the Markov property: where we are going depends on the current state and not where we came from. Its transition matrix is $\frac{T+I}{2}$, so we can apply Theorem 26. The largest eigenvalue and its corresponding eigenvector do not change, so the Markov chain still converges to the same distribution.

Here we state and prove a theorem on the mixing time of Markov chains and FPAUS algorithms.

Theorem 28. *Let #A be a counting problem in #P, and let x denote a problem instance in #A with size n. Assume that the following holds:*

(a) a solution of x can be constructed in $O(poly(n))$ time,

(b) there is a Markov chain with transition matrix T that converges to the uniform distribution of the solutions of x, and for its second-largest eigenvalue it holds that

$$\frac{1}{1-\lambda_2} = O(poly(n)), \tag{1.18}$$

(c) there is a random algorithm that for any solution y, draws an entry from the conditional distribution $T(\cdot|y)$ in polynomial time.

Then #A is in FPAUS.

Proof. Let x be a problem instance of #A, and let $\varepsilon > 0$. Since #A is in #P, there is a constant $c > 1$ and a polynomial $poly_1$ such that the number of solutions of x is less than or equal to $c^{poly_1(n)}$. Indeed, any witness can be verified in polynomial time, and the witnesses are described using a fixed alphabet (the alphabet depends on only the problem and not the problem instance). To verify a solution, it must be read. What follows is that the number of solutions cannot be more than $|\Sigma|^{poly(n)}$, where Σ is the alphabet used to describe the solutions and $poly()$ is the natural or given polynomial upper bound on the running time to verify a solution.

Having said these, it is easy to show that the following algorithm is an FPAUS:

1. Construct a solution y of x.

2. Using y as the starting point of the Markov chain, do

$$\frac{2}{1-\lambda_2}\left(poly_1(n)\log(c)+\log\left(\frac{1}{\varepsilon}\right)\right) \tag{1.19}$$

number of steps in the lazy version of the Markov chain.

3. Return with the last state of the Markov chain.

Indeed, the state that the algorithm returns follows a distribution that satisfies Equation (1.7), since

$$\tau_y(\varepsilon) \leq \frac{2}{1-\lambda_2}\left(\log\left(\frac{1}{\pi(y)}\right)+\log\left(\frac{1}{\varepsilon}\right)\right) \leq$$
$$\frac{2}{1-\lambda_2}\left(\mathrm{poly}_1(n)\log(c)+\log\left(\frac{1}{\varepsilon}\right)\right). \tag{1.20}$$

The first inequality comes from Theorem 25 and from Corollary 27. The second inequality comes from the observation that $\frac{1}{\pi(y)}$ is the size of the solution space, since π is the uniform distribution, and we showed that $c^{poly_1(n)}$ is an upper bound of it. The running time of the algorithm is $O(poly(n,-\log(\varepsilon)))$, since the initial state y can be constructed in polynomial time, there are $poly(n,-\log(\varepsilon))$ number of steps in the Markov chain, and each of them can be performed in $O(poly(n))$ time. □

This theorem explains the following definition.

Definition 29. *Let #A be a counting problem in #P. Let $\mathcal{M}$ be a class of Markov chains, such that for each problem instance x of #A, it contains a Markov chain converging to the uniform distribution of witnesses of x. Let M_x denote this Markov chain, and let $\lambda_{2,x}$ denote its second-largest eigenvalue. We say that $\mathcal{M}$ is* rapidly mixing *if*

$$\frac{1}{1-\lambda_{2,x}} = O(poly(|x|)). \tag{1.21}$$

Similarly, we can say that a Markov chain is slowly or torpidly mixing *if*

$$\frac{1}{1-\lambda_{2,x}} = \Omega(c^{|x|}) \tag{1.22}$$

for some $c > 1$.

The difference between the largest eigenvalue, 1 and the second largest eigenvalue is called the *spectral gap*.

1.1.7 Techniques to prove rapid mixing of Markov chains

Multicommodity flow

There are several methods to prove rapid mixing of a Markov chain via proving upper bounds on the inverse of the spectral gap. First, we introduce the multicommodity flow technique developped by [82]. We denote by $T(\cdot|\cdot)$ the transition probabilities of the Markov chain.

The Markov graph $G(V, E)$ of a Markov chain on matchings is a directed graph whose vertices are the states of the Markov chain, and there is an arc between two states u and v if there is a transition from u to v.

Definition 30. *We define the* load *of an edge $e = (u, v)$ as*

$$Q(e) := T(u|v)\theta(u)$$

A path system Γ *in a Markov graph is a set of distributions of paths for each ordered pair (x, y), $x, y \in V$. We will denote the distribution of paths defined for (x, y) by $\Gamma_{x,y}$, and then*

$$\Gamma := \cup_{(x,y)\in V\times V}\Gamma_{x,y}$$

Let $p_{(x,y)}(\gamma)$ denote the probability of a path γ in the distribution $\Gamma_{x,y}$ of a path system Γ.

Let

$$\kappa_\Gamma := \max_{e=(u,v)\in E} \sum_{(x,y)\in V\times V} \sum_{\gamma\in\Gamma_{(x,y)}:e\in\gamma} \theta(x)\theta(y)p_{x,y}(\gamma)\frac{|\gamma|}{Q(e)} \tag{1.23}$$

The κ_Γ of a path system is called the *Poincaré coefficient*. We have the following inequality.

Theorem 31. *[82] For any path system Γ,*

$$\frac{1}{1-\lambda_2} \leq \kappa_\Gamma, \tag{1.24}$$

where λ_2 is the second eigenvalue of the transition matrix of the Markov chain.

What follows is that the Markov chain can be used for an FPAUS if κ_Γ is bounded by a polynomial in the size of the data.

Sometimes we create path systems such that there is a unique path between any ordered pair (x, y). In such case, the method is referred as *canonical path system method* .

The Cheeger inequalities

Cheeger's inequalities connect the conductance of a Markov chain and its second-largest eigenvalue. First we define the conductance and then state the inequalities.

Definition 32. *The* capacity *of a subset $S \subseteq X$ of the state space is defined as*

$$\pi(S) := \sum_{x \in S} \pi(x). \tag{1.25}$$

The ergodic flow *of a subset $S \subseteq X$ of the state space is defined as*

$$F(S) := \sum_{\substack{x \in S \\ y \in \overline{S}}} \pi(x) T(y|x). \tag{1.26}$$

The conductance *of a Markov chain is*

$$\Phi := \min \left\{ \frac{F(S)}{\pi(S)} \middle| S \subset X, 0 < \pi(S) \leq \frac{1}{2} \right\}. \tag{1.27}$$

Theorem 33. Cheeger's inequality *The second-largest eigenvalue λ_2 of a Markov chain satisfies the following inequalities:*

$$1 - 2\Phi \leq \lambda_2 \leq 1 - \frac{\Phi^2}{2}. \tag{1.28}$$

The proof can be found in [15], Chapter 6. The two inequalities in Equation (1.28) will be referred to as the left and right Cheeger's inequality. This theorem says that a Markov chain is rapidly mixing if and only if its conductance is large. We can prove the torpid mixing of a Markov chain by finding a subset whose ergodic flow is negligible compared to its capacity.

In the definition of conductance, there might be double exponentially many subsets to be considered. Indeed, the size of the state space might be an exponential of the input size, and the number of subsets with at most half probability might be also an exponential function of the base set. In spite of this, surprisingly, we can prove rapid mixing of a Markov chain using the right Cheeger's inequality.

1.2 Mathematical models

1.2.1 Graphs, networks, discrete tomography

Degree sequences

Definition 34. *The* degree sequence $D = d_1, d_2, \ldots, d_n$ *is a sequence of non-negative integers. We say that a vertex-labeled simple graph* $G = (V, E)$ *is a* realization *of* D *if for all* $v_i \in V$, $d(v_i) = d_i$, *where* $d(v)$ *denotes the degree of* v, *that is, the number of edges incident to* v. *Here the labeling of the vertices are identified with the indexes of the vertices.*

The bipartite degree sequence $D_b = (d_{1,1}, d_{1,2}, \ldots, d_{1,n}), (d_{2,1}, d_{2,2}, \ldots, d_{2,m})$ *is an ordered pair of sequences of non-negative integers. We say that a vertex-labeled bipartite graph* $G = (U, V, E)$ *is a* realization *of* D_b *if for all* $u_i \in U$, $d(u_i) = d_{1,i}$ *and for all* $v_j \in V$, $d(v_j) = d_{2,j}$.

The directed degree sequence $\vec{D} = (d_1^+, d_1^-), (d_2^+, d_2^-), \ldots, (d_n^+, d_n^-)$ *is a sequence of ordered pair of non-negative integers. We say that a vertex-labeled directed graph* $\vec{G} = (V, E)$ *is a* realization *of* $\vec{G}$ *if for all* $v_i \in V$, $d^+(v_i) = d_i^+$ *and* $d^-(v_i) = d_i^-$, *where* $d^+(v)$ *is the in-degree of* v *and* $d^-(v)$ *is the out-degree of* v.

For any type of degree sequence, we say that the degree sequence is graphic *if there is a realization of it.*

The graphicality of degree sequences are well known problems in graph theory. Erdős and Gallai gave necessary and sufficient conditions when a degree sequence is graphic [31]. Gale and Ryser gave necessary and sufficient conditions for bipartite degree sequences [37, 75]. Havel and Hakimi proved the following theorem:

Theorem 35. *(Havel-Hakimi, [43, 47]) Let* $D = d_0, d_1 \geq d_2 \geq \ldots \geq d_n$ *be a degree sequence. Then* D *is graphic if and only if* $D' = d_1 - 1, d_2 - 1, \ldots, d_{d_0} - 1, d_{d_0+1}, \ldots d_n$ *is graphic.*

The proof of the theorem is based on the following observation. If D is graphic then also a realization of D exists in which v_0 is the neighbor of the first d_0 largest degree vertices. Any realization can be transformed into such a realization with a series of so-called *switch operations*. A switch operation removes two edges (v_1, v_2) and (v_3, v_4) and creates two new edges (v_1, v_3) and (v_2, v_4). The Havel-Hakimi theorem can be iterated on D', and this leads to a *canonical* or *Havel-Hakimi realization* in which v_0 is the neighbor of the

first d_0 largest degree vertices, then on $G \setminus \{v_0\}$, v_1 is the neighbor of the first $\tilde{d}_1$ largest degree vertices, where $\tilde{d}_1$ is the degree of v_1 in $G \setminus \{v_0\}$, etc. Any realization can be transformed to this canonical realization by switch operations. Since the inverse of a switch operation is also a switch operation, the corollary of the Havel-Hakimi theorem is the following.

Corollary 36. *Let G_1 and G_2 be two realizations of the same degree sequence. Then G_1 can be transformed to G_2 by a finite series of switch operations.*

Similar theorem and corollary for bipartite degree sequences are folklore. For sake of completeness we state these theorems here.

Theorem 37. *Let $D_b = (d_{1,1}, d_{1,2}, \ldots, d_{1,n}), (d_{2,1} \geq d_{2,2} \geq \ldots \geq d_{2,m})$ be a bipartite degree sequence. Then D_b is graphic if and only if $D'_b = (d_{1,2}, \ldots, d_{1,n}), (d_{2,1} - 1, d_{2,2} - 1, \ldots, d_{2,d_{1,1}} - 1, d_{2,d_{1,1}+1}, \ldots, d_{2,m})$ is graphic.*

Corollary 38. *Let G_1 and G_2 be two realizations of the same bipartite degree sequence D_b. Then G_1 can be transformed via realizations of D_b to G_2 by a finite series of switch operations.*

Given a degree sequence D (respectively, a bipartite degree sequence D_b, directed degree sequence $\vec{D}$) we can ask how many vertex labeled simple graph realizations of D (respectively, vertex labeled bipartite graph realizations of D_b, vertex labeled directed graph realizations of $\vec{D}$) are there. We also would like to generate random realizations following a distribution of the set of realizations that is very close to the uniform one, that is, we would like an FPAUS sampler.

Joint Degree Matrix

There are other properties that characterize networks. One of them is the *assortativity*. A network is said to be assortative if vertices with similar degrees tends to be neighbors. On the other hand, the neighbors in dissortative networks typically have different degrees. For example, social networks tend to be assortative while regulatory networks in gene regulation systems tend to be dissortative. A possible way to describe the assortativity of a network is the *Joint Degree Matrix*, also denoted by JDM.

Definition 39. *The matrix $\mathcal{J}(G) = [\mathcal{J}_{ij}]$ $\{1 \leq i, j \leq k\}$ is the* joint degree matrix *of the (simple) graph G, where for all i, $\mathcal{J}_{ii}$ is the number of edges*

spanned by V_i and for all $i \neq j$, $\mathcal{J}_{ij}$ is the number of edges between V_i and V_j, where V_ℓ is the set of vertices with degree ℓ. If for $k \times k$ matrix M there exists a graph s.t. its joint degree matrix is equal to M then the latter is called a graphic JDM.

Remark 1. *It is clear that the degree sequence of the graph is fully determined by its JDM. Indeed, we have:*

$$|V_i| = \frac{1}{i}\left(\mathcal{J}_{ii} + \sum_{\ell=1}^{\infty} \mathcal{J}_{i\ell}\right). \qquad \square$$

Beyond deciding if a JDM has a realization, we also would like to generate a random realization of a JDM following an almost uniform distribution, that is, we would like an FPAUs sampler.

Discrete tomography

The discrete tomography problem is to construct binary or multicolor images from a few of their projections. One of the possible tasks in discrete tomography is the following. Given a list of k bipartite degree sequences of the same vertex set, decide if these degree sequences have disjoint common realizations. That is, a realization for each degree sequence such that for any pair of vertices u and v, (u, v) is an edge in at most one realization. This problem is generally NP-complete even for $k = 2$ [42]. However, there are positive results for some special sets of degree sequences. Above constructing one realization, it is also an important task to construct an FPAUS sampler for the possible realizations.

1.2.2 Genome rearrangement

Below we give a mathematical model of genomes.

Definition 40. *A* genome *is an edge labeled, directed graph $\vec{G} = (V, E, L, l)$, where l is a labeling function $l : E \to L$ such that each vertex has a total degree 1 or 2. That is, the sum of the in and out degrees is either 1 or 2. Both parallel edges and loops are allowed.*

The labels are unique, that is, the labeling function $l : E \to L$ is an injection. The vertices with total degree 1 are called telomers*, the vertices*

with total degree 2 *are called* adjacencies. *The edges are called* genes *or* synteny blocks.

The components of a genome are called chromosomes. *Each component is either a (not necessarily directed) line or a (not necessarily directed) cycle. In this way, we talk about* linear *and* circular *chromosomes. When* $\vec{G}$ *is connected (not necessarily strongly connected), the genome is called* unichromosomal. *A disconnected genome is called* multichromosomal.

If an edge is going from vertex u *to vertex* v*, then its end at* u *is called* tail *and its end at* v *is called* head. *The joint name of heads and tails is* extremities.

In genome rearrangement models, we consider the edges (that is, the genes or synteny blocks) as building blocks. That is, a genome rearrangement operation changes the *vertices* of a genome and not the edges. In a genome rearrangement problem, two genomes, $\vec{G_1} = (V_1, E_1, L, l_1)$ and $\vec{G_2} = (V_2, E_2, L, l_2)$ are given such that $l_1(E_1) = l_2(E_2)$, and we are interested in the series of operations that transforms $\vec{G_1}$ into $\vec{G_2}$. Usually, the *parsimony principle* is followed. That is, we are looking for the minimum number of necessary operations to transform G_1 into G_2. Typically, there are several shortest sequences of operations transforming one genome into another one. We can ask how many such shortest sequences are there and we might want to (almost) uniform sample one of them. We will call these sequences *shortest rearrangement paths* or *shortest rearrangement scenarios*.

The genome rearrangement models differ in what kind of genomes and rearrangement operations are considered. Below we list those considered in this thesis.

Double Cut and Join (DCJ)

In the Double Cut and Join (or frequently called, DCJ model), there is no restriction on the possible genome: a genome might be unichromosomal or multichromosomal and any chromosome might be linear or circular. Furthermore, in case of a multichromosomal genomes, the chromosome types might be mixed, that is, it is allowed that a genome contain both linear and circular chromosomes. With other words, a genome might be an arbitrary directed graph in which each vertex has a total degree 1 or 2. Usually, the label of the edges are the positive integers, and we denote by $h1$ the head of the first edge, we denote by $t1$ the tail of the first edge, etc. The vertices are labeled

by the extremities they are incident to. In this way, for example, $(h2, t5)$ denotes the vertex of degree 2 incident to the head of the second edge and the tail of the fifth edge, etc. General extremities are denoted by x, y, z and q.

A DCJ operation takes at most two vertices and replaces them into at most two vertices. That is, a DCJ operation is any of the following.

- It takes two vertices of degree 2, (x, y) and (z, q), and creates two new vertices (x, z) and (y, q). We will denote this by $(x, y|z, q)$. Note that $(x, y|z, q) \equiv (q, z|y, x) \equiv (y, x|q, z) \equiv (z, q|x, y)$.
- It takes a vertex of degree 2, (x, y), and a vertex with degree 1, (z), and creates two new vertices (x, z) and (y). We will denote it by $(x, y|z, 0) \equiv (0, z|y, x) \equiv (y, x|0, z) \equiv (z, 0|x, y)$.
- It takes a vertex of degree 2, (x, y), and creates two vertices of degree 1, (x) and (y). We will denote it by $(x, y|0, 0) \equiv (0, 0|x, y) \equiv (y, x|0, 0) \equiv (0, 0|y, x)$.
- It takes two vertices of degree 1, (x) and (y) and creates a new vertex of degree 2, (x, y). We will denote it by $(0, x|0, y) \equiv (x, 0|y, 0) \equiv (0, y|0, x) \equiv (y, 0|x, 0)$.

Computing the DCJ distance between two genomes can be easily obtained using the adjacency graph defined below. See also Figure 1.2.

Definition 41. *The* adjacency graph $G(V_1 \cup V_2, E)$ *of two genomes* G_1 *and* G_2 *is a bipartite multigraph in which* V_1 *is the set of adjacencies and telomeres of* G_1 *and* V_2 *is the set of adjacencies and telomeres of* G_2*. The number of edges between* $u \in V_1$ *and* $v \in V_2$ *is the number of extremities they share.*

Each vertex of the adjacency graph has either degree 1 or 2, and thus, the adjacency graph falls into disjoint cycles and paths. Each path has one of the following three types:

- *odd path*, containing an odd number of edges and an even number of vertices,
- *W-shaped path*, which is an even path with two endpoints in V_1,
- *M-shaped path*, which is an even path with two endpoints in V_2.

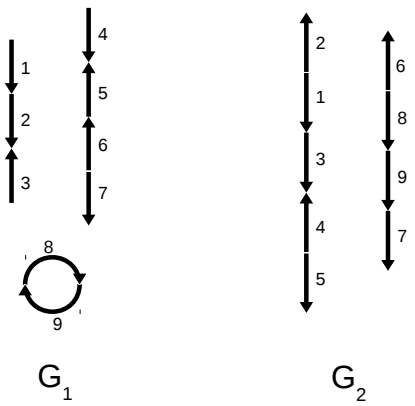


Figure 1.1: Two genomes with 9 genes sharing the same set of labels.

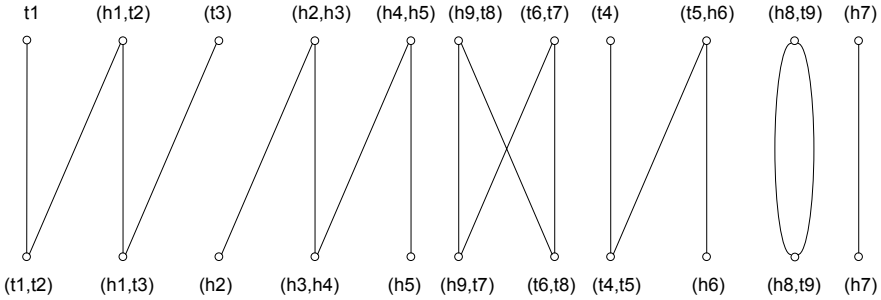


Figure 1.2: The adjacency graph of the two genomes on Fig. 1.1

In addition we call *trivial components* the cycles with two edges and the paths with one edge. An adjacency graph example can be seen on Figure 1.2.

Yancopoulus *et al.* introduced the DCJ model, and gave the first polynomial running time algorithm to compute the minimum number of DCJ operations necessary to transform one genome into another one [98]. Bergeron *et al.* gave a linear time algorithm constructing one scenario [13]. Below we present their theorem.

Theorem 42. *For the DCJ distance of two genomes, G_1 and G_2 it holds that*

$$d_{\mathrm{DCJ}}(G_1, G_2) = N - \left(C + \frac{I}{2}\right) \tag{1.29}$$

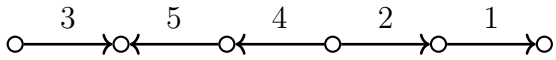
where N is the number of edges in each genome, C is the number of cycles in the adjacency graph of G_1 and G_2, and I is the number of odd paths in the adjacency graph of G_1 and G_2.

Since calculating C and I is easy, computing the DCJ distance clearly has a polynomial running time algorithm. Bergeron *et al.* [13] also give a linear time algorithm to find one scenario of length $d_{\mathrm{DCJ}}(G_1, G_2)$. It is also possible to count the number of most parsimonious scenarios in polynomial time if the genomes share the same telomeres [14, 70]. Braga and Stoye also gave an algorithm to count the number of most parsimonious scenarios for the general case, however, it runs in exponential time [14]. The complexity of the counting problem in the general case remains unknown and is conjectured to be #P-complete.

Sorting by Reversals

The *Sorting by Reversals* model considers unichromosomal, linear genomes. The operations are reversals that can be considered as those DCJ operations that keep the genome unichromosomal and linear. That is, a consecutive segment of the current chromosome is taken and reverted. These unichromosomal, linear genomes and the reversals acting on them can be modeled in the following way. The labels of the edges are positive integer numbers from 1 to n. One of the vertices of degree 1 is defined as the beginning of the chromosome and the other is defined as the end. A directed edge is considered to have a positive direction if its head is towards the end of the chromsome, otherwise it is considered to have a negative direction. In

this way, any unichromosomal, linear genome can be defined as a *signed permutation*. Based on the choice of beginning and end of a chromsome, each unichromosomal, linear genome can be represented by a signed permutation in two different ways. For example, the following genome



can be represented as

$$+3 \quad -5 \quad -4 \quad +2 \quad +1$$

and

$$-1 \quad -2 \quad +4 \quad +5 \quad +3.$$

A reversal takes a consecutive segment of the permutation, and flips the order of the numbers as well as their signs. A single number as a segment can also be selected, and then the reversal acting on it simply changes its sign.

Example 1. *For example, the permutation* $-1 \quad -2 \quad -3 \quad -4$ *can be transformed to the* $+1 \quad +2 \quad +3 \quad +4$ *permutation in* 26 *different most parsimonious ways, two of them are*

$$\underline{-1} \quad -2 \quad -3 \quad -4$$

$$+1 \quad \underline{-2} \quad -3 \quad -4$$

$$+1 \quad +2 \quad \underline{-3} \quad -4$$

$$+1 \quad +2 \quad +3 \quad \underline{-4}$$

$$+1 \quad +2 \quad +3 \quad +4$$

and

$$\underline{-1 \quad -2 \quad -3} \quad -4$$

$$+3 \quad \underline{+2 \quad +1 \quad -4}$$

$$\underline{+3 \quad +4 \quad -1} \quad -2$$

$$+1 \quad \underline{-4 \quad -3 \quad -2}$$

$$+1 \quad +2 \quad +3 \quad +4$$

Given two genomes, one of them can be considered as the $+1 \quad +2 \quad \ldots +n$ permutation, and then the problem of transforming one signed permutation into another one can be considered as sorting a signed permutation into the $+1 \quad +2 \quad \ldots +n$ permutation. Still, an unichromosomal, linear genome can be represented as a signed permutation in two different ways, however, the sorting problem can be solved for both cases, and the representation yielding the smaller number of necessary reversals can be chosen. The minimum number of reversals necessary to sort a signed permutation π is called its *reversal distance* and denoted by $d_{REV}(\pi)$.

The following combinatorial objects play a central role at computing the reversal distance. To construct the *graph of desire and reality*, first a signed permutation of size n is transformed into an unsigned permutation of $2n$. Each positive number $+i$ is replaced by $2i-1,\ 2i$ and each negative number -1 is replaced by $2i,\ 2i-1$. The so-obtained numbers are framed between 0 and $2n+1$. For example, the permutation

$$+3 \quad -5 \quad -4 \quad +2 \quad +1$$

becomes

$$0 \quad 5 \quad 6 \quad 10 \quad 9 \quad 8 \quad 7 \quad 3 \quad 4 \quad 1 \quad 2 \quad 11.$$

A vertex is assigned to each number and we identify these vertices with their corresponding numbers. Starting with 0, every second vertex is connected with a straight line. In the above example, 0 is connected to 5, 6 to 10, etc. These edges are called the *reality edges*. Indeed, they represent the reality, that is, 0 is connected to 5 because the permutation starts with $+3$, 6 is connected to 10 since $+3$ is followed by -5, etc.

Also starting with 0, each even number $2i$ is connected to $2i+1$ with an arc (semicircle). These are called the *desire edges*. Indeed, this is what we want: we want 0 to be followed by 1, that is, we want the permutation to start with $+1$, etc.

The graph of desire and reality is a drawn graph, in which the vertices are drawn along a straight line, the reality edges are drawn as line segments and the desire edges are drawn as arcs (semicircles). It is also a multigraph, since there might be two edges between two vertices when the "reality meets the desire", that is, if $2i$ is followed by $2i+1$. Since each vertex has degree 2, the graph of desire and reality can be unequivocally decomposed into cycles.

The *overlap graph* is a vertex-colored simple graph. The vertices of the overlap graph are identified with the desire edges of the graph of desire and

reality. A vertex is colored by black if its corresponding desire edge spans over an odd number of vertices in the graph of desire and reality. Two vertices in the overlap graph are neighbors if their corresponding desire edges overlap, that is, their spanned intervals intersect, but none of them contains the other. For an example of graph of desire and reality and overlap graph, see Figure 1.3.

We have the following theorem.

Theorem 43. *([45]) Let π be a signed permutation of size n, and let $c(\pi)$ denote the number of cycles in its graph of desire and reality. If each non-trivial component in the overlap graph of π contains at least one black vertex, then the reversal distance of π is*

$$d_{REV}(\pi) = n + 1 - c(\pi).$$

For example, the $+3 \ -5 \ -4 \ +2 \ +1$ permutation satisfies that each non-trivial component in its overlap graph contains a black vertex, therefore its reversal distance is $5 + 1 - 3 = 3$.

If the overlap graph of a permutation contains non-trivial all-white components, some of them are so-called *hurdles*. For such genomes, the reversal distance is always greater than $n + 1 - c(\pi)$. We consider only *hurdle-free* permutations in this thesis, the interested reader is referred to [45] or [12].

We say that a reversal *acts* on a black vertex if it flips the segment delimited by the neighbor reality edges of the desire arc the black vertex represent. For example, the reversal acting on the $(10, 11)$ vertex of the overlap graph of permutation $+3 \ -5 \ -4 \ +2 \ +1$ flips the segment $-5 \ -4 \ +2 \ +1$. It can be shown (see, for example, [12]) that a reversal acting on a black vertex that maximizes the number of its white neighbors minus the number of its black neighbors is always a *sorting reversal*, that is, it decreases the reversal distance. In this way, it is easy to construct one sorting scenario. For example, a sorting scenario of $+3 \ -5 \ -4 \ +2 \ +1$ is

$$\begin{array}{rrrrr}
+3 & \underline{-5} & \underline{-4} & \underline{+2} & \underline{+1} \\
\underline{+3} & \underline{-1} & -2 & +4 & +5 \\
+1 & \underline{-3} & \underline{-2} & +4 & +5 \\
+1 & +2 & +3 & +4 & +5
\end{array}$$

It is also easy to show [12] that the reversal acting on a black vertex v has the following effect on the graph of desire and reality:

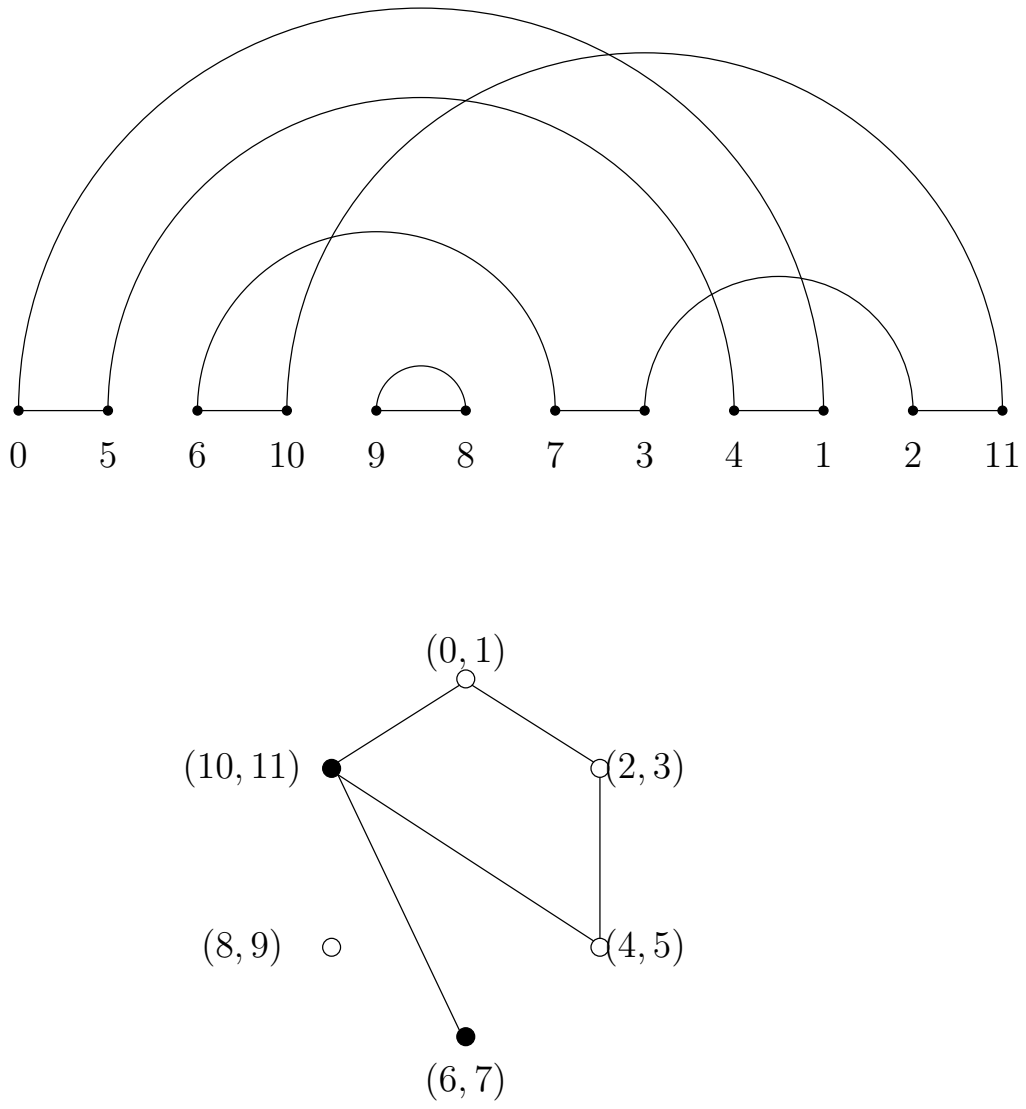


Figure 1.3: The graph of desire and reality as well as the overlap graph of the permutation $+3-5-4+2+1$

- All neighbors of v change color, that is, black vertices become white, white vertices become black.

- All pair of neighbors of v changes neighborhood. That is, for all pair

$u, w \in \mathcal{N}(v)$ if (u, w) is an edge in the overlap graph before the act of the reversal, (u, w) will not be an edge after the act of the reversal and vice versa.

- The vertex v becomes an isolated, white vertex.

A special class of signed permutations are those whose graph of desire and reality contains only cycles of length 4 and 2, and each non-trivial component in their overlap graph contains at least one black vertex. There are two important properties of this class. First, all reversals in all most parsimonious sorting by reversals act on black vertices. The second important property is that each extremity is effected by reversals at most once in all most parsimonious sorting by reversals. Biologists call this phenomena the *infinite site model* [63]. In the infinite site model, mutations (reverals) happen randomly. The length of the genome (the number of positions or sites where a mutation can happen) tends to infinity while the rate of the mutations per site tends to 0 in such a way that the expected number of mutations during a fixed time is constant. In this model, no reversals happen twice at a given site almost surely.

Although efficient algorithms exists to compute the reversal distance for any signed permutations (and not only those whose overlap graph contains only trivial components and non-trivial components with at least one black vertex) [45, 87], we know little on the number of solutions. It is conjectured that computing the number of most parsimonious sortings by reversals is #P-complete.

Single Cut or Join

The Single Cut or Join model is one of the simplest possible genome rearrangement models. Arbitrary genomes are allowed in this model, just like in the DCJ model, however, only two types of DCJ operations are allowed: the one that cuts a vertex of degree 2 into two vertices of degree 1, and the one that merges two vertices of degree 1 into a vertex of degree 2. The model was introduced by Feijão and Meidanis [32]. They computed the SCJ distance between two genomes. Let G_1 and G_2 be two genomes with the same edge labels, and let A_1 and A_2 be the set of the vertices of degree 2 in G_1 and G_2. We consider two vertices to be the same if they are incident to the same extremities. Feijão and Meidanis proved that

$$d_{SCJ}(G_1, G_2) = |A_1 \Delta A_2| \tag{1.30}$$

where Δ denotes the symmetric difference of two sets.They also showed that the *small parsimony problem* can be solved in polynomial time for the SCJ model. The small parsimony problem is the following. Given an unrooted binary tree $T = (V, E)$, and let $E = L \uplus I$ where L are the leaves of T and I are the internal nodes. Let a set of objects X are defined on which some distance $d : X \times X \to \mathbb{R}^{\geq 0}$ is defined. Furthermore, let a function $f : L \to X$ is given, The small parsimony problem is the following: extend f to a $g : V \to X$ such that

$$\sum_{(u,v)\in E} d(g(u), g(v))$$

is minimized. In the SCJ-small parsimony problem, X is a set of genomes with the same labels, and d is the SCJ distance. If a tree is a star with 3 leaves, then the small parsimony problem is also called the median problem. It is shown that the DCJ median and the reversal median problems are already NP-hard [17, 88].

Enumeration problems in genome rearrangement

In the joint work with Heather Smith [122], we discussed several counting problems that can be asked in any genome rearrangement model. Given any rearrangement model, we can define the following five counting problems:

1. **Pairwise rearrangement problem** Given two genomes, G_1 and G_2, and one of the rearrangement models, M, how many most parsimonious rearrangement scenarios exist that transform G_1 into G_2? We will denote this number by $n_M(G_1, G_2)$.

2. **Most parsimonious median problem** Given a series of genomes, G_1, $G_2 \ldots G_k$, and one of the rearrangement models, M, how many genomes G_m exist that minimize $\sum_{i=1}^{k} d_M(G_i, G_m)$, where $d_M(G, G')$ denotes the minimum number of operations needed to transform G into G' under the model M. We call each G_m an optimal median. The set of optimal medians will be denoted $\mathcal{O}_M(G_1, G_2, \ldots G_k)$.

3. **Most parsimonious median scenarios** Given a series of genomes, G_1, $G_2 \ldots G_k$, and one of the rearrangement models, M, how many optimal median scenarios exist. That is, count for all optimal medians

the number of possible rearrangement scenarios. With a formula, we are looking for

$$\sum_{G_m \in O_M(G_1,G_2,\ldots,G_k)} \prod_{i=1}^{k} n_M(G_i, G_m).$$

4. **Most parsimonious labeling of evolutionary trees** Given one of the rearrangement models, M, a rooted binary tree, $T(V, E)$, where V is the disjoint union of leaves L and internal nodes I. Furthermore, given a function $f : L \to \mathcal{G}$ that labels the leaves, where $\mathcal{G}$ denotes the set of possible genomes. We are looking for how many functions $g : V \to \mathcal{G}$ exists such that for any $v \in L$, $g(v) = f(v)$ and g minimizes

$$\sum_{(u,v)\in E} d_M(g(u), g(v)).$$

We denote this set of functions by $\mathcal{O}'_M(T, f)$.

5. **Most parsimonious scenarios on evolutionary trees** Given one of the rearrangement models, M, a rooted binary tree $T(V, E)$ and a labeling function f as described above, we are looking for

$$\sum_{g\in\mathcal{O}'_M(T,f)} \prod_{(u,v)\in E} n_M(g(u), g(v)).$$

1.2.3 Biological sequences

We will use the following notations. If Γ is a finite set of symbols, then Γ^* denotes the set of finite sequences from Γ. For $A \in \Gamma^*$, $|A|$ denote the length of the sequence. Indexed small case letters denote the characters of a sequence denoted by a capital letter. For example, $A = a_1a_2 \ldots a_n$. The *prefix* of A is $A_i := a_1a_2 \ldots a_i$, and a *suffix* of A is $A^i := a_{i+1}a_{i+2\ldots a_n}$. That is, if writing sequences next to each other denotes *sequence concatenation*, then for any A and i,

$$A = A_iA^i.$$

DNA sequences can be modeled as sequences over the alphabet $\{A, C, G, T\}$, RNA sequences can be modeled as sequences over the alphabet $\{A, C, G, U\}$. These letters encode small molecules called *nucleotides* or *bases*. Nucleotides

can make covalent bonds to each other thus form a linear macromolecule. Proteins can be modeled as sequences over the alphabet of 20 amino acids. The DNA sequences are double stranded, the two strands are the reverse complement of each other. RNA and protein sequences are single stranded molecules, which can form complex 3-dimensional structures.

Obtaining the sequences of the biological sequences is quite cheap, however, finding their structure is still a tedious, money and time-consuming laboratory work that needs lots of man-power, too. Therefore, the *in silico* structure prediction is an important task in bioinformatics. The biological structures can be modelled with stochastic models like Hidden Markov Models, Stochastic Regular Grammars and Stochastic Context-free Grammars. The most likely generation is considered as the predicted structure. The RNA structures are also have a biochemical model called the Zucker-Tinoco energy model. In this model, the predicted structure is the one which has the minimal free energy. For all the above-mentioned prediction methods, dynamic programming algorithms are available. Still, there were a few open algorithmic problems whose solutions are introduced in this thesis. In this chapter, we introduce the mathematical models for the structure prediction in details.

Biological structure prediction

Stochastic transformational grammars are frequently used in biological structure prediction. Transformational grammars have been invented by Noam Chomsky [18]. Regular grammars are one of the simplest transformational grammars as they are in the lowest level of the Chomsky hierarchy [19]. Stochastic versions of regular grammars are related to Hidden Markov Models [84, 10].

Definition 44. *A* regular grammar *is a tuple* $(T, N, \mathbb{S}, R)$*, where* T *is a finite set called* terminal characters*,* N *is a finite set called* non-terminal characters*. The sets* T *and* N *are disjoint.* $\mathbb{S} \in N$ *is a special non-terminal character, called starting non-terminal.* R *is a finite set of* rewriting rules*, each in one of the following forms*

$$W \rightarrow xW' \tag{1.31}$$
$$W \rightarrow x \tag{1.32}$$
$$W \rightarrow \epsilon \tag{1.33}$$

where $W, W' \in N$, $x \in T$, ϵ denotes the empty sequence. The shorthand for the rules (1.31)-(1.33) is

$$W \to xW' \mid x \mid \epsilon.$$

A generation *is a finite series of transformations*

$$\mathbb{S} = X_0 \to X_1 \to \ldots \to X_k \in T^* \tag{1.34}$$

where for each $i = 1, \ldots, k-1$, there exists a rule $W \to \beta$ and a word $X_p \in T^*$ such that $X_i = X_p W$ and $X_{i+1} = X_p \beta$. (Here β might be any sequence appearing at the right-hand side of a rewriting rule.) T^* denotes the finite sequences from T. The language *$L_G \subseteq T^*$ contains those sequences that can be generated by the grammar. A grammar is said to be* unambiguous *if any sequence $X \in L_G$ can be generated in exactly one way. An* ambiguous *grammar contains at least one sequence in its language that can be generated in at least two different ways.*

A generation is possible if there is a non-terminal in the intermediate sequence. Once the sequence contains only terminal characters, the generation is terminated. This is the rationale behind the naming of terminals and non-terminals.

Given a sequence $X \in T^*$, the following questions can be asked:

(a) Is $X \in L_G$?

(b) How many generations are there which produce X?

(c) Given a function $w : R \to \mathbb{R}^{\geq 0}$, which generation $\mathcal{G}$ maximizes

$$\prod_{i=0}^{k-1} w(W_i \to \beta_i)$$

where the rewriting rule $W_i \to \beta_i$ is applied in the ith step of the generation $\mathcal{G}$ generating X in k steps?

(d) Given a function $w : R \to \mathbb{R}^{\geq 0}$, compute

$$\sum_{\mathcal{G}_i} \prod_{j=0}^{k_i-1} w(W_{i,j} \to \beta_{i,j})$$

where the rewriting rule $W_{i,j} \to \beta_{i,j}$ is applied in the jth step of the generation $\mathcal{G}_i$ generating X in k_i steps.

As can be seen, non-terminals play the role of "memory" in generating sequences. Above the non-terminals at the end of the intermediate sequences, the generation is memoryless. We can consider the stochastic version of regular grammars, and the memoryless property provides that the stochastic versions of regular grammars are Markov processes. The stochastic regular grammars can be defined in the following way.

Definition 45. *A* stochastic regular grammar *is a tuple* $(T, N, \mathbb{S}, R, \pi)$*, where* $(T, N, \mathbb{S}, R)$ *is a regular grammar and* $\pi : R \to \mathbb{R}^+$ *is a probability distribution for each non-terminal, that is, for each* $W \in N$*, the equality*

$$\sum_{\beta|(W\to\beta)\in R} \pi(W \to \beta) = 1 \tag{1.35}$$

holds. (Recall that β *is any of the sequences that might appear in the right-hand side of a rewriting rule of a regular grammar, including the empty sequence.) A stochastic regular grammar makes random generations*

$$S = X_1 \to X_2 \to \dots$$

where in each rewriting step, the rewriting rule $W \to \beta$ *is chosen randomly following the distribution* π*.*

The random generation in a stochastic regular grammar can be viewed as a random process, in which the states are the intermediate sequences. This process indeed has the Markovian property. That is, what the intermediate sequence X_{i+1} is depends only on X_i and does not depend on any X_j, $j < i$. One can ask for a given sequence $X \in T^*$ what the most likely generation and the total probability of the generation are. This latter is the sum of the probabilities of the possible generations. Both questions can be answered by algebraic dynamic programming algorithms introduced in Chapter 2.

Stochastic regular grammars are closely related to Hidden Markov Models.

Definition 46. *A* Hidden Markov Model *is a tuple* $(\vec{G}, START, END, \Gamma, T, e)$*, where* $\vec{G} = (V, E)$ *is a directed graph in which loops are allowed but parallel edges are not,* $START$ *and* END *are distinguished vertices in* $\vec{G}$*,* $START$ *has* 0 *in-degree,* END *has* 0 *out-degree,* Γ *is a finite set of symbols, called an*

alphabet, $T : E \to \mathbb{R}^+$ *is the transition probability function satisfying for all* $u \neq END$

$$\sum_{v|(u,v)=e \in E} T(e) = 1.$$

and $e : \Gamma \times (V \setminus \{START, END\}) \to \mathbb{R}^{\geq 0}$ *is the emission probability function satisfying for all* $v \in V \setminus \{START, END\}$

$$\sum_{x \in \Gamma} e(x, v) = 1$$

The vertices of $\vec{G}$ *are called states. A random walk on the states is defined by* $\vec{G}$ *and the transition probabilities. The random walk starts in the state* $START$ *and ends in the state* END*. During such a walk, states emit characters according to the emission distribution* e*. In case of loops, the random walk might stay in one state for several steps. A random character is emitted in each step. The process is hidden in the sense that an observer can see only the emitted characters and cannot observe the random walk itself. An* emission path *is a random walk together with the emitted characters. The probability of an emission path is the product of its transition and emission probabilities.*

If (u, v) is an edge, then the notation $T(v|u)$ is also used, emphasizing that T is a conditional distribution. Indeed $T(v|u)$ is the probability that the Markov process will be in state v in the next step given that it is in the state u in the current step.

One can ask, for an $X \in \Gamma^*$, what the most likely emission path and the total emission probability are. This latter is the sum of the emission path probabilities that emit X. These questions are equivalent with those that can be asked for the most likely generation and the total generation probability of a sequence in a stochastic regular grammar, stated by the following theorem.

Theorem 47. *For any Hidden Markov Model* $H = (\vec{G}, START, END, \Gamma, T, e)$*, there exists a stochastic regular grammar* $G = (T, N, \mathbb{S}, R, \pi)$ *such that* $\Gamma = T$*,* L_G *is exactly the set of sequences that* H *can emit, and for any* $X \in L_G$*, the probability of the most likely generation in* G *is the probability of the most likely emission path in* H*, and the total generation probability of* X *in* G *is the total emission probability of* X *in* H*. Furthermore, the running time needed to construct* G *is a polynomial function of the size of* $\vec{G}$*.*

The proof can be found, for example, in [115].

Due to their similarities, similar problems will be tractable for HMMs than for stochastic regular grammars. The algorithms solving these problems are well known in the scientific literature. The algorithm finding the most likely emission path is known as the Viterbi algorithm [93, 36] and the algorithm summing the probabilities of possible emission paths is called the Forward algorithm [9, 11]. It is also well-known in the Hidden Markov Model literature that "the Viterbi algorithm is similar [...] in implementation to the forward calculation" [72].

DNA sequences have a fixed chemical structure (the double stranded helix structure), however, they still might have (functional) structure. For example, DNA sequences have coding and non-coding regions. There are $3^4 = 64$ possible triplets of the letters $\{A, C, G, T\}$. Only 61 of them codes an amino acid (this is a surjective but not injective mapping from the 61 coding triplets to the 20 amino acids), the remaining 3 are called *stop codons* and encode the end of the coding region. A typical protein sequence contains several hundreds of amino acids, thus, in the corresponding DNA coding regions, there are several hundreds of consecutive coding triplets without any stop codon. This is a statistical signal that provides the prediction of coding regions with Hidden Markov Models. When a new species is sequenced, its DNA sequence is obtained. Biologist would like to predict the protein sequences of this new species by predicting the coding regions of its DNA sequence and translating the coding triplets into amino acid sequences, that is, into proteins.

Also, some parts of the DNA sequences are rich in CpG motifs, called *CpG islands*. A CpG motif is simply a CG substring in the DNA sequence, the 'p' notation is for emphasising that the two characters are on the same strand of the DNA sequence. The *non-CpG island* parts are richer in other substrings of length 2 due to chemical reasons not explained here. This is again a statistical signal that provides the prediction of CpG islands. The CpG islands play a central role in gene regulation.

Protein sequences have complex three dimensional structures, that can be decomposed into smaller structural parts. We can distinguish α-helices, β-sheets, and unstructured elements. These are called secondary structure types. The amino-acid distribution in these three structural types are different, and again, this allows the prediction of secondary structures of protein sequences with Hidden Markov Models.

RNA sequences can fold back and make base-pairs using hydrogen bonds. Above the standard, so-called Watson-Crick basepairs – A-U, C-G, G-C, and

U-A – two further basepairs, G-U and U-G are possible. These base-pairs stabilizes the structure.

Definition 48. *Let S be an RNA sequence of length n, that is, a finite long sequence over the alphabet $\{A, C, G, U\}$. A* pseudoknot-free RNA secondary structure *is a collection of index pairs (i, j), $i < j$ such that any index is in at most one pair, for all pairs (i, j), (s_i, s_j) is one of the following pairs: (A, U), (C, G), (G, C), (U, A), (G, U), (U, G), and for all (i_1, j_1) and (i_2, j_2) such that $i_1 < i_2$ either $j_1 < i_2$ or $j_2 < j_1$. That is, the pairs are either nested or separated. A basepair (i, j) is called* outermost *on the substring from b to e if there is no basepair (i', j') with $i' < i$ and $j < j'$.*

We consider in this thesis only pseudoknot-free RNA secondary structures and we simply call them RNA secondary structures *or* RNA structures*.*

We can construct a graph of an RNA structure, $G(V, E)$, which is an edge colored planar graph. The colors are represented by straight and dotted lines. The vertex set V are the bases of the RNA sequence $b_1 b_2 \dots b_n$. There are two types of edges. The edges with straight line represent the covalent bonds, thus, for all $1 \leq i < n$, $(b_i, b_{i+1}) \in E$. The dotted lines represent the hydrogen bonds. Therefore, for all basepairs (i, j), $(b_i, b_j) \in E$.

It is easy to see that $G(V, E)$ is a planar graph. Indeed, if the vertices are drawn along a line, the straight edges will be all along this line. The dotted lines can be drawn by arcs (semicircles), and due to the definition of RNA structures, they cannot cross each other (and they cannot cross the straight lines, too). Furthermore, $G(V, E)$ can unequivocally decomposed into cycles and a path called *null cycle* such that each dotted line is in exactly two cycles (or in a cycle and in the path). It can be obtained iteratively in the following way.

1. The null cycle contains the basepairs which are outermost on the whole RNA sequence and the covalent bonds between b_i and b_{i+1} such that no basepair (i', j') exists with $i' \leq i$ and $i + 1 \leq j'$. This null cycle initiate the iteration, the iteration goes for each outermost basepair in the current cycle.

2. Now for each basepair (i, j) that was outermost in the previous segment, there is a cycle that contains (b_i, b_j), all basepairs (i', j') that are outermost on the segment from i to j and all covalent bonds between b_k and b_{k+1} with $i \leq k$, $k+1 \leq j$ such that no basepair $(i", j")$ exists with

i" $\leq k$ and $k+1 \leq j$". We can classify these cycles in the following way:

(a) If there is no outermost basepair (and thus, no basepair above (i,j) in the current segment), then the cycle is called *hairpin cycle*.

(b) If there is one outermost basepair (i',j') and $i+1=i'$ and $j'+1=j$, then the cycle is called *stacking pair*.

(c) If there is one outermost basepair (i',j') and $i+1=i'$ or $j'+1=j$ but not both equation holds, then the cycle is called *bulge*.

(d) If there is one outermost basepair (i',j') and neither $i+1=i'$ nor $j'+1=j$, then the cycle is called *internal loop*.

(e) If there are more than one outermost basepairs, then the cycle is called *multiloop*.

In the Zucker-Tinoco energy model [91], a free energy is assigned to each cycle, and the free energy of the RNA structure is the sum of these free energies. To get an energy model in which the minimal free energy can be computed in $O(n^3)$ time, the functions assigning a free energy to internal loops, bulges and multiloops are simplified in the following way.

Lyngsø *et al.* [62] considered that the free energy assigned to an internal loop or bulge with basepairs (i,j) and $(i',j+)$ can be decomposed as

$$\begin{aligned} eL(i,j,i',j') &= \text{size}(i'-i+j-j'-2)+ \\ &\quad \text{asymmetry}(i'-i-1,j-j'-1)+ \\ &\quad \text{stacking}(b_i,b_j)+\text{stacking}(b'_i,b'_j). \end{aligned} \tag{1.36}$$

Here, the stacking$(\cdot,\cdot)$ function depends on only the type of the bases and not their positions. Furthermore, it is assumed that

$$\text{asymmetry}(k+1,l+1)=\text{asymmetry}(k,l)+f(k+1). \tag{1.37}$$

The free energy of a multiloop M is considered as

$$eM(M)=M_c+M_I\times\text{interiorpair}(M)+M_B\times\text{unpaired}(M) \tag{1.38}$$

where M_c, M_I and M_B are constants, interiorpair(M) is the number of outermost basepairs in the internal loops and unpaired(M) are the number of unpaired bases in the internal loop [97].

1.3 Results presented in this thesis

The results in this thesis are presented from the computational complexity point of view. The thesis has four parts. It introduces problems for which exact computations are possible in polynomial time and also problems for which efficient approximations are possible. It introduces positive engineering work, that is, Markov chains with non-trivial kernels and small diameters and negative results: torpidly mixing Markov chains, #P-complete and non-approximable problems. We introduce first the negative results, since in the light of these negative results, it is easier to understand why the engineering work can be considered as positive results.

1. **Counting problems in FP.** Most of the counting problems solvable in polynomial time have a dynamic programming solution. We give an algebraic framework of dynamic programming and show that many computational problems can be described as computing the homomorph image of a certain semiring. We give two non-trivial applications: computing the moments of the Boltzmann distribution of RNA secondary structures and performing Baum-Welch training in linear memory. We also give solutions of two counting problems under the SCJ model. Counting the most parsimonious scenarios between two genomes under the SCJ model can be reduced to counting alternating permutations. Counting most parsimonious medians can be reduced to counting matchings in graphs with maximum degree 2.

2. **Counting problems in FPRAS via rapidly mixing Markov chains.** The switch Markov chain is conjectured to be rapidly mixing for any degree sequences. The degree sequences can be factorized using the Thyskevich-decomposition and we showed that it is sufficient to prove rapid mixing for each factor. We prove that the switch Markov chain is rapidly mixing for P-stable degree sequences, and any degree sequence is P-stable if the degrees are between some linear bounds. We show that this bound is tight, that is, there are non-P-stable degree sequences above these linear bounds. The counterexample is given by Thyskevich-product of certain factors. We show that one of the factor or its complement appears as an induced subgraph of a realization in any class of non-P-stable sequences. We can prove that the switch Markov chain is rapidly mixing on the realizations of this non-P-stable factor. We show that the switch Markov chain is irreducible on the

balanced realizations of a JDM, and this Markov chain is rapidly mixing. Finally, we give an FPRAS to approximate the number of most parsimonious rearrangement scenarios under the DCJ model.

3. **Negative results: torpid mixing, #P-complete and non-approximable problems**

   We show that the only known irreducible Markov chain on the most parsimonious reversal scenarios is torpidly mixing. The torpid mixing is caused by the necessity of large perturbations to get an irreducible Markov chain.

   We also give two negative results on SCJ rearrangement problems. We show the non-approximability of counting the most parsimonious rearrangement scenarios on an evolutionary tree and also prove its #P-completeness. We show that the problem remains #P-complete on star trees, although we do not know whether or not the problem remains non-approximable. Our results hold even for the case when the genomes do not contain so-called conflicting adjacencies. Two adjacencies are in conflict if they are different but share an extremity.

4. **Non-trivial kernels and diameters of Markov chains.** We have a conjecture that a certain Markov chain with small perturbations is irreducible on the most parsimonious reversal rearrangement scenarios under the infinite site model. We prove this conjecture for the class of permutations whose overlap graph has maximum degree 2.

   We show how to cool down most parsimonious rearrangement scenarios under the DCJ model to most parsimonious rearrangemnet scenarios under the reversal model. Defining a topology using small perturbations and using an energy model in which the minimum energy rearrangement scenarios are the reversal scenarios, we show that for a large class of permutations, every local minimum is a global one. We show that polynomial number of parallel chains are sufficient to achieve high communication rate between parallel chains while having the uniform distribution in the hottest chain (that is, the uniform distribution of DCJ scenarios) and significant fraction of reversal scenarios in the coldest chain.

   We give a Gibbs sampler of optimal SCJ labeling on an arbitrary evolutionary tree.

We give necessary and sufficient perturbations of half-regular factorizations of the complete bipartite graph as well as necessary and sufficient perturbations of edge-colorings of bipartite graphs. In some sense, these perturbations are large as they might affect arbitrary number of vertices in the non-regular vertex class or arbitrary number of edges in the edge coloring. On the other hand, these perturbations are small since they perturbe at most three vertices of the regular vertex class or at most three colors of the edge coloring. We prove that the inverse of the acceptance ratio in the Metropolis-Hastings algorithm is polynomially bounded. Also, we show that in both cases, the Markov chain has a small diameter.

Part I

Counting problems in FP

dc_2046_22

Chapter 2

Algebraic Dynamic Programming

In this chapter we introduce the concept of the algebraic dynamic programming and show two non-trivial applications: computing moments of the Boltzmann distribution of RNA sequences and performing Baum-Welch training in linear memory. This chapter is based on the second chapter of the monograph Miklós, I. (2019) Computational Complexity of Counting and Sampling, Chapman and Hall/CRC, ISBN 9781138035577 - CAT# K31733 and two further publications: Miklós, I., Meyer, I.M., Nagy, B. (2005) Moments of the Boltzmann distribution for RNA secondary structures Bul. Math. Biol., 67(5):1031–1047 and Miklós, I., Meyer, I.M. (2005) A linear memory algorithm for Baum-Welch training. BMC Bioinformatics 6:231. In both paper, the author of this thesis proved the main theorems (Theorems 50 and 51, Lemma 52 and Theorem 53).

2.1 Introduction to Algebraic Dynamic Programming

The idea that the recursion and the actual computation can be separated in dynamic programming algorithms appeared shortly after introducing the concept of dynamic programming. The Aho-Ullman-Hopcroft textbook on algorithms published in 1974 [1] already mentions that in optimization problems, the minimization and addition form a semiring. This is known today as the *tropical semiring*, although this name was coined only in the '80-s

[80]. Jerrum and Snir in a paper published in 1982 talks about "universal algorithms" where "the family of monotone computations are essentially the same in any semiring" [53]. Helman and Rosenthal [48] partitioned problems into "problem structure" and "optimization problem", and they stated that "we can use the same main program and simply redefine a single function". Although this method to dynamic programming is still not widespread, the Cormen textbook on algorithms [23] mentions that some basic shortest path algorithms like the Floyd-Warshall algorithm [35, 74, 95] can be considered as matrix multiplication over the tropical semiring. The textbook discusses other semirings, too, for example, the min-max semiring that can be used to solve the widest path problem.

Bioinformaticians, especially those that worked with RNA structure prediction also rediscovered that the problem structure can be separated from the actual computations. They coined the name *algebraic dynamic programming* [40, 41]. A generalization to this approach was presented in the monograph on computational complexity by the author of this thesis [115]. Here we give a simplified version of that work.

Let X be a finite alphabet and let X^* denote the set of finite sequences over X, including the empty sequence that we will denote by ε. Let $\circ$ denote the sequence concatenation, that is,

$$a_1 a_2 \dots a_n \circ b_1 b_2 \dots b_m := a_1 a_2 \dots a_n b_1 b_2 \dots b_m$$

Clearly, the sequence concatenation is an associative operation on X^* and ε is both a left and a right unit, therefore $(X^*, \circ)$ is a non-commutative monoid. This monoid is known in abstract algebra as the *free non-commutative monoid* generated by the alphabet X.

Let $\mathbb{X}$ be the set of finite multisets of X^*. We define $\oplus$ on $\mathbb{X}$ as the union of the corresponding sets, that is, for all $\mathcal{A}, \mathcal{B} \in \mathbb{X}$, let

$$\mathcal{A} \oplus \mathcal{B} := \mathcal{A} \cup \mathcal{B},$$

where $\cup$ is the multiset union. Further, we define $\odot$ as the "Minkowski product" of two (multi)sets, that is, for all $\mathcal{A}, \mathcal{B} \in \mathbb{X}$, let

$$\mathcal{A} \odot \mathcal{B} := \{A \circ B | A \in \mathcal{A},\ B \in \mathcal{B}\}.$$

Here any sequence in $\mathcal{A}$ or $\mathcal{B}$ is considered with multiplicity, that is, if A is in $\mathcal{A}$ n_1 times, and B is in $\mathcal{B}$ n_2 times, then $\mathcal{A} \odot \mathcal{B}$ contains $A \circ B$ $n_1 \times n_2$ times.

It is easy to see that $(\mathbb{X}, \oplus)$ is a commutative monoid in which $\emptyset$ is a unit, and $(\mathbb{X}, \odot)$ is a non-commutative monoid in which $\{\varepsilon\}$ is the unit. Also, the associative rules

$$\mathcal{A} \odot (\mathcal{B} \oplus \mathcal{C}) = (\mathcal{A} \odot \mathcal{B}) \oplus (\mathcal{A} \odot \mathcal{C})$$

and

$$(\mathcal{B} \oplus \mathcal{C}) \odot \mathcal{A} = (\mathcal{B} \odot \mathcal{A}) \oplus (\mathcal{C} \odot \mathcal{A})$$

holds. Furthermore, for all $\mathcal{A} \in \mathbb{X}$,

$$\mathcal{A} \odot \emptyset = \emptyset \odot \mathcal{A} = \emptyset,$$

therefore, $(\mathbb{X}, \oplus, \odot)$ is a semiring.

Theorem 49. *Let $(R, +, \otimes)$ be an arbitrary semiring, and let $f : X^* \to R$ be a multiplicative function, satisfying that for all $A, B \in X^*$,*

$$f(A \circ B) = f(A) \otimes f(B).$$

Then $\varphi : \mathbb{X} \to R$ defined as

$$\varphi(\mathcal{A}) := \sum_{A \in \mathcal{A}} f(A)$$

is a semiring-homomorphism.

Proof. It is easy to see that both the addition and the multiplication is preserved. Indeed,

$$\varphi(\mathcal{A} \oplus \mathcal{B}) = \sum_{C \in \mathcal{A} \cup \mathcal{B}} f(C) = \sum_{A \in \mathcal{A}} f(A) + \sum_{B \in \mathcal{B}} f(B) = \varphi(\mathcal{A}) + \varphi(\mathcal{B}).$$

Similarly,

$$\varphi(\mathcal{A} \odot \mathcal{B}) = \sum_{A,B \mid A \in \mathcal{A},\ B \in \mathcal{B}} f(A \circ B) = \sum_{A,B \mid A \in \mathcal{A},\ B \in \mathcal{B}} f(A) \otimes f(B) =$$

$$= \sum_{A \in \mathcal{A}} \sum_{B \in \mathcal{B}} f(A) \otimes f(B) = \left(\sum_{A \in \mathcal{A}} f(A) \right) \otimes \left(\sum_{B \in \mathcal{B}} f(B) \right) = \varphi(\mathcal{A}) \otimes \varphi(\mathcal{B}).$$

□

For example, if R is the semiring of the natural numbers, and

$$f(A) \equiv 1,$$

then $\varphi(\mathcal{A})$ computes the size of $\mathcal{A}$. If R is the tropical semiring, and

$$f(A) = |A|$$

for all $A \in X^*$, where $|A|$ is the length of the sequence, then $\varphi(\mathcal{A})$ computes the length of the shortest sequence in $\mathcal{A}$. If R is the polynomial semiring over the natural numbers, $\mathbb{N}[x]$, and

$$f(A) = x^{|A|}$$

then $\varphi(\mathcal{A})$ computes the length statistics of $\mathcal{A}$. That is, if

$$\varphi(\mathcal{A}) = \sum_k a_k x^k$$

then a_k is the number of sequences of length k in $\mathcal{A}$ (with multiplicity).

Algebraic dynamic programming demonstrated on regular grammars

In algebraic dynamic programming, the solution space is built as a (multi)set of sequences. For example, consider a regular grammar (T, N, S, R), where T is the finite set of terminal characters, N is the finite set of non-terminal characters, S is the start non-terminal (sometimes called as axiom), and R is the set of rewriting rules all in form $W \to aW'|\varepsilon$, where $a \in T$ and $W, W' \in N$. Given $A \in T^*$, we are interested in the possible generation of $A = a_1 a_2 \dots a_n$ via a series of rewriting:

$$S \to a_1 W_1 \to a_1 a_2 W_2 \to \dots \to a_1 a_2 \dots a_n W_n \to a_1 a_2 \dots a_n.$$

Every such series of rewritings can be identified with the sequence of rewritings, that is, a sequence over R^*. For each $k \in [0, n]$ and $W \in N$, let $s(k, W)$ denote the set of series of rewritings from S to $a_1 a_2 \dots a_k W$. Then the initial conditions are

$$s(0, W) = \begin{cases} \{\varepsilon\} & \text{if } W = S \\ \emptyset & \text{otherwise} \end{cases} \tag{2.1}$$

The main recursion is

$$s(k, W) = \bigcup_{W'|(W' \to a_k W) \in R} s(k-1, W') \odot \{W' \to a_k W\}. \tag{2.2}$$

The termination is

$$s(n, \varepsilon) = \bigcup_{W|(W \to \varepsilon) \in R} s(n, W) \odot \{W \to \varepsilon\}. \tag{2.3}$$

Now let B be the Boolean semiring, $(\{True, False\}, \vee, \wedge)$, and for all $C \in R^*$,

$$f(C) \equiv True,$$

Then for the corresponding $\varphi : R^* \to B$,

$$\varphi(s(n, \varepsilon)) = True$$

if and only if A is the part of the language generated by the grammar (T, N, S, R). We can perform the computations in the Boolean semiring B. That is, if $\varphi(s(k, W))$ is denoted by $v(k, W)$, then the Equations 2.1-2.3 becomes

$$v(0, W) = \begin{cases} True & \text{if } W = S \\ False & \text{otherwise} \end{cases} \tag{2.4}$$

$$v(k, W) = \bigvee_{W'|(W' \to a_k W) \in R} v(k-1, W') \wedge True \tag{2.5}$$

$$v(n, \varepsilon) = \bigvee_{W|(W \to \varepsilon) \in R} v(n, W) \wedge True. \tag{2.6}$$

In these equations, the $\wedge True$ is written for didactic reasons, highlighting that the $\odot$ operation is replaced by the $\wedge$ operation and any non-empty set is replaced by the $True$ value. The Equations 2.4-2.6 are called the *Viterbi recursions*.

Let $w : R \to \mathbb{R}^{\geq 0}$ be a weight function. This is frequently a probability distribution function such that for all $W \in N$,

$$w(W \to \varepsilon) + \sum_{a, W'|(W \to aW') \in R} w(W \to aW') = 1. \tag{2.7}$$

Now we consider the semiring of the non-negative real numbers $(\mathbb{R}^{\geq 0}, +, \times)$ and we define for any $r_1 r_2 \ldots r_k \in R^*$,

$$f(r_1 r_2 \ldots r_k) := \prod_{i=1}^{k} w(r_i).$$

This is clearly a multiplicative function. The corresponding homomorphism $\varphi : R \to \mathbb{R}^{\geq 0}$ computes the sum of the weights of the generations of the given $A \in T^*$. If the weights are probability distributions in the sense of Equation 2.7, then (T, N, S, R, w) is called a *stochastic regular grammar*, and $\varphi(s(n, \varepsilon))$ computes the generation probability of sequence A. Again, we can perform the computations expressed in the Equations 2.1-2.3 in the homomorph image. That is, if $\varphi(s(k, W))$ is denoted by $f(k, W)$, then

$$f(0, W) = \begin{cases} 1 & \text{if } W = S \\ 0 & \text{otherwise} \end{cases} \tag{2.8}$$

$$f(k, W) = \sum_{W' | (W' \to a_k W) \in R} v(k-1, W') \times w(W' \to a_k W) \tag{2.9}$$

$$f(n, \varepsilon) = \sum_{W | (W \to \varepsilon) \in R} v(n, W) \times w(W \to \varepsilon). \tag{2.10}$$

The dynamic programming algorithm expressed in Equations 2.8-2.10 is known as the *Forward algorithm* of the stochastic regular grammars [27].

We can also consider the semiring $(\mathbb{R}^{\geq 0} \cup \{-\infty\}, \max, \times)$. Then for a weight function

$$f(r_1 r_2 \ldots r_k) := \prod_{i=1}^{k} w(r_i).$$

if w is a probability distribution, then the corresponding homomorphism computes the most likely generation of sequence A. We still can perform the computations in the homomorph image. That is, if $\varphi(s(k, W))$ is denoted by $v(k, W)$, then

$$v(0, W) = \begin{cases} 1 & \text{if } W = S \\ -\infty & \text{otherwise} \end{cases} \tag{2.11}$$

$$v(k, W) = \max_{W' | (W' \to a_k W) \in R} \{v(k-1, W') \times w(W' \to a_k W)\} \tag{2.12}$$

$$v(n, \varepsilon) = \max_{W | (W \to \varepsilon) \in R} \{v(n, W) \times w(W \to \varepsilon)\}. \tag{2.13}$$

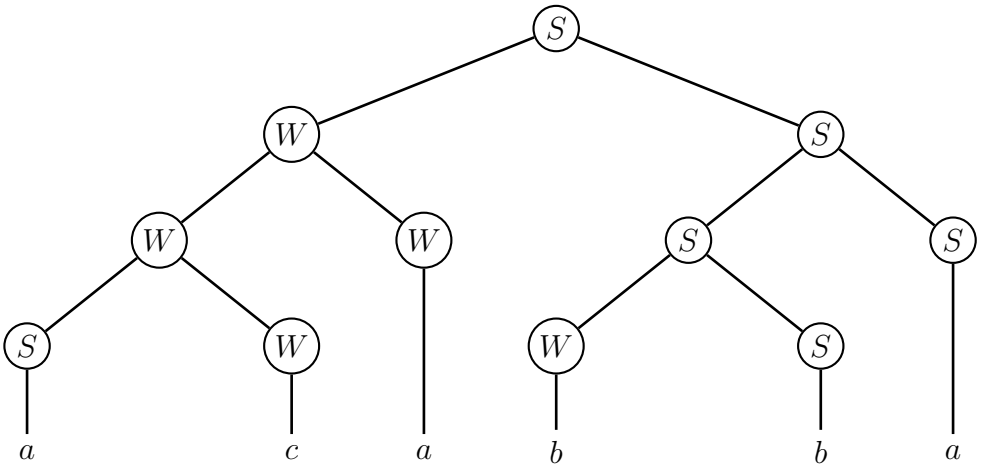


Figure 2.1: The parse tree of a possible generation of sequence *acabba* by the context-free grammar with rewriting rules $S \longrightarrow WS|SS|a|b$, $W \longrightarrow WW|SW|a|b|c$.

Equations 2.11-2.13 is known as the *Viterbi algorithm* of the stochastic regular grammars [27].

Algebraic dynamic programming demonstrated on context free grammars

Consider a context-free grammar $G = (T, N, S, R)$, where T is the finite set of terminal characters, N is the finite set of non-terminal characters, S is the starting non-terminal and R is a finite set of rules in form $W \to \beta$, where $W \in N$ $\beta \in (T \cup N)^* \setminus \{\varepsilon\}$. A context-free grammar is in *Chomsky Normal Form* if all rules are in form $W \to W_1W_2|a$, where $W, W_1, W_2 \in N$ and $a \in T$. It can be shown that any context-free grammar can be rewritten into Chomsky Normal Form such that the rewritten form generates the same language.

Let $A \in T^*$, and we are interested in the possible generations of A by G. Any generation can be described as a *parse tree*, see for example Figure 2.1. The nodes of the parse tree can be visited in a dept first search, and thus can be listed unequivocally. For example, the parse tree in Figure 2.1 corresponds

to the sequence

$$S \to WS,\ W \to WW,\ W \to SW,\ S \to a,\ W \to c,$$

$$W \to a,\ S \to SS,\ S \to WS, W \to b,\ S \to b,\ S \to a.$$

Then the parse trees can be represented as sequences from R^*. Let $s(i, j, W)$ denote the set of sequences from R^* corresponding to parse trees that generate the substring $a_i a_{i+1} \dots a_j$ and is rooted in W. Then the initial conditions are

$$s(i, i, W) = \begin{cases} \{W \to a_i\} & \text{if } (W \to a_i) \in R \\ \emptyset & \text{otherwise} \end{cases} \tag{2.14}$$

The main recursion is

$$\begin{aligned} &s(i, j, W) = \\ &\bigcup_{i \le k < j} \quad \bigcup_{W_1, W_2 | (W \to W_1 W_2) \in R} [\\ &s(i, k, W_1) \odot s(k + 1, j, W_2) \odot \{W \to W_1 W_2\}] \end{aligned} \tag{2.15}$$

The set of parse trees that generates A are in $s(1, n, S)$, where $n = |A|$. We can again plug-in several semirings into Equations 2.14 and 2.15 to decide if A is part of the language generated by the context free grammar, to count the parse trees generating A, to find the best scored parse tree or compute the sum of multiplicative scores of parse trees that generate A. The decision/optimization algorithm is also known as the *CYK algorithm* and the counting/summing algorithm is also known as the *Inside algorithm* in the scientific literature [27].

2.2 Moments of the Boltzmann distribution of RNA secondary structures

The *Boltzmann distribution* of RNA structures is defined such that the probability of a structure S at temperature T measured in Kelvin is

$$P(S) = \frac{e^{-\frac{G(S)}{RT}}}{Z}$$

where $G(S)$ is the free energy of the structure measured in J/mol, R is the universal gas constant, $8.314\frac{J}{mol\times K}$, and Z is called the *partition function* defined as

$$Z := \sum_{S'} e^{-\frac{G(S')}{RT}},$$

where the summation is over all possible structures S' of the given RNA sequence. We are interested in computing the minimum free energy structure, the partition function and the moments of the Boltzmann distribution. The partition function and the moments of the Boltzmann distribution are always calculated for a given temperature T.

Due to the Equations 1.36-1.38, the dynamic programming algorithm of RNA structures can be rewritten into a context free grammar $G = (T, N, S, R)$ satisfying the following properties:

1. The grammar is in Chomsky Normal Form.

2. There is a one-to-one correspondence between the possible RNA structures and the generations of the RNA sequence by the grammar.

3. The rewriting rules of G can be scored by a function $g : R \to \mathbb{R}$ in such a way that for each parse tree generating A, the sum of the scores of the rewriting rules is the free energy of the corresponding structure.

This allows to plug in appropriate semirings into the recursions of the context free grammar. When we are interested in the minimum free energy, the semiring is the tropical semiring. The score funtion f from R to $\mathbb{R}$ is set $f \equiv g$. When the partition function is to be calculated, the semiring is the real number semiring, and

$$f(r) = e^{-\frac{g(r)}{RT}}$$

Algorithms for computing the minimum free energy structure and the partition function were well known in the scientific literature, see for example [62]. Miklós, Meyer and Nagy presented a method to compute arbitrary moments of the Boltzmann distribution. The expected value in the Boltzmann distribution by definition is

$$E_B[G] := \sum_{S} \frac{e^{-\frac{G(S)}{RT}} G(S)}{Z},$$

and the variance is

$$V_B[G] := \sum_S \frac{e^{-\frac{G(S)}{RT}}(G(S) - E_B[G])^2}{Z},$$

where the summation is over all the possible structures S that the RNA sequence can take. By introducing the variables

$$X := \sum_S e^{-\frac{G(S)}{RT}} G(S)$$

and

$$Y := \sum_S e^{-\frac{G(S)}{RT}} G(S)^2,$$

we get that

$$E_B[G] = \frac{X}{Y} \tag{2.16}$$

and

$$V_B[G] = E_B[G^2] - (E_B[G])^2 = \frac{Y}{Z} - \frac{X^2}{Z^2} \tag{2.17}$$

Our aim is to give dynamic programming algorithms that compute X and Y. For this, we introduce the following algebraic structure and prove that it is a (semi)ring.

Theorem 50. *Let* $R = \underbrace{\mathbb{R} \times \mathbb{R} \times \ldots \times \mathbb{R}}_{d} = \mathbb{R}^d$. *Let the operation* $\oplus$ *defined as the coordiante-wise addition, that is*

$$(x_0, x_1, \ldots, x_{d-1}) \oplus (y_0, y_1, \ldots, y_{d-1}) := (x_0 + y_0, x_1 + y_1, \ldots, x_{d-1} + y_{d-1}).$$

Let the operation $\otimes$ *be defined such that if the product of* $x = (x_0, x_1, \ldots, x_{d-1})$ *and* $y = (y_0, y_1, \ldots, y_{d-1})$ *is* $z = (z_0, z_1, \ldots, z_{d-1})$, *then for all* $k = 0, 1, \ldots, d-1$,

$$z_k := \sum_{i=0}^{k} \binom{k}{i} x_i y_{k-i}.$$

Then $(R, \oplus, \otimes)$ *is a commutatative ring with additive unite* $(0, 0, \ldots, 0)$ *and multiplicative unit* $(1, 0, 0, \ldots, 0)$.

Proof. The coordinatewise addition is clearly a commutative group with $(0, 0, \ldots, 0)$ being the additive unit. The multiplication is clearly a commutative operation due to the commutativity of the multiplication of real numbers and the symmetry of the binomial coefficients, that is for all $i \leq k$, $\binom{k}{i} = \binom{k}{k-i}$. It is also clear that $(1, 0, 0, \ldots, 0)$ is a multiplicative unit.

The associativity is not obvious, but can be shown by straightforward calculation. Consider the product

$$(x \otimes y) \otimes z := w.$$

Then on the k^{th} coordinate we have that

$$w_k = \sum_{j=0}^{k} \binom{k}{j} \left(\sum_{i=0}^{j} \binom{j}{i} x_i y_{j-i} \right) z_{k-j} = \sum_{0 \leq i \leq j \leq k} \binom{k}{k-j, j-i, i} x_i y_{j-i} z_{k-j} =$$
$$\sum_{i', j' | 0 \leq i' + j' \leq k} \binom{k}{i', j', k-i'-j'} x_{i'} y_{j'} z_{k-i'-j'},$$

with $i' = i$ and $j' = j - i$ in the last sum. This expression is symmetric for x, y and z, therefore, we get that

$$(x \otimes y) \otimes z = (y \otimes z) \otimes x = x \otimes (y \otimes z),$$

where the first equality is due to the symmetry, and the second equality is due to the commutativity of the multiplication.

Proving the distributivity is also a straightforward calculation. In the product

$$w := x \otimes (y \oplus z)$$

on the k^{th} coordinate we have

$$w_k = \sum_{i=0}^{k} x_i (y_{k-i} + z_{k-i}) = \sum_{i=0}^{k} x_i y_{k-i} + \sum_{i=0}^{k} x_i z_{k-i},$$

which is indeed the k^{th} coordinate in

$$w := (x \otimes y) \oplus (x \otimes z).$$

□

We use $(R, \oplus, \otimes)$ with $d = 3$ to compute X and Y. With $d > 3$, we can compute further moments of the Boltzmann distribution. For all rewriting rules $r \in R$, we define

$$f(r) := \left(e^{-\frac{g(r)}{RT}}, g(r)e^{-\frac{g(r)}{RT}}, \ldots, g^{n-1}(r)e^{-\frac{g(r)}{RT}}\right). \tag{2.18}$$

Theorem 51. *With the f function defined in Equation 2.18, for any $C \in R^*$, we have that*

$$\bigotimes_{r \in C} f(r) = \left(e^{-\frac{\sum_{r \in C} g(r)}{RT}}, \sum_{r \in C} g(r)e^{-\frac{\sum_{r \in C} g(r)}{RT}}, \ldots, \left(\sum_{r \in C} g(r)\right)^{n-1} e^{-\frac{\sum_{r \in C} g(r)}{RT}}\right)$$

Proof. By induction. The statement is clearly holds for $|C| = 1$. Assume that it holds for $|C| = m$, we would like to prove it for $|C| = m + 1$. Let $|C| = m + 1$, then $C = C' \circ r$ with $|C'| = m$. Then

$$\bigotimes_{r \in C} f(r) = \left(\bigotimes_{r' \in C'} f(r')\right) \otimes f(r).$$

On the k^{th} coordinate, we have

$$\sum_{i=0}^{k} \binom{k}{i} \left(\sum_{r' \in C'} g(r')\right)^{i} e^{-\frac{\sum_{r' \in C'} g(r')}{RT}} g(r)^{k-i} e^{-\frac{g(r)}{RT}} =$$

$$e^{-\frac{g(r)+\sum_{r' \in C'} g(r')}{RT}} \sum_{i=0}^{k} \binom{k}{i} \left(\sum_{r' \in C'} g(r')\right)^{i} g(r)^{k-i} =$$

$$e^{-\frac{g(r)+\sum_{r' \in C'} g(r')}{RT}} \left(\sum_{r' \in C'} g(r') + g(r)\right)^{k} =$$

$$e^{-\frac{\sum_{r \in C} g(r)}{RT}} \left(\sum_{r \in C} g(r)\right)^{k}.$$

□

Thus, according to Theorem 49, we can plug-in the (semi)ring presented in Theorem 50 into the Equations 2.14-2.15, and then $s(1, n, S)$ computes a vector in which the k^{th} coordinate is

$$\sum_{S} e^{-\frac{G(S)}{RT}} G(S)^k.$$

From these values, the partition function, the expected value and the variance can be computed as indicated by the Equations 2.16 and 2.17.

2.3 Linear memory Baum-Welch training

The *Baum-Welch training* is an *Expectation-Maximization algorithm* for stochastic grammars and Hidden Markov Models. Here we introduce it for Hidden Markov Models. Let $(\vec{G}, START, END, \Gamma, T, e)$ be a Hidden Markov Model, and let $A \in \Gamma^*$, $|A| = n$. Then the Forward values are defined as

$$f(0, START) = 1, \quad f(0, v) = 0 \ \forall v \in \vec{G}(V) \setminus \{START\},$$

$$f(i, v) = \sum_{u \in \vec{G}(V)} f(i-1, u) \times T(u, v) \times e(a_i, v).$$

The Backward values are defined as

$$b(n, END) = 1, \quad b(n, v) = T(v, END), \ \forall v \in \vec{G}(V) \setminus \{END\},$$

$$b(i, u) = \sum_{v \in \vec{G}(V)} b(i+1, v) \times T(u, v) e(a_{i+1}, v).$$

The meaning of the Forward values is the $f(i, v)$ is the sum of emission path probabilities emitting prefix A_i such that the model is in state v, and v already emitted character a_i. The meaning of the Backward values is that $f(i, u)$ denote the sum of emission path probabilities emitting suffix A^i, such that the model starts in u and u already emitted character a_i.

The Baum-Welch training updates transition and emission probabilities. The estimations for the transition and emission probabilities are

$$\hat{T}(u, v) = \frac{\sum_{i=1}^{n-1} f(i, u) T(u, v) e(a_{i+1}, v) b(i+1, v)}{\sum_{i=1}^{n} f(i, u) b(i, u)} \tag{2.19}$$

and

$$\hat{e}(a, v) = \frac{\sum_{i|a_i=a} f(i, v) b(i, v)}{\sum_{i=1}^{n} f(i, v) b(i, v)}. \tag{2.20}$$

It can be proved that the probability that the Hidden Markov Model emits sequence A using probabilities $\hat{T}$ and $\hat{e}$ is always larger or equal than

the probability that the Hidden Markov Model emits sequence A using probabilities T and e [27].

The Baum-Welch training in the form of Equations 2.19 and 2.20 requires $f(i,u)$ and $b(i,u)$ in the same time. It needs a large amount of memory especially in gene prediction where A is a large genomic sequence: It needs $O(n|\vec{G}(V)|)$ memory, and $O(n|\vec{G}(V)|(\bar{T}+|\Gamma|))$ time where $\bar{T}$ is the average number of outgoing edges in the Hidden Markov Model.

Tarnas and Hughey [89] and Wheeler and Hughey [96] introduced a reduced space Baum-Welch training using a divide & conquer technique. Their method needs $O(\log(n)|\vec{G}(V)|)$ memory and $O(n\log(n)|\vec{G}(V)|\bar{T}+n|\vec{G}(V)||\Gamma|))$ time. Here we introduce a method that needs only $|\vec{G}(V)|$ memory and $O(n|\vec{G}(V)|\bar{T}(|T|+|\vec{G}(V)||\Gamma|))$ time. The method can be easily extended to pair-HMMs, where states emit characters into two sequences, and the observer can see only the emitted sequences and not their co-emission pattern. In that case, the original Baum-Welch training requires memory proportional to the product of the two sequence lengths. Our method can reduce it to linear memory, that's why its name [67, 27].

Is is easy to see that for the denominator of the fractions in in Equations 2.19 and 2.20 it holds that

$$\sum_{i=1}^{n} f(i,u)b(i,u) = \sum_{p\sim A} P(p)n(p,u), \tag{2.21}$$

where $p \sim A$ denotes that path p emits A, $P(p)$ is the probability of the path (the product of the transition and emission probabilities in it with multiplicity) and $n(p,i)$ is the number of times u appears in p. Similarly, we have that

$$\sum_{i=1}^{n-1} f(i,u)T(u,v)e(a_{i+1},v)b(i+1,v) = \sum_{p\sim A} P(p)n(p,u\to v), \tag{2.22}$$

where $n(p,u\to v)$ is the number of transitions from u to v in path p, and

$$\sum_{i|a_i=a} f(i,u)b(i,u) = \sum_{p\sim A} P(p)n(p,u\sim a), \tag{2.23}$$

where $n(p,u\sim a)$ is the number of times u emits a in path p.

Using algebraic dynamic programming, we might move the computations to the polynomial ring $\mathbb{R}[x]$, and assign to each path a monomial $P(p)x^{n(p,u)}$

(and similarly, $P(p)x^{n(p,u\to v)}$ and $P(p)x^{n(p,u\sim a)}$. Then using the universal yield algebra $|T|+|\vec{G}(V)||\Gamma|$ times, we could compute polynomials for which we are looking for the substitution $x=1$ in their first derivative. In this way, we arrive to an algorithm that runs in $|\vec{G}(V)|\times M$ memory where M is the memory requirement to store a polynomial needed in the computations, since in the recursion for all i, we need the values with indices $i-1$.

However, this approach would be be neither time efficient nor memory efficient as the operations with polynomials cannot be performed in constant time and the polynomials cannot be stored in constant memory. However, we do not really need the polynomial only its substitution $x=1$ in their first derivative. It turns out that we can perform the computation in a homomorph image, where each step takes only constant time and needs constant memory[1]. We state it in the following lemma.

Lemma 52. *Let $R_1=(\mathbb{R}[x],+,\times)$ be the (semi)ring of real polynomials. Let $R_2=(\mathbb{R}[x]\times\mathbb{R}[x],+,\times)$ be an algebra with addition*

$$(p_1,q_1)+(p_2,q_2):=(p_1+p_2,q_1+q_2)$$

and multiplication

$$(p_1,q_1)\times(p_2,q_2):=(p_1\times q_2+p_2\times q_1,q_1\times q_2).$$

Let $R_3=(\mathbb{R}\times\mathbb{R},+,\times)$ be an algebra with addition

$$(x_1,y_1)+(x_2,y_2):=(x_1+x_2,y_1+y_2)$$

and multiplication

$$(x_1,y_1)\times(x_2,y_2):=(x_1\times y_2+x_2\times y_1,q_1\times q_2).$$

Then both R_2 and R_3 are (semi)rings, and

$$\varphi_1:R_1\to R_2,\varphi_1(p)=(p',p)$$

and

$$\varphi_2:R_2\to R_3,\varphi_2((p,q))=(p[1],q[1])$$

are both semi(ring)-homomorphisms, where p' is the first derivative of p.

[1]Actually, there is a hidden $poly(\log(|A|))$ factor both in the memory and the running time due to the necessary number of digits of the numbers appearing in the recursions, however, in practice, these factors can be omitted

Proof. Proving that R_2 and R_3 are rings can be done in the same way. Here we show it for R_2.

It is trivial to check the commutativity and the associativity of the addition in R_2. The associativity of the multiplication in R_2 is a bit tedious but straightforward:

$$[(p_1, q_1) \times (p_2, q_2)] \times (p_3, q_3) =$$

$$= [[(p_1 \times q_2 + p_2 \times q_1, q_1 \times q_2)] \times (p_3, q_3) =$$

$$(p_1 \times q_2 \times q_3 + p_2 \times q_1 \times q_3 + p_3 \times q_1 \times q_2, q_1 \times q_2 \times q_3).$$

Since it is symmetric for the three factors and it is easy to check that the multiplication is commutative, we get that for any $r_1, r_2, r_3 \in R_2$,

$$(r_1 \times r_2) \times r_3 = (r_3 \times r_2) \times r_1 = r_1 \times (r_2 \times r_3).$$

The distributive rule is also straightforward to verify:

$$(p_1 + q_1) \times [(p_2, q_2) + (p_3, q_3)] =$$

$$= (p_1, q_1) \times (p_2 + p_3, q_2 + q_3) = (p_1 \times (q_2 + q_3) + q_1 \times (p_2 + p_3), q_1 \times (q_2 + q_3)) =$$

$$(p_1 \times q_2 + q_1 \times p_2 + p_1 \times q_3 + q_1 \times p_3 =$$

$$[(p_1 + q_1) \times (p_2, q_2)] + [(p_1 + q_1) \times (p_3, q_3)] .$$

The mapping φ_1 is a homomorphism since it holds for the derivation that

$$(p + q)' = p' + q'$$

and

$$(p \times q)' = p' \times q + p \times q'.$$

The mapping φ_2 is a homomorphism since the addition and multiplication in R_2 and R_3 defined in the same way, and substituting a value into a polynomial in $\mathbb{R}[x]$ is a homomorphism from $\mathbb{R}[x]$ to $\mathbb{R}$. □

Therefore, we get that we can compute the right hand sides of Equations 2.21-2.23 in the homomorph image R_3. In R_3 both an addition and a multiplication can be done in constant time. That is, we can use the recursions for a specific state w given by the following Theorem.

Theorem 53. *Let w be a state in a Hidden Markov Model. Then $\sum_{p\sim A} P(p)n(p,w)$ can be computed by the following recursion.*

$$f(0,START) = 1, \quad f(0,v) = 0 \ \forall v \in \vec{G}(V) \setminus \{START\},$$

$$f'_w(0,v) = 0 \ \forall v \in \vec{G}(V),$$

$$f(i,v) = \sum_{u\in\vec{G}(V)} f(i-1,u) \times T(u,v) \times e(a_i,v),$$

$$f'_w(i,v) = \sum_{u\in\vec{G}(V)} f'_w(i-1,u) \times T(u,v) \times e(a_i,v), \quad \forall v \neq w,$$

$$f'_w(i,w) = f(i,w) + \sum_{u\in\vec{G}(V)} f'_w(i-1,u) \times T(u,w) \times e(a_i,w).$$

Then

$$\sum_{p\sim A} P(p)n(p,w) = \sum_{u\in\vec{G}(V)} f'_w(n,u) \times T(u,END).$$

Proof. It is the straight consequence of Lemma 52, and the fact the we are looking for the evaluation of the derivative of the appropriate generating function at $x = 1$. □

Similar recursions hold for $\sum_{p\sim A} P(p)n(p,u \to v)$ and $\sum_{p\sim A} P(p)n(p,u \sim a)$.

Chapter 3

Two easy counting problems under the SCJ model

In this chapter, we show that two of the five counting problems with genome rearrangements models presented in the Preliminaries are in FP if the rearrangement model is the SCJ model. Computing the number of shortest SCJ rearrangement scenarios was a joint work with Eric Tannier and Sándor Kiss. Eric Tannier suggested the research based on the observation that matching also play an important role in case of SCJ rearrangement scenarios as they also play a central role in DCJ rearrangement scenarios (see also Chapter 6). The author of this thesis developed a dynamic programming algorithm and also observed that the number of SCJ scenarios sorting a component of the adjacency graph are related to alternating (or zig-zag) permutations. The original work has been published in Theoretical Computer Science, 552:83–98, DOI: 10.1016/j.tcs.2014.07.027.

Comnputing the number of most parsimonious medians under the SCJ model was a joint work with Heather Smith. The original work was publised in BMC Bioinformatics, 16(Suppl 14): S6., `https://doi.org/10.1186/1471-2105-16-S14-S6`.

3.1 Pairwise rearrangement problem under the SCJ model

3.1.1 A dynamic programming solution

According to Equation 1.30, it is easy to see that any most parsimonious scenario transforming G_1 into G_2 has to cut all the adjacencies in $G_1 \setminus G_2$ and add all the adjacencies in $G_2 \setminus G_1$, and there are no more SCJ operations. Drawing one solution is easy: first cut all adjacencies in $G_1 \setminus G_2$, then join all adjacencies in $G_2 \setminus G_1$. But if we want to explore the solution space, we have to observe that if an adjacency (a, b) exists in $G_1 \setminus G_2$ and an adjacency (a, c) exists in $G_2 \setminus G_1$, then first adjacency (a, b) must be cut to create telomere (a), and then telomere (a) can be connected to telomere (c). Similarly, if extremity c belongs to an adjacency in $G_1 \setminus G_2$, then it must be also cut before connecting the two telomeres. Therefore there are restrictions on the order of cuts and joins. If two adjacencies share an extremity, we call them *conflicting adjacencies*.

The allowed order of cuts and joins can be read from the adjacency graph (see the Preliminaries), introduced by [13] to compute the DCJ distance between two genomes. It can be used to study SCJ scenarios as well.

When an SCJ operation acts on G_1 and thus creates G_1', it also acts on the adjacency graph of G_1 and G_2 by transforming it into the adjacency graph of G_1' and G_2. Therefore the transformation of G_1 into G_2 can be seen as a transformation of the adjacency graph into trivial components. Recall that trivial components of an adjacency graph are the cycles of length 2 and paths with a single edge. We say that such an SCJ scenario *sorts* the adjacency graph. As any SCJ operation in a most parsimonious scenario acts on a single component, we say that the subsequence of SCJ operations acting on that component *sorts* it if it transforms it into trivial components.

We first give the way of computing the number of scenarios for sorting one component. Then the number of scenarios for several components will be deduced by a combination of scenarios from each component.

Let $W(i)$ (respectively $M(i)$, $O(i)$ and $C(i)$) denote the number of most parsimonious SCJ scenarios sorting a W-shaped path (respectively M-shaped path, odd path, cycle) with i adjacencies in G_1. The following dynamic programming algorithm allows to compute all these numbers.

For a trivial component, no SCJ operation is needed so there is only one

solution: the empty sequence. This gives

$$C(1) = 1 \tag{3.1}$$
$$O(0) = 1 \tag{3.2}$$

The smallest W-shaped path has 0 adjacency in G_1 and one in G_2. There is a unique solution sorting it: add the adjacency. This gives

$$W(0) = 1 \tag{3.3}$$

A scenario of any other component starts with cutting an adjacency in G_1. For a W-shaped path, this results in two W-shaped paths. For an M-shaped path, this results in two odd paths. For an odd path, this results in an odd path and a W-shaped path. For a cycle, this results in a W-shaped path. Each emerging component has fewer adjacencies in G_1, and hence, a dynamic programming recursion can be applied: the resulting components must be sorted and in case of two resulting components, the sorting steps on the components must be merged. Hence the dynamic programming recursions are

$$C(i) = i \times W(i-1) \tag{3.4}$$
$$W(i) = \sum_{j=1}^{i} \binom{2i}{2j-1} W(j-1)W(i-j) \tag{3.5}$$
$$M(i) = \sum_{j=1}^{i} \binom{2i-2}{2j-2} O(j-1)O(i-j) \tag{3.6}$$
$$O(i) = \sum_{j=1}^{i} \binom{2i-1}{2j-2} O(j-1)W(i-j) \tag{3.7}$$

These dynamic programming recursions can be used for counting and sampling by the classical Forward-Backward phases: in the Forward phase the number of solutions is calculated, and in the Backward phase one random solution is chosen based on the numbers in the sums.

So it is possible to compute $W(i)$, $M(i)$, $O(i)$ and $C(i)$ in polynomial time and to sample one scenario from the uniform distribution. We can then count and sample for several components by introducing a multinomial coefficient. We arrived to the following theorem.

Theorem 54. *Let G_1 and G_2 be two genomes with adjacency graph AG. Assume AG contains i M-shaped paths, with respectively $m_1, m_2, \ldots, m_i$ adjacencies in G_1; AG contains j W-shaped paths, with respectively $w_1, w_2, \ldots, w_j$ adjacencies in G_1; AG contains k odd paths, with respectively $v_1, v_2, \ldots, v_k$ adjacencies in G_1; and AG contains l cycles, with respectively $c_1, c_2, \ldots, c_l$ adjacencies in G_1. The number of most parsimonious* SCJ *scenarios from G_1 to G_2 is*

$$\frac{\left(\sum_{n=1}^{i}(2m_n-1)+\sum_{n=1}^{j}(2w_n+1)+\sum_{n=1}^{k}(2v_n)+\sum_{n=1}^{l}(2c_n)\right)!}{\prod_{n=1}^{i}(2m_n-1)!\prod_{n=1}^{j}(2w_n+1)!\prod_{n=1}^{k}(2v_n)!\prod_{n=1}^{l}(2c_n)!} \times \times \prod_{n=1}^{i} M(n) \prod_{n=1}^{j} W(n) \prod_{n=1}^{k} O(n) \prod_{n=1}^{l} C(n) \quad (3.8)$$

Sampling a scenario from the uniform distribution is then achieved by generating a random permutation with different colors and indices, one color for each component, and then wipe down the indices in order to get a permutation with repeats. For each component, its sorting steps must be put into the joint scenario indicated by the color of the component.

We can then state the following theorem settling the complexity of the comparison of two genomes by SCJ. We will denote by MPSCJ the optimization problem to find the minimum number of SCJ operations transforming one genome into another. Here MP stands for most parsimonious. The counting counterpart problem #MPSCJ is to count the most parsimonious scenarios between two genomes.

Theorem 55. #MPSCJ *is in* FP *and there is a polynomial algorithm sampling from the exact uniform distribution of the solution space of an* MPSCJ *problem.*

3.1.2 Alternating permutations

The solutions to #MPSCJ for single components are linked to the number of alternating permutations, for which finding a formula is an old open problem. An *alternating permutation* of size n is a permutation $c_1, \ldots, c_n$ of $\{1, \ldots, n\}$ such that $c_{2i-1} < c_{2i}$ and $c_{2i} > c_{2i+1}$ for all i [3]. For example, if $n = 4$, the permutation $1, 3, 2, 4$ is an alternating permutation but $1, 3, 4, 2$ is not because 3 is less than 4. The number of alternating permutations of size n is denoted by A_n and finding these numbers is known as André's problem.

We show that computing SCJ scenarios is closely related:

Theorem 56.

$$M(k) = A_{2k-1}$$

$$W(k) = A_{2k+1}$$

$$O(k) = A_{2k}$$

$$C(k) = k \times A_{2k-1}$$

Proof. We prove only the first line, the second and the third lines can be proved the same way. The proof of the last line comes from the fact that a cycle with k adjacencies can be opened in k different ways into a W-shaped component with $k-1$ adjacencies. Let the adjacencies in the G_1 part of the M-shaped component be (x_1, x_2), (x_3, x_4), $\ldots, (x_{2k-1}, x_{2k})$. Any SCJ scenario sorting these must cut all these adjacencies and must create adjacencies (x_2, x_3), (x_4, x_5), $\ldots, (x_{2k-2}, x_{2k-1})$. Let us index the SCJ operations in a scenario, and let π_{2i-1} be the index of the SCJ step which cuts the adjacency (x_{2i-1}, x_{2i}), and let π_{2i} be the index of the SCJ step which joins x_{2i} and x_{2i+1}.

In any most parsimonious SCJ sorting the M-shaped component, $\pi_{2i-1} < \pi_{2i}$ and $\pi_{2i+1} < \pi_{2i}$, so π is an alternating permutation. Hence the number of sorting scenarios is at most A_{2k-1}.

On the other hand, for any alternating permutation of size $2k-1$, we can construct a sorting scenario in which the indexes come from the alternating permutation. Since the sorting scenarios for different alternating permutations are different, the number of SCJ scenarios is at least A_{2k-1}. □

3.2 Most parsimonious medians

Feijão and Meidanis [32] proved that there is a unique optimal SCJ median for 3 genomes. Their proof trivially extends to show that the optimal median remains unique for arbitrary odd number of genomes: the optimal median contains the set of adjacencies that can be found in the majority of the genomes. Indeed, the SCJ distance between two genomes G_1 and G_2 is simply $|\Pi_1 \Delta \Pi_2|$, where Π_i is the set of adjacencies in G_i. The key observation is that it is impossible that two conflicting adjacencies are presented in more than half of the genomes, therefore the genome that contains exactly the

adjacencies that are presented in the majority of the genomes is a valid genome.

When the number of genomes is even, each extremity might be in at most two adjacencies that is presented in exactly half of the genomes. It is easy to see that the optimal median contains the set of adjacencies that are present in more than half of the genomes and any conflict-free subset of the adjacencies that are present in exactly half of the genomes. The number of optimal medians can be counted in the following way.

Given a set of genomes $\mathcal{G} = \{G_1, G_2, \dots G_{2k}\}$ having the same synteny blocks, we define the conflict graph $C(V, E)$ in the following way: The vertices V are the set of extremities presented in $\mathcal{G}$ and there is an edge between v_1 and v_2 if and only if the adjacency (v_1, v_2) is presented in exactly half of the genomes.

Observation 57. *The maximum degree of the vertices in C is 2.*

Proof. This follows from the fact that any extremity can be in at most two adjacencies which are present in exactly half of the genomes. □

The consequence of Observation 57 is that C can be decomposed into isolated vertices, paths and cycles. Any conflict-free subset of the adjacencies is a matching (non-necessary maximum matching and possibly empty) of C. The number of matchings is the product of the number of matchings on each component. Therefore it suffices to calculate this number. It is well-known [61] that the number of matchings in a path of length n is

$$\sum_{k=1}^{\lfloor \frac{n}{2} \rfloor} \binom{n-k}{k}$$

and the number of matchings in a cycle of length n is

$$\sum_{k=1}^{\lfloor \frac{n}{2} \rfloor} \frac{n}{n-k} \binom{n-k}{k}.$$

Since obtaining the conflict graph, decomposing it into paths and cycles, estimating the number of matchings on each component and multiplying these numbers all can be done in polynomial time, we can announce the following theorem:

Theorem 58. *The number of optimal medians under the SCJ model is in FP.*

Although calculating the number of optimal medians is easy, Miklós and Smith proved that the number of most parsimonious median scenarios is in #P-complete (see Chapter 8). The proof uses a technique (modulo prime number calculations) that is typically used in those #P-complete problems that admit an FPRAS approximation. On the other hand, Miklós and Smith also proved that a simple Markov chain that walks on the optimal median genomes by adding or removing a random adjacency and converges to the distribution proportional to the number of scenarios that the median genome has by applying the Metropolis-Hastings algorithm [66, 44] is torpidly mixing even if the number of genomes are fixed to 4, and only the size of the genomes are allowed to grow (unpublished result). Therefore it is absolutely unclear whether the number of most parsimonious median scenarios under the SCJ model has an FPRAS approximation or an FPRAS approximation would imply RP = NP. If the problem is in FPRAS, deeper understanding of the solution space is necessary that is to be incorporated into a sophisticated Markov chain method.

Part II

Counting problems in FPRAS via rapidly mixing Markov chains

dc_2046_22

Chapter 4

Sampling realizations of bipartite degree sequences

In this chapter, we consider the following class of Markov chains on the realizations of bipartite degree sequences. Let D_b be a bipartite degree sequence, then the M_{D_b} Markov chain on the realizations of D_b is defined in the following way. Let $G = (U, V, E)$ be the current realization of D_b. With probability $\frac{1}{2}$ do nothing (that is, the Markov chain is Lazy). With probability $\frac{1}{2}$, select $\{u_1, u_2\} \in \binom{U}{2}$ uniformly and $\{v_1, v_2\} \in \binom{V}{2}$ uniformly. If both $(u_1, v_1) \in E$ and $(u_2, v_2) \in E$ (or alternatively, both $(u_1, v_2) \in E$ and $(u_2, v_1) \in E$) and both $(u_1, v_2) \notin E$ and $(u_2, v_1) \notin E$ (alternatively, both $(u_1, v_1) \notin E$ and $(u_2, v_2) \notin E$), then delete the existing edges and add the non-existing edges. Otherwise, do nothing.

It is elementary to prove that the Markov chain M_{D_b} converges to the uniform distribution of the realizations of D_b. Indeed, according to the Corollary 38, M_{D_b} is irreducible. It is also aperiodic since it is Lazy. For any two realizations of D_b, G_1 and G_2, and for the transition probabilities of M_{D_b} it holds that

$$P(G_2|G_1) = P(G_1|G_2). \tag{4.1}$$

Indeed, $P(G_2|G_1) \neq 0$ if and only if G_2 can be obtained from G_1 with a switch operation. The same holds for $P(G_2|G_1)$. The probability of any switch operation has probability

$$\frac{1}{2\binom{|U|}{2}\binom{|V|}{2}},$$

thus the Equation 4.1 holds. Since M_{D_b} is irreducible, aperiodic and reversible w.r.t. the uniform distribution (due to Equation 4.1), it converges to the uniform distribution from any starting point, according to Theorem 22.

It is conjectured that this class of Markov chain is rapidly mixing. In a series of publications, we proved rapid mixing for several classes of degree sequences. Here we prove the followings.

1. If a degree sequence can be decomposed into the so-called Tyshkevich product of degree sequences (see Section 4.1), then it is enough to prove rapid mixing on each factor. The original work has been published in Erdős, L.P., Miklós, I., Toroczkai, Z. (2018) Combinatorics, Probability and Computing, 27(2):186-207, `https://doi.org/10.1017/S0963548317000499`. The author of this thesis proved Theorem 62.

2. The above-defined switch Markov chain is rapidly mixing on realizations of P-stable degree sequences (see Section 4.2). This is a large class of degree sequences, it contains for example, all degree sequences on $n+n$ vertices in which all degrees are between $\frac{1}{4}n$ and $\frac{3}{4}n$. The original work has been published in Erdős, E.L. Mezei, T., Miklós, I., Soltész, D. (2018) PLoS ONE, 13(8): e0201995, `https://doi.org/10.1371/journal.pone.0201995` and Erdős, E.L., Greenhill, C., Mezei, T. R., Miklós, I., Soltész, D., Soukup, L. (2022) Eur. J. Comb., 99:103421, `https://doi.org/10.1016/j.ejc.2021.103421`. The author of the thesis proved Theorems 64, 65 and 66 and also Theorem 67. In fact, first rapid mixing of the switch Markov chain on the realizations of degree sequences satisfying Equations 4.8 and 4.9 presented in Theorem 65 was proved, and then it was observed that the generalization to Theorem 64 is straightforward.

3. We also proved rapid mixing of the Markov chain for a class of degree sequences that are not P-stable. The original work has been published in Erdős, E.L., Győri, E., Mezei, T. R., Miklós, I., Soltész, D. (2021) Electronic Journal of Combinatorics, 28:3 #P3.7, DOI: 10.37236/9652. The two young postdocs of the paper generalized the results presented in this thesis to any class of degree sequences $\mathcal{H}_k$ (see Definition 68), here we show only the elementary proof for $\mathcal{H}_1$.

Large part of the work presented in this chapter was done using the financial support of NKFIH under the contract number KH126853. This grant

allowed hiring two postdocs, Tamás R. Mezei and Dániel Soltész who worked under the joint supervision of the author of this thesis and Péter L. Erdős. Their contribution is clearly indicated throughout this chapter. They also worked further on this topic leading to several theorems, especially generalizations of the presented results to simple and directed degree sequences. The results of their work is briefly summarised at the end of the chapter.

4.1 Tyshkevich-decompositions

Definition 59. *Let* $D_b = (d_{1,1}, d_{1,2}, \ldots, d_{1,n}), (d_{2,1}, d_{2,2}, \ldots, d_{2,m})$ *and* $D'_b = (d'_{1,1}, d'_{1,2}, \ldots, d'_{1,n'}), (d'_{2,1}, d'_{2,2}, \ldots, d'_{2,m'})$ *be two bipartite degree sequences. The* Tyshkevich product *of* D_b *and* D'_b *is*

$$\begin{aligned} D_b \circ D'_b \;:=\; & (d_{1,1}, d_{1,2}, \ldots, d_{1,n}, d'_{1,1}+m, d'_{1,2}+m, \ldots, d'_{1,n'}+m), \\ & (d_{2,1}+n', d_{2,2}+n', \ldots, d_{2,m}+n', d'_{2,1}, d'_{2,2}, \ldots, d'_{2,m'}) \end{aligned} \quad (4.2)$$

It is easy to see that the Tyschevich product is associative, but not commutative, thus it forms a non-commutative monoid with the empty bipartite degree sequence being the unit.

Observation 60. *If both* D_b *and* D'_b *are graphic then* $D_b \circ D'_b$ *is also graphic. Furthermore, if both* D_b *and* D'_b *are graphic, then every realization* $\tilde{G} = (U \cup U', V \cup V', \tilde{E})$ *of* $D_b \circ D'_b$ *is such that the induced subgraphs on* (U, V) *and* (U', V') *are realizations of* D_b *and* D'_b*, respectively, the induced subgraph on* (U', V) *is the complete bipartite graph* $K_{n',m}$ *and the induced subgraph on* (U, V') *is the empty graph.*

Proof. First we prove that the graphicality of D_b and D'_b implies the graphicality of $D_b \circ D'_b$. Let $G = (U, V, E)$ be a realization of D_b and let $G' = (U', V', E')$ be a realization of D'_b. Then let $\tilde{G} = (U \cup U', V \cup V', \tilde{E})$ is a graph whose edge set is $\tilde{E} = E \cup E' \cup \{(u, v') | u \in U \ \wedge \ v' \in V'\}$. Clearly, this is a realization of $D_b \circ D'_b$.

Now consider any switch operation on $\tilde{G}$. Since the induced subgraph on (U', V) is the complete subgraph, on the other hand, the induced subgraph on (U, V') is the empty graph, it follows that any switch operation operates either on the subgraph induced by (U, V) or on the subgraph induced by (U', V'). What follows is that the graph obtained by a switch operation still contains the complete subgraph on (U', V) and the empty subgraph on

(U, V'). Since any realization is obtainable from any realization of the same degree sequence by a series of switch operations, all realizations of $D_b \circ D'_b$ contain the induced subgraphs as stated in the theorem. □

A corollary of this observation is the following.

Corollary 61. *Let $D_b = D_{b,1} \circ D_{b,2} \circ \ldots \circ D_{b,k}$. Furthermore for all $i = 1, 2, \ldots, k$, let N_k be the number of realizations of $D_{b,i}$. Then the number of realizations of D_b is*

$$\prod_{i=1}^{k} N_k.$$

Furthermore, there is a bijection between the realizations of D_b and the direct product of the set of realizations of the degree sequences $D_{b,i}$. The bijection is given by the Thyskevich product of the factors in the direct product.

When a state space is a direct product of smaller spaces, and the Markov chain changes only one coordinate in one step, then the Markov chain is rapidly mixing if the number of coordinates is not large and the Markov chains restricted on each coordinate are rapidly mixing. Below we state this theorem precisely and prove it.

Theorem 62. *Let $\mathcal{M}$ be a class of Markov chains whose state space is a k-dimensional direct product of spaces, and the problem size of a particular chain is denoted by n.*

Any transition of the Markov chain $M \in \mathcal{M}$ changes only one coordinate. Coordinate i is chosen with probability p_i, where we assume that $\frac{1}{p_i} = O(poly_1(n))$ (and thus, we indirectly assume a polynomial upper bound on k, too). Furthermore, the transition probabilities do not depend on the other coordinates. The transitions on each coordinate form irreducible, aperiodic Markov chains (denoted by $M_1, M_2, \ldots M_k$), which are reversible with respect to a distribution π_i. Furthermore, each of $M_1, \ldots M_k$ are rapidly mixing, i.e., with the relaxation time $\frac{1}{1-\lambda_{2,i}}$ is bounded by a $O(\mathrm{poly}_2(n))$ for all i. Then the Markov chain M converges rapidly to the direct product of the π_i distributions, and the second-largest eigenvalue of M is

$$\lambda_{2,M} = \max_i \{1 - p_i + p_i \lambda_{2,i}\}.$$

and thus the relaxation time of M is also polynomially bounded:

$$\frac{1}{1 - \lambda_{2,M}} = O(\mathrm{poly}_1(n)\mathrm{poly}_2(n)).$$

Proof. The transition matrix of M can be described as

$$\sum_{i=1}^{k} \left[\bigotimes_{j=1}^{i-1} \mathbf{I}_j \right] \otimes p_i \mathbf{M}_i \otimes \bigotimes_{j=i+1}^{k} \mathbf{I}_j$$

where $\otimes$ denotes the usual tensor product from linear algebra, $\mathbf{M}_i$ denotes the transition matrix of the Markov chain on the ith coordinate, and $\mathbf{I}_j$ denotes the identical matrix with the same size as $\mathbf{M}_j$. Since all pairs of terms in the sum above commute, the eigenvalues of M are

$$\left\{ \sum_{i=1}^{k} p_i \lambda_{j_i,i} : 1 \leq j_i \leq |\Omega_i| \right\}$$

where Ω_i is the state space of the Markov chain M_i on the ith coordinate. The second-largest eigenvalue of M is then obtained by combining a second-largest eigenvalue with the other largest eigenvalues, i.e., with all others being 1s:

$$\sum_{j \neq i} p_j + p_i \lambda_{2,i} = 1 - p_i + p_i \lambda_{2,i} \ .$$

If g denotes the smallest spectral gap, ie., $g = 1 - \max_i \{\lambda_{2,i}\}$, then from the above, the second-largest eigenvalue of M is

$$1 - p_i(1 - \lambda_{2,i}) \leq 1 - p_i g$$

namely, the second-largest eigenvalue of M is at most $O(poly_2(n))$ times closer to 1 than the maximal second-largest eigenvalue of the individual Markov chains. □

It is easy to see that Theorem 62 can be applied to Tyshkevich products of degree sequences on which the Markov chain is rapidly mixing. Indeed, the inverse of the probability of choosing a switch operation on any factor of the Tyshkevich product is polynomial bounded.

4.2 P-stable degree sequences

In this section, we define P-stable sequences, prove that the switch Markov chain is rapidly mixing on realizations of P-stable sequences, and define a class of non-P-stable sequences that can be find as the degree sequences of induced subgraphs of realizations of non-P-stable sequences. First we define P-stability.

Definition 63. *Let $\mathcal{D}_b$ be a set of bipartite degree sequences. We say that $\mathcal{D}_b$ is* P-stable *if there exists a polynomial $p \in \mathbb{R}[x]$ such that for any x and any bipartite degree sequence $D_b = (D_1, D_2) \in \mathcal{D}_b$ on $n + m = x$ vertices we have*

$$\left| \mathbb{G}(D_b) \cup \left(\bigcup_{i \in [n],\ j \in [m]} \mathbb{G}(D_1 + \mathbb{1}_i, D_2 + \mathbb{1}_j) \right) \right| \leq p(x) |\mathbb{G}(D_b)|, \tag{4.3}$$

where $\mathbb{G}(D_s)$ is the set of realizations of the bipartite degree sequence D_b, and $\mathbb{1}_i$ is the vector that contains 1 at coordinate i and 0 everywhere else.

Theorem 64. *Let $\mathcal{D}_b$ be a class of P-stable bipartite degree sequences. Then the class of Markov chains $\mathcal{M} = \{M(D_b) | D_b \in \mathcal{D}_b\}$ is rapidly mixing.*

Proof. The proof uses the multicommodity flow technique (Theorem 31). To construct the appropriate path system given in Definition 30, we first introduce the sweep of an alternating cycle [55].

Let G_1 and G_2 be realizations of the same bipartite degree sequence such that $G_1 \Delta C = G_2$, where C is a cycle and Δ denotes symmetric difference. That is, G_2 can be obtained from G_1 by flipping the edges and non-edges along an alternating cycle of edges and non-edges in G_1. We show how to transform $G_1 = (V, U, E_1)$ into G_2 with $(|C| - 2)/2$ switch operations. Let the vertices of C be $u_1, v_1, u_2, v_2, \ldots, u_\ell, v_\ell$. Let the ordering of the vertices in C be such that $(u_1, v_1) \notin E_1$. We will call u_1 the cornerstone of the sweeping. The meaning ot the name is that all switch operations contains this vertex. Let v_i be the first vertex along C such that $(u_1, v_i) \in E$. Such a vertex exists because $(u_1, v_\ell) \in E_1$. We can perform a series of switch operations on the quadruple of vertices $(u_1, v_i, u_i, v_{i-1}), (u_1, v_{i-1}, u_{i-1}, v_{i-2}), \ldots, (u_1, v_2, u_2, v_1)$. Indeed, $(u_1, v_1) \in E_1$ and $(u_1, v_{i-1}) \notin E_1$ due to the definition of v_i. Furthermore, $(v_i, u_i) \notin E_1$ and $(u_i, v_{i-1} \in E_1)$ since C is an alternating cycle of edges and non-edges. After the switch operation on $(u_1, v_{i-1}, u_{i-1}, v_{i-2})$, $(u_1, v_{i-1}) \in E_1$, therefore the the next switch operation can be performed, etc. After $i-1$ switch operations we arrive to a realization G_1' which is either G_2 (if $i = \ell$) or differs from G_2 in a cycle of length $2\ell - 2i - 2$. Observe that G can be transformed into G' with a single switch operation if $G \Delta G'$ is a cycle of length 4, therefore by induction, $(|C| - 2)/2$ switch operations is sufficient to transform G_1 into G_2.

Now we can give the multicommodity flow. Let X and Y be the realizations of the same P-stable bipartite degree sequence D_b. The cardinalities of

the two vertex classes are denoted by n and m. Let $H = X\Delta Y$. For each vertex w in H, there are $\left(\frac{d(w)}{2}\right)!$ ways to pair the edges of X to the edges of Y incident to v, where $d(w)$ denotes the degree of w in H. For each possible pairing on each vertex in H, we define a cycle decomposition on H. Let Φ denote a fixed ensemble of pairings, and let $\varphi_v(e)$ denote the pair of e in Φ on vertex v. Let u_i be the smallest index vertex in H that does not have degree 0. Let v_j be the smallest index vertex for which (u_i, v_j) is in $H \bigcap Y$. Let (u_i, v_j) be denoted by e. Then define a circuit that starts with e, ends with $\varphi_{u_i}(e)$, and contains edges

$$e = e_1, e_2, \ldots, e_l$$

where for each $e_k = (u_k, v_k)$, e_{k+1} is defined as $\varphi_{v_k}(e_k)$. Denote the so-defined circuit by $\mathcal{C}_1$. If $H \setminus \mathcal{C}_1$ is not the empty graph, repeat the same on $H \setminus \mathcal{C}_1$ to get a circuit $\mathcal{C}_2$. The process is iterated till $H \setminus (\mathcal{C}_1 \cup \mathcal{C}_2 \cup \ldots \cup \mathcal{C}_s)$ is the empty graph. Then each $\mathcal{C}_i$ is decomposed into cycles $C_{i,1}, C_{i,2}, C_{i,j_i}$. The cycle $C_{i,1}$ is the cycle between the first and second visit of w, where w is the first revisited vertex in $\mathcal{C}_i$ (note that w might be both in U and V). Then $C_{i,2}$ is defined in the same way in $\mathcal{C}_i \setminus C_{i,1}$, etc. The path from X to Y is defined by processing the cycles

$$C_{1,1}, C_{1,2}, \ldots C_{1,j_1}, C_{2,1}, \ldots C_{s,s_j}$$

applying the sequence of switch operations sweeping the current cycle. Observe that the sweeping process can be obtained using any vertex as a corner stone. Let $u_1 \in U$ be the corner stone that is in U and has minimum degree in the induced subgraph on C (the current cycle on which the sweeping process is applied) in the vertex class U.

For the so-obtained path γ, we define

$$f(\gamma) := \frac{\pi(X)\pi(Y)}{\prod_{w \in V(H)} \left(\frac{d(w)}{2}\right)!}.$$

Let the number of realizations of D_b be N. Observe that the length of any path is less than nm due to the upper limit on the length of the sweeping process. Furthermore, π is the uniform distribution, therefore, it holds that

$$\frac{1}{Q(e)} \sum_{\gamma \ni e} f(\gamma)|\gamma| < \frac{\binom{n}{2}\binom{m}{2} nm}{N \prod_{w \in V(H)} \left(\frac{d(w)}{2}\right)!} \sum_{\gamma \ni e} 1.$$

Therefore, if the number of paths in the path system going through any edge is less than

$$poly(n,m)N \prod_{w \in V(H)} \left(\frac{d(w)}{2}\right)!$$

for some $poly(n,m)$, then the switch Markov chain is rapidly mixing. Let Z be a realization on a path γ going from X to Y obtained from the ensemble of pairings Φ, and let e be the transition from Z to Z'. Let M_G denote the adjacency matrix of a bipartite graph G, and let

$$\hat{M} := M_X + M_Y - M_Z. \tag{4.4}$$

Miklós, Erdős and Soukup proved that the path γ and thus X and Y can be unequivocally obtained from $\hat{M}$, Φ, $e = (Z, Z')$ and $O(\log(nm))$ bits of information. The crutial point to prove this is Theorem 5.10 in [117] proved by Lajos Soukup. The corollary is that the number of paths going through a particular $e = (Z, Z')$ is upper bounded by

$$poly(n,m)|\mathcal{M}_Z| \prod_{v \in V(H)} \left(\frac{d(v)}{2}\right)! \tag{4.5}$$

where $\mathcal{M}_Z$ are the set of possible $\hat{M}$ matrices defined in Equation (4.4). Therefore, it is sufficient to show that the number of $\hat{M}$ matrices is upper bounded by

$$poly(n,m)N.$$

To prove this, first observe that $\hat{M}$ has the same row and column sums as the adjacency matrix of any realization of D_b, and it might contain at most 3 values which are not 0 or 1, at most two of them are 2 and at most one of them is -1. Indeed, if

$$Z = X\Delta C_1 \Delta C_2 \Delta \ldots \Delta C_k$$

then $\hat{M}$ is a 0-1 matrix, and thus is an adjacency matrix of a realization of D_b. If Z is a realization during processing a cycle C, then there might be 3 chords of C (entries of $\hat{M}$ for some pair u_1 and v_j in C) whose corresponding values in $\hat{M}$ are not 0 or 1. Particularly, there might be two 2 values and one -1. A value 2 might appear when Z does not contain an edge which is presented both in X and Y, and a -1 might appear when Z contains an

edge which is neither in X nor in Y. Furthermore these "bad" values are in the same line corresponding to the cornerstone of the sweeping process.

Assume that $\hat{m}_{1,j} = 2$. There must be an i' such that $\hat{m}_{i',j} = 0$ (otherwise the row sum in $\hat{M}$ would be larger than $|V|$ which is absurd). Since $d(u_1)$ is minimal in the induced subgraph on C in vertex class U, there must be a j' such that $\hat{m}(1, j') < \hat{m}(i', j')$. Now we create a $\hat{M'}$ by subtracting 1 from $\hat{m}(1, j)$ and $\hat{m}(i', j')$ and adding 1 to $\hat{m}(i', j)$ and $\hat{m}(1, j')$. Observe that this does not change the row and column sums. Also, $\hat{m'}(1, j) = 1$, $\hat{m'}(i', j) = 1$. If $\hat{m'}(i', j') = 0$, then $\hat{m'}(1, j')$ must be -1. In that case, the -1 value is moved to the row i'. In all other cases, either -1 remained in the same row, or even might disappear if $\hat{m'}(1, j') = -1$ and $\hat{m'}(i', j') = 1$. In conclusion, $\hat{M'}$ contains one less 2's in its entries.

If there is another entry 2 in $\hat{M'}$, then we can do the same procedure. Therefore, after modifying at most 8 entries in $\hat{M}$, we arrive to a matrix that has the same row and column sums than X and Y, and has at most one -1 entry, all other entries are 0's and 1's.

If there is a -1 in the so-obtained matrix, then change it to 0. Then we get a matrix which is the adjacency matrix in $\mathbb{G}(D_1 + \mathbb{1}_x, D_2 + \mathbb{1}_y)$ for some x and y. Otherwise, we get a matrix which is the adjacency matrix of a realization of D_b.

So how many $\hat{M}$ might be there? We know that any matrix $\hat{M}$ is at most a Hamming distance 9 from

$$\tilde{\mathbb{G}} := \mathbb{G}(D_b) \cup \left(\bigcup_{i \in [n],\ j \in [m]} \mathbb{G}(D_1 + \mathbb{1}_i, D_2 + \mathbb{1}_j) \right),$$

such that at the differing places it contains $0, 1, 2$ or -1. But then it follows that the number of $\hat{M}$ matrices is upper bounded by some $poly(n, m)N$ for there can be at most

$$\sum_{i=1}^{9} \binom{nm}{i}^{i} |\tilde{\mathbb{G}}|$$

matrices being at most Hamming distance 9 from the set $\tilde{\mathbb{G}}$ with entries $0, 1, 2$ and -1. (This is a crude estimation as transforming $\hat{M}$ into a matrix in $\tilde{\mathbb{G}}$ is done not arbitrarily changing at most 9 entries; however, for our purposes, this estimation suffices.) Since $|\tilde{\mathbb{G}}| = O(poly(n, m))N$, the claim follows.

Therefore, the number of paths going through on a particular edge in the

Markov graph is upper bounded by the value in Equation (4.5), and thus

$$\frac{1}{Q(e)}\sum_{\gamma\ni e} f(\gamma)|\gamma| \;<\; \frac{\binom{n}{2}\binom{m}{2}nm}{N\prod_{w\in V(H)}\left(\frac{d(w)}{2}\right)!}\sum_{\gamma\ni e} 1 \le poly(nm). \qquad (4.6)$$

By Theorems 25 and 31, this proves the rapid mixing of the Markov chain. □

We denote by $\overline{D}_b$ the complement of the bipartite degree sequence $D_b = ((d_{1,1}, d_{1,2}, \ldots, d_{1,n}), (d_{2,1}, d_{2,2}, \ldots, d_{2,m}))$ such that for all i, $\overline{d}_{1,i} := m - d_{1,i}$ and for all j, $\overline{d}_{2,j} := n - d_{2,j}$. Observe that there is a bijection between the realizations of D_b and $\overline{D}_b$ obtained by complementation. A consequence is that we can define *down-P-stability* by subtracting 1 in both degree sequences and modifying Equation 4.3 as

$$\left|\mathbb{G}(D_b) \cup \left(\bigcup_{i\in[n],\ j\in[m]} \mathbb{G}(D_1 - \mathbb{1}_i, D_2 - \mathbb{1}_j)\right)\right| \le p(x)|\mathbb{G}(D_b)|, \qquad (4.7)$$

and prove that the switch Markov chain is rapidly mixing on any class of down-P-stable degree sequences.

We can say even more. In fact, a class of bipartite degree sequences are P-stable if and only if they are down-P-stable. This theorem for bipartite degree sequences and also an extension to simple degree sequences were proved by Tamás R. Mezei (unpublished work).

The applicability of Theorem 64 is based on the fact that we can prove for a large class of bipartite degree sequences that they are P-stable. The following theorem is about the P-stability of linearly bounded degree sequences.

Theorem 65. *Let $\mathcal{D}_b$ be a class of bipartite degree sequences such that for any degree sequence $D_b = (D_1, D_2) = ((d_{1,1}, d_{1,2}, \ldots, d_{1,n}), (d_{2,1}, d_{2,2}, \ldots, d_{2,m})) \in \mathcal{D}_b$ the following holds. There are constants $0 < c_1 \le c_2 < n$ and $0 < d_1 \le d_2 < m$ satisfying the following properties:*

$$\begin{aligned} c_1 \le d_{1,i} \le c_2, \qquad & \forall i \in [n] \\ d_1 \le d_{2,j} \le d_2, \qquad & \forall j \in [m]. \end{aligned} \qquad (4.8)$$

Furthermore, assume that

$$(c_2 - c_1 - 1)\cdot(d_2 - d_1 - 1) < 1 + \max\{c_1(m - d_2), d_1(n - c_2)\} \qquad (4.9)$$

holds. Then $\mathcal{D}_b$ is P-stable.

Proof. Let G be a realization of $(D_1 + \mathbb{1}_i, D_2 + \mathbb{1}_j)$ for some i and j. We show that the adjacency matrix of G is at most a Hamming distance 8 from the adjacency matrix of a realization of D_b. If the adjacency matrix of G, denoted by A, for the vertex pair (u_i, v_j) is 1 then by removing it, we get an adjacency matrix of a realization of D_b. Otherwise, by changing a_{u_i,v_j} from 0 to -1, we get an adjacency matrix whose row and column sums are the same than the row and column sums of the adjacency matrix of any realization of D_b. By rearranging the rows and columns (relabeling the vertices of G), we may assume that now $a_{u_1,v_1} = -1$.

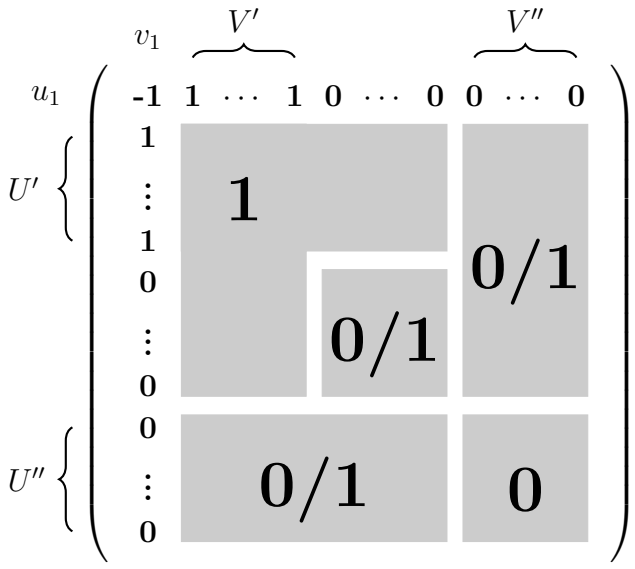


Figure 4.1: A is shown; each of the entries in the regions marked with 0/1 may be 0 or 1.

Let $U' = \{u \in U \mid a_{u,v_1} = 1\}$ and $V' = \{v \in V \mid a_{u_1,v} = 1\}$. If $\exists (u, v) \in U' \times V'$ such that $a_{u,v} = 0$, then by adding 1 to a_{u_1,v_1} and $a_{u,v}$ and removing 1 from $a_{u_1,v}$ and a_{u,v_1} transforms A into an adjacency matrix. We will call such change (adding 1 to one of the diagonals of a 2×2 square and removing one from the other diagonal) as the switch operation on u_1, u and v_1, v.

Therefore suppose from now on, that $\forall (u, v) \in U' \times V'$ we have $a_{u,v} = 1$. Let

$$U'' = \{u \in U \mid \exists v \in V' \,:\, a_{u,v} = 0\} \quad \text{and} \quad V'' = \{v \in V \mid \exists u \in U' \,:\, a_{u,v} = 0\}.$$

Clearly, $U'' \cap U' = V'' \cap V' = \emptyset$. Suppose there $\exists (u_2, v_2) \in U'' \times V''$ such that $a_{u_2,v_2} = 1$. By definition, there $\exists (u_1, v_1) \in U' \times V'$ such that $a_{u_2,v_1} = 0$ and $a_{u_1,v_2} = 0$. Clearly, at first the switch operation on u_1, u_2 and v_1, v_2, and then the switch operation on u_0, u_1 and v_0, v_1 transforms A into an adjacency matrix.

Lastly, suppose that $\forall (u, v) \in U'' \times V''$ we have $a_{u,v} = 0$. This case is shown in Figure 4.1.

We have

$$|U''| \cdot (m - d_1) \geq \left|\left\{(u, v) \in U'' \times V \;\middle|\; a_{u,v} = 0\right\}\right| = \\ = |U'' \times V''| + \left|\left\{(u, v) \in U'' \times (V \setminus V'') \;\middle|\; a_{u,v} = 0\right\}\right|.$$

The right hand side can be estimated from below as follows. Since the row and column sums of A are the same as the adjacency matrix of any realization of D_b, we have

$$|U'| \geq c_1 - a_{u_1,v_1} = c_1 + 1, \quad \text{and} \quad |V'| \geq d_1 - a_{u_1,v_1} = d_1 + 1.$$

For any $v \in V'$ and $u \in U \setminus U''$, we have $a_{u,v} = 1$. Also, for any $u \in U'$ and $v \in V \setminus V''$, we have $a_{u,v} = 1$. Therefore

$$n - c_1 - 2 \geq |U \setminus U' \setminus \{u_0\}| \geq |U''| \geq n - c_2,$$
$$m - d_1 - 2 \geq |V \setminus V' \setminus \{u_0\}| \geq |V''| \geq m - d_2.$$

Clearly, if $c_2 \leq c_1 + 1$ or $d_2 \leq d_1 + 1$ (i.e., G is half-regular), we already have a contradiction. We also have

$$\left|\left\{(u, v) \in U'' \times (V \setminus V'' \setminus \{v_1\}) \middle| \; a_{u,v} = 0\right\}\right| \geq \\ \geq (n - c_2)(m - |V''| - 1) - |U \setminus U' \setminus U''| \cdot |V \setminus V' \setminus V'' \setminus \{v_1\}|.$$

Moreover, $a_{u,v_1} = 0$ for any $u \in U''$. Combining these inequalities, we get

$$\begin{aligned} |U''| \cdot (m - d_1) \;\; \geq \;\; & (n - c_2) \cdot (m - |V''|) + (c_2 - c_1 - 1) + |U''|(m - d_1 - 1) + \\ & + |V''|(n - c_1 - 1) - (n - c_1 - 1) \cdot (m - d_1 - 1). \end{aligned}$$

Further simplying:

$$\begin{aligned} c_1 + 1 - c_2 + |U''| \;\; \geq \;\; & |V''|(c_2 - c_1 - 1) - c_2 m + (c_1 + 1)m + \\ & + (d_1 + 1)n - (c_1 + 1)(d_1 + 1). \end{aligned}$$

Since we may suppose that $c_2 \geq c_1 + 2$, we can substitute $|V''| \geq m - d_2$ and $|U''| \leq n - c_1 - 2$ into the inequality.

$$(c_2 - c_1 - 1)(d_2 - d_1 - 1) \geq d_1(n - c_2) + 1.$$

Symmetrically, a similar derivation gives

$$(c_2 - c_1 - 1)(d_2 - d_1 - 1) \geq c_1(n - d_2) + 1.$$

The last two inequalities clearly contradict the inequality in Equation 4.9. Therefore, there must exists $u_2 \in U''$ and $v_2 \in V''$ such that $a_{u_2,v_2} = 1$. Then two switch operations are sufficient to transform A into an adjacency matrix of a realization of D_b.

□

To see that the inequality in Equation 4.8 is indeed a linear bound on the degrees, observe that $c_1 = \frac{1}{4}n$, $c_2 = \frac{3}{4}n$, $d_1 = \frac{1}{4}m$ and $d_2 = \frac{3}{4}m$ satisfies the inequality. It can be seen that this bound is sharp, that is, for any $\varepsilon > 0$ the bounds $c_1 = \frac{1}{4}n$, $c_2 = \frac{3}{4}n$, $d_1 = \frac{1}{4}n$ and $d_2 = \left(\frac{3}{4} + \varepsilon\right) n$ does not give P-stable degree sequences. The counterexample can be given by the Tyshkevich product of three bipartite degree sequences. These degree sequences are the following (in order of their product):

1. a $\frac{1}{4}n$-regular degree sequence on $\left(\frac{1}{2} - \varepsilon\right) n + \left(\frac{1}{2} - \varepsilon\right) n$ vertices,
2. a bipartite degree sequence $(1, 2, \ldots, \varepsilon n), (1, 2, \ldots, \varepsilon n)$ on $\varepsilon n + \varepsilon n$ vertices
3. a $\frac{1}{4}n$-regular degree sequence on $\frac{1}{2}n + \frac{1}{2}n$ vertices.

Then the degrees in the first sequence are between $\frac{1}{4}n$ and $\frac{3}{4}n$ and in the second sequence between $\frac{1}{4}n$ and $\left(\frac{3}{4} + \varepsilon\right) n$. Observe that the second degree sequence has exactly one realization. On the other hand, Dániel Soltész proved that the bipartite degree sequence

$$(2, 2, 3, \ldots, \varepsilon n), (2, 2, 3, \ldots, \varepsilon n) \tag{4.10}$$

has $\theta\left(\frac{3+\sqrt{5}}{2}\right)^{\varepsilon n}$ realizations. Since the number of realizations of a Tyshkevich product is the product of the number of realizations of the factors, the class of bipartite degree sequences containing the above-constructed degree sequences are not P-stable.

We can also prove the following theorem.

Theorem 66. *Let $\mathcal{D}_b$ be a class of degree sequences with the following property. There exists a $c \in \mathbb{N}$ such that for all $D_b \in \mathcal{D}_b$, for all realizations $G = (U, V, E)$ of D_b and for all $u \in U$, $v \in V$ such that $(u, v) \notin E$, there is a an alternating edge-non-edge path of length at most c from u to v starting with an edge (and thus, ending with an edge). Then $\mathcal{D}_b$ is P-stable.*

Proof. Let $D_b = (D_1, D_2) \in \mathcal{D}_b$ on $n+m$ vertices, and let $x \in [n]$ and $y \in [m]$. Consider any realization $G = (U, V, E)$ of $(D_1 + \mathbb{1}_x, D_2 + \mathbb{1}_y)$. We prove that the adjacency matrix A of G has a Hamming distance at most c from the adjacency matrix of a realization of D_b. If $(u_x, v_y) \in E$, then by removing that edge, we get a realization of D_b, thus A is from a Hamming distance 1 from an adjacency matrix of a realization of D_b. Consider the realization G' of D_b whose adjacency matrix A' minimizing its Hamming distance from A. We claim that it has a Hamming distance at most $c + 1$ from A and it is an alternating path of edges and non-edges from u_x to v_y starting with an edge and thus ending with an edge.

First we show that the symmetric difference of A and A' but be a single alternating path of edges and non-edges (0's and 1's). Clearly, $A \Delta A'$ can be decomposed into alternating cycles and an alternating path of edges and non-edges (in sense of the corresponding realizations.) Then switching any alternating cycle in G' creates another realization of D_b with smaller Hamming distance. Therefore $G \Delta G'$ is an alternating path P_1 from u_x to v_y and in G' it starts and ends with a non-edge. Due to the condition of the theorem, there is an alternating path P_2 of length at most c from u_x to v_y starting and ending with an edge. Due to their different starts, P_1 and P_2 are edge-disjoint (although they might not be vertex disjoint). Flipping both P_1 and P_2 in G' we get a realization that is a Hamming distance at most c from G.

Since for any $D_b = (D_1, D_2) \in \mathcal{D}_b$, any x and y, and any realization G of $(D_1 + \mathbb{1}_x, D_2 \mathbb{1}_y)$, the adjacency matrix of G is a Hamming distance at most c from an adjacency matrix of a realization of D_b, $\mathcal{D}_b$ is not P-stable. □

The reverse of Theorem 66 is interesting. Assume that $\mathcal{D}_b$ is not P-stable. Then for any $d \in \mathbb{N}$ and $a \in \mathbb{R}$, there is a degree sequence $D_b = (D_1, D_2) \in \mathcal{D}_b$ on $n + m$ vertices and $i \in [n], j \in [m]$ such that

$$\frac{|\mathbb{G}(D_1 + \mathbb{1}_i, D_2 + \mathbb{1}_j)|}{|\mathbb{G}(D_b)|} > a(n+m)^d. \tag{4.11}$$

Map each realization of $(D_1+\mathbb{1}_i, D_2+\mathbb{1}_j)$ to a realization of D_b that minimizes its Hamming distance. The symmetric difference must be an alternating path from u_i to v_j that both starts and ends with an edge in the realization of $(D_1 + \mathbb{1}_i, D_2 + \mathbb{1}_j)$. That is, we map each realization of $(D_1 + \mathbb{1}_i, D_2 + \mathbb{1}_j)$ to a realization of D_b by flipping a shortest alternating path from u_i to v_j both starting and ending with an edge. Such path exists since D_b is graphic. If all such shortest paths were upper bounded by a small constant (computable from a and d), then it would contradict to the inequality in Equation 4.11. Indeed, in that case only a polynomial number of realizations of $(D_1 + \mathbb{1}_i, D_2 + \mathbb{1}_j)$ would be the inverse image of any realizations of D_b. If the bound of the length on the shortest path were sufficiently small, then this polynomial number would be a smaller polynomial than the one in the right hand side of Equation 4.11, a contradiction.

Thus, for any $c \in \mathbb{N}$, there must be a D_b, a realization $G = (U, V, E)$ of D_b, and a $u \in U$, $v \in V$ such that $(u, v) \notin E$, and the smallest alternating path from u to v starting and ending with an edge has length $> c$. This smallest alternating path is $u = u_1, v_1, u_2, v_2, \ldots, u_\ell, v_\ell = v$. This path together with the non-edge (u, v) is an alternating cycle that will be denoted by C. What is the degree sequence of the induced subgraph on C? For all $i = 1, 2, \ldots$ and $j = \ell, \ell - 1, \ldots$, $i < j$ $(u_i, v_j) \notin E$, otherwise there would be a shorter alternating path from u to v. Similarly, for all $i = 1, 2, 3, \ldots$ and $j = \ell, \ell - 1, \ldots$, $i < j$, $(v_i, u_j) \in E$. Therefore, $d(u_1) = d(v_\ell) = 1$, $d(u_2) = d(v_{\ell-1}) = 1$, $d(u_3) = d(v_{\ell-2}) = 2$, $d(u_4) = d(v_{\ell-3} = 3, \ldots$, $d(u_{\ell-1}) = d(v_2) = \ell - 2$, $d(u_\ell) = d(v_1) = \ell - 1$. Thus, we arrived to the following Theorem:

Theorem 67. *Let $\mathcal{D}_b$ be a class of degree sequences which is not P-stable. Then for all $c \in \mathbb{N}$, there exists a $D_b \in \mathcal{D}_b$ and a realization $G(U, V, E)$ of D_b, such that $G(U, V, E)$ contains a cycle C, and the induced subgraph on C has degree sequence $((1, 1, 2, 3, \ldots, \ell - 1), (1, 1, 2, 3, \ldots, \ell - 1))$, where $2\ell \geq c$ is the length of C.*

Observe that $((1, 1, 2, 3, \ldots, \ell-1), (1, 1, 2, 3, \ldots, \ell-1))$ is the complement of the degree sequence in Equation 4.10 (with $\varepsilon n + 1 = \ell$). This highlights why this degree sequence is interesting on its own. Dániel Soltész proved that the class containing this degree sequence for all ℓ is non-P-stable. However, the switch Markov chain on the realizations of it is still rapidly mixing[107].

4.3 Rapid mixing of the switch Markov chain on a family of non-stable degree sequences

In this section, we prove that the switch Markov chain is rapidly mixing on the realizations of the bipartite degree sequence $((1, 2, \ldots, n-2, n-1, n-1), (1, 2, \ldots, n-2, n-1, n-1))$.

We start with the definition of a class of degree sequences.

Definition 68. *Let us define the bipartite degree sequences and class of bipartite sequences:*

$$\begin{aligned} \boldsymbol{h}_k(n) &:= \begin{pmatrix} 1 & 2 & 3 & \cdots & n-2 & n-1 & n-k \\ n-k & n-1 & n-2 & \cdots & 3 & 2 & 1 \end{pmatrix} \\ \mathcal{H}_k &:= \left\{ \boldsymbol{h}_k(n) \;\middle|\; n \in \mathbb{N} \right\} \end{aligned}$$

Let $U_n = \{u_1, \ldots, u_n\}$ and $V_n = \{v_1, \ldots, v_n\}$, often denoted simply U and V. We label the vertices of $\boldsymbol{h}_0(n)$ such that U is the first and V is the second vertex class, with $\deg_{\boldsymbol{h}_0(n)}(u_i) = n + 1 - i$ and $\deg_{\boldsymbol{h}_0(n)}(v_i) = i$ for $i \in [1, n]$. The unique realization $H_0(n)$ is displayed on Figure 4.2.

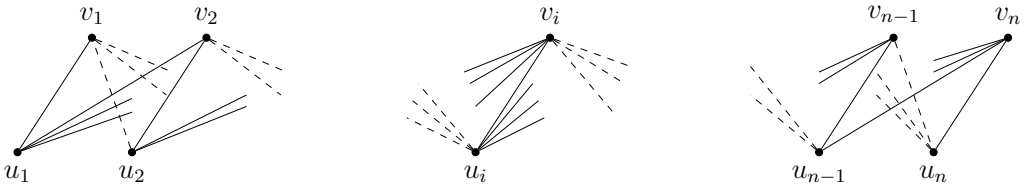


Figure 4.2: The unique realization $H_0(n)$ of $\boldsymbol{h}_0(n)$. Edges are denoted by straight lines, non-edges are denoted by dashed lines.

The next observation is that $\boldsymbol{h}_1(n)$ is the degree sequence of our interest and the symmetric difference of any realization of $\boldsymbol{h}_1(n)$ and the unique realization of $\boldsymbol{h}_0(n)$ is an alternating path from u_1 to v_n. Therefore, we can identify any realization of $\boldsymbol{h}_1(n)$ by this alternating path. Furthermore, this alternating path is monotonously increasing: when it contains vertices $u_1 = u_{i_1}, v_{i_2}, \ldots, u_{i_\ell}, v_{i_\ell} = v_n$, then for all odd j, $i_j \leq i_{j+1}$ and for all even j, $i_j < i_{j+1}$.

Even further, any switch operation *i*) moves one vertex in the alternating path or *ii*) skips two vertices or *iii*) (as an inverse of the previous move) adds

two vertices to a path. Indeed, let $H_0(n)$ denote the unique realization of $\boldsymbol{h}_0(n)$ and let $H_0(n)\Delta P_1$ and $H_0(n)\Delta P_2$ be two realizations of $\boldsymbol{h}_1(n)$, where P_1 and P_2 are two paths from u_1 to v_n and Δ denotes the symmetric difference. If the two realizations differ by a switch operation, then their symmetric difference is a cycle of length 4, C_4, thus we get that

$$C_4 = (H_0(n)\Delta P_1)\Delta(H_0(n)\Delta P_2) = P_1\Delta P2.$$

If the symmetric difference of two paths is a C_4, then they differ by either two-two edges in both paths or one edge in one of the paths and three in the other. It is impossible that all four edges comes from one path as the paths are monotonously increasing, so no vertex is revisited. When the C_4 contains two-two edges from the two paths, the two paths differ by one vertex, when C_4 contains one edge from one path and three from the other, then one of the paths have two consecutive vertices less than the other one, and all other vertices are the same.

We need the following key lemma.

Lemma 69. *Let X and Y be two realizations of $\boldsymbol{h}_1(n)$, let $P_X := H_0(n)\Delta X$, let $P_Y := H_0(n)\Delta Y$ and let $x_1, x_2, \ldots, x_k$ and $y_1, y_2, \ldots, y_\ell$ denote the vertices in the path P_X and P_Y, respectively. Then for all $i = 2, 3, \ldots, n-1$ there exists a realization Z of $\boldsymbol{h}_1(n)$ such that if the vertices of $P_Z := H_0(n)\Delta Z$ are $z_1, z_2, \ldots, z_m$, then $z_1 = y_1$, $z_2 = y_2$, $\ldots$, up to all vertices whose indices are smaller than i and going backward, $z_m = x_k, z_{m-1} = x_{k-1}$, $\ldots$, while the indices of the vertices of P_Z are larger than i.*

Proof. Let $y_1, y_2, \ldots, y_s$ be the vertices of P_Y with indices smaller than i, and let $x_t, x_{t+1}, \ldots, x_k$ be the vertices of P_X with indices larger than i. If y_s and x_t are in different vertex classes, then let $P_Z := y_1, y_2, \ldots, y_s, x_t, x_{t+1}, \ldots, x_m$. Otherwise let $P_Z := y_1, y_2, \ldots, y_s, w_i, x_t, x_{t+1}, \ldots, x_m$, where $w_i = u_i$ if $y_s, x_t \in V$ and $w_i = v_i$ if $y_s, x_t \in U$. □

We will denote by $P_Y \circ_i P_X$ the path P_Z constructed in the proof of Lemma 69. We will also define $P_Y \circ_1 P_X := P_X$ and $P_Y \circ_n P_X := P_Y$.

We prove another lemma on these paths.

Lemma 70. *Let X and Y be two realizations of $\boldsymbol{h}_1(n)$, let $P_X := H_0(n)\Delta X$, let $P_Y := H_0(n)\Delta Y$. Then for all $i = 1, 2, \ldots, n-1$, either $P_Y \circ_i P_X = P_Y \circ_{i+1} P_X$ or $P_Y \circ_{i+1} P_X$ can be obtained from $P_Y \circ_i P_X$ by at most two switch operations.*

Proof. If $i = 1$, then define $y_s = u_1$. Observe that $P_Y \circ_1 P_X$ contains y_s, and $y_s = u_1$ in $P_Y \circ_2 P_X$. Therefore, both $P_Y \circ_i P_X$ and $P_Y \circ_{i+1} P_X$ contains y_s for all i. Let $x_{t'} = v_n$ if $i + 1 = n$ and let $x_{t'}$ be the first vertex in the path whose index is larger than $i + 1$. ($x_{t'}$ might be x_{t+2} if both u_{i+1} and v_{i+1} is in P_X.) Observe that $x_{t'} = v_n$ in $P_Y \circ_{n-1} P_X$ and $P_Y \circ_n P_X$ contains v_n. Therefore, both $P_Y \circ_i P_X$ and $P_Y \circ_{i+1} P_X$ contains $x_{t'}$ for all i.

What follows is that the symmetric difference of $P_Y \circ_i P_X$ and $P_Y \circ_{i+1} P_X$ can be restricted to the vertices $y_s, u_i, v_i, u_{i+1}, v_{i+1}$ and $x_{t'}$. That is, the symmetric difference of the two paths consists of at most two paths on these six vertices. It is easy to see that two switch operations is sufficient, the situation can be modeled on paths in $\boldsymbol{h}_1(3)$. □

We are ready to state and prove the following theorem.

Theorem 71. *The switch Markov chain is rapidly mixing on $\mathcal{H}_1$.*

Proof. We use the canonical path system method. Let X and Y be two realizations of $\boldsymbol{h}_1(n)$. We denote by Z_i the realization $H_0(n)\Delta(P_Y \circ_i P_X)$, and let $\tilde{Z}_i$ denote the intermediate realization between Z_i and Z_{i+1} if Z_{i+1} can be obtained from Z_i by two switch operations and let $\tilde{Z}_i$ be Z_i if Z_{i+1} can be obtained from Z_i by at most 1 switch operation.

We construct the series of realizations represented by their symmetric difference with $H_0(n)$:

$$X = Z_1, \tilde{Z}_1, Z_2, \tilde{Z}_2, \ldots, \tilde{Z}_{n-1}, Z_n = Y.$$

Clearly, any two consecutive realizations differ by at most 1 switch operation. By removing realizations being identical with the previous realization in the series, we get a canonical path between X and Y.

Now, we are going to prove that there is no edge in the Markov graph that is heavily loaded, that is, in the Markov graph, the number of paths going through on any edge (Z_s, Z_{s+1}) is upper bounded by a polynomial of n times the number of realizations. To prove this, we introduce M_i as $P_X \circ_i P_Y$. For both Z_i and Z_{i+1} we will call M_i the *auxiliary structure* of the realization of entry in the canonical path.

We have to prove that from Z_s, Z_{s+1}, M_i and from $O(poly(\log(n)))$ bits of information, we can reconstruct X and Y. But this is obvious as we need only the information of i, the information if P_X and/or P_Y contained any vertex from $\{u_i, v_i\}$ and the switch operation from Z_i to $\tilde{Z}_i$ if $Z_s = \tilde{Z}_i$ for some i.

We get that we can reconstruct X and Y from M_i, Z_s, Z_{s+1} and logarithmic bit of information. It follows that the number of paths going through (Z_s, Z_{s+1}) in the Markov graph is upper bounded by a polynomial of n times the number of realizations of $\boldsymbol{h}_1(n)$. Since the target distribution of the Markov chain is the uniform distribution, and the length of any path is upper bounded by a linear function of n, and the inverse of the probability of a switch operation is also upper bounded by a polynomial of n we get that the Markov chain is rapidly mixing due to Theorem 31. Indeed, Equation 1.23 simplifies to

$$\kappa_\Gamma := \max_{e=(u,v)\in E} \sum_{(x,y)\in V\times V} \sum_{\gamma\in\Gamma_{(x,y)}:e\in\gamma} \pi(\gamma)\frac{|\gamma|}{P}, \tag{4.12}$$

where P is the probability of a switch operation and π is the probability of a realization in the uniform distribution, that is, the inverse of the number of realizations. Since we have a polynomial upper bound on $\frac{|\gamma|}{P}$, we get that

$$\kappa_\Gamma \leq \frac{poly(n)}{|\mathbb{G}(\boldsymbol{h}_1(n))|} \max_{e=(u,v)\in E} \sum_{(x,y)\in V\times V} \sum_{\gamma\in\Gamma_{(x,y)}:e\in\gamma} 1. \tag{4.13}$$

Since the number of terms in the summation is upper bounded by a $poly(n)|\mathbb{G}(\boldsymbol{h}_1(n))|$, we get a polynomial upper bound on κ_Γ, thus proved that the switch Markov chain is rapidly mixing on the realizations of $\boldsymbol{h}_1(n)$.

□

4.4 Further results

Here we list results extending and generalizing the results presented in this thesis. The author of this thesis contributed to these results, however, the main contributions were given by the above mentioned postdocs, Tamás R. Mezei and Dániel Soltész.

- The P-stability has been extended to simple and directed degree sequences. We can define that a class of arbitrary degree sequences is P-stable if any degree sequence $\boldsymbol{d}$ of n vertices in it satisfies

$$|\{G|G \in \mathbb{G}(\boldsymbol{d'}) \;\; L_1(\boldsymbol{d'}, \boldsymbol{d}) \leq 2\}| \leq poly(n)|\mathbb{G}(\boldsymbol{d})|.$$

  Then we can prove the following theorem:

Theorem 72. *[106] The switch Markov chain is rapidly mixing on P-stable unconstrained, bipartite, and directed degree sequence classes.*

We remark that in case of directed degree sequences, triple switches are needed in the Markov chain that flips an alternating cycle of edges and non-edges of a cycle of length 3.

- Theorem 65 can be extended to simple and directed degree sequences. That is, the switch Markov chain on the realizations of the class of linearly bounded simple and directed degree sequences is rapidly mixing [106].

- Tamás Mezei proved [106, 109] that a randomly generated Erdős-Rényi graph $G(n,p)$ is almost surely P-stable if

$$p, 1-p \geq \frac{5\log n}{n-1},$$

and a randomly generated Erdős-Rényi bipartite graph is almost surely P-stable if

$$p, 1-p \geq 4\sqrt{\frac{2\log n}{n}}.$$

- Dániel Soltész gave the asymptotic growth of $|\boldsymbol{h}_k(n)|$ for all k, and showed that for all k, $\mathcal{H}_k$ is a non-P-stable class of degree sequences. Still, Theorem 71 can be extended to any $\mathcal{H}_k$ [107] (main theorems were proved jointly by Tamás R. Mezei and Dániel Soltész).

Chapter 5

A decomposition based proof for fast mixing of a Markov chain over balanced realizations of a joint degree matrix

It turns out that a graphic JDM always have a *balanced* realization, that is, a realization in which all induced bipartite subgraphs of pair of vertex classes as well as all induced subgraphs on any vertex class are almost regular. It also turns out that the switch Markov chain remains irreducible on these realizations. Here we prove the rapid mixing of this Markov chain. The proof is based on a general theorem on the mixing of Markov chains on factorized state spaces. This is a joint work with Péter L. Erdős and Zoltán Toroczkai (Erdős, P.L., Miklós, I., Toroczkai, Z. (2015) SIAM Journal on Discrete Mathematics, 29:481–499.). The auxiliary graph in Figure 5.4 was given by E.L.P. The author of this thesis proved Theorems 74 and 78.

5.1 Mixing of Markov chains on factorized state spaces

Intuitively, when there is a partitioning of the state space of a Markov chain, the Markov chain is rapidly mixing on the partitions, and the Markov chain is rapidly mixing among the partitions, too, then the Markov chain is rapidly mixing on the whole space. In this section we provide a mathematically

rigorous description of this intuitive observation and prove it. The theorem and its proof were given by Erdős, Miklós and Toroczkai [110].

First, we need a generalization of the Cheeger inequality (the lower-bound).

Lemma 73. *For any reversible Markov chain, and any subset S of its state space,*

$$\frac{1-\lambda_2}{2}\min\{\pi(S),\pi(\overline{S})\} \leq \sum_{x\in S, y\in\overline{S}} \pi(x)T(y|x)\ . \tag{5.1}$$

Proof. The right-hand side of Equation (5.1) is symmetric due to the reversibility of the chain. Thus, if $\pi(S) > \frac{1}{2}$, then S and $\overline{S}$ can be switched. If $\pi(S) \leq \frac{1}{2}$, the inequality is simply a rearrangement of the Cheeger inequality (the left inequality in Theorem 33.). Indeed,

$$1 - 2\frac{\sum_{x\in S, y\in\overline{S}} \pi(x)T(y|x)}{\pi(S)} \leq 1 - 2\Phi \leq \lambda_2\ . \tag{5.2}$$

Rearranging the two ends of the inequality in Equation (5.2), we get the inequality in Equation (5.1). □

Now we are ready to state and prove a general theorem on rapidly mixing Markov chains on factorized state spaces.

Theorem 74. *Let $\mathcal{M}$ be a class of reversible, irreducible and aperiodic Markov chains whose state space Y can be partitioned into disjoint classes $Y = \cup_{x\in X} Y_x$ by the elements of some set X. The problem size of a particular chain is denoted by n. For notational convenience we also denote the element $y \in Y_x$ via the pair (x, y) to indicate the partition it belongs to. Let T be the transition matrix of $M \in \mathcal{M}$, and let π denote the stationary distribution of M. Moreover, let π_X denote the marginal of π on the first coordinate that is, $\pi_X(x) = \pi(Y_x)$ for all x. Also, for arbitrary but fixed x let us denote by π_{Y_x} the stationary probability distribution restricted to Y_x, i.e., $\pi(y)/\pi(Y_x)$, $\forall y \in Y_x$. Assume that the following properties hold:*

i For all x, the transitions with x fixed form an aperiodic, irreducible and reversible Markov chain denoted by M_x with stationary distribution π_{Y_x}. This Markov chain M_x has transition probabilities as Markov chain M for all transitions fixing x, except loops, which have increased probabilities such that the transition probabilities sum up to 1. All transitions

that would change x have 0 probabilities. Furthermore, this Markov chain is rapidly mixing, i.e., for its second-largest eigenvalue $\lambda_{M_x,2}$ it holds that

$$\frac{1}{1-\lambda_{M_x,2}} \leq \mathrm{poly}_1(n).$$

ii There exists a Markov chain M' with state space X and with transition matrix T' which is aperiodic, irreducible and reversible w.r.t. π_X, and for all x_1, y_1, x_2 it holds that

$$\sum_{y_2 \in Y_{x_2}} T((x_2,y_2)|(x_1,y_1)) \geq T'(x_2|x_1). \tag{5.3}$$

Furthermore, this Markov chain is rapidly mixing, namely, for its second-largest eigenvalue $\lambda_{M',2}$ it holds that

$$\frac{1}{1-\lambda_{M',2}} \leq \mathrm{poly}_2(n).$$

Then M is also rapidly mixing as its second-largest eigenvalue obeys:

$$\frac{1}{1-\lambda_{M,2}} \leq \frac{256\mathrm{poly}_1^2(n)\mathrm{poly}_2^2(n)}{\left(1-\frac{1}{\sqrt{2}}\right)^4}$$

Proof. For any non-empty subset S of the state space $Y = \bigcup_x Y_x$ of M we define

$$X(S) := \{x \in X \mid \exists y, (x,y) \in S\}$$

and for any given $x \in X$ we have

$$Y_x(S) := \{(x,y) \in Y \mid (x,y) \in S\} = Y_x \cap S.$$

We are going to prove that the ergodic flow $F(S)$ (see Equation (1.26)) from any $S \subset Y$ with $0 < \pi(S) \leq 1/2$ cannot be too small and therefore, neither the conductance of the Markov chain will be small. We cut the state space into two parts $Y = Y^l \cup Y^u$, namely the lower and upper parts using the following definitions (see also Fig. 5.1): the partition $X = L \sqcup U$ is defined as

$$\begin{aligned} L &:= \left\{ x \in X \,\middle|\, \frac{\pi(Y_x(S))}{\pi(Y_x)} \leq 1/\sqrt{2} \right\}, \\ U &:= \left\{ x \in X \,\middle|\, \frac{\pi(Y_x(S))}{\pi(Y_x)} > 1/\sqrt{2} \right\}. \end{aligned}$$

Furthermore, we introduce:

$$Y^l := \bigcup_{x \in L} Y_x \quad \text{and} \quad Y^u := \bigcup_{x \in U} Y_x \,,$$

and finally let

$$S_l := S \cap Y^l \quad \text{and} \quad S_u := S \cap Y^u.$$

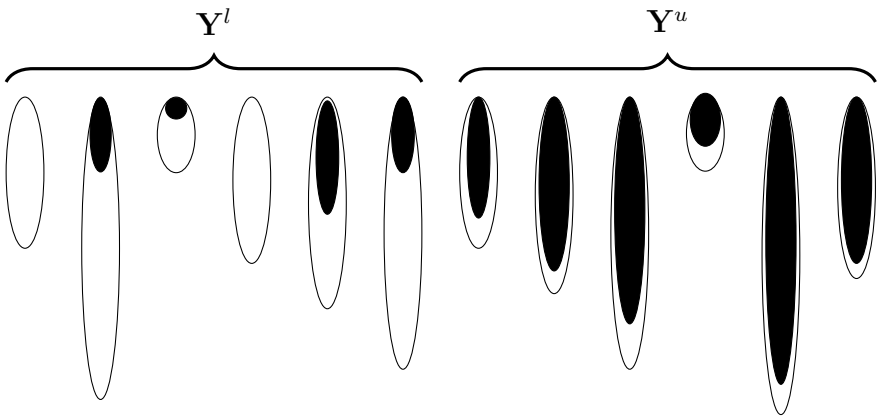


Figure 5.1: The structure of $Y = Y^l \sqcup Y^u$. A non-filled ellipse (with a simple line boundary) represents the space Y_x for a given x. The solid black ellipses represent the set S with some of them (the S_l) belonging to the lower part Y^l, and the rest (the S_u) belonging to the upper part (Y^u).

Since M' is rapidly mixing we can write (based on Theorem 33):

$$1 - 2\Phi_{M'} \leq \lambda_{M',2} \leq 1 - \frac{1}{\text{poly}_2(n)} \,,$$

or

$$\Phi_{M'} \geq \frac{1}{2\text{poly}_2(n)} \,.$$

Without loss of generality, we can assume that $\text{poly}_2(n) > 1$ for all positive n, a condition that we need later on for technical reasons. We use this lower bound of conductance to define two cases regarding the lower and upper part of S.

1. We say that the lower part S_l is not a negligible part of S when

$$\frac{\pi(S_l)}{\pi(S_u)} \geq \frac{1}{4\sqrt{2}\mathrm{poly}_2(n)}\left(1-\frac{1}{\sqrt{2}}\right). \tag{5.4}$$

2. We say that the lower part S_l is a negligible part of S when

$$\frac{\pi(S_l)}{\pi(S_u)} < \frac{1}{4\sqrt{2}\mathrm{poly}_2(n)}\left(1-\frac{1}{\sqrt{2}}\right). \tag{5.5}$$

Our plan is the following: the ergodic flow $F(S)$ is positive on any non-empty subset and it obeys:

$$F(S) = F'(S_l)\frac{\pi(S_l)}{\pi(S)} + F'(S_u)\frac{\pi(S_u)}{\pi(S)},$$

where

$$F'(S_l) := \frac{1}{\pi(S_l)} \sum_{x\in S_l, y\in \bar{S}} \pi(x)T(y|x)$$

and

$$F'(S_u) := \frac{1}{\pi(S_u)} \sum_{x\in S_u, y\in \bar{S}} \pi(x)T(y|x).$$

In other words, $F'(S_l)$ and $F'(S_u)$ are defined as the flow going from S_l and S_u and leaving S.

The value $F(S)$ cannot be too small, if at least one of $F'(S_l)$ or $F'(S_u)$ is big enough (and the associated fraction $\pi(S_l)/\pi(S)$ or $\pi(S_u)/\pi(S)$). In Case 1 we will show that $F'(S_l)$ itself is big enough. To that end it will be sufficient to consider the part which leaves S_l but not Y^l (this guarantees that it goes out of S, see also Fig. 5.2). For Case 2 we will consider $F'(S_u)$, particularly that part of it which goes from S_u to $Y^l \setminus S_l$ (and then going out of S, not only S_u, see also Fig. 5.3).

In **Case** 1, the flow going out from S_l within Y^l is sufficient to prove that the conditional flow going out from S is not negligible. We know that for any particular x, we have a rapidly mixing Markov chain M_x over the second coordinate y. Let their smallest conductance be denoted by Φ_X. Since all these Markov chains are rapidly mixing, we have that

$$\max_x \lambda_{M_x,2} \leq 1 - \frac{1}{\mathrm{poly}_1(n)}$$

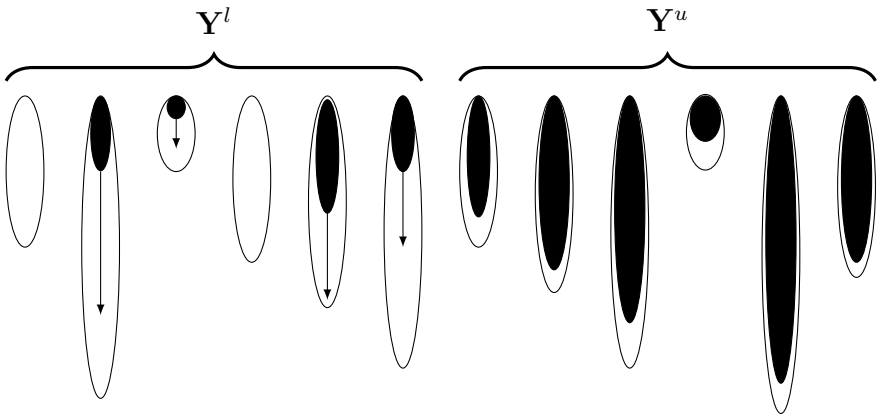


Figure 5.2: When S_l is not a negligible part of S, there is a considerable flow going out from S_l to within Y^l, implying that the conditional flow going out from S cannot be small. See text for details and rigorous calculations.

or, equivalently:

$$\Phi_X \geq \frac{1}{2\mathrm{poly}_1(n)}.$$

However, in the lower part, for any particular x one has:

$$\pi_{Y_x}(Y_x(S)) = \frac{\pi(Y_x(S))}{\pi(Y_x)} \leq \frac{1}{\sqrt{2}},$$

so for any fixed x belonging to L it holds that

$$\frac{1}{2\mathrm{poly}_1(n)} \min\left\{\pi_{Y_x}(Y_x(S)), \left(1 - \frac{1}{\sqrt{2}}\right)\right\} \leq$$
$$\leq \sum_{(x,y)\in S,(x,y')\in\bar{S}} \pi_{Y_x}((x,y))T((x,y')|(x,y))$$

using the modified Cheeger inequality (Lemma 73). Observing that

$$\pi_{Y_x}((x,y)) = \frac{\pi((x,y))}{\pi(Y_x)},$$

we obtain:

$$\begin{aligned}
&\frac{1}{2\text{poly}_1(n)}\pi(Y_x(S))\left(1-\frac{1}{\sqrt{2}}\right) \leq \\
&\leq \frac{1}{2\text{poly}_1(n)}\min\left\{\pi(Y_x(S)), \pi(Y_x)\left(1-\frac{1}{\sqrt{2}}\right)\right\} \leq \\
&\leq \sum_{(x,y)\in S,(x,y')\in\bar{S}} \pi((x,y))T((x,y')|(x,y)) \; .
\end{aligned}$$

Summing this for all the x's belonging to L, we deduce that

$$\begin{aligned}
&\pi(S_l)\frac{1}{2\text{poly}_1(n)}\left(1-\frac{1}{\sqrt{2}}\right) \leq \\
&\sum_{x|Y_x(S)\subseteq S_l}\left(\sum_{(x,y)\in S,(x,y')\in\bar{S}} \pi((x,y))T((x,y')|(x,y))\right) . \qquad (5.6)
\end{aligned}$$

Note that the flow on the right-hand side of Equation 5.6 is not only going out from S_l but also from the entire S. Therefore, we have that

$$F(S) \geq \frac{\pi(S_l)}{\pi(S)} \times \frac{1}{2\text{poly}_1(n)}\left(1-\frac{1}{\sqrt{2}}\right) .$$

Either $\pi(S_l) \leq \pi(S_u)$, which then yields

$$\frac{\pi(S_l)}{\pi(S)} = \frac{\pi(S_l)}{\pi(S_l)+\pi(S_u)} \geq \frac{\pi(S_l)}{2\pi(S_u)} \geq \frac{1}{8\sqrt{2}\text{poly}_2(n)}\left(1-\frac{1}{\sqrt{2}}\right)$$

after using Equation 5.4, or $\pi(S_l) > \pi(S_u)$, in which case we have

$$\frac{\pi(S_l)}{\pi(S)} > \frac{1}{2} \geq \frac{1}{8\sqrt{2}\text{poly}_2(n)}\left(1-\frac{1}{\sqrt{2}}\right) .$$

(Note that $\text{poly}_2(n) > 1$.) Thus in both cases the following inequality holds:

$$F(S) \geq \frac{1}{8\sqrt{2}\text{poly}_2(n)}\left(1-\frac{1}{\sqrt{2}}\right) \times \frac{1}{2\text{poly}_1(n)}\left(1-\frac{1}{\sqrt{2}}\right) .$$

In **Case** 2, the lower part of S is a negligible part of S. We have that

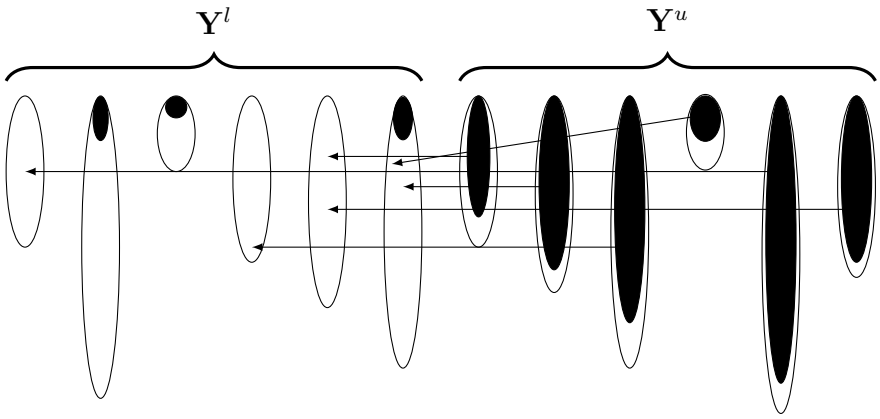


Figure 5.3: When S_l is a negligible part of S, there is a considerable flow going out from S_u into $Y^l \setminus S_l$. See text for details and rigorous calculations.

$$\pi_X(X(S_u)) \leq \frac{1}{\sqrt{2}}$$

otherwise $\pi(S_u) > 1/2$ would happen (due to the definition of the upper part), and then $\pi(S) > 1/2$, a contradiction.

Hence in the Markov chain M', based on the Lemma 73, we obtain for $X(S_u)$ that

$$\frac{1}{2\text{poly}_2(n)} \min\left\{\pi_X(X(S_u)), \left(1 - \frac{1}{\sqrt{2}}\right)\right\} \leq \sum_{\substack{x' \in \overline{X(S_u)} \\ x \in X(S_u)}} \pi_X(x) T'(x'|x). \quad (5.7)$$

For all y for which $(x, y) \in S_u$, due to Equation (5.3), we can write:

$$T'(x'|x) \leq \sum_{y'} T((x', y')|(x, y)) \ .$$

Multiplying this by $\pi((x, y))$, then summing for all suitable y:

$$\pi(Y_x(S))T'(x'|x) \leq \sum_{y|(x,y)\in S_u} \sum_{y'} \pi((x, y))T((x', y')|(x, y))$$

(note that $x \in U$ and thus $Y_x(S) = Y_x(S_u)$) and thus

$$T'(x'|x) \leq \frac{\sum_{y|(x,y)\in S_u} \sum_{y'} \pi((x, y))T((x', y')|(x, y))}{\pi(Y_x(S))}.$$

Inserting this into Equation 5.7, we find that

$$\frac{1}{2\text{poly}_2(n)} \min\left\{\pi_X(X(S_u)), \left(1-\frac{1}{\sqrt{2}}\right)\right\} \leq$$
$$\leq \sum_{x\in X(S_u), x'\in\overline{X(S_u)}} \frac{\pi_X(x)}{\pi(Y_x(S))} \sum_{y|(x,y)\in S_u} \sum_{y'} \pi((x,y))T((x',y')|(x,y)).$$

Recall that $\pi_X(x) = \pi(Y_x)$, and thus $\frac{\pi_X(x)}{\pi(Y_x(S))} \leq \sqrt{2}$ for all $x \in X(S_u)$. Therefore we can write that

$$\frac{1}{2\text{poly}_2(n)} \min\left\{\pi_X(X(S_u)), \left(1-\frac{1}{\sqrt{2}}\right)\right\} \leq$$
$$\sqrt{2} \sum_{(x,y)\in S_u} \left(\sum_{(x',y')|x'\in\overline{X(S_u)}} \pi((x,y))T((x',y')|(x,y)) \right).$$

Note that $\pi(S_u) \leq \pi_X(X(S_u)) < 1$, and since both items in the minimum taken in the LHS are smaller than 1, their product will be smaller than any of them. Therefore we have

$$\frac{1}{2\sqrt{2}\text{poly}_2(n)} \pi(S_u) \left(1-\frac{1}{\sqrt{2}}\right) \leq$$
$$\leq \sum_{(x,y)\in S_u} \left(\sum_{(x',y')|x'\in\overline{X(S_u)}} \pi((x,y))T((x',y')|(x,y)) \right).$$

This flow is going out from S_u, and it is so large that at most half of it can be picked up by the lower part of S (due to reversibility and due to Equation 5.5), and thus the remaining part, i.e., at least half of the flow, must go out of S. Therefore:

$$\frac{\pi(S_u)}{\pi(S)} \times \frac{1}{4\sqrt{2}\text{poly}_2(n)} \left(1-\frac{1}{\sqrt{2}}\right) \leq F(S) .$$

However, since S_u dominates S, namely, $\pi(S_u) > \frac{\pi(S)}{2}$, we have that

$$\frac{1}{8\sqrt{2}\text{poly}_2(n)} \left(1-\frac{1}{\sqrt{2}}\right) \leq F(S).$$

Comparing the bounds from Case 1 and Case 2, for all S satisfying $0 < \pi(S) \leq \frac{1}{2}$, we can write:

$$\frac{1}{16\sqrt{2}\text{poly}_2(n)\text{poly}_1(n)}\left(1-\frac{1}{\sqrt{2}}\right)^2 \leq F(S).$$

And thus, for the conductance of the Markov chain M (which is the minimum over all possible S)

$$\frac{1}{16\sqrt{2}\text{poly}_2(n)\text{poly}_1(n)}\left(1-\frac{1}{\sqrt{2}}\right)^2 \leq \Phi_M.$$

Applying this to the Cheeger inequality, one obtains

$$\lambda_{M,2} \leq 1 - \frac{\left(\frac{1}{16\sqrt{2}\text{poly}_2(n)\text{poly}_1(n)}\left(1-\frac{1}{\sqrt{2}}\right)^2\right)^2}{2}$$

and thus

$$\frac{1}{1-\lambda_{M,2}} \leq \frac{256\text{poly}_1^2(n)\text{poly}_2^2(n)}{\left(1-\frac{1}{\sqrt{2}}\right)^4}$$

which is what we wanted to prove. □

Martin and Randall [64] have developed a similar theorem. They assume a disjoint decomposition of the state space Ω of an irreducible and reversible Markov chain defined via the transition probabilities $P(y|x)$. They require that the Markov chain be rapidly mixing when restricted onto each partition Ω_i ($\Omega = \cup_i \Omega_i$) and furthermore, another Markov chain, the so-called projection Markov chain $\overline{P}(i|j)$ defined over the indices of the partitions be also rapidly mixing. If all these hold, then the original Markov chain is also rapidly mixing. For the projection Markov chain they use the normalized conditional flow

$$\overline{P}(j|i) = \frac{1}{\pi(\Omega_i)} \sum_{x\in\Omega_i, y\in\Omega_j} \pi(x)P(y|x) \tag{5.8}$$

as transition probabilities. This can be interpreted as a weighted average transition probability between two partitions, while in our case, Equation (5.3) requires only that the transition probability of the lower bounding

Markov chain is not more than the minimum of the sum of the transition probabilities going out from one member of the partition (subset Y_{x_1}) to the other member of the partition (subset Y_{x_2}) with the minimum taken over all the elements of Y_{x_1}. Obviously, it is a stronger condition that our Markov chain must be rapidly mixing, since a Markov chain is mixing slower when each transition probability between any two states is smaller. (The latter statement is based on a comparison theorem by Diaconis and Saloff-Coste [25].) Therefore, from that point of view, our theorem is weaker. On the other hand, the average transition probability (Equation (5.8)) is usually hard to calculate, and in this sense our theorem is more applicable. Note that Martin and Randall have also resorted in the end to using chain comparison techniques (Sections 2.2 and 3 in their paper) employing a Metropolis-Hastings chain as a lower bounding chain instead of the projection chain above. Our theorem, however, provides a direct proof of a similar statement.

5.2 Balanced realizations of a JDM

A symmetric matrix J with non-negative integer elements is the *joint degree matrix* (JDM) of an undirected simple graph G if the element $J_{i,j}$ gives the number of edges between the class V_i of vertices all having degree i and the class V_j of vertices all with degree j in the graph. In this case we also say that J is *graphic* and that G is a *graphic realization* of J. Note that there can be many different graphic realizations of the same JDM.

Given a JDM, the number of vertices $n_i = |V_i|$ in class i is obtained from:

$$n_i = \frac{J_{i,i} + \sum_{j=1}^{k} J_{i,j}}{i}, \tag{5.9}$$

where k denotes the maximum number of degrees. This implies that a JDM also uniquely determines the degree sequence, since we have obtained the number of nodes of given degrees for all possible degrees. For sake of uniformity we consider all vertex classes V_i for $i = 1, \ldots, k$; therefore we consider empty classes with $n_i = 0$ vertices as well. A necessary condition for J to be graphic is that all the n_i-s are integers. Let n denote the total number of vertices. Naturally, $n = \sum_i n_i$ and it is uniquely determined via Equation (5.9) for a given graphic JDM. The necessary and sufficient conditions for a given JDM to be graphic are provided in the following theorem

Theorem 75. **[104]** *A $k \times k$ matrix J is a graphic JDM if and only if the following conditions hold:*

1. *For all $i = 1, \ldots, k$*

$$n_i := \frac{J_{i,i} + \sum_{j=1}^{k} J_{i,j}}{i}$$

   is integer.

2. *For all $i = 1, \ldots, k$*

$$J_{i,i} \leq \binom{n_i}{2}.$$

3. *For all $i = 1, \ldots, k$ and $j = 1, \ldots, k$, $i \neq j$,*

$$J_{i_j} \leq n_i n_j.$$

Let $d_j(v)$ denote the number of edges such that one end-vertex is v and the other end-vertex belongs to V_j, i.e., $d_j(v)$ is the degree of v in V_j. The vector consisting of the $d_j(v)$-s for all j is called the *degree spectrum* of vertex v. We introduce the notation

$$\Theta_{i,j} = \begin{cases} 0, & \text{if } n_i = 0\,, \\ \frac{J_{i,j}}{n_i}, & \text{otherwise,} \end{cases}$$

which gives the average number of neighbors of a degree-i vertex in vertex class V_j. Then a realization of the JDM is *balanced* iff for every i and all $v \in V_i$ and all j, we have

$$|d_j(v) - \Theta_{i,j}| < 1\ .$$

The following theorem is proven in paper [104] as Corollary 5:

Theorem 76. *Every graphic JDM admits a balanced realization.*

A restricted switch operation (RSO) takes two edges (x, y) and (u, v) with x and u from the same vertex class and switches them with two non-edges (x, v) and (u, y). The RSO preserves the JDM, and in fact forms an irreducible Markov chain over all its realizations [104]. An RSO Markov chain restricted to *balanced realizations* can be defined as follows:

Definition 77. *Let J be a JDM. The state space of the RSO Markov chain consists of all the balanced realizations of J. It was proved by Czabarka et al.* [104] *that this state space is connected under restricted switch operations. The transitions of the Markov chain are defined in the following way. With probability* $1/2$*, the chain does nothing, so it remains in the current state (we consider a lazy Markov chain). With probability* $1/2$ *the chain will choose four, pairwise disjoint vertices, v_1, v_2, v_3, v_4 from the current realization (the possible choices are order dependent) and check whether v_1 and v_2 are chosen from the same vertex class, and furthermore whether the*

$$E \setminus \{(v_1, v_3), (v_2, v_4)\} \cup \{(v_1, v_4), (v_2, v_3)\}$$

switch operation is feasible. If this is the case then our Markov chain performs the switch operation if it leads to another balanced JDM realization. Otherwise the Markov chain remains in the same state. (Note that exactly two different orders of the selected vertices will provide the same switch operation, since the roles of v_1 and v_2 are symmetric.) *Then there is a transition with probability*

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

between two realizations iff there is a RSO transforming one into the other.

Here we prove that such a Markov chain is rapidly mixing. The convergence of a Markov chain is measured as a function of the input data size. Here we note that the size of the data is the number of vertices (or number of edges, they are polynomially bounded functions of each other) and not the number of digits to describe the JDM. This distinction is important as, for example, one can create a 2×2 JDM with values $J_{2,2} = J_{3,3} = 0$ and $J_{2,3} = J_{3,2} = 6n$, which has $\Omega(n)$ number of vertices (or edges) but it needs only $O(\log(n))$ number of digits to describe (except in the unary number system). Alternatively, one might consider the input is given in unary.

Formally, we state the rapid mixing property via the following theorem:

Theorem 78. *The RSO Markov chain on balanced JDM realizations is a rapidly mixing Markov chain, namely, for the second-largest eigenvalue λ_2 of this chain, it holds that*

$$\frac{1}{1-\lambda_2} = O(\text{poly}(n))$$

where n is the number of vertices in the realizations of the JDM.

Note that the expression on the LHS is called, with some abuse of notation, the *relaxation time*: it is the time is needed for the Markov chain to reach its stationary distribution. The proof is based on the special structure of the state space of the balanced JDM realizations. This special structure allows the following proof strategy: if we can prove that some auxiliary Markov chains are rapidly mixing on some sub-spaces obtained from decomposing the above-mentioned specially structured state space, then the Markov chain on the whole space is also rapidly mixing. We are going to prove the rapid mixing of these auxiliary Markov chains, as well as give the proof of the general theorem, that a Markov chain on this special structure is rapidly mixing, hence proving our main Theorem 78.

In order to describe the structure of the space of balanced JDM realizations, we first define the almost semi-regular bipartite and almost regular graphs.

Definition 79. *A bipartite graph $G(U, V; E)$ is* almost semi-regular *if for any $u_1, u_2 \in U$ and $v_1, v_2 \in V$*

$$|d(u_1) - d(u_2)| \leq 1$$

and

$$|d(v_1) - d(v_2)| \leq 1.$$

Definition 80. *A graph $G(V, E)$ is* almost regular*, if for any $v_1, v_2 \in V$*

$$|d(v_1) - d(v_2)| \leq 1.$$

It is easy to see that the restriction of any balanced realization of the JDM to vertex classes $V_i, V_j, i \neq j$ can be considered as the coexistence of two almost regular graphs (one on V_i and the other on V_j), and one almost semi-regular bipartite graph on the vertex class pair V_i, V_j. More generally, the collection of these almost semi-regular bipartite graphs and almost regular graphs completely determines the balanced JDM realization. Formally:

Definition 81 (Labeled union)**.** *Any balanced JDM realization can be represented as a set of almost semi-regular bipartite graphs and almost regular graphs. The realization can then be constructed from these factor graphs as their* labeled union*: the vertices with the same labels are collapsed, and the edge set of the union is the union of the edge sets of the factor graphs.*

It is useful to construct the following auxiliary graphs. For each vertex class V_i, we create an auxiliary bipartite graph, $\mathcal{G}_i(V_i, U; E)$, where U is a set of "super-nodes" representing all vertex classes V_j, including V_i. There is an edge between $v \in V_i$ and super-node u_j representing vertex class V_j iff

$$d_j(v) = \lceil \Theta_{i,j} \rceil \ ,$$

i.e., iff node v carries the ceiling of the average degree of its class i toward the other class j. (For sake of uniformity, we construct these auxiliary graphs for all $i = 1, \ldots, k$, even if some of them have no edge at all. Similarly, all super-nodes are given, even if some of them have no incident edge.) We claim that these k auxiliary graphs are *half-regular*, i.e., each vertex in V_i has the same degree (the degrees in the vertex class U might be arbitrary). Indeed, the vertices in V_i all have the same degree in the JDM realization, therefore, the number of times they have the ceiling of the average degree toward a vertex class is constant in a balanced realization.

Let Y denote the space of all balanced realizations of a JDM and just as before, let k denote the number of vertex classes (some of them can be empty). We will represent the elements of Y via a vector y whose $k(k+1)/2$ components are the k almost regular graphs and the $k(k-1)/2$ almost regular bipartite graphs from their labeled union decomposition, as described in Definition 81 above. Given an element $y \in Y$ (i.e., a balanced graphic realization of the JDM) it has k associated auxiliary graphs $\mathcal{G}_i(V_i, U; E)$, one for every vertex class V_i (some of them can be empty graphs). We will consider this collection of auxiliary graphs for a given y as a k-dimensional vector x, where $x = (\mathcal{G}_1, \ldots, \mathcal{G}_k)$.

For any given y we can determine the corresponding x (so no particular y can correspond to two different xs), however, for a given x there can be several y's with that same x. We will denote by Y_x the subset of Y containing all the y's with the same (given) x and by X the set of all possible induced x vectors. Clearly, the x vectors can be used to define a disjoint partition on Y: $Y = \bigcup_{x \in X} Y_x$. For notational convenience we will consider the space Y as pairs (x, y), indicating the x-partition to which y belongs. This should not be confused with the notation for an edge, however, this should be evident from the context. A restricted switch operation might fix x, in which case it will make a move only within Y_x, but if it does not fix x, then it will change both x and y. For any x, the RSOs moving only within Y_x form a Markov chain. On the other hand, tracing only the x's from the pairs (x, y) is not

a Markov chain: the probability that an RSO changes x (and thus also y) depends also on the current y not only on x. However, the following theorem holds:

Theorem 82. *Let (x_1, y_1) be a balanced realization of a JDM in the above mentioned representation.*

i Assume that (x_2, y_2) balanced realization is derived from the first one with one restricted switch operation. Then, either $x_1 = x_2$ or they differ in exactly one coordinate, and the two corresponding auxiliary graphs differ only in one switch operation.

ii Let x_2 be a vector differing only in one coordinate from x_1, and furthermore, only in one switch within this coordinate, namely, one switch within one coordinate is sufficient to transform x_1 into x_2. Then there exists at least one y_2 such that (x_2, y_2) is a balanced JDM realization and (x_1, y_1) can be transformed into (x_2, y_2) with a single RSO.

Proof. (i) This is just the reformulation of the definitions for the (x, y) pairs. (ii) (See also Fig. 5.4) By definition there is a degree $i, 1 \leq i \leq k$ such that auxiliary graphs $x_1(\mathcal{G}_i)$ and $x_2(\mathcal{G}_i)$ are different and one switch operation transforms the first one into the second one. More precisely there are vertices $v_1, v_2 \in V_i$ such that the switch transforming $x_1(\mathcal{G}_i)$ into $x_2(\mathcal{G}_i)$ removes edges (v_1, U_j) and (v_2, U_k) (with $j \neq k$) and adds edges (v_1, U_k) and (v_2, U_j). (The capital letters show that the second vertices are super-vertices.) Since the edge (v_1, U_j) exists in the graph $x_1(G_1)$ and (v_2, U_j) does not belong to graph $x_1(G_i)$, therefore $d_j(v_1) > d_j(v_2)$ in the realization (x_1, y_1). This means that there is at least one vertex $w \in V_j$ such that w is connected to v_1 but not to v_2 in the realization (x_1, y_1). Similarly, there is at least one vertex $r \in V_k$ such that r is connected to v_2 but not to v_1 (again, in realization (x_1, y_1)). Therefore, we have a required RSO on nodes v_1, v_2, w, r. □

Thus any RSO on a balanced realization yielding another balanced realization either does not change x or changes x exactly on one coordinate (one auxiliary graph), and this change can be described with a switch, taking one auxiliary graph into the other.

We are going to apply Theorem 74 to prove that the RSO Markov chain is rapidly mixing on the balanced JDM realizations. We partition its state space according to the vectors x of the auxiliary graph collections (see Definition 81 and its explanations). The following result will be used to prove that all

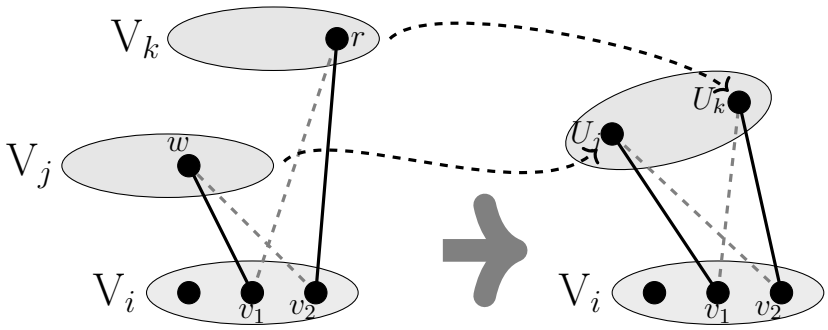


Figure 5.4: Construction of the auxiliary bipartite graph $\mathcal{G}_i$ and a RSO $\{(v_1, w), (v_2, r)\} \mapsto \{(v_1, r), (v_2, w)\}$ taking (x_1, y_1) into (x_2, y_2).

derived (marginal) Markov chains M_x are rapidly mixing. Next, we announce two theorems that are direct extensions of statements for fast mixing switch Markov chains for regular degree sequences (Cooper, Dyer and Greenhill [22]) and for half-regular bipartite degree sequences (Erdős, Kiss, Miklós and Soukup [108]).

Theorem 83. *The switch Markov chain on the realizations of almost regular degree sequences is rapidly mixing.*

Theorem 84. *The switch Markov chain on the realizations of almost half-regular bipartite degree sequences is rapidly mixing.*

We are now ready to prove the main theorem.

Proof. (Theorem 78) We show that the RSO Markov chain on balanced realizations fulfills the conditions in Theorem 74. First we show that condition (i) of Theorem 74 holds. When restricted to the partition Y_x (that is with x fixed), the RSO Markov chain over the balanced realizations walks on the union of almost semi-regular and almost regular graphs. By restriction here we mean that all probabilities which would (in the original chain) leave Y_x are put onto the shelf-loop probabilities. Since an RSO changes only one coordinate at a time, independently of other coordinates, all the conditions in Theorem 62 are fulfilled. Thus the relaxation time of the RSO Markov chain restricted onto Y_x is bounded from above by the relaxation time of the chain restricted onto that coordinate (either an almost semi-regular bipartite

or an almost regular graph) on which this restricted chain is the slowest (the smallest gap). However, based on Theorems 83 and 84, all these restrictions are fast mixing, and thus by Theorem 62 the polynomial bound in (i) of Theorem 74 holds. (Here $K = \frac{k(k+1)}{2}$, see Definition 81 and note that an almost semi-regular bipartite graph is also an almost half-regular bipartite graph.)

Next we show that condition (ii) of Theorem 74 also holds. The first coordinate is the union of auxiliary bipartite graphs, all of which are half-regular. The M' Markov chain corresponding to Theorem 74 is the switch Markov chain on these auxiliary graphs. Here each possible switch has a probability

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

and by Theorem 82 it is guaranteed that condition 5.3 is fulfilled. Since, again all conditions of Theorem 62 are fulfilled (mixing is fast within any coordinate due to Theorems 83 and 84), the M' Markov chain is also fast mixing. The condition in Equation (5.3) holds due to Theorem 82. Since all conditions in Theorem 74 hold, the RSO switch Markov chain on balanced realizations is also rapidly mixing. □

Chapter 6

Approximating the number of Double Cut-and-Join scenarios

In this chapter, we give an FPAUS algorithm to almost uniformly sample most parsimonious DCJ sorting scenarios transforming one genome into another. We obtain it in three steps. First, we decompose the problem into an "easy" and "hard" part (Lemma 86), and we show that this hard part has the same computational complexity than the whole problem (Theorem 86). In the second step, we show that an FPAUS can be achieved via sampling matchings of a complete bipartite graph following a certain distribution (Theorem 94). In the third step, we prove the rapid mixing of a Markov chain converging to this certain distribution of matchings (Theorem 99). The proving technique is the canonical path method by Sinclair [82], and is based on upper and lower bounds on the number of certain types of most parsimonious DCJ sorting scenarios (Theorem 93). We use these estimations to give upper bound on the so-called Poincaré coefficient defined in Equation 1.23 using Lemma 98.

Since counting the most parsimonious DCJ sorting scenarios is a self-reducible counting problem, the number of most parsimonious DCJ scenarios can also be approximated by an FPRAS algorithm.

This was a joint work with Eric Tannier. Eric Tannier proposed the work, and initially we worked on the easy part, following the results of Braga and Stoye [14]. Surprisingly, proving rapid mixing for the easy part seems to be harder, and it is still an open problem. The author of this thesis proved Lemma 98, which is the key lemma to prove the rapid mixing of the Markov chain as well as Theorem 93 that is used in the key lemma. The original

work has been published in Theoretical Computer Science, 2012, 439:30-40, doi: 10.1016/j.tcs.2012.03.006.

6.1 Preliminaries

First, we define the computational problem.

Definition 85. *The* Most Parsimonious DCJ sorting scenario *(*MPDCJ*) problem for two genomes G_1 and G_2 is to compute $d_{\mathrm{DCJ}}(G_1, G_2)$, that is, the minimum number of DCJ operations necessary to transform G_1 into G_2. The* #MPDCJ *problem asks for the number of scenarios of length $d_{\mathrm{DCJ}}(G_1, G_2)$, denoted by* #MPDCJ$(G_1, G_2)$.

MPDCJ is an optimization problem, which has a natural corresponding decision problem asking if there is a scenario with a given number of DCJ operations. So we may write that #MPDCJ $\in$ #P, which means that #MPDCJ asks for the number of witnesses of the decision problem "Is there a scenario for G_1 and G_2 of size $d_{\mathrm{DCJ}}(G_1, G_2)$?".

A DCJ operation on a genome G_1 which decreases the DCJ distance to a genome G_2 is called a *sorting* DCJ for G_1 and G_2. It is possible to characterize the effect of a sorting DCJ on the adjacency graph of genomes G_1 and G_2 (see subsection 1.2.2 for the adjacency graph and its components). It acts on the vertex set V_1 and has one of the following effects ([14]):

- splitting a cycle into two cycles,
- splitting an odd path into a cycle and an odd path,
- splitting an M-shaped path into a cycle and an M-shaped path,
- splitting an M-shaped path into two odd paths,
- splitting a W-shaped path into a cycle and a W-shaped path,
- merging the two ends of a W-shaped path, thus transforming it into a cycle,
- combining an M-shaped and a W-shaped path into two odd paths.

Note that trivial components are never affected by these operations, and all but the last type of DCJ operations act on a single component of the adjacency graph. The last type of DCJ acts on two components, which are M and W-shaped paths.

In this context, *sorting* a component involving the set of adjacencies and telomers in G_2, denoted by A, means applying a sequence of sorting DCJ operations to vertices of this component in G_1 so that the resulting adjacency graph has only trivial components involving the adjacencies and telomers of A. In a minimum length DCJ scenario, every component is sorted independently, except M and W-shaped paths, which can be sorted together. If in a DCJ scenario one operation acts on both an M and a W-shaped path, we say that they are sorted *jointly*; otherwise we say that they are sorted *independently*.

6.2 Decomposing the #MPDCJ problem

The complexity status of #MPDCJ in general is not known. It is solvable in polynomial time when the genomes are co-tailed [14, 70], or more generally in the absence of M and W-shaped paths. Therefore the hard part seems to be dealing with M and W-shaped paths. We show here that for the general case, we may restrict ourselves to this hard part, and suppose that there are only M and W-shaped paths in the adjacency graph.

Given two genomes G_1 and G_2 with the same label set, let AG be the adjacency graph of G_1 and G_2. Denote by G_1^* the genome which has the adjacencies and telomeres of G_1 whenever they are in an M or W-shaped paths of AG, and those of G_2 when they are in another component of AG. By definition the adjacency graph between G_1 and G_1^* has no M and W-shaped paths, while the adjacency graph between G_1^* and G_2 has only trivial components and M and W-shaped paths.

Lemma 86. $d_{\mathrm{DCJ}}(G_1, G_2) = d_{\mathrm{DCJ}}(G_1, G_1^*) + d_{\mathrm{DCJ}}(G_1^*, G_2)$.

Proof. Recall the characterization of the effect of DCJ operations on the adjacency graph implies that in a minimum length DCJ scenario between G_1 and G_2, a DCJ operation never acts on two vertices involved in different components of AG, except if these two components are M and W-shaped paths. This implies that the DCJ operations of a minimum length scenario

are of two kinds: those which act on the vertices involved in M and W-shaped paths, and the others.

The subsequence of DCJ operations of the first kind transforms G_1 into G_1^*, and the complementary subsequence transforms G_1^* into G_2. This proves the lemma. □

Definition 87. *The* #MPDCJ$_{MW}$ problem *asks for the number of* DCJ *scenarios between two genomes when their adjacency graph contains only trivial components and M and W-shaped paths.*

The correspondence between solutions for #MPDCJ$_{MW}$ and #MPDCJ is stated by the following lemma.

Lemma 88.

$$\begin{aligned}\#\text{MPDCJ}(G_1, G_2) &= \frac{d_{\text{DCJ}}(G_1, G_2)!}{d_{\text{DCJ}}(G_1^*, G_2)! \prod_i (c_i - 1)! \prod_j (l_j - 1)!} \\ &\times \prod_i c_i^{c_i - 2} \prod_j l_j^{l_j - 2} \times \#\text{MPDCJ}_{MW}(G_1^*, G_2) \quad (6.1)\end{aligned}$$

where i indexes the cycles of the adjacency graph of G_1 and G_2, c_i denotes the number of vertices in vertex set V_1 belonging to the ith cycle, j indexes the odd paths of the adjacency graph, l_j is the number of vertices in vertex set V_1 belonging to the jth odd path.

Proof. As M and W-shaped paths and other components are always treated independently, we have

$$\begin{aligned}\#\text{MPDCJ}(G_1, G_2) &= \binom{d_{\text{DCJ}}(G_1, G_2)}{d_{\text{DCJ}}(G_1^*, G_2)} \\ &\times \#\text{MPDCJ}(G_1, G_1^*) \times \#\text{MPDCJ}_{MW}(G_1^*, G_2)\end{aligned}$$

For the genomes G_1 and G_1^*, whose adjacency graph do not contain M and W-shaped paths, we have from [14] and [70] that

$$\#\text{MPDCJ}(G_1, G_1^*) = \prod_i c_i^{c_i - 2} \prod_j l_j^{l_j - 2} \times \frac{d_{DCJ}(G_1, G_1^*)!}{\prod_i (c_i - 1)! \prod_j (l_j - 1)!}.$$

These two equations together with Lemma 86 give the result. □

The following theorem says that the hardness of the #MPDCJ problem is the same as the $\#\text{MPDCJ}_{MW}$ problem.

Theorem 89.

$$\#\text{MPDCJ}_{MW} \in \text{FP} \iff \#\text{MPDCJ} \in \text{FP} \quad (6.2)$$

$$\#\text{MPDCJ}_{MW} \in \#\text{P-}\textit{complete} \iff \#\text{MPDCJ} \in \#\text{P-}\textit{complete} \quad (6.3)$$

$$\#\text{MPDCJ}_{MW} \in \text{FPRAS} \iff \#\text{MPDCJ} \in \text{FPRAS} \quad (6.4)$$

$$\#\text{MPDCJ}_{MW} \in \text{FPAUS} \iff \#\text{MPDCJ} \in \text{FPAUS} \quad (6.5)$$

Proof. Both the multinomial factor and the two products in Equation 6.1. can be calculated in polynomial time. Thus the transformation between the solutions to the two different counting problems is a single multiplication or division by an exactly calculated number. This proves that $\#\text{MPDCJ}_{MW}$ is in FP if and only if #MPDCJ is in FP, as well as $\#\text{MPDCJ}_{MW}$ is in #P-complete if and only if #MPDCJ is in #P-complete.

Such a multiplication and division keeps the relative error when the solution of one of the problems is approximated. This proves that $\#\text{MPDCJ}_{MW}$ is in FPRAS if and only if #MPDCJ is in FPRAS.

Concerning the last equivalence, the $\Leftarrow$ part is trivial because $\#\text{MPDCJ}_{MW}$ is a particular case of #MPDCJ. Now we prove that $\#\text{MPDCJ}_{MW} \in \text{FPAUS} \Rightarrow \#\text{MPDCJ} \in \text{FPAUS}$. Suppose an FPAUS exists for $\#\text{MPDCJ}_{MW}$, and let G_1 and G_2 be two arbitrary genomes. The following algorithm gives a FPAUS for #MPDCJ.

- Draw a DCJ scenario between G_1^* and G_2 following a distribution p satisfying

  $$d_{TV}(p, U) \leq \epsilon$$

  where U is the uniform distribution over all possible most parsimonious DCJ scenarios between G_1^* and G_2.

- Generate a DCJ scenario between G_1 and G_1^*, following the uniform distribution. This scenario can be sampled sharply uniformly in polynomial time: (1) there are only cycles and odd paths in the adjacency graph of G_1 and G_1^*, so the number of scenarios can be calculated in polynomial time; (2) there is a polynomial number of sorting DCJ steps on each component, and a sorting DCJ operation results in an adjacency graph that also only has cycles and odd paths.

- Draw a sequence of 0s and 1s, containing $d_{\mathrm{DCJ}}(G_1^*, G_2)$ 1s and $d_{\mathrm{DCJ}}(G_1, G_1^*)$ 0s, uniformly from all $\binom{d_{\mathrm{DCJ}}(G_1,G_2)}{d_{\mathrm{DCJ}}(G_1^*,G_2)}$ such sequences.
- Merge the two paths contructed at the two first steps, according to the drawn sequence of 0s and 1s.

Note that the DCJ scenario obtained transforms G_1 into G_2. Let us denote the distribution of paths generated by this algorithm by p', and the uniform distribution over all possible DCJ scenarios between G_1 and G_2 by U'. Let X_s denote the set of all possible scenarios drawn by the above algorithm using a specific scenario s between G_1^* and G_2. Then

$$\sum_{s' \in X_s} |p'(s') - U'(s')| = |p(s) - U(s)| \tag{6.6}$$

Using Equation 6.6 we get that

$$d_{TV}(p', U') = \frac{1}{2} \sum_{s} \sum_{s' \in X_s} |p'(s') - U'(s')| = \frac{1}{2} \sum_{s} |p(s) - U(s)| = d_{TV}(p, U) \tag{6.7}$$

This proves that the above algorithm is an FPAUS for #MPDCJ, proving the left-to-right direction in Equation 6.5. □

We will show that $\#\mathrm{MPDCJ}_{MW}$ is in FPAUS, thus #MPDCJ is in FPAUS. As MPDCJ is a self-reducible problem (there is a polynomial time reduction of the decision problem to the search problem), the FPAUS implies the existence of an FPRAS [54]. The FPAUS algorithm for $\#\mathrm{MPDCJ}_{MW}$ will be defined via a rapidly mixing Markov chain. And first, we have to recall or prove some properties on the number of independent and joint sortings of M and W-shaped paths.

6.3 Independent and joint sorting of M and W-shaped paths

Our goal is to show that the number of DCJ scenarios in which an M and a W-shaped path are sorted independently is a significant fraction of the total number of scenarios sorting these M and W-shaped paths (independently or jointly). We build on the following results by [14].

Theorem 90. *[14]*

- *The number of minimum length* DCJ *scenarios sorting a cycle with $k > 1$ vertices in G_1 is k^{k-2}.*
- *The number of minimum length* DCJ *scenarios sorting an odd path with $k > 1$ vertices in G_1 is k^{k-2}.*
- *The number of minimum length* DCJ *scenarios sorting a W-shaped path with $k > 1$ vertices in G_1 is k^{k-2}.*
- *The number of minimum length* DCJ *scenarios sorting an M-shaped path with $k > 0$ vertices in G_1 is $(k+1)^{k-1}$.*

Theorem 91. *The number of* DCJ *scenarios that independently sort a W and an M-shaped path is $\binom{k_1+k_2-1}{k_1-1} k_1^{k_1-2}(k_2+1)^{k_2-1}$ where k_1 and k_2 are the number of vertices of G_1 in the W and M-shaped paths, respectively.*

Proof. It is a consequence of the previous theorem. The W-shaped path is sorted in k_1-1 operations, and the M-shaped path is sorted in k_2 operations. Thus there are $\binom{k_1+k_2-1}{k_1-1}$ ways to merge two scenarios. □

Theorem 92. *The number of* DCJ *scenarios that jointly sort a W and an M-shaped path is less than $2(k_1+k_2)^{k_1+k_2-2}$, where k_1 and k_2 are the number of vertices of G_1 in the W and M-shaped paths, respectively.*

Proof. Let t_1^W and t_2^W be the two telomeres of the W-shaped path, and t_1^M and t_2^M be the two telomeres of the M-shaped path. Let G_1' and G_2' be constructed from genomes G_1 and G_2 by adding a gene g, with extremities g^h and g^t, and replacing the telomeres of G_1 by adjacencies $(t_1^W, g^t), (t_2^W, g^h)$ and the telomeres of G_2 by adjacencies $(t_1^M, g^t), (t_2^M, g^h)$. In addition let G_1'' and G_2'' be constructed from G_1 and G_2 also by adding a gene g, and replacing the telomeres of G_1 by adjacencies $(t_1^W, g^t), (t_2^W, g^h)$ and the telomeres of G_2 by adjacencies $(t_1^M, g^h), (t_2^M, g^t)$. In both cases the M and W-shaped paths in G_1 and G_2 are transformed into a cycle with $k_1 + k_2$ adjacencies in both genomes. Call the cycles C' for the first case, and C'' for the second.

We prove that any scenario that jointly sort the W and M-shaped paths has a corresponding distinct scenario either sorting the cycle C' or sorting the cycle C''. This proves the theorem, because there are $(k_1 + k_2)^{k_1+k_2-2}$ scenarios sorting each cycle.

A scenario jointly sorting the M and W-shaped paths can be cut into two parts: the first contains DCJ operations which act only on the M or only on the W-shaped path; the second part starts with a DCJ operation transforming an M and a W-shaped path into two odd paths, and continues with operations independently sorting the two odd paths.

In the first part, either a DCJ operation acts on two adjacencies of the M or W-shaped path, and the corresponding operation acts on the same two adjacencies on C' or C'', or it acts on an adjacency and a telomere of the W-shaped path, and the corresponding operation acts on two adjacencies of C' or C'', one of them containing an extremity of g. So there is a correspondance between being a telomere in the W-shaped path, and being adjacent to an extremity of g in C' or C''.

Now the corresponding operation of the DCJ transforming the two paths into two odd paths has to create two cycles from C' or C''. Choose C' or C'' so that it is the case. Now sorting an odd path exactly corresponds to sorting a cycle, by replacing between being a telomere in the path by being adjacent to an extremity of g in the cycle.

So two different scenarios jointly sorting the M and W-shaped paths correspond to two different scenarios sorting either C' or C''. Then the number of scenarios jointly sorting the M and W-shaped paths is less than $2(k_1 + k_2)^{k_1+k_2-2}$.

□

Theorem 93. *Let $T(k_1, k_2)$ denote the number of* DCJ *scenarios jointly sorting a W and an M-shaped path with respectively k_1 and k_2 vertices G_1. Let $I(k_1, k_2)$ denote the number scenarios independently sorting the same paths. We have that*

$$\frac{T(k_1, k_2)}{I(k_1, k_2)} = O\left(\frac{k_1^{1.5} k_2^{1.5}}{(k_1 + k_2)^{1.5}}\right) \tag{6.8}$$

$$\frac{I(k_1, k_2)}{T(k_1, k_2)} = O\left(k_1 + k_2\right) \tag{6.9}$$

Proof. To prove Equation 6.8 it is sufficient to show that

$$\frac{2(k_1 + k_2)^{k_1+k_2-2}}{\binom{k_1+k_2-1}{k_1-1} k_1^{k_1-2} (k_2 + 1)^{k_2-1}} = O\left(\frac{k_1^{1.5} k_2^{1.5}}{(k_1 + k_2)^{0.5}}\right) \tag{6.10}$$

Using Stirling's formula, we get on the left hand side of Equation 6.10

$$\frac{2\sqrt{2\pi(k_1-1)}\left(\frac{k_1-1}{e}\right)^{k_1-1}\sqrt{2\pi(k_2)}\left(\frac{k_2}{e}\right)^{k_2}(k_1+k_2)^{k_1+k_2-2}}{\sqrt{2\pi(k_1+k_2-1)}\left(\frac{k_1+k_2-1}{e}\right)^{k_1+k_2-1}k_1^{k_1-2}(k_2+1)^{k_2-1}} \tag{6.11}$$

After simplifications and algebraic rearrangement, we get

$$2\sqrt{\frac{2\pi(k_1-1)k_2}{k_1+k_2-1}}\left(\frac{k_1+k_2}{k_1+k_2-1}\right)^{k_1+k_2-1}\left(\frac{k_1-1}{k_1}\right)^{k_1-1}\left(\frac{k_2}{k_2+1}\right)^{k_2}\left(\frac{k_1(k_2+1)}{k_1+k_2}\right) \tag{6.12}$$

from which Equation 6.10 follows with applying $(1+1/n)^n$ tends to e, and $(1-1/n)^n$ tends to $1/e$.

To prove Equation 6.9 consider the subset of DCJ scenarios jointly sorting the W and M-shaped paths, and starting with a DCJ operation which acts on a telomere of the W-shaped path, and on an adjacency which is link with a telomere of the M-shaped path. The result is two odd paths with respectively k_1 and k_2 adjacencies and telomeres in G_1. They can be sorted in respectively k_1-1 and k_2-1 steps, in $k_1^{k_1-2}$ and $k_2^{k_2-2}$ different ways. Since we can combine any two particular solutions in $\binom{k_1+k_2-2}{k_1-1}$ ways, $\frac{I(k_1,k_2)}{T(k_1,k_2)}$ is bounded by

$$\frac{\binom{k_1+k_2-1}{k_1-1}k_1^{k_1-2}(k_2+1)^{k_2-1}}{\binom{k_1+k_2-2}{k_1-1}k_1^{k_1-2}k_2^{k_2-2}}. \tag{6.13}$$

After minor algebraic simplification, this expression is equal to

$$\frac{k_1+k_2+1}{k_2}\left(1+\frac{1}{k_2}\right)^{k_2-1}k_2, \tag{6.14}$$

which is clearly $O(k_1+k_2)$. □

6.4 The Markov chain on DCJ scenarios

Assume that there are n W-shaped paths and m M-shaped paths, and consider the complete bipartite graph $K_{n,m}$. Let $\mathcal{M}$ be a matching of $K_{n,m}$, which might range from the empty graph up to any maximum matching. A DCJ scenario is said to be $\mathcal{M}$-compatible when an M-shaped and a W-shaped path are sorted jointly if and only if they are connected by an edge of $\mathcal{M}$.

We denote by $\{P_i\}_i$ set of degree 0 vertices in $\mathcal{M}$, and by $\{M_iW_i\}_i$ the set of edges in $\mathcal{M}$. Let $l(P_i)$ be the minimum length of a DCJ scenario independently sorting P_i, and $l(M_iW_i)$ be the minimum length of a DCJ scenario jointly sorting M_i and W_i. We can calculate $N(M_i, W_i)$, the number of joint sortings of M_i and W_i, in polynomial time [14]. Denote by $N(P_i)$ the number of independent sortings of a path P_i. The number of $\mathcal{M}$-compatible scenarios is

$$f(\mathcal{M}) = \left(\frac{(\sum_i l(M_iW_i) + \sum_i l(P_i))!}{l(M_i, W_i)!, \ldots, l(P_i)!} \right) \Pi_i N(M_i, W_i) \Pi_i N(P_i),$$

and we can compute it in polynomial time. Define a distribution θ over the set of all matchings of the complete bipartite graph $K_{n,m}$ as

$$\theta(\mathcal{M}) \propto f(\mathcal{M}) \tag{6.15}$$

We first show that sampling DCJ scenarios from the uniform distribution is equivalent to sampling matchings of $K_{n,m}$ from the distribution θ.

Theorem 94. *Let a distribution q over the scenarios of n W-shaped paths and m M-shaped paths be defined by the following algorithm.*

- *Draw a random matching $\mathcal{M}$ of $K_{n,m}$ following a distribution p.*
- *Draw a random $\mathcal{M}$-compatible* DCJ *scenario from the uniform distribution of all $\mathcal{M}$-compatible ones.*

Then

$$d_{TV}(p, \theta) = d_{TV}(q, U) \tag{6.16}$$

where θ is the distribution defined in Equation 6.15, and U denotes the uniform distribution over all DCJ *scenarios.*

Proof.

$$d_{TV}(q, U) = \frac{1}{2} \sum_{x \ scenario} |q(x) - U(x)|$$

We may decompose this sum into

$$\frac{1}{2} \sum_{(\mathcal{M} \ matching \ of \ K_{n,m})} \sum_{(x \ \mathcal{M}-compatible \ scenario)} |q(x) - U(x)|$$

$\sum_{(x\ \mathcal{M}-compatible\ scenario)} q(x)$ is $p(\mathcal{M})$ since x is drawn uniformly among the scenarios compatible with $\mathcal{M}$, and $\sum_{(x\ \mathcal{M}-compatible\ scenario)} U(x)$ is $\theta(\mathcal{M})$. Furthermore, both $q(x)$ and $U(x)$ are constant for a particular matching $\mathcal{M}$, thus

$$\frac{1}{2} \sum_{(\mathcal{M}\ matching\ of\ K_{n,m})} \sum_{(x\ \mathcal{M}-compatible\ scenario)} |q(x) - U(x)| \quad = \quad \frac{1}{2} \sum_{(\mathcal{M}\ matching\ of\ K_{n,m})} |p(\mathcal{M}) - \theta(\mathcal{M})| = d_{TV}(p, \theta) \tag{6.17}$$

yielding the result. □

So we are going to define an MCMC on matchings of $K_{n,m}$ converging to θ. The rapid convergence of this MCMC will imply that $\#\text{MPDCJ}_{WM}$ admits an FPAUS, and hence $\#\text{MPDCJ} \in \text{FPAUS}$, and then $\#\text{MPDCJ} \in \text{FPRAS}$. The primer Markov chain walks on the matchings of $K_{n,m}$ and is defined by the following steps: suppose the current state is a matching $\mathcal{M}$, and

- with probability 1/2, the next state of the Markov chain is the current state $\mathcal{M}$;
- with probability 1/2, draw a random $i \sim U[1, n]$ and $j \sim U[1, m]$; if $ij \in \mathcal{M}$, then remove ij from $\mathcal{M}$; else if $\deg_{\mathcal{M}}(i) = 0$ and $\deg_{\mathcal{M}}(j) = 0$, then add ij to $\mathcal{M}$.

It is easy to see that this Markov chain is irreducible and aperiodic. We apply the standard Metropolis-Hastings algorithm on this chain [66], namely, when we are in state $\mathcal{M}$, we propose the next state $\mathcal{M}_{new}$ according to the primer Markov chain, and accept the proposal with probability

$$\min\left\{1, \frac{f(\mathcal{M}_{new})}{f(\mathcal{M})}\right\} \tag{6.18}$$

The obtained Markov chain is reversible and converges to the distribution θ defined in Equation 6.15. Furthermore, this is a lazy Markov chain (due to remaining in the current state with at least probability 1/2 in each step), providing that all its eigenvalues are positive real numbers (see also Theorem 26).

An important property of this Markov chain is that

Observation 95. *The non-zero transition probabilities as well as their inverses are polynomially bounded.*

Indeed, the transition probability from $\mathcal{M}$ to $\mathcal{M}_{new}$, if non zero, is at least

$$\frac{1}{2 \times n \times m} \frac{f(\mathcal{M}_{new})}{f(\mathcal{M})}.$$

$\mathcal{M}$ and $\mathcal{M}_{new}$ vary by at most one edge $M_i W_i$, and on this edge, according to Theorem 93, the ratio of number of scenarios jointly and independently sorting M_i and W_i is polynomial. Furthermore, the combinatorial factors appearing in $f(\mathcal{M})$ and $f(\mathcal{M}_{new})$ due to merging the sorting steps on different components are the same. So $\frac{f(\mathcal{M}_{new})}{f(\mathcal{M})}$ as well as its inverse are polynomially bounded.

We now prove the rapid convergence of this Markov chain using the multicommodity flow technique [82], see also Section 1.1.7.

6.5 Fast convergence of the MCMC

In this section, we prove that the constructed Markov chain rapidly converges to its stationary distribution. From its construction, this distribution is θ as defined in Equation 6.15.

To prove that the Markov chain we defined on bipartite matchings has a polynomial relaxiation time, we need to construct a path system Γ on the set of matchings of $K_{n,m}$, such that κ_Γ is bounded by a polynomial in N, the number of markers in G_1 and G_2.

In our case the path system between two matchings $\mathcal{X}$ and $\mathcal{Y}$ is a unique path with probability 1. Here is how we construct it.

Fix a total order on the vertex set of $K_{n,m}$. Take the symmetric difference of $\mathcal{X}$ and $\mathcal{Y}$, denoted by $\mathcal{X}\Delta\mathcal{Y}$. It is a set of disjoint paths and cycles. Define an order on the components of $\mathcal{X}\Delta\mathcal{Y}$, such that a component C is smaller than a component D if the smallest vertex in C is smaller than the smallest vertex in D. Now we orient each component in the following way: the beginning of each path is its extremity with the smaller vertex. The starting vertex of a cycle is its smallest vertex, and the direction is going towards its smaller neighbour.

We transform $\mathcal{X}$ to $\mathcal{Y}$ by visiting the components of $\mathcal{X}\Delta\mathcal{Y}$ in increasing order. Let the current component be C, and the current matching is $\mathcal{Z}$ (at

first $\mathcal{Z} = \mathcal{X}$). If C is a path or cycle starting with an edge in $\mathcal{X}$, then the transformation steps are the following: delete the first edge of C from $\mathcal{Z}$, delete the third edge of C from $\mathcal{Z}$, add the second edge of C to $\mathcal{Z}$, delete the 5th edge of C from $\mathcal{Z}$, add the 4th edge of C to $\mathcal{Z}$, etc.

If C is a path or cycle starting with an edge in $\mathcal{Y}$, then the transformation steps are the following: delete the second edge of C from $\mathcal{Z}$, add the first edge of C to $\mathcal{Z}$, delete the 4th edge of C from $\mathcal{Z}$, add the third edge of C to $\mathcal{Z}$, etc.

This path has length at most nm, and κ_Γ can be written:

$$\kappa_\Gamma \leq nm \max_{e=(u,v)\in E} \sum_{(x,y)\in V\times V: e\in\Gamma_{x,y}} \frac{\theta(x)\theta(y)}{Q(e)}.$$

By Observation 95, the inverse of the transition probabilities is bounded by a polynomial in N, so we get

$$\kappa_\Gamma \leq O(\text{poly}(N)) \max_{e=(u,v)\in E} \sum_{(x,y)\in V\times V: e\in\Gamma_{x,y}} \frac{\theta(x)\theta(y)}{\theta(u)}. \tag{6.19}$$

We then have to show that $\sum \frac{\theta(x)\theta(y)}{\theta(u)}$ can be bounded by a polynomial in N. Let $\mathcal{Z} \to \mathcal{Z}'$ be an edge on the path from $\mathcal{X}$ to $\mathcal{Y}$. We define

$$\widehat{\mathcal{M}} := \mathcal{X}\Delta\mathcal{Y}\Delta\mathcal{Z} \tag{6.20}$$

Lemma 96. *The couple $\widehat{\mathcal{M}}$ and $\mathcal{Z} \to \mathcal{Z}'$ determines $\mathcal{X}$ and $\mathcal{Y}$.*

Proof. It is obvious that

$$\widehat{\mathcal{M}}\Delta\mathcal{Z} = \mathcal{X}\Delta\mathcal{Y} \tag{6.21}$$

hence, $\mathcal{Z}$ and $\widehat{\mathcal{M}}$ determine the symmetric difference of $\mathcal{X}$ and $\mathcal{Y}$. From the transition $\mathcal{Z} \to \mathcal{Z}'$, we can trace back which transition steps have been already made in the following way. The order of the components of $\mathcal{X}\Delta\mathcal{Y}$ is determined, and from the transition $\mathcal{Z} \to \mathcal{Z}'$ we know the current component. We also know the beginning and the direction of the component, be it either a path or a cycle, hence, we know which edges have been changed in the component so far, and which ones not yet. From these, we can reconstruct $\mathcal{X}$ and $\mathcal{Y}$. □

Lemma 97. *A matching can be obtained from $\widehat{\mathcal{M}}$ by deleting at most two edges.*

Proof. On each component in $\mathcal{X}\Delta\mathcal{Y}$, we delete at most two edges before putting back one. Hence $\widehat{\mathcal{M}}$ contains at most either 4 consecutive edges along a path or 2 pair of edges, and all remaining edges are independent. Therefore it is sufficient to delete at most two edges from $\widehat{\mathcal{M}}$ to get a matching. □

Denote this matching by $\widetilde{\mathcal{M}}$.

Lemma 98.

$$\frac{\theta(\mathcal{X})\theta(\mathcal{Y})}{\theta(\mathcal{Z})} = O(\text{poly}(N))\theta(\widetilde{\mathcal{M}}) \tag{6.22}$$

Proof. We prove that

$$\frac{f(\mathcal{X})f(\mathcal{Y})}{f(\mathcal{Z})f(\widetilde{\mathcal{M}})} = O(\text{poly}(N)) \tag{6.23}$$

It proves the lemma, as $\theta(\cdot)$ and $f(\cdot)$ differ only by a normalizing constant. $\widetilde{\mathcal{M}}\Delta\mathcal{Z}$ differs at most in two edges from $\mathcal{X}\Delta\mathcal{Y}$. These edges appear in $\mathcal{X}\Delta\mathcal{Y}$, but not in $\widetilde{\mathcal{M}}\Delta\mathcal{Z}$. The two vertices of any missing edges correspond to components which are independently sorted either in $\mathcal{Z}$ or $\widetilde{\mathcal{M}}$, but jointly in either $\mathcal{X}$ or $\mathcal{Y}$. Amongst these two vertices, one of them correspond to a W-shaped component $\mathcal{A}$, the other to an M-shaped component $\mathcal{B}$. Let k_1 be the number of adjacencies and telomeres of G_1 in $\mathcal{A}$, and k_2 the number of adjacencies and telomeres of G_1 in $\mathcal{B}$. The ratio on the left-hand side of Equation 6.23 due to such difference is

$$\frac{T(k_1,k_2)}{(k_1+k_2+1)!} \Big/ \frac{I(k_1)I'(k_2)}{k_1!(k_2+1)!} \tag{6.24}$$

where $I(x)$ denotes the independent sorting of a W-shaped component of size x, and $I'(x)$ denotes the independent sorting of an M-shaped component of size x. However, it is polynomially bounded, since

$$\frac{I(k_1)I'(k_2)}{\binom{k_1+k_2+1}{k_1}} = I(k_1,k_2) \tag{6.25}$$

and we can apply Theorem 93. □

These results together lead to the following theorem:

Theorem 99. *The Metropolis-Hastings Markov chain on the matchings defined above converges rapidly to θ.*

Proof. From Lemma 98, Equation 6.19 may be written

$$\kappa_\Gamma \leq O(\text{poly}(N)) \max_{e=(u,v)\in E} \sum_{(x,y)\in V\times V : e\in\Gamma_{x,y}} \theta(\widetilde{\mathcal{M}}).$$

By Lemmas 96 and 97, a matching $\widetilde{\mathcal{M}}$ may appear only a polynomial number of times in this sum. So

$$\kappa_\Gamma \leq O(\text{poly}(N)) \sum_{\widetilde{\mathcal{M}}} \theta(\widetilde{\mathcal{M}}),$$

and as $\sum_{\widetilde{\mathcal{M}}} \theta(\widetilde{\mathcal{M}}) = 1$, κ_Γ is bounded by a polynomial in N. This proves the theorem. □

Using this result, we can prove the following theorem

Theorem 100. $\#\text{MPDCJ}_{MW} \in \text{FPAUS}$

Proof. The above defined Markov chain on partial matchings is an aperiodic, irreducible and reversible Markov chain, with only positive eigenvalues. Furthermore, a step can be performed in running time polymonial with the size of the graph. For any start state i, $\log(1/\theta(i))$ is polynomially bounded with the size of the corresponding genomes G_1^* and G_2, since there are $O(N^2)$ DCJ operations, the length of the DCJ paths is less than N, thus the number of sorting DCJ paths are $O(N^{2N})$, and the inverse of the probability of any partial matching is less than this. Thus, the relaxation time is polynomial in both N and $\log(1/\epsilon)$, according to Theorem 25. This means that in fully polynomial running time (polynomial both in N and $-\log(\epsilon)$) a random partial matching can be generated from a distribution p satisfying

$$d_{TV}(p,\theta) \leq \epsilon \tag{6.26}$$

But then a random DCJ path can be generated in fully polynomial running time following a distribution q satisfying

$$d_{TV}(q,U) \leq \epsilon \tag{6.27}$$

according to Theorem 94. This is what we wanted to prove. □

Now we are ready to conclude by our main theorem:

Theorem 101. $\#\text{MPDCJ} \in \text{FPRAS}$

Proof. $\#\text{MPDCJ}_{MW} \in \text{FPAUS}$ according to Theorem 100. Then $\#\text{MPDCJ} \in \text{FPAUS}$ according to Theorem 89. Since #MPDCJ is a self-reducible counting problem, it is in FPRAS [54]. □

6.6 Conclusion

Sampling from reversal scenarios has been conjectured to be #P-complete [124], but almost all counting problems on genome rearrangement scenarios have an open complexity status. We conjecture that sampling from DCJ scenarios is also #P-complete, and we proved in this chapter that it admits an FPRAS. Braga and Stoye [14] proved that altering three consecutive steps in a DCJ sorting path is sufficient to get an irreducible Markov chain. Such a Markov chain can be also used in a Metropolis-Hastings algorithm to converge to the uniform distribution of all DCJ sorting scenarios. We conjecture that it is also a rapidly mixing Markov chain, which would give a more direct proof of our results. Interestingly, proving the rapid mixing of this chain seems to be a technically very hard task even for genomes whose adjacency graph does not contain any M or W-shaped path. This is surprising given that for these cases exact enumeration results are known.

Part III

Negative results: torpid mixing, #P-complete and non-approximable problems

dc_2046_22

Chapter 7

The Metropolized Partial Importance Sampling MCMC mixes slowly on minimum reversal rearrangement paths

Markov chain Monte Carlo has been the standard technique for inferring the posterior distribution of genome rearrangement scenarios under a Bayesian approach. We present here a negative result on the rate of convergence of a generally used Markov chain in genome rearrangement. We prove that the relaxation time of the Markov chains walking on the most parsimonious reversal sorting scenarios might grow exponentially with the size of the signed permutations, namely, with the number of syntheny blocks.

This was a joint work with Bence Mélykuti and Krister Swenson. KS showed a permutation similar to the first part of the permutation on Figure 7.1. In a previous work with Timothy Brooks Paige, we found permutations whose reversal sorting solution space contained big gaps (see the text in this chapter for details), however, with a much more complicated structure of their solution space. The permuations we found with KS have a very simple solution space providing an easy proof of the main theorem presented in this work. BM contributed in the discussions and careful reading and correcting of this manuscript. The original manuscript has been published in IEEE/ACM Transactions in Computational Biology and Bioinformatics, 2010, 4(7):763-767, doi: 10.1109/TCBB.2009.26.

7.1 Partial Importance Sampling

Sometimes each point in the state space of a Markov chain Monte Carlo can be represented as a vector, and the primary Markov chain modifies the current state, x_t, by changing a subset (or window) of its coordinates, w. Let w' denote the newly drawn coordinates of y proposed from x_t. Then the Equation 1.12 in the Preliminaries becomes

$$\frac{\pi(y)T(x_t, w'|y)}{\pi(x_t)T(y, w|x_t)} \tag{7.1}$$

where $T(a, w|b)$ tells the probability of proposing a from b by choosing and modifying the coordinates w. When the newly drawn coordinates of y do not depend on the respective coordinates of x_t, the algorithm is called *Metropolized Partial Importance Sampling*.

In the case of genome rearrangements, the state space of MCMC is the set of allowed transition paths (or rearrangement scenarios) between two genomes, g_1 and g_2, typically restricted to shortest rearrangement scenarios. Such a state space can be considered as being comprised of $(d(g_1, g_2) + 1)$-tuples of genomes, where $d(g_1, g_2)$ is the distance between genomes g_1 and g_2. (g_1, intermediate genomes connecting g_1 to g_2, and g_2). A Metropolized Partial Importance Sampler cuts out a subpath from the current path, which is framed by genomes g_k and g_ℓ and draws a new subpath transforming g_k into g_ℓ. This subpath is drawn from a distribution that does not depend on the cut out subpath. In published implementations [28, 58, 59, 114, 118, 100], the new subpath is drawn step by step, drawing a new intermediate genome by considering the list of mutations that act on the current intermediate genome. If the allowed transition paths are the minimum reversal sorting paths, then the next intermediate genome is drawn by applying a random, uniformly distributed sorting reversal on the current intermediate genome. Formally, we can define this Markov chain in the following way.

Definition 102. *Let π be a signed permutation. The state space of the $M(\pi)$ Markov chain is the sortest reversal sorting paths of π, represented by the sequences of signed permutations, $\pi = \pi_0, \pi_1, \pi_2, \ldots, \pi_k = id$, where k is the reversal distance of π. Let p be an arbitrary, but fixed distribution of the substrings of $0, 1, 2 \ldots, k$ of length at least 3. The primary transitions of the Markov chain are defined by the following algorithm:*

1. *Select a substring of $0, 1, 2, \ldots, k$ following the distribution of p, $i, i + 1, \ldots, j$.*

2. *Let* $\sigma_0 := \pi_j^{-1}\pi_i$, *and let* $m = 0$.

3. *While* $\sigma_m \neq id$, *choose a sorting reversal of* σ_m, tr_m *following the uniform distribution of the sorting reversals of* σ_m. *Let* $\sigma_{m+1} = \sigma_m \circ tr_m$, *and increase* m *by* 1.

4. *The proposed new sorting path is*

$$\pi_0, \pi_1, \ldots, \pi_i, \pi_i \circ tr_0, \pi_i \circ tr_0 \circ tr_1, \ldots, \pi_i \circ tr_0 \circ tr_1 \circ \ldots \circ tr_{j-i} = p_j, p_{j+1}, \ldots, \pi_k.$$

5. *The Metropolis-Hastings algorithm is applied on the primary Markov chain so it converges to the uniform distribution of all shortest sorting paths.*

In the next section, we prove that this kind of MCMC mixes slowly in the worst case.

7.2 ParIS mixes slowly on minimum reversal paths

In this section, we prove the torpid mixing of the above defined Markov chain using the Cheeger inequality defined in Equation 1.28.

Theorem 103. *The Markov chain* $M(\pi)$ *defined in Definition 102 is torpidly mixing in the sense as defined in Definition 29 for any window distribution* p.

Proof. For each $n \in \mathbb{N}$ we construct a signed permutation of length $13n - 2$. Fig. 7.1. shows the general structure of the permutation from our example. Its graph of desire and reality can be split into two parts. The first part is a single component that consists of $4n - 2$ rainbow motifs, each chained to the next, with a cycle of length 6 chained to the end. The second part contains n repeats of cycles of length 10 being equivalent to the -1, -2, -3, -4 permutation. Such permutation exists for every n. The general permutation of the first part is shown in Fig. 7.2, the second part contains the numbers in the identical order, one positive sign is followed by four negative signs, namely, the second part of the permutation is

$$8n - 1,\ -(8n),\ -(8n + 1),\ -(8n + 2),\ -(8n + 3),\ 8n + 4, \ldots$$

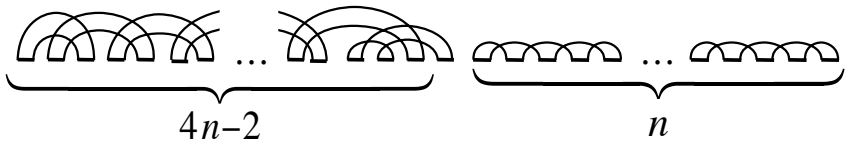


Figure 7.1: The general structure of the graph of desire and reality of the signed permutation that we generated. See main text for details.

$$6n\text{–}2, 6n\text{–}3, 1, 6n\text{–}4, 6n\text{–}1, 6n\text{–}5, 2, \ldots k, 6n\text{–}2k\text{–}2, 6n\text{+}k\text{–}2, 6n\text{–}2k\text{–}3, k\text{+}1, \ldots$$
$$2n-2, 2n+2, 8n-4, 2n+1, 2n-1, -(8n-2), 2n, 8n-3, 8n-1$$

Figure 7.2: The general form of the signed permutation for the first component on Fig. 7.1

It is easy to show that the first part of the permutation needs $4n$ reversals to get sorted, and it has exactly two most parsimonious sorting paths by reversals. Moreover, these two sorting paths have only the start and end genome in common, all the intermediate genomes of the two sorting paths are different. Indeed, there are two cycle-increasing reversals on the cycle of length 6, both of them are sorting reversals. It causes the last rainbow cycle becomes a twisted cycle and the cycle of length 6 is split into a trivial cycle and a twisted cycle. In the second step, however, only the twisted cycle that was a rainbow cycle at the beginning can be the subject of a sorting reversal, since the reversal on the other cycle causes no black vertex in the overlap graph. That is, it is not a sorting reversal. It causes a rainbow cycle becomes twisted and the other twisted cycle becomes a rainbow cycle. The only sorting reversal in the third step is applied on the twisted cycle, and causes two twisted cycles overlapping at the end of the chain. This patterns continues: in every even step, there are two twisted cycles (except in the last step), and only on one of them there is a sorting reversal. In every odd step, there is one twisted cycle on which a sorting reversal might act. In the last step, there is only one twisted cycle, and the reversal on it sorts the permutation.

Each cycle of length 10 in the second part of the permutation needs 4 reversals to get sorted, and each of them has 26 most parsimonious sorting paths. Of these 26, $4! = 24$ paths reverse single numbers one by one, and they form a four-dimensional hypercube, i.e. they have 14 common intermediate genomes in addition to the start and end genomes. The remaining two

sorting paths reverse the first or last three numbers of the permutation alternately, twice each (see also Example 1 on page 32). The Hannenhalli-Pevzner theorem says that all sorting paths of a permutation are combinations of the sorting paths over its components, therefore there are

$$|I_D| = 2 \times 26^n \times \frac{(8n)!}{(4n)!(4!)^n} \tag{7.2}$$

sorting paths of the nth member of the series. This set of paths can be partitioned into two, equal size parts based on which path they use for sorting the first component. Let S_D be one of these sets. We are going to show that $F(S_D)/\pi_D(S_D)$ converges to 0 exponentially fast with n, and hence, exponentially fast with $|D| = 13n - 2$.

The first observation is that

$$\frac{F(S_D)}{\pi_D(S_D)} = \frac{1}{|S_D|} \sum_{x \in S_D, y \in I_D \setminus S_D} P_D(y|x) \tag{7.3}$$

since π_D is the uniform distribution. We proceed by cutting S_D into three parts such that the first two parts are 'negligibly' small (that is, exponentially small compared to the whole solution space), and the third contains an ergodic flow towards the complement of S_D that is too small. Let $S_{D,1}$ be the subset of S_D which contains the paths in which there are less than $\left(7 + \frac{9}{11}\right) n$ intermediate genomes between the first and last sorting reversals of the first component. Since each sorting path contains $8n$ reversals and $4n$ reversals sort the first component, there exists a $c_1 > 1$ for which

$$\frac{|S_{D,1}|}{|S_D|} = O\left(\frac{1}{c_1^n}\right) \tag{7.4}$$

since the reversals sorting the first component can be positioned without constraints in $\binom{8n}{4n}$ ways into each complete sorting path, and in less than $\binom{(7+\frac{9}{11})n}{4n}$ ways if all these mutations must be put in a window of length less than $\left(7 + \frac{9}{11}\right) n$, and the number of possible windows in a series of reversals of length $8n$ is $O(n^2)$.

The remaining set $S_D \setminus S_{D,1}$ contains sorting paths in which the complete sorting of at least $\frac{9}{11}n$ cycles of length 10 are between the first and the last sorting reversals of the large component. Let $S_{D,2}$ be the subset of $S_D \setminus S_{D,1}$ that contains paths in which at most $\frac{3}{4}n$ cycles of length 10 are sorted with

single number reversals between the first and the last sorting reversals of the large component. It is obvious that there exists a $c_2 > 1$ for which

$$\frac{|S_{D,2}|}{|S_D|} = O\left(\frac{1}{c_2^n}\right) \tag{7.5}$$

since the number of cycles of length 10 that are sorted with single number reversals are binomially distributed with mean $\frac{24}{26}k$ with $k \geq \frac{9}{11}n$. Hence $\frac{3}{4}n < \frac{24}{26}k$, and we can apply the Chernoff inequality.

Let $S_{D,3}$ be $S_D \backslash (S_{D,1} \cup S_{D,2})$. We have

$$\begin{aligned}\frac{F(S_D)}{\pi_D(S_D)} = \frac{1}{|S_D|} \quad & \Big(\textstyle\sum_{x\in S_{D,1}, y\in I_D\setminus S_D} P_D(y|x) + \\ & \textstyle\sum_{x\in S_{D,2}, y\in I_D\setminus S_D} P_D(y|x) + \\ & \textstyle\sum_{x\in S_{D,3}, y\in I_D\setminus S_D} P_D(y|x)\Big)\end{aligned} \tag{7.6}$$

$|S_{D,1}|$ and $|S_{D,2}|$ are upper bounds for the first and the second sum, hence

$$\frac{F(S_D)}{\pi_D(S_D)} = O\left(\frac{1}{\min\{c_1, c_2\}^n}\right) + \frac{1}{|S_D|} \sum_{x\in S_{D,3}, y\in I_D\setminus S_D} P_D(y|x) \tag{7.7}$$

Recall that

$$\begin{aligned}P_D(y|x) &= \sum_w T_D(y, w|x) \min\left\{1, \frac{\pi_D(y) T_D(x, w'|y)}{\pi_D(x) T_D(y, w|x)}\right\} \\ &= \sum_w \min\left\{T_D(y, w|x), T_D(x, w'|y)\right\}\end{aligned} \tag{7.8}$$

and hence, $P_D(y|x)$ can be bounded by

$$P_D(y|x) \leq \sum_w T_D(x, w'|y) \tag{7.9}$$

Let $c = \min\{c_1, c_2\}$, and we have

$$\frac{F(S_D)}{\pi_D(S_D)} \leq O\left(\frac{1}{c^n}\right) + \frac{1}{|S_D|} \sum_w \sum_{\substack{x\in S_{D,3},\\ y\in I_D\setminus S_D}} T_D(x, w'|y) \tag{7.10}$$

where the first sum runs only on windows w that contain the first and the last reversal sorting step of the large component. The inner sum sums for all y the probability that such a subpath is proposed in the w' window that transforms y into in the $S_{D,3}$ set. For a particular y, there is a $c_3 > 1$ such that the probability of the transformation towards any $x \in S_{D,3}$, namely, $\sum_{x \in S_{D,3}} T_D(x, w'|y)$ is $O\left(\frac{1}{c_3^n}\right)$. This is because at least $\frac{3}{4}n$ cycles of length 10 should be sorted by single number reversals for a successful transition. However, in the proposal distribution the number of cycles of length 10 that are sorted by single number reversals is binomially distributed with mean $\frac{4}{6}k = \frac{2}{3}k$ (since 4 of the possible 6 sorting reversals on $-1, -2, -3, -4$ are single number reversals) for k smaller than n and we can again apply the Chernoff bound. The number of ys in the subset $I_D \backslash S_D$ is exactly $|S_D|$, the number of possible windows is only $O(n^2)$, hence for some $1 < c_3^* < c_3$

$$\frac{F(S_D)}{\pi_D(S_D)} = O\left(\left(\frac{1}{\min\{c_1, c_2, c_3^*\}}\right)^n\right) \tag{7.11}$$

□

7.3 Discussion and Conclusion

In this chapter we showed that the Metropolized Partial Importance Sampler might mix slowly on the set of minimum reversal paths. The cause of slow mixing are the big gaps in the most parsimonious sorting paths, like the gaps between the two most parsimonious sorting paths of the large component in our example. Due to these big gaps, large portions of the actual sorting path should be replaced in the proposal to get an irreducible chain. The large changes cause small acceptance ratios, and eventually slow mixing. One might argue that the Metropolized Partial Importance Sampling could be improved on the above mentioned example if it resampled mutations only on one component (whose mutations might not be consecutive on the current path). However, big gaps are common in genome rearrangements paths, for example, it can be shown that hurdle-cutting and hurdle merging [45] sorting paths are disjoint except for the start and the end genome. Both the hurdle-cutting and the hurdle-merging paths might be numerous, and we conjecture that the Metropolized Partial Importance Sampler might mix slowly even on sorting two hurdles.

Our result does not prove but suggests that the similar MCMC methods on the posterior distribution of all sorting paths [28, 58, 59, 114, 118, 100] might also mix slowly. Indeed, the key point in our proof is that the backproposal probability is vanishingly small for the majority of the set of paths S_D, and we saw similar behavior in the case of the posterior distribution of rearrangement paths. The BADGER software [81, 59] has a pre-burn-in phase in which the proposal and backproposal probabilities are omitted from the Metropolis-Hastings ratio, and this makes the likelihood improve significantly. If that pre-burn-in phase is switched off, the burn-in phase remains at low likelihood values and no convergence is obtained. Indeed, our experiments [105] showed that without this pre-burn-in phase, the Markov chain does not converge on Yersinia phylogenies. Therefore we had to use the BADGER software instead of our software, which does not apply this pre-burn-in trick [118].

However, this proof does not imply in any sense that no fast mixing Markov chain exists for sampling from the uniform distribution of minimum reversal sorting paths or posterior distributions of genome rearrangement paths under a Bayesian framework. Indeed, there are at least two possible ways to improve the mixing of Markov chains: with novel proposals that might destroy bottlenecks and with parallel chains that exchange information. We show one example for each.

- Let a reversal be described as a double cut-and-join (DCJ) mutation. The DCJ representation of a reversal tells which adjacencies are changed in the signed permutation. Let sorting paths be described by their series of reversals in DCJ representation. For example, the sorting path:
  $+3, +4, -1, -2 \rightarrow +1, -4, -3, -2 \rightarrow +1, +2, +3, +4$
  is represented by $(0, t3|t1, h2)$ $(h1, h4|t2, 5)$, where h and t denote the head and tail, that is, the end and the beginning of a directed edge. This means that before the first reversal, the beginning of gene 3 was at the beginning of the permutation (represented as 0), the beginning of gene 1 was in adjacency with the end of gene 2, and the first reversal swapped the positions $t3$ and $t1$. Similarly, the second reversal breaks the adjacencies between $h1$ and $h4$ and between $t2$ and the end of the permutation by swapping $h4$ and $t2$. Note that $(a, b|c, d)$ means the same reversal as $(d, c|b, a)$, but differs from, for example, $(b, a|c, d)$.

  Let the vertices of a graph be the minimum reversal paths of a signed

permutation. Let two points of this graph be connected iff at most four, not necessarily consecutive reversals can be removed from each of their DCJ representations such that the remaining patterns will be the same (note that the remaining representations of DCJ mutations might not represent valid DCJ operations). Our conjecture is that the graph will always be connected if the signed permutation contains only oriented components. Above this conjecture, it is an open question if such fixed number of removals holds for all signed permutations, and if so, the so-obtained Markov chain (namely, remove a fixed number of not necessarily consecutive reversals and put back reversals not necessarily to the same place) can be transformed into a rapidly mixing MCMC. The hope that such a Markov chain might be rapidly mixing is due to the fact that in such Markov chain there is a polynomial lower bound for the backproposal probabilities (and hence for the acceptance ratio) while the diameter of the Markov chain will grow also polynomially with the problem size.

The conjecture is proved for a class of permutations in Chapter 9.

- For a signed permutation of length n that can be sorted in k steps, we create a Markov chain whose states are k+1-tuples. The first coordinate of any element in the state space contains the signed permutation, and the lth coordinate contains a transformation path from the given signed permutation to an other signed permutation that can be sorted in $l-1$ steps.

  We define a Markov chain on this set that changes two consecutive coordinates, the lth and $l+1$st in the following way. The new lth coordinate is the shortened path in the old $l+1$st coordinate, and the new $l+1$st coordinate is a random extension of the old lth coordinate. Applying the appropriate Metropolis-Hastings ratio, the Markov chain will converge to the uniform distribution. We could prove that this Markov chain on its own generally will not converge rapidly to the equilibrium distribution [116], however, the mixing is rapid on Yersinia data if the following inside-swapping step is also added to the transition kernel of the Markov chain. The inside-swapping step swaps two consecutive commuting reversals on one of the sorting paths. To do such a step, we first choose a random i between 2 and $k+1$, then we count all the neighboring reversals in the sorting path in the ith coordinate that

can be swapped. We select a random pair, calculate how many commuting reversal neighbors there are after swapping them, and calculate the corresponding Metropolis-Hastings ratio with which we accept the change. We compared this Markov chain with the Importance Sampling method of Ajana *et al.* [4], and showed that this latter method explores only a negligible part of the possible sorting reversals. Since the Partial Importance Sampling method applies the same Importance Sampling transition kernel, this again suggests that the slow convergence of the Markov chain we described in this manuscript might be a general problem in case of real data, not only for the example we gave.

We also would like to highlight that a commonly used method, Parallel Tempering [39], also known as $(MC)^3$ [73] will not work. Indeed, in this chapter, we showed that an MCMC might mix slowly even if the target distribution is the uniform one, and the uniform distribution cannot be further heated.

Chapter 8

Hardness results on SCJ problems

In this chapter, we prove that two counting problems from the five on rearrangement models defined in the Preliminaries are hard under the SCJ model. We prove that counting most parsimonious evolutionary histories on an evolutionary tree under the SCJ model is #P-complete, and does not have an FPRAS approximation if RP $\neq$ NP. Miklós, Tannier and Kiss proved the non-approximability in Theoretical Computer Science, 552:83–98, DOI: 10.1016/j.tcs.2014.07.027. Here we present a modified proof based on the work of Miklós and Smith. This modified proof also proves the #P-completeness.

We also prove that counting the most parsimonios median scenarios is #P-complete. This was a joint work with Heather Smith. The author of the thesis suggested to prove the theorem using the Chinese reminder theorem. The detailed computations presented in Table 8.2 was done by Heather Smith using the guidance of the author of this thesis. The original work has been published in Advances in Applied Mathematics102:18–82.

8.1 Counting the most parsimonious substitution histories on an evolutionary tree

In this section, we are going to introduce a #P-completeness proof based on the work of Miklós, Tannier and Kiss [126] and Miklós and Smith [123]. The small parsimony problem is defined in the following way.

Definition 104. (SP-Tree) *Given a finite alphabet Γ, a rooted binary tree $T = (V, E)$, where $L \subset V$ denotes the leaves of the tree and $f : L \to \Gamma^k$, a function assigning a sequence of length k to each leaf of the tree. The SP-tree problem is to compute a function $g : V \to \Gamma^k$ such that $g(v) = f(v)$ for all $v \in L$, and the score*

$$\sum_{(u,v)\in E} H(g(u), g(v)) \tag{8.1}$$

is minimized, where H denotes the Hamming distance of the sequences, that is, the number of positions in which the two sequences differ.

The SP-Tree problem is known to be an easy optimization problem, and the number of functions g minimizing the score in Equation (8.1) can be found in polynomial time. However, the counting problem becomes hard if we would like to obtain the number of most parsimonious scenarios instead of labelings of the internal nodes. There are $H(g(u), g(v))!$ number of ways to transform the sequence labeling vertex u to the sequence labeling vertex v using $H(g(u), g(v))$ substitutions (a substitution changes one character in a sequence). Therefore, if $\mathcal{G}$ denotes the set of functions minimizing the score in Equation (8.1), then we would like to compute

$$\sum_{g\in\mathcal{G}} \prod_{(u,v)\in E} H(g(u), g(v))! \tag{8.2}$$

We will denote this counting problem #SPS-Tree (small parsimony scenario on trees). We are going to show that computing this number is #P-complete even if the problem is restricted to an alphabet of size 2. First we need the following lemma. Recall that a *conjunctive normal form* is a logical function expressed as a conjunction of disjunctive *clauses*, each clause is a disjunction of *literals*, a literal is either a logical variable or its negate. When each clause is a disjunction of 3 literals, we call the logical function a *3CNF* .

Lemma 105. *For any 3CNF Φ, there exists a 3CNF Φ' such that the following hold:*

1. *Φ' can be constructed in polynomial time, particularly, the size of Φ' is a polynomial function of the size Φ.*

2. *Φ and Φ' have the same number of satisfying assignments.*

3. *Φ' contains an even number of variables, and in any satisfying assignments of Φ', exactly half of the variables have a logical value TRUE.*

Proof. Let $x_1, x_2, \ldots x_n$ be the variables in Φ. Φ' contains the variables $x_1, x_2, \ldots, x_n, y_1, y_2, \ldots y_n$, and is defined as

$$\Phi' := \Phi \wedge \bigwedge_{i=1}^{n} \quad ((x_i \vee y_i \vee x_{i+1}) \wedge (\overline{x}_i \vee \overline{y}_i \vee x_{i+1}) \wedge (x_i \vee y_i \vee \overline{x}_{i+1}) \wedge (\overline{x}_i \vee \overline{y}_i \vee \overline{x}_{i+1})) \tag{8.3}$$

where the index of x_{i+1} is taken modulo n, that is, x_{n+1} is defined as x_1.

The assignments of the x_i variables in any satisfying assignment of Φ' also satisfies Φ, therefore, it is sufficient to show that any satisfying assignment of Φ can be extended to exactly one satisfying assignment of Φ'. But this is obvious: the conjunctive form in Equation (8.3) forces that y_i must take the value of $\overline{x}_i$. This provides also that in any satisfying assignment of Φ', exactly half of the values will take the TRUE value. □

We need the following theorem.

Theorem 106. *The #3SAT problem, that is, counting the satisfying assignments of a 3CNF is #P-complete.*

The proof can be found, for example, in [115]. To prove that computing the quantity in Equation (8.2) is #P-complete, we give a polynomial running time algorithm which for any 3CNF formula Φ, constructs a problem instance $p \in \#\mathrm{SPS} - \mathrm{TREE}$ with the following property: The number of solutions of p can be written as $a + by$, where y is the number of satisfying assignments of Φ, b is an easy-to-calculate positive integer, and $0 < a \ll b$. Thus, if s is the number of solutions of p, then $\left\lfloor \frac{s}{b} \right\rfloor$ is the number of satisfying assignments of Φ.

Let Φ be a 3CNF, and let Φ' be the 3CNF that has as many satisfying assignments as Φ and in all satisfying assignments, the number of TRUE and FALSE values are the same. Let n denote the number of logical variables in Φ' and let k denote the number of clauses in Φ'. We are going to construct a tree denoted by T'_Φ, and to label its leaves with sequences over the alphabet $\{0, 1\}$. The first n characters of each sequence correspond to the logical variables x_i, and there are further, auxiliary characters. The number of auxiliary characters are $148k \left(\left\lceil (k \log(n!) + n \log(2)) / \log(\frac{2^{20}}{3^{12}}) \right\rceil + 1 \right)$. The

construction is such that there will be 2^n most parsimonious labelings of the internal nodes, one for each possible logical assignment. Each labeling is such that the labeling of the root completely determines the labelings at the other internal nodes. The corresponding assignment is such that the value of the logical variable x_i is TRUE if there is a character 1 in the sequence at the root of the tree in position i. The characters in the auxiliary positions are 0 in all the most parsimonious labelings.

If an assignment is a satisfying assignment, then the corresponding labeling has many more scenarios than the labelings corresponding to non-satisfying assignments. Furthermore, for each satisfying assignment, the corresponding labelings have the same, easy-to-compute number of scenarios.

For each clause c_j, we construct a subtree T_{c_j}. The construction is done in three phases, illustrated on Figure 8.1. First, we create a constant-size subtree, called the unit subtree, using building blocks we call elementary subtrees. Then in the blowing-up phase, this unit subtree is repeated several times, and in the third phase it is amended with another constant-size subtree. The reason for this construction is the following: the unit subtree is constructed in such a way that if a clause is satisfied, the number of scenarios on this subtree is large, and is always the same number not depending on how many literals provide satisfaction of the clause. When the clause is not satisfied, the number of scenarios is a smaller number. The blowing up is necessary for sufficiently separating the number of solutions for satisfying and non-satisfying assignments. Finally, the amending is necessary for achieving 2^n most parsimonious labelings on each T_{c_j} and to guarantee that the number of most parsimonious scenarios is the same for each satisfying assignment. The amending is slightly different for those clauses that come from Φ and those that are in $\Phi' \setminus \Phi$.

We detail the construction of the subtree for the clause $c_j = x_1 \vee x_2 \vee x_3$, denoted by T_{c_j}. Subtrees for the other kinds of clauses are constructed similarly. The unit subtree is built from 76 smaller subtrees that we will call *elementary subtrees*. On each elementary subtree, the sequences labeling the leaves contains 0 at almost all the positions, except the positions corresponding to the literals of the clause and the position of the possible auxiliary character. Only 14 different types of elementary subtrees are in a unit subtree, but several of them have given multiplicity, and the total count of them is 76, see also Table 8.1. Some of the elementary subtrees are cherry motives (two leaves connected via an internal node, see also Fig. 8.2.a)) for which we arbitrarily identify a left and a right leaf. For some of these cherries, we

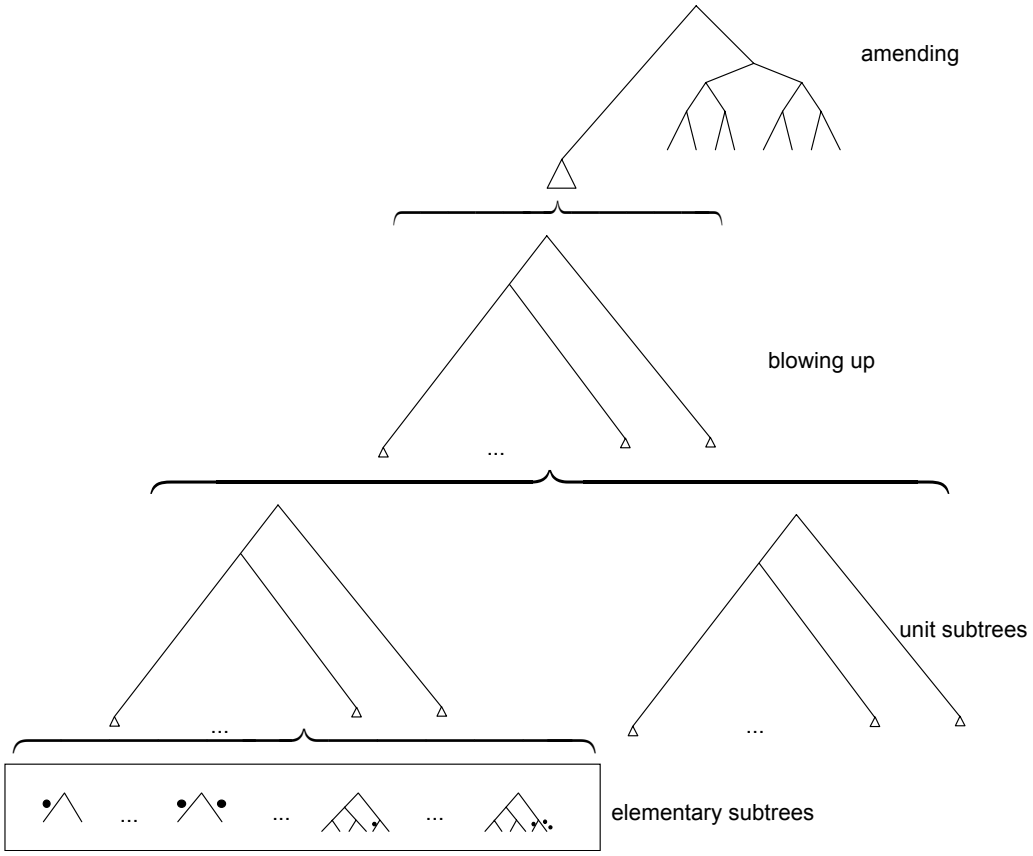


Figure 8.1: Constructing a subtree T_{c_j} for a clause c_j. The subtree is built in three phases. First, elementary subtrees are connected with a comb to get a unit subtree. In the second phase the same unit subtree is repeated several times, "blowing up" the tree. In the third phase, the blown-up tree is amended with a constant size, depth 3 fully balanced tree. The smaller subtrees constructed in the previous phase are denoted with a triangle in the next phase. See also text for details.

introduce one or more auxiliary characters, which are 1 only on the indicated leaf of the cherry and 0 everywhere else in the tree. So the edges connecting these leaves to the rest of the entire tree $T_{\Phi'}$ will contain one or more additional substitutions in all the most parsimonious solutions.

The constructed unit subtree will be such that if the clause is not satisfied, the number of possible most parsimonious scenarios for the corresponding labeling on this unit subtree is $2^{136} \times 3^{76}$, and if the clause is satisfied, then the number of possible most parsimonious scenarios for each corresponding labeling is $2^{156} \times 3^{64}$. The ratio of the two numbers is $2^{20}/3^{12} > 1$. We will denote this number by γ.

Below we detail the construction of the elementary subtrees and also give the number of most parsimonious scenarios on them since the number of scenarios on the unit subtree is simply the product of these numbers. This part is quite technical, however, the careful reader might observe the following. The number of scenarios for a fixed labeling on a unit tree is the product of the number of scenarios on the elementary trees. These numbers are always in $2^x 3^y$ form, and we need a (linear) combination of unit trees such that the sum of the exponents both on 2 and 3 is the same for all satisfying assignments and different for the non-satisfying assignment; furthermore, the number of solutions for the non-satisfying assignment is smaller than for any of the satisfying assignments. Such combinations can be found by some linear algebraic considerations not presented here; below we just show one possible solution.

For each elementary tree, we give the characters at positions of the three literals. The elementary trees which are cherries are the following:

- There are four cherries on which the left leaf contains 1 in an extra position, and the characters in the positions of the three literals on the left and right leaf are given by

  011, 100
  101, 010
  110, 001
  000, 111.

  The first column shows the characters in the positions corresponding to the literals on the left leaf, while the second column shows those

characters on the right leaf. Observe that there are 16 most parsimonious labelings of the root of the cherry motif and each needs 4 substitutions. However, 8 of them contain a character 1 in the auxiliary position, which will not be a most parsimonious labeling on $T_{\Phi'}$, as we discussed. So we have to consider only the other 8 labelings, where the characters are all 0, except the positions corresponding to the literals.

The number of scenarios on one cherry is 24 if the sequence at the root of the cherry is the same as on the right leaf. Indeed, in that case, 4 substitutions are necessary on the left edge, and they can be performed in any order. If the number of substitutions are 3 and 1, respectively, on the left and right edges, or *vice versa*, the number of solutions is 6. Finally, if both edges have 2 SCJ operations, then the number of solutions is 4.

- There is one cherry motif without any extra adjacency, and the characters in the positions corresponding to the literals are

  000, 111.

  There are 8 most parsimonious labelings at the root, and each needs 3 substitutions. If the labeling at the root corresponds to a non-satisfying assignment, the number of scenarios on this cherry is 6; if all logical values are true, the number of scenarios is still 6; in any other case, the number of scenarios is 2.

  This elementary subtree is repeated 3 times.

- Finally, there are 3 types of cherry motifs with a character 1 at one-one auxiliary positions on both leaves. These are two different adjacencies, so both of them need one extra substitution on their incoming edge. The characters in the positions corresponding to the literals are

  011, 100
  101, 010
  110, 001.

There are 8 possible labelings of the root which are most parsimonious in $T_{\Phi'}$, and each needs 5 substitutions. If all substitutions at the positions corresponding to the 3 literals falls onto one edge, then the number of scenarios is 24, otherwise the number of solutions is 12.

Each of these elementary subtrees is repeated 15 times.

The remaining elementary subtrees contain 3 cherry motifs connected with a comb, that is, a completely unbalanced tree, see also Figure 8.2. For the cherry at the right end of this elementary subtree, there is one or more auxiliary positions that have character 1 at one of the leaves and 0 everywhere else in T_{Φ}.

There are 3 elementary subtrees of this type which have only one auxiliary position. On these trees, the sequence at the right leaf of the rightmost cherry is all 0, and the sequence at the left leaf of the rightmost cherry motif is all 0 except at the auxiliary position and exactly 2 positions amongst the 3 positions corresponding to the literals.

The remaining leaves of these elementary subtrees are constructed in such a way that there are 8 most parsimonious labelings, each needing 7 substitutions, see the example in Figure 8.2. The number of substitutions is 0 or 1 at each edge except the two edges of the rightmost cherry motif. Here the number of substitutions might be 3 and 0, 2 and 1, or 1 and 2, yielding 6 or 2 scenarios, see also Table 8.1.

Each of these elementary subtrees are repeated 3 times.

Finally, there are 3 elementary subtrees of this type which have one auxiliary position for the left leaf of the rightmost cherry motif, and there are 2 auxiliary positions for the right leaf of the rightmost cherry motif. The sequence at the right leaf of the rightmost cherry is all 0 except at the 2 auxiliary positions, and the sequence at the left leaf of the rightmost cherry motif is all 0 except at the auxiliary positions and exactly 2 positions amongst the 3 positions corresponding to the literals.

The remaining leaves of these elementary subtrees are constructed in such a way that there are 8 most parsimonious labelings, each needing 9 substitutions, see the example in Figure 8.2. The number of substitutions is 0 or 1 on each edge except the two edges of the rightmost cherry motif. Here the number of substitutions might be 1 and 4, 2 and 3, or 3 and 2, yielding 24 or 12 scenarios, see also Table 8.1.

Each of these elementary subtrees are repeated 5 times.

	 011	 101	 110	000	 011	101	 110	 000	 011	 101	 110	 011	 101	110
#	1	1	1	1	3	3	3	3	5	5	5	15	15	15
000	6	6	6	6	6^3	6^3	6^3	6^3	12^5	12^5	12^5	12^{15}	12^{15}	12^{15}
100	24	4	4	4	6^3	2^3	2^3	2^3	12^5	12^5	12^5	24^{15}	12^{15}	12^{15}
010	4	24	4	4	2^3	6^3	2^3	2^3	12^5	12^5	12^5	12^{15}	24^{15}	12^{15}
110	6	6	6	6	2^3	2^3	2^3	2^3	12^5	12^5	24^5	12^{15}	12^{15}	24^{15}
001	4	4	24	4	2^3	2^3	6^3	2^3	12^5	12^5	12^5	12^{15}	12^{15}	24^{15}
101	6	6	6	6	2^3	2^3	2^3	2^3	12^5	24^5	12^5	12^{15}	24^{15}	12^{15}
011	6	6	6	6	2^3	2^3	2^3	2^3	24^5	12^5	12^5	24^{15}	12^{15}	12^{15}
111	4	4	4	24	2^3	2^3	2^3	6^3	24^5	24^5	24^5	12^{15}	12^{15}	12^{15}

Table 8.1: The number of scenarios on different elementary subtrees of the unit subtree of the subtree T_{c_j} for clause $c_j = x_1 \vee x_2 \vee x_3$. Columns represent the 14 different elementary subtrees, the topology of the elementary subtree is indicated on the top. The black dots mean extra substitutions on the indicated edge due to the characters in the auxiliary positions; the numbers represent the presence/absence of adjacencies on the left leaf of a particular cherry motif, see text for details. The row starting with # indicates the number of repeats of the elementary subtrees. Further rows represent the logical true/false values of the literals, for example, 001 means x_1 = FALSE, x_2 = FALSE, x_3 = TRUE. The values in the table indicate the number of scenarios, raised to the appropriate power due to multiplicity of the elementary subtrees. It is easy to check that the product of the numbers in the first line is $2^{136} \times 3^{76}$ and in any other lines is $2^{156} \times 3^{64}$.

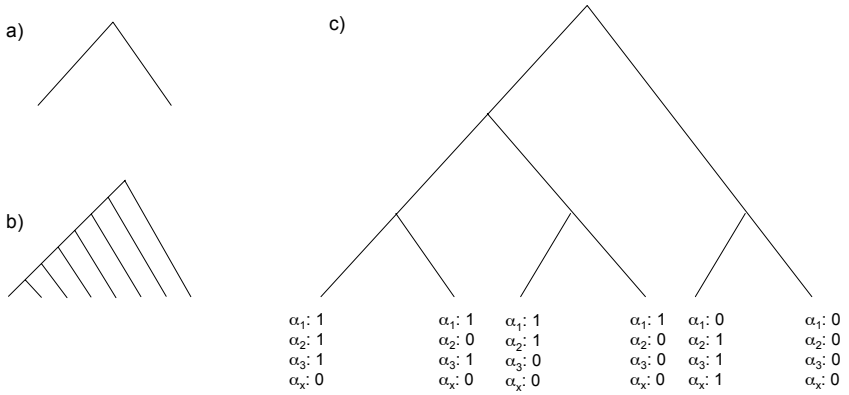


Figure 8.2: **a)** A cherry motif, i.e., two leaves connected with an internal node. **b)** A comb, i.e., a fully unbalanced tree. **c)** A tree with 3 cherry motifs connected with a comb. The assignments for 4 adjacencies, α_1, α_2, α_3 and α_x are shown at the bottom for each leaf. α_i, $i = 1, 2, 3$ are the adjacencies related to the logical variables b_i, and α_x is an extra adjacency. Note that Fitch's algorithm gives ambiguity for all adjacencies α_i at the root of this subtree.

In this way, the roots of all 76 elementary subtrees have 8 most parsimonious labelings corresponding to the 8 possible assignments of the literals in the clause. We connect the 76 elementary subtrees with a comb, and thus, there are still 8 most parsimonious labelings at the root of the entire subtree, which is the unit subtree. If the labeling at the root corresponds to a satisfying assignment of the clause, the number of scenarios is $2^{156} \times 3^{64}$, if the clause is not satisfied, the number of scenarios is $2^{136} \times 3^{76}$, as can be checked on Table 8.1. The ratio of them is indeed $2^{20}/3^{12} = \gamma$. The number of leaves on this unit subtree is 248, and 148 auxiliary positions are introduced.

This was the construction of the constant size unit subtree. In the next step, we "blow up" the system. Similar blowing up can be found in the seminal paper by Jerrum, Valiant and Vazirani [54], in the proof of Theorem 5.1. We repeat the above described unit subtree $\lceil (k \log(n!) + n \log(2))/\log(\gamma) \rceil + 1$ times, and connect all of them with a comb (completely unbalanced tree). It is easy to see that there are still 8 most parsimonious labelings. For a solution satisfying the clause, the number of scenarios on this blown-up subtree is

$$X = \left(2^{156} \times 3^{64}\right)^{\left\lceil \frac{k \log(n!) + n \log(2)}{\log(\gamma)} \right\rceil + 1} \tag{8.4}$$

and the number of scenarios if the clause is not satisfied is

$$Y = \left(2^{136} \times 3^{76}\right)^{\left\lceil \frac{k\log(n!)+n\log(2)}{\log(\gamma)} \right\rceil + 1}. \tag{8.5}$$

It is easy to see that

$$\frac{X}{Y} = \gamma^{\left\lceil \frac{k\log(n!)+n\log(2)}{\log(\gamma)} \right\rceil + 1} \geq \gamma^{\frac{\log(n!)^k)+\log(2^n)}{\log(\gamma)}} = n!^k 2^n, \tag{8.6}$$

and

$$\frac{X}{Y} = \gamma^{\left\lceil \frac{k\log(n!)+n\log(2)}{\log(\gamma)} \right\rceil + 1} \leq \gamma^{\frac{\log(n!)^k)+\log(2^n)}{\log(\gamma)}+2} = n!^k 2^n \gamma^2, \tag{8.7}$$

since for any positive number x,

$$x^{\frac{1}{\log(x)}} = e. \tag{8.8}$$

Let all adjacencies not participating in the clause be 0 on this blown-up subtree.

We are close to the final subtree T_{c_j} for one clause, c_j. In the third phase, we amend the so-far obtained tree with a constant-size subtree. The amending is slightly different for clauses coming from Φ and for those that are in $\Phi' \setminus \Phi$. We detail the amending for both cases.

If the clause contains only x logical variables, say, the clause is $x_1 \vee x_2 \vee x_3$, then construct two copies of a fully balanced depth 6 binary tree, on which the root has 64 most parsimonious labelings corresponding to the 64 possible assignments of the literals participating in the clause and their corresponding logical variables of the y type (namely, y_1, y_2 and y_3). This can be done with a construction similar to the left part of the tree on Figure 8.2.c).

In one of the copies, all other characters corresponding to logical variables not participating in the clause are 1 on all leaves, and thus, in each most parsimonious labelings of the root. In the other copy, those characters should be all 0.

In the copy, where all other characters are 0, the construction should be done in such a way that going from the root of the tree, first the y logical variables must be separated, then the x ones. Namely, characters at the position corresponding to y_1 should be the same (say, 0) on each leaf of the left subtree of the root and should be the other value on each leaf of the right subtree of the root. Similalry, for each of the four grandchildren of the root, the leaves must take the same value at the position corresponding to

y_2, and these values must be different for the siblings. The same rule must be applied for the grand-grandchilden of the root. There is an internal node of this subtree such that on all of its leaves, each character at each position corresponding to y variables is 0. Replace the subtree at this position with the blown-up subtree. Connect the two copies with a common root. The so obtained tree is T_{c_j}.

Observe that there are 2^n possible most parsimonious labelings of T_{c_j}. We have the following lemma on them.

Lemma 107. *For any most parsimonious labelings, if Φ' and thus, particularly, the clause c_j is satisfied, then the number of scenarios on T_{c_j} is*

$$X \times \left(\left(\frac{n-6}{2}\right)!\right)^2 \geq Y \times (n!)^k \times 2^n \times \left(\left(\frac{n-6}{2}\right)!\right)^2. \tag{8.9}$$

If the clause c_j is not satisfied, then the number of scenarios is at most $Y \times (n-6)!$. If the clause c_j is satisfied, however, Φ' is not satisfied, then the number of scenarios is at most $X \times (n-6)!$.

Proof. There are 3 logical x variables in the clause and there are 3 corresponding y variables. For the remaining $n-6$ variables, there are $n-6$ substitutions on the two edges of the root. If Φ' is satisfied, then for each i, exactly one in the couple (x_i, y_i) has the TRUE value and the other has the FALSE value. Therefore, there are $\frac{n-6}{2}$ substitutions on both edges of the root. On all remaining edges of the amending, there is either 0 or 1 substitution. Finally, the number of scenarios on the blown-up tree is X. Therefore, the number of scenarios is indeed $X \times \left(\left(\frac{n-6}{2}\right)!\right)^2$ if Φ' is satisfied. The inequality in Equation (8.9) comes from Equation (8.6).

If the clause is not satisfied, then the number of scenarios on the blown-up tree is Y. The substitutions on the two edges of the root might be arbitrarily distributed, however, in any cases, the number of scenarios is at most $(n-6)!$. This extremity is taken when all the substitutions fall onto the same edge.

If c_j is satisfied, however, Φ' is not, then the number of scenarios on the blown-up tree is X, and the number of scenarios on the two edges of the root is at most $(n-6)!$. □

If the clause is in the form $x_i \vee y_i \vee x_{i+1}$ (some of the literals might be negated), then the amending is the following. Construct two copies of a fully balanced depth 4 binary tree, on which the root has 16 most parsimonious

labelings corresponding to the 16 possible assignments of logical variables x_i, y_i, x_{i+1} and y_{i+1}. On one of the copies, all other characters are 0, while on the other copy, all other characters must be 1. On the copy, where all other characters are 0, the construction should be such that there must be an internal node such that at all of its leaves, the characters at the position corresponding to y_{i+1} are 0 and the subtree has depth 3. Replace this subtree with the blown-up tree. Connect the two copies with a common root. This is the final T_{c_j} tree.

For this tree, a lemma similar to Lemma 107 can be proved.

Lemma 108. *For any most parsimonious labelings, if Φ' and thus, particularly, the clause c_j is satisfied, then the number of scenarios on T_{c_j} is*

$$X \times \left(\left(\frac{n-4}{2}\right)!\right)^2 \geq Y \times (n!)^k \times 2^n \times \left(\left(\frac{n-4}{2}\right)!\right)^2. \tag{8.10}$$

If the clause c_j is not satisfied, then the number of scenarios is at most $Y \times (n-4)!$. If the clause c_j is satisfied, however, Φ' is not satisfied, then the number of scenarios is at most $X \times (n-4)!$.

Proof. The proof is similar to the proof of Lemma 107, just now there are $n-4$ substitutions that must be distributed on the two edges of the root. □

For all k clauses, construct such a subtree and connect all of them with a comb. This is the final tree $T_{\Phi'}$ for the 3CNF Φ'. It is easy to see that $T_{\Phi'}$ has 2^n most parsimonious labelings corresponding to the 2^n possible assignments of the logical variables. For these labelings, we have the following theorem.

Theorem 109. *If a labeling corresponds to a satisfying assignment, then the number of scenarios is*

$$\begin{aligned} &X^k \times \left(\left(\frac{n-4}{2}\right)!\right)^{2n} \times \left(\left(\frac{n-6}{2}\right)!\right)^{k-2n} \geq \\ &Y^k \times \left(n!^k \times 2^n\right)^k \times \left(\left(\frac{n-4}{2}\right)!\right)^{2n} \times \left(\left(\frac{n-6}{2}\right)!\right)^{k-2n}. \end{aligned} \tag{8.11}$$

If a labeling corresponds to a non-satisfying assignment, then the number of scenarios is at most

$$\begin{aligned} &X^{k-1} \times Y \times (n-4)!^{2n} \times (n-6)!^{k-2n} \leq \\ &Y^k \times \left(n!^k \times 2^n \times \gamma^2\right)^{k-1} \times (n-4)!^{2n} \times (n-6)!^{k-2n}. \end{aligned} \tag{8.12}$$

Particularly, the number of scenarios corresponding to non-satisfying assignments is at most

$$Y^k \times \left(n!^k \times \gamma^2\right)^{k-1} \times (2^n)^k \times (n-4)!^{2n} \times (n-6)!^{k-2n} \leq$$
$$Y^k \times \left(n!^k \times 2^n\right)^k \ll$$
$$Y^k \times \left(n!^k \times 2^n\right)^k \times \left(\left(\frac{n-4}{2}\right)!\right)^{2n} \times \left(\left(\frac{n-6}{2}\right)!\right)^{k-2n}. \quad (8.13)$$

Proof. If Φ' contains n logical variables, then there are $2n$ clauses in $\Phi' \setminus \Phi$ and $k-2n$ clauses in Φ. Based on this, the number of scenarios for any labelings corresponding to a satisfying assignment can be easily calculated from Lemmas 107 and 108.

If Φ' is not satisfied, then at least one of the clauses is not satisfied causing a smaller number of scenarios on the corresponding subtree. However, the number of scenarios on other subtrees corresponding to other clauses might be higher due to the uneven distribution of the substitutions falling onto the two edges of the root of the subtrees. The upper bounds are based on Equation (8.7) considering that $\gamma = \frac{2^{20}}{3^{12}} < 2$ and $n \geq 6$. □

What follows is that

$$\left\lfloor \frac{s}{X^k \times \left(\left(\frac{n-4}{2}\right)!\right)^{2n} \times \left(\left(\frac{n-6}{2}\right)!\right)^{k-2n}} \right\rfloor \quad (8.14)$$

is the number of satisfying assignments of Φ where s is the number of most parsimonious scenarios on $T_{\Phi'}$. Since both the size of the tree $T_{\Phi'}$ and the length of the sequences labeling the leaves of $T_{\Phi'}$ is a polynomial function of size Φ, furthermore, $T_{\Phi'}$ together with the sequences labeling its leaves can be constructed in polynomial time, we get the following theorem.

Theorem 110. *The counting problem #SPS-TREE is in #P-complete.*

8.2 Counting the most parsimonious substitution histories on a star tree

We learned in Section 8.1 that counting the most parsimonious scenarios on an evolutionary tree is in #P-complete. Here we show that the problem remains #P-complete if the binary tree is replaced to a star tree. Below we define this problem.

Definition 111. (#SPS-Star) *Given a multiset of sequences of the same length over the same alphabet $\mathcal{S} = \{A_1, A_2, \ldots A_n\}$. The #SPS-Star problem is to compute the value defined as*

$$\sum_{M \in \mathcal{M}} \prod_{i=1}^{n} H(A_i, M)! \tag{8.15}$$

where $\mathcal{M}$ is the set of sequences that minimizes the sum of Hamming distances from the sequences in $\mathcal{S}$. Namely, for any $M \in \mathcal{M}$,

$$\sum_{i=1}^{n} H(A_i, M) \tag{8.16}$$

is minimal.

Surprisingly, this problem is #P-complete, even if the size of the alphabet is 2, although finding the size of $\mathcal{M}$ is trivial. It is easy to see that $\mathcal{M}$ consists of the sequences that contain the majority character for each position. The majority character might not be unique; the size of $\mathcal{M}$ is the product of the number of majority characters in each position. If the size of the alphabet is 2, say, it is $\{0, 1\}$, then the size of $\mathcal{M}$ is 2^m, where m is the number of positions where half of the sequences in $\mathcal{S}$ contain 0 and the other half of them contain 1. Sequences in M are called *median sequences*. For each median sequence M, $\prod_{i=1}^{n} H(A_i, M)!$ is the number of corresponding scenarios.

Below we present a proof that #SPS-Star is in #P-complete based on the work of Miklós and Smith [123]. The proof is based on reducing #3SAT to #SPS-Star using modulo prime number calculations.

Let Φ be a 3CNF with n variables and k clauses, and let p be a prime number between $\min 300, n+5$ and $5 \min 300, n+5$. We are going to construct a multiset $\mathcal{S}$ containing $2+2n+50k$ sequences, each of them of length $2n + 2(q+4) + 2n(q+3) + k(75+50)$, where $q = p - n + 5$. Each sequence is in the form

$$a_1 b_1 a_2 b_2 \ldots a_n b_n e_1 e_2 \ldots e_{t(p)}, \tag{8.17}$$

where $t(p) = 2(q+4) + 2n(q+3) + k(75+50q)$. The a_i and b_i characters correspond to the logical variable x_i in Φ, and the e_j characters are additional characters. In these additional positions, all sequences contain character 0 except one of them. We will say that a sequence *contains x additional ones*, which means that there are x additional positions where the sequence

contains 1s. The sequences come in pairs such that they are the complement of each other in the first $2n$ positions. What follows is that there are 2^{2n} median sequences. The sequences are the following.

1. There is a sequence that contains all 0 characters in the first $2n$ positions and has $q + 4$ additional ones. Furthermore, there is a sequence that contains all 1s in the first $2n$ positions and contains $q+4$ additional ones. We denote these sequences as A and $\overline{A}$.

2. For each index $i = 1, \ldots n$, there are a couple of sequences. For one of them $a_i = b_i = 1$, and for all other $j \neq i$ $a_i = b_i = 0$, and the sequence contains $q+3$ additional ones. The other sequence is the complement of the first one in the first $2n$ positions and also contains $q + 3$ additional ones. We denote these sequences by A_i and $\overline{A}_i$.

3. For each clause, there are 50 sequences, see Table 8.2. Each sequence differs in the characters corresponding the logical variables participating in the clause, in the characters corresponding to other logical variables and in the number of additional ones. In Table 8.2, column **A** gives the characters $a_{i_1}, b_{i_1}, a_{i_2}, b_{i_2}, a_{i_3}$ and b_{i_3} for each sequence if the clause is $(x_{i_1} \vee x_{i_2} \vee x_{i_3})$. If some of the literals are negated, the corresponding a and b values must be swapped. In each sequence, for all $j \neq i_1, i_2, i_3$, all characters a_j and b_j are the same. Column **B** tells if it is 1 or 0. Each sequence has q additional ones plus the number that can be found in column **C**. These 3 columns completely describe the sequences; the remaining columns in the table are explained later.

   It is easy to see that there are 2^{2n} median sequences; the first $2n$ characters might be arbitrary, and the characters in the additional positions must be 0. We set up three properties on the medians.

   Property 1. Exactly n of the characters are 1 in the median.

   Property 2. For each i, $a_i + b_i = 1$.

   Property 3. For each i, $a_i + b_i = 1$, and the assignment

$$x_i = \begin{cases} TRUE & \text{if } a_i = 1 \\ FALSE & \text{if } a_i = 0 \end{cases} \tag{8.18}$$

   satisfies Φ.

It is easy to see that these properties are nested, namely, if a median sequence has Property i, then it also has Property j for each $j < i$. We prove the following on the median sequences.

If a median sequence M does not have Property 1, then the number of corresponding scenarios can be divided by p. Indeed, in such a case, either $H(M, A) \geq p$ or $H(M, \overline{A}) \geq p$ and thus either $H(M, A)!$ or $H(M, \overline{A})!$ can be divided by p.

A	**B**	**C**	**M1** 1 1 1	**M2** 1 1 0	**M3** 1 0 1	**M4** 0 1 1	**M5** 1 0 0	**M6** 0 1 0	**M7** 0 0 1	**M8** 0 0 0
01 00 00	0	+3	$p-1$	$p-1$	$p-1$	$p-3$	$p-1$	$p-3$	$p-3$	$p-3$
00 01 00	0	+3	$p-1$	$p-1$	$p-3$	$p-1$	$p-3$	$p-1$	$p-3$	$p-3$
00 00 01	0	+3	$p-1$	$p-3$	$p-1$	$p-1$	$p-3$	$p-3$	$p-1$	$p-3$
10 11 11	1	+0	$p-6$	$p-6$	$p-6$	$p-4$	$p-6$	$p-4$	$p-4$	$p-4$
11 10 11	1	+0	$p-6$	$p-6$	$p-4$	$p-6$	$p-4$	$p-6$	$p-4$	$p-4$
11 11 10	1	+0	$p-6$	$p-4$	$p-6$	$p-6$	$p-4$	$p-4$	$p-6$	$p-4$
10 10 00	0	+2	$p-5$	$p-5$	$p-3$	$p-3$	$p-3$	$p-3$	$p-1$	$p-1$
10 00 10	0	+2	$p-5$	$p-3$	$p-5$	$p-3$	$p-3$	$p-1$	$p-3$	$p-1$
00 10 10	0	+2	$p-5$	$p-3$	$p-3$	$p-5$	$p-1$	$p-3$	$p-3$	$p-1$
10 10 00	0	+2	$p-5$	$p-5$	$p-3$	$p-3$	$p-3$	$p-3$	$p-1$	$p-1$
10 00 01	0	+2	$p-3$	$p-5$	$p-3$	$p-1$	$p-5$	$p-3$	$p-1$	$p-3$
00 10 01	0	+2	$p-3$	$p-5$	$p-1$	$p-3$	$p-3$	$p-5$	$p-1$	$p-3$
10 01 00	0	+2	$p-3$	$p-3$	$p-5$	$p-1$	$p-5$	$p-1$	$p-3$	$p-3$
10 00 10	0	+2	$p-5$	$p-3$	$p-5$	$p-3$	$p-3$	$p-1$	$p-3$	$p-1$
00 01 10	0	+2	$p-5$	$p-1$	$p-5$	$p-3$	$p-3$	$p-1$	$p-5$	$p-3$
01 10 00	0	+2	$p-3$	$p-3$	$p-1$	$p-5$	$p-1$	$p-5$	$p-3$	$p-3$
01 00 10	0	+2	$p-3$	$p-1$	$p-3$	$p-5$	$p-1$	$p-3$	$p-5$	$p-3$
00 10 10	0	+2	$p-5$	$p-3$	$p-3$	$p-5$	$p-1$	$p-3$	$p-3$	$p-1$
10 01 00	0	+2	$p-3$	$p-3$	$p-5$	$p-1$	$p-5$	$p-1$	$p-3$	$p-3$
10 00 01	0	+2	$p-3$	$p-5$	$p-3$	$p-1$	$p-5$	$p-3$	$p-1$	$p-3$
00 01 01	0	+2	$p-1$	$p-1$	$p-3$	$p-1$	$p-5$	$p-3$	$p-3$	$p-5$
01 10 00	0	+2	$p-3$	$p-3$	$p-1$	$p-5$	$p-1$	$p-5$	$p-3$	$p-3$
01 00 01	0	+2	$p-1$	$p-3$	$p-1$	$p-3$	$p-3$	$p-5$	$p-3$	$p-5$
00 10 01	0	+2	$p-3$	$p-5$	$p-1$	$p-3$	$p-3$	$p-5$	$p-1$	$p-3$
01 01 00	0	+2	$p-1$	$p-1$	$p-3$	$p-3$	$p-3$	$p-3$	$p-5$	$p-5$
01 00 10	0	+2	$p-3$	$p-1$	$p-3$	$p-5$	$p-1$	$p-3$	$p-5$	$p-3$

Continued on next page

Table 8.2 – continued from previous page

A	**B**	**C**	**M1** 1 1 1	**M2** 1 1 0	**M3** 1 0 1	**M4** 0 1 1	**M5** 1 0 0	**M6** 0 1 0	**M7** 0 0 1	**M8** 0 0 0
00 01 10	0	+2	$p-3$	$p-1$	$p-5$	$p-3$	$p-3$	$p-1$	$p-5$	$p-3$
10 10 11	1	+1	$p-6$	$p-6$	$p-4$	$p-4$	$p-4$	$p-4$	$p-2$	$p-2$
10 11 01	1	+1	$p-4$	$p-6$	$p-4$	$p-2$	$p-6$	$p-4$	$p-2$	$p-4$
11 10 01	1	+1	$p-4$	$p-6$	$p-2$	$p-4$	$p-4$	$p-6$	$p-2$	$p-4$
10 01 11	1	+1	$p-4$	$p-4$	$p-6$	$p-2$	$p-6$	$p-2$	$p-4$	$p-4$
10 11 10	1	+1	$p-6$	$p-4$	$p-6$	$p-4$	$p-4$	$p-2$	$p-4$	$p-2$
11 01 10	1	+1	$p-4$	$p-2$	$p-6$	$p-4$	$p-4$	$p-2$	$p-6$	$p-4$
01 10 11	1	+1	$p-4$	$p-4$	$p-2$	$p-6$	$p-2$	$p-6$	$p-4$	$p-4$
01 11 10	1	+1	$p-4$	$p-2$	$p-4$	$p-6$	$p-2$	$p-4$	$p-6$	$p-4$
11 10 10	1	+1	$p-6$	$p-4$	$p-4$	$p-6$	$p-2$	$p-4$	$p-4$	$p-2$
10 01 11	1	+1	$p-4$	$p-4$	$p-6$	$p-2$	$p-6$	$p-2$	$p-4$	$p-4$
10 11 01	1	+1	$p-4$	$p-6$	$p-4$	$p-2$	$p-6$	$p-4$	$p-2$	$p-4$
11 01 01	1	+1	$p-2$	$p-4$	$p-4$	$p-2$	$p-6$	$p-4$	$p-4$	$p-6$
01 10 11	1	+1	$p-4$	$p-4$	$p-2$	$p-6$	$p-2$	$p-6$	$p-4$	$p-4$
01 11 01	1	+1	$p-2$	$p-4$	$p-2$	$p-4$	$p-4$	$p-6$	$p-4$	$p-6$
11 10 01	1	+1	$p-4$	$p-6$	$p-2$	$p-4$	$p-4$	$p-6$	$p-2$	$p-4$
01 01 11	1	+1	$p-2$	$p-2$	$p-4$	$p-4$	$p-4$	$p-4$	$p-6$	$p-6$
01 11 10	1	+1	$p-4$	$p-2$	$p-4$	$p-6$	$p-2$	$p-4$	$p-6$	$p-4$
11 01 10	1	+1	$p-4$	$p-2$	$p-6$	$p-4$	$p-4$	$p-2$	$p-6$	$p-4$
01 01 11	1	+1	$p-2$	$p-2$	$p-4$	$p-4$	$p-4$	$p-4$	$p-6$	$p-6$
01 11 01	1	+1	$p-2$	$p-4$	$p-2$	$p-4$	$p-4$	$p-6$	$p-4$	$p-6$
11 01 01	1	+1	$p-2$	$p-4$	$p-4$	$p-2$	$p-6$	$p-4$	$p-4$	$p-6$
01 01 01	0	+1	$p-2$	$p-3$	$p-3$	$p-1$	$p-5$	$p-5$	$p-5$	$p-7$
10 10 10	1	+2	$p-6$	$p-4$	$p-4$	$p-4$	$p-2$	$p-2$	$p-2$	p

Table 8.2: Constructing the 50 sequences for a clause. See text for explanation.

If a median sequence M has Property 1, but does not have Property 2, then the number of corresponding scenarios can be divided by p. Indeed, let i be such that $a_i + b_i = 0$ or $a_i + b_i = 2$. Then either $H(M, A_i) = p$ or $H(M, \overline{A}_i) = p$, making the corresponding factorial dividable by p.

If a median sequence M has Properties 1 and 2, but does not have Prop-

erty 3, then the number of corresponding scenarios can be divided by p. Assume that $c_j = (x_{i_1} \vee x_{i_2} \vee x_{i_3})$ is the clause that is not satisfied by the assignment defined in Equation (8.18). Then $a_{i_1} = a_{i_2} = a_{i_3} = 0$ and $b_{i_1} = b_{i_2} = b_{i_3} = 1$. In that case, the Hamming distance between M and the sequence that is defined for clause c_j in the last row of Table 8.2 is p. It follows that the number of corresponding scenarios can be divided by p. If some of the literals are negated in a clause not satisfied by the assignment defined in Equation (8.18), the same arguing holds, since both in the constructed sequences and in M, some of the a and b values are swapped.

If a median sequence M satisfies Property 3, then the number of corresponding scenarios are

$$(p-6)!^{7k}(p-5)!^{6k}(p-4)!^{12k}(p-3)!^{12k}(p-2)!^{6k+2n}(p-1)!^{7k+2} \qquad (8.19)$$

which cannot be divided by p. Indeed, if Property 3 holds, then $H(M, A) = H(M, \overline{A}) = p-1$ and for each i, $H(M, A_i) = H(M, \overline{A}_i) = p-2$. Since a clause $c_j = (x_{i_1} \vee x_{i_2} \vee x_{i_3})$ is satisfied, characters a_{i_1}, a_{i_2} and a_{i_3} in M are one of the combinations that can be found in the 7 columns in Table 8.2 labeled by **M1**–**M7** and the corresponding b characters are the complements. Then the Hamming distances between M and the 50 sequences defined for c_j are the values indicated in the appropriate column. It is easy to verify that in each column from **M1** to **M7**, there are $7-7$ $(p-6)$ and $(p-1)$, $6-6$ $(p-5)$ and $(p-2)$, and $12-12$ $(p-4)$ and $(p-3)$. If some of the literals are negated in a clause, then both in the constructed sequences and in the median sequences, the corresponding a and b values are swapped, and the same reasoning holds.

What follows is that only those median sequences contribute to the number of scenarios modulo p that have Property 3. Therefore,

$$\sum_{M \in \mathcal{M}} \prod_{A \in \mathcal{S}} H(M, A)! \equiv s(\Phi)(p-6)!^{7k}(p-5)!^{6k}(p-4)!^{12k} \times$$
$$\times (p-3)!^{12k}(p-2)!^{6k+2n}(p-1)!^{7k+2} \mod p \qquad (8.20)$$

where $s(\Phi)$ is the number of satisfying assignments of Φ. Since all the calculations in the reduction can be done in polynomial time, we get the following theorem.

Theorem 112. *The counting problem #SPS-STAR is in #P-complete.*

Proof. For each 3CNF Φ with n logical values and prime number $p > n$, we can create a problem instance in #SPS-STAR such that from the solution of the #SPS-STAR problem instance, the number of satisfying assignments of Φ modulo p can be computed in polynomial time. The number of satisfying assignmnts of Φ is between 0 and 2^n. Therefore if for sufficiently many prime numbers p, we compute the number of satisfying assignments modulo p, there will be only one solution between 0 and 2^n modulo the product of the prime numbers due to the Chinese Reminder Theorem. It is well known that the product of the prime numbers up to t is at least $\sqrt{2}^t$ and at most 4^t [30]. Therefore, the product of the prime numbers between n and $6n$ is surely at least 2^n. These prime numbers can be found in polynomial time, as well as the computations needed in the Chinese Reminder Theorem can be done in polynomial time. That is, the #3SAT problem is polynomial time reducible to the #SPS-STAR problem, therefore, the #SPS-STAR problem is #P-complete. □

Part IV

Non-trivial kernels and diameters of Markov chains

dc_2046_22

Chapter 9

Proving the pressing game conjecture for linear graphs

The negative result presented in Chapter 7 motivated the search of alternative Markov chains which are irreducible on the shortest reversal sorting paths. One possible way is the approach presented at Section 7.3 that considers the shortest reversal sorting paths as a sequence of DCJ operations. The kernel of a Markov chain perturbs a sequence of DCJ operations s into a sequence of DCJ operations s' such that the longest common subsequence of s and s' is at least their common length minus 4. The *four reversal conjecture* is that such a Markov chain will be irreducible. A weaker conjecture is that the conjecture holds for signed permutations coming from the infinite site model (see Definition 113 below). The biological relevance of the infinite site model was discussed on page 36 in the Preliminaries.

In this chapter, we prove the conjecture for signed permutations from the infinite site model whose overlap graphs are linear, that is, the components of their overlap graphs are paths. This was a common work with two undergraduate students of the BSM, Eliot Bixby and Toby Flint. The author of this thesis suggested the presented proving strategy, the students worked out the details. The original paper was published in Involve, 9(1):41-56, DOI: 10.2140/involve.2016.9.41.

Definition 113. *A signed permutation π is called a* permutation from the infinite site model *if its graph of desire and reality contains only cycles of length* 4 *and length* 2*, and furthermore, each component of its overlap graph contains at least one black vertex.*

The vertices of the simplified overlap graph *of a permutation π from the inifinite site model are the cycles of length 4. Two vertices are neighbors if their corresponding cycles overlap. A vertex is black if the desire edges in its corresponding cycles overlap, otherwise it is white.*

Let $G = (V, E)$ be a vertex colored simple graph, the vertices are colored by black and white. The pressing *of $v \in V$ means that*

- *all neighbors of v change color, that is, black vertices become white, white vertices become black,*
- *all pair of neighbors of v changes neighborhood. That is, for all pair $u, w \in \mathcal{N}(v)$ if (u, w) is an edge in the graph before the pressing, (u, w) will not be an edge after the pressing and vice versa,*
- *the vertex v is replaced by two isolated, white vertices.*

A successful pressing sequence *of a G is a series of pressing of its black vertices transforming G into an all-white, empty graph (empty means no edges).*

Theorem 114. *Let π be a signed permutation from the infinite site model. Then there is a one-to-one correspondence between its shortest reversal sorting paths and successful pressing sequences of its simplified overlap graph. The bijection between a shortest reversal sorting path $\pi = g_0, g_1, \ldots, g_k = id$ and a pressing sequence $v_1, v_2, \ldots, v_k$ is that for each i, the reversal transforming g_{i-1} to g_i acts on the desire edges of the cycle corresponding to v_i.*

Proof. First of all, we observe the following. If G is the simplified overlap graph of a signed permutation π from the infinite site model, and v is a black vertex in G, then pressing vertex v yields a simplified overlap graph of permutation π' that can be obtained by applying a reversal on the desire edges of the cycle corresponding to v. This is proved, for example, in [12].

Let A be the set of shortest reversal sorting paths of π, and let B be the set of successful pressing sequences of G. We show that the mapping $f : A \to B$ that maps each shortest reversal sorting path $\pi = g_0, g_1, \ldots, g_k = id$ to a sequence $v_1, v_2, \ldots, v_k$ such that for all i, v_i represents the cycle on which the reversal transforming g_{i-1} to g_i is indeed a mapping from A to B, and in fact, is an injection. Since $d_{REV}(\pi) = n + 1 - c(\pi)$, where n is the length of the signed permutation π, and $c(\pi)$ is the number of cycles in its graph of desire and reality, and the $c(id) = n + 1$, each sorting reversal must increase

the number of cycles by 1. This can be achieved only by applying a reversal on the desire edges of a cycle whose desire edges intersect. Therefore, this cycle and its corresponding vertex v exists. Such a reversal creates another signed permutation from the inifinite site model, thus, so exists the sequence $v_1, v_2, \ldots, v_k$. Since the simplified overlap graph of id is the all-white, empty graph, $v_1, v_2, \ldots, v_k$ is indeed a successful pressing sequence. Therefore f is indeed a mapping from A to B. Since a shortest reversal sorting path is unequivocally determined by its sequence of reversals, the mapping is an injection.

Now we set a mapping $g : B \to A$ such that the image of $v_1, v_2, \ldots v_k$ is $\pi = g_0, g_1, \ldots, g_k = id$ such that for each i, g_i is obtained by applying the reversal indicated by v_i on the permutation g_{i-1}. Since id is the only permutation whose simplified overlap graph is the all-white, empty graph, g_k is indeed id, and thus $\pi = g_0, g_1, \ldots, g_k = id$ is a reversal sorting path. Since in each step, the number of cycles is increased by 1, it is indeed a shortest reversal sorting path, that is, g is indeed a mapping from B to A. This mapping is also an injection, since a shortest reversal sorting path is unequivocally determined by its sequence of reversals. Furthermore, $g = f^{-1}$, thus f (and so g) is indeed a bijection. □

The correspondence between the shortest reversal sorting paths and successful pressing sequences motivated the study of successful pressing sequences on black and white graphs.

Let $G = (V, E)$ be vertex colored simple graph with colors black and white and consider the set of vertices as an alphabet. Any sequence over this alphabet is called a *pressing sequence*. It is a *valid* pressing sequence when each vertex is black when it is pressed, and it is *successful* if it is valid and leads to the all-white, empty graph. The length of the pressing sequence is the number of vertices pressed in it. The following theorem is also true.

Theorem 115. *Let G be a black-and-white graph such that each component contains at least one black vertex. Then every successful pressing sequence of G has the same length.*

The proof can be found in [46]. We are ready to state the pressing sequence conjecture.

Conjecture 2. *Let G be a black-and-white graph such that each component contains at least one black vertex. Construct a metagraph, M whose vertices*

are the successful pressing sequences on G. Connect two vertices if the length of the longest common subsequence of the pressing sequences they represent is at least the common length of the pressing sequences minus 4. *The conjecture is that M is connected.*

The conjecture means that with small alterations, we can transform any pressing sequence into any other pressing sequence, regardless of the underlying graph. By "small alteration" we mean that we remove at most 4 (not necessarily consecutive) vertices from a pressing sequence, and add at most 4 vertices, not necessarily to the same places where vertices were removed, and not necessarily to consecutive places.

In this paper, we prove the pressing game conjecture for linear graphs. In addition, we can prove the metagraph will be already connected if we require that neighboring vertices have a longest common subsequence at least the common length of their pressing sequences minus 2.

9.1 Proof of the Conjecture on Linear Graphs

The proof of our main theorem is recursive, and for this, we need the following notations. Let G be a black-and-white graph, and v a black vertex in it. Then Gv denotes the graph we get by pressing vertex v. Similarly, if P is a valid pressing sequence of G (namely, each vertex is black when we want to press it, but P does not necessarily yield the all-white, empty graph), then GP denotes the graph we get after pressing all vertices in P in the indicated order. Finally, let P^k denote the suffix of P starting in position $k+1$.

The convenience of linear graphs is their simple structure and furthermore, their self-reducibility:

Observation 116. *Let G be a linear black-and-white graph and v a black vertex in it. Then Gv consists of a linear graph and the separated white vertex v.*

Since any separated white vertex does not have to be pressed again, it is sufficient to consider $Gv \setminus \{v\}$, which is a linear graph. We are ready to state and prove our main theorem.

Theorem 117. *Let G be an arbitrary, finite, linear black-and-white graph, and let M be the following graph. The vertices of M are the successful pressing sequences on G, and two vertices are adjacent if the length of the longest*

common subsequence of the pressing sequences they represent is at least the common length of the pressing sequences minus 2*. Then* M *is connected.*

Proof. It is sufficient to show that for any successful pressing sequences X and $Y = v_1v_2 \dots v_k$, there is a series $X_1, X_2, \dots X_m$ such that for any $i = 1, 2, \dots m-1$, the length of the longest common subsequence of X_i and X_{i+1} is at least the common length of the sequences minus 2, and X_m starts with v_1. Indeed, then both X_m and Y start with v_1, and both X_m^1 and Y^1 are successful pressing sequences on $Gv_1 \setminus \{v_1\}$. We can use induction to transform X_m into a pressing sequence which starts v_2, then we consider its suffix which is a successful pressing sequence on $Gv_1v_2 \setminus \{v_1, v_2\}$, etc.

Furthermore, it is sufficient to show that v_1 can be moved to some earlier position in some series of small alterations of the sequence, provided the intermediaries are also valid pressing sequences.

We first show that if v_1 is not in X, there exists some valid X' containing v_1, and X' differs from X by exactly one vertex. This is true for any vertex in any arbitrary overlap graph G and we state it in a separate lemma since we are going to use it again later.

Lemma 118. *Assume that* X *is a successful pressing sequence on* G *and that vertex* v *is not a separated vertex in* G*. Then either* v *is in* X *or there exists some valid* X' *containing* v*, and* X' *differs from* X *by exactly one vertex.*

Proof. Let $X = u_1u_2 \dots u_k$. Assume that v is not in X. Vertex v has at least one neighbor in G and none in GX, therefore there exists at least one vertex in X which, when pressed, is adjacent to v. Consider the last such vertex, which is in position i, and call it u_i; by definition none of the vertex pressings in X^i affect the adjacencies or color of v, so after pressing u_i, v must be a white disconnected vertex. It follows that in $Gu_1 \dots u_{i-1}$, v and u_i have exactly the same neighbors, and as such $u_1 \dots u_{i-1}vu_{i+1} \dots u_k$ is a valid pressing sequence. □

We now assume that v_1 is part of the current pressing sequence, which we denote by $P_1w_1v_1P_2$, where both P_1 and P_2 might be empty.

Case 1. If w_1 and v_1 are not neighbors in GP_1, then $P_1v_1w_1P_2$ is also a valid pressing sequence, and one of the longest common subsequences of $P_1w_1v_1P_2$ and $P_1v_1w_1P_2$ is $P_1w_1P_2$, one vertex less than the original pressing sequences. In this way, we can move v_1 to a smaller index position in the pressing sequence, and this is what we want to prove.

Case 2. If w_1 and v_1 are neighbors in GP_1, then v_1 is white in GP_1, and then pressing w_1 makes it black again. However, v_1 is black in G, since it is the first vertex in the valid pressing sequence Y. As such there must exist at least one vertex in P_1 which was adjacent to a black v_1 when pressed. Let w_2 be the last such vertex in P_1, and let us denote $P_1 = P_{1a}w_2P_{1b}$.

We claim that none of the vertices in P_{1b} are neighbors of w_2 in GP_{1a}. Indeed if there were such a neighbor, call it w_3, after pressing w_2, w_3 would be adjacent to v_1. Note that w_3 cannot have already been adjacent to v_1 by linearity of GP_{1a}. As such, pressing w_3, would change the color of v_1, meaning either v_1 was black prior to pressing w_1 – a contradiction – or there were further vertices in P_{1b} which were adjacent to a black v_1 when pressed, another contradiction.

Since P_{1b} does not contain a vertex which is a neighbor of w_2 in GP_{1a}, we move w_2 next to w_1. The new pressing sequence $P_{1a}P_{1b}w_2w_1v_1P_2$ is still a valid and successful pressing sequence and the longest common subsequence of P and $P_{1a}P_{1b}w_2w_1v_1P_2$ is $P_{1a}P_{1b}w_1v_1P_2$, one vertex less than the common length of the sequences.

For sake of simplicity, we denote $P_{1a}P_{1b}$ by P_1' and now we can assume the pressing sequence is of the form $P_1'w_2w_1v_1P_2$, with P_1' and P_2 both potentially empty. Since after pressing w_2, w_1 and v_1 become neighbors with w_1 being black and v_1 being white, the topology and colors of w_2, w_1 and v_1 in GP_1' is one of the following:

Case 2a. Assume that P_2 is not empty. The $\{w_1, w_2, v_1\}$ triplet has at least one neighbor (and at most two) in GP_1'; call them u_1 and u_2. Furthermore, either (1) one of u_1 and u_2 is pressed in P_2, or (2) we can replace some vertex in P_2 with u_1 or u_2, such that the resulting sequence is still valid, and successful on $GP_1'w_2w_1v_1$, due to Lemma 118. As such, we can assume that at least one neighbor of the $\{w_1, w_2, v_1\}$ triplet is pressed in P_2.

Without loss of generality, say u_1 is pressed before u_2 in P_2 and let $P_2 = P_{2a}u_1P_{2b}$. Note that we can press v_1 instead of $w_2w_1v_1$, and the resulting sequence $GP_1'v_1P_{2a}$ will be valid, as none of the vertices in P_{2a} are neighbors of w_2, w_1, or v_1. Next note from Figure 9.1, that the colors of u_1 and u_2 are identically altered in the pressing of either v_1 or $w_2w_1v_1$, and so we can press u_1. Figure 9.2 shows that the color of u_2 and a possible second neighbor of

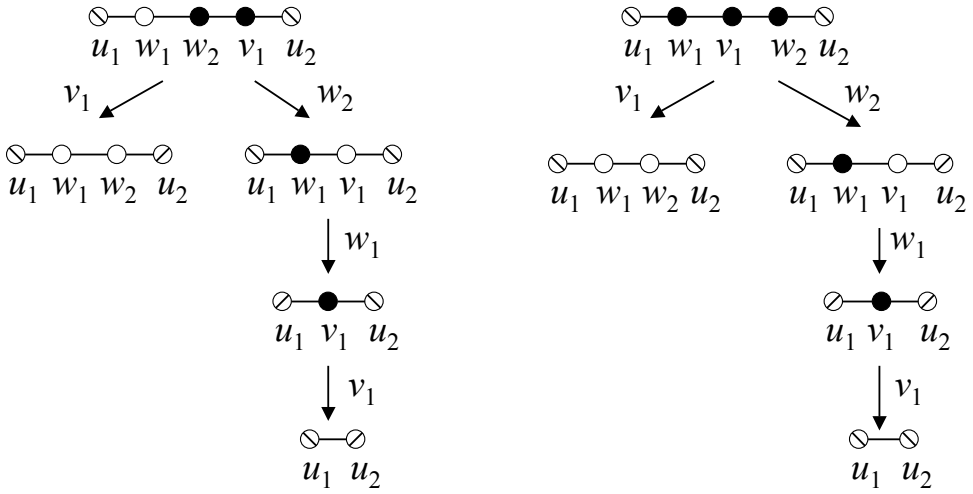


Figure 9.1: In the indicated two configurations, the neighbors of the $\{w_1, w_2, v_1\}$ triplet, u_1 and u_2, change color in the same way by pressing only v_1 and pressing $w_2w_1v_1$. The color change on u_1 and u_2 is indicated with the flipping of their crossing line.

u_1 denoted by u_3 will be the same in $GP_1'w_2w_1v_1P_{2a}u_1$ and $GP_1'v_1P_{2a}u_1w_1w_2$. Therefore $P_1'v_1P_{2a}u_1w_1w_2P_{2b}$ will also be a successful pressing sequence on G, since no more vertices are affected by the given alteration of the pressing sequence. One of the longest common subsequences of $P_1'w_2w_1v_1P_{2a}u_1P_{2b}$ and $P_1'v_1P_{2a}u_1w_1w_2P_{2b}$ is $P_1'v_1P_{2a}u_1P_{2b}$, 2 vertices less than the entire pressing sequences. As intended, we have shown that v_1 is in a smaller index position of the pressing sequence.

Case 2b Finally, assume that P_2 is empty. Then $GP_1'w_2w_1v_1$ is the all-white empty graph, and thus, $GP_1'w_2w_1$ contains the separated black v_1 and all separated white vertices, or contains a black v_1 connected to another black vertex and all separated and white vertices.

What follows is that GP_1' contains at most 4 non-isolated vertices, 3 of which are w_2, w_1, and v_1. Call the fourth u. If u exists, it must be black and adjacent to v_1 when v_1 is pressed. There are only 4 such cases, given the possible topologies for w_2, w_1, and v_1. If w_1 and w_2 are adjacent, then u is either black and adjacent to v_1 in GP_1' or it is adjacent to w_2 and is white. If w_2 and w_1 are not adjacent, then u can be adjacent to either w_2 or w_1, and must be white in both cases.

Note that all of these topologies can be described as follows; all neighbors

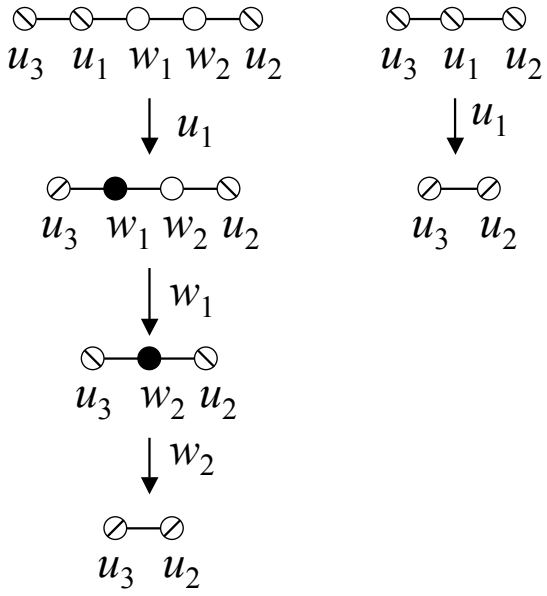


Figure 9.2: The color of u_2 and u_3 changes in the same way on the two indicated configurations. See text for details.

of v_1 are black, v_1 is black, and all other vertices are white. This motivates the following lemma:

Lemma 119. *If GP is such that all neighbors of v_1 are black, v_1 is black, and all other vertices are white, and furthermore, there is a successful pressing sequence on G that starts with v_1, then there exists at least one vertex u in P such that when u is pressed u is not adjacent to v_1.*

Proof. Suppose instead that every vertex in P is adjacent to v_1 when pressed. P cannot be empty since then GP would be G and pressing v_1 in G would create an all-white non-trivial graph, contradicting that there exists a successful pressing sequence starting with pressing v_1. Furthermore if all vertices in P are neighbors of v_1 when pressed, then P must contain an even number of vertives since v_1 is black both in G and GP.

Let $P = P_1'u_2u_1$. In order for u_1 and u_2 to be adjacent to v_1 when pressed, and for GP to fit the given criteria, GP_1' must also have v_1 and all neighbors black, and all other vertices white. By repeated application, we see that G must also fit these criteria. By assumption then, there are no black vertices not adjacent to v_1, and as such, pressing v_1 results in an all-white non-trivial

graph. However this is a contradiction, as there exists a successful pressing sequence for G in which v_1 is pressed first. □

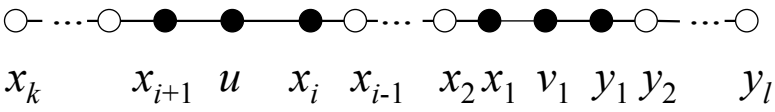
From the above lemma, we have that there exists some vertex in P_1' not adjacent to v_1 when pressed, and there are vertices which are adjacent to v_1 when pressed. For technical reasons, we have to separate them in the pressing sequence, which is doable due to the following lemma.

Lemma 120. *Let Pxu be a valid pressing sequence on G such that x is a neighbor of some v in GP and u is not a neighbor of v in GPx. Then Pux is a valid pressing sequence on G and $GPxu = GPux$.*

Proof. It is sufficient to show that x and u are not neighbors in GP. If x and u were neighbors, then the two neighbors of x would be u and v, causing u and v to become neighbors in GPx, a contradiction. □

Due to Lemma 120 it is possible to 'bubble up' vertices that are not neighbors of v_1 in the pressing sequence so that the pressing sequence becomes $P_uP_nv_1$ where P_u contains the vertices that are not neighbors of v_1 when pressed and P_u contains the vertices that are neighbors of v_1 when pressed. Each bubbling up step is allowed as the length of the longest common subsequence of two consecutive sorting sequences is their common length minus 1. We know that neither P_u nor P_n is empty due to Lemma 119 and due to the fact that w_1 and w_2 are in P_n.

Let u be the last vertex in P_u and let $P_u = P_u'u$. Without loss of generality, we can assume that u is on the left hand side of v_1 in GP_u' and then GP_u' is

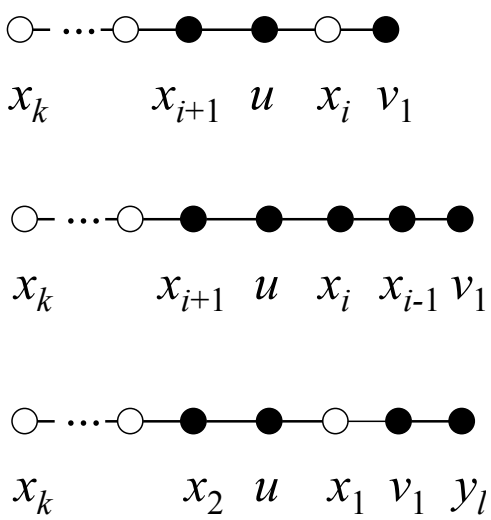


The vertices on the left hand side of v_1 are denoted by $x_1, x_2 \dots x_k$ and we distinguish u amongst them. The vertices on the right hand side of v_1 are denoted by $y_1, y_2, \dots y_l$.

Obviously, no x is a neighbor of any y when pressed, so we can bubble up the y vertices in P_n such that first the y vertices are pressed and then the x vertices. After a finite number of allowed alterations, $P_n = y_1y_2 \dots y_lx_1x_2 \dots x_k$.

Similarly to the previous cases, we can move down vertex u in the pressing sequence before x_i. We know that v_1 is black in $GP_u'u$ since it is black in G

and neither of its neighbors is pressed in $P'_u u$. We are going to press some of the vertices amongst the x and y vertices provided that v_1 will be black after that series of pressing. We consider the graph $GP'_u y_1 \dots y_l x_1 \dots x_{i-1}$ if v_1 is black in it (the runs of x vertices might be empty if $i = 1$), and otherwise the graph $GP'_u y_1 \dots y_l x_1 \dots x_{i-2}$ (also the runs of x vertices might be empty if $i = 2$) or $GP'_u y_1 \dots y_{l-1}$ if $i = 1$ and the number of y vertices is odd (if $i = 1$ and the number of y vertices is even, then v_1 will be black in $GP'_u y_1 \dots y_l$). We have one of the following graphs



on which $ux_i \dots x_k v_1$, $ux_{i-1} \dots x_k v_1$, $y_l u x_1 \dots x_k v_1$ is the current successful pressing sequence, respectively.

A successful pressing sequence replacing $ux_i \dots x_k v_1$ is $v_1 x_i \dots x_k u$, as can be seen on the left hand side of Figure 9.3. The length of the longest common subsequence of the two pressing sequences is 2 less than their common length, as required. The pressing sequence $y_l u x_1 \dots x_k v_1$ can be replaced by $ux_1 y_l x_2 \dots x_k v_1$ since y_l is a neighbor of neither u nor x_1. Then this pressing sequence can be replaced by $v_1 x_1 y_l x_2 \dots x_k u$, as can be seen on the right hand side of Figure 9.3. The length of the longest common subsequence of $ux_1 y_l x_2 \dots x_k v_1$ and $v_1 x_1 y_l x_2 \dots x_k u$ is again 2 less than their common length.

Finally, the pressing sequence $ux_{i-1} \dots x_k v_1$ can be replaced in two steps, first it is changed to $x_i x_{i+1} u x_{i-1} x_{i+2} \dots x_k v_1$, then to $x_i x_{i+1} v_1 x_{i-1} x_{i+2} \dots x_k u$, as can be checked in Figure 9.4. In both steps, the length of the longest common subsequences of two consecutive pressing sequences is 2 less than their common length as required.

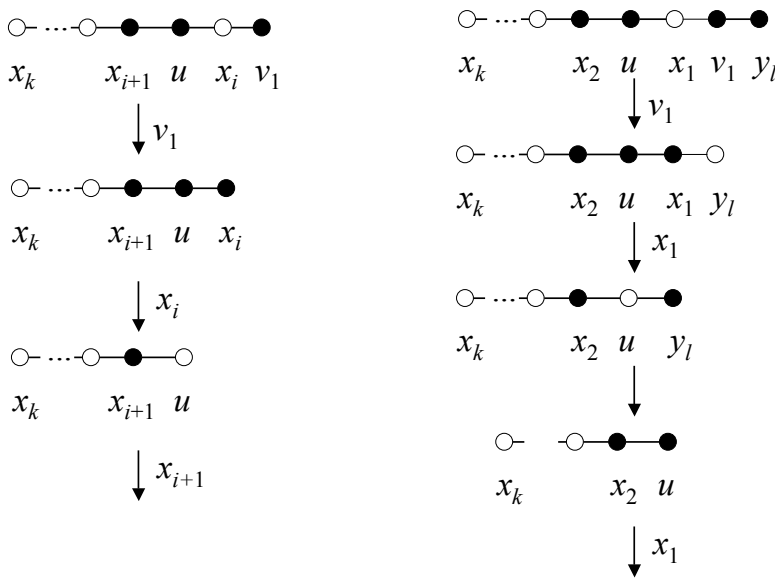


Figure 9.3: Alternative pressing sequences for two cases. See text for details.

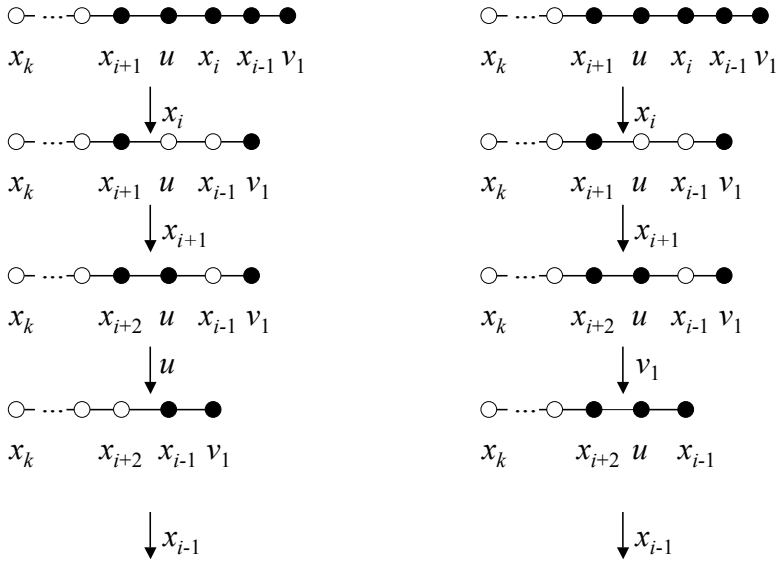


Figure 9.4: Changing the pressing sequence $ux_{i-1} \dots x_k v_1$ in two steps such that v_1 is in a smaller index position. See text for details.

We proved that in any case, v_1 can be moved into a smaller index position with a finite series of allowed perturbations. Iterating this, we can move v_1 to the first position. Then we can do the same thing with v_2 on the graph $Gv_1 \setminus \{v_1\}$, and eventually transform X into Y with allowed perturbations. □

9.2 Discussion and Conclusions

In this chapter, we proved the pressing game conjecture for linear graphs. Although the linear graphs are very simple, this proof technique provides a direction for proving the general case. Indeed, it is generally true that if a vertex v is not in a successful pressing sequence P, then a successful pressing sequence P' exists which contains v and the length of the longest common subsequence of P and P' is only 1 less than their common length. Case 1 in the proof of Theorem 117 holds for arbitrary graphs, and in an unpublished paper, we were able to prove that the conjecture is true for Case 2a using linear algebraic techniques similar to that one used in [46]. The only missing part is Case 2b, which seems to be very complicated for general graphs, for example, Lemma 120 cannot be generalized for arbitrary graphs.

A stronger theorem holds for the linear case that is conjectured for the general case. One possible direction above proving the general conjecture is to study the emerging Markov chain on the solution space of the pressing game on linear graphs. We proved that a Markov chain that randomly removes two vertices from the current pressing sequence, adds two random vertices to it, and accepts it if the result is a successful pressing sequence is irreducible. Using Markov chain Monte Carlo, it is easy to set the jumping probabilities of the Markov chain such that it converges to the uniform distribution of the solutions. The remaining question is the speed at which this Markov chain converges.

Chapter 10

Cooling down DCJ scenarios to reversal scenarios

One of the main criticism of the DCJ model is that it allows mixed genomes containing both linear and circular chromosomes. Although such genomes do exist, most of the species contain either all circular or all linear chromosomes. Particularly, the vast majority of multicelluar Eukaryotes contains only linear chromosomes. Therefore, biologists prefer a genome rearrangement model that does not create mixed genomes. We can clearly see here a trade-off between model fidelity/accuracy and its computability. Indeed, the DCJ model describes the genome rearrangement less accurately, however, there is an FPAUS algorithm to sample almost uniformly most parsimonious DCJ scenarios between two genomes (see also Chapter 6). The reversal model is more accurate, however, the complexity status of almost uniformly sampling most parsimonious reversal scenarios is unknown.

In this chapter, we present an idea that the DCJ model can actually be used to sample almost uniformly shortest reversal sorting scenarios. The original work was a joint work with Eric Tannier and was published in Bioinformatics, 26: 3012-3019, `https://doi.org/10.1093/bioinformatics/btq574`. The author of this thesis conjectured that all local minima in the energy surface defined below are global minima, in which Eric Tannier did not believe first. He set up cases that seemed to be local but not global minima, and the author of this thesis wrote a computer program that used stochastic search to find a direction from those points to the global minima. From these examples, they proved the conjecture together. Theorem 125 was proved by the author of this thesis.

We introduce a parallel Markov chain method, in which a series of Markov chains converges to the Boltzmann distribution of most parsimonious DCJ scenarios between two linear, unichromosomal, hurdle-free, co-tailed genomes, Π and Γ. Hurdle-free means that in the overlap graph of the signed permutation $\Gamma^{-1}\Pi$, there is no non-trivial, all-white component. Co-tailed means that the two genomes have the same telomers. For such genomes,

$$d_{DCJ}(\Pi, \Gamma) = d_{REV}(\Gamma^{-1}\Pi). \tag{10.1}$$

That is, the shortest reversal sorting paths of $\Gamma^{-1}\Pi$ is a subset of the shortest DCJ sorting paths from Π to Γ.

The hypothetical "energy" of a most parsimonious DCJ scenario is set in such a way that the minimum energy scenarios are the shortest reversal sorting paths of $\Gamma^{-1}\Pi$. Since the two genomes have the same telomers, the adjacency graph of Γ and Π contains only cycles. Therefore, any sorting DCJ operation cuts two adjacencies and creates two new ones. In terms of manipulating the chromosomes, it is either a reversal or cutting out a circular chromosome from a chromosome or fusing a circular chromosome into another chromosome. If a DCJ sorting path contains only reversals, then there is no circular chromosome presented in any of the intermediate genomes. This motivates the following definition.

Definition 121. *For a DCJ scenario S between genomes Π and Γ, denote by $S(i)$ the DCJ at the i^{th} position on S. Let Π/s denote the genome obtained by applying the DCJ operation s on genome Π, and let $\Pi_i^S := \Pi/S(1)/\ldots/S(i)$ for $0 \leq i \leq d_{DCJ}$ ($\Pi_0^S = \Pi$ and $\Pi_{d_{DCJ}}^S = \Gamma$). Let*

$$c(S) := \sum_{i=1}^{d_{DCJ}(\Pi,\Gamma)-1} circ(\Pi_i^S) \tag{10.2}$$

be the energy *of the scenario S, where $circ(\Pi)$ is the number of circular chromosomes in genome Π.*

For any DCJ scenario S between two unichromosomal co-tailed genomes, $c(S) \geq 0$ and S is a scenario of reversals if and only if $c(S) = 0$.

On co-tailed, unichromosomal linear genomes, DCJs are either reversals, or fusions or fissions of chromosomes involving at least one circular chromosome. A fission of a chromosome into two, immediately followed by the fusion

of these two chromosomes (with different points of fusions) is called a *generalized block-interchange*. In the literature of genome rearrangements, the block-interchange means the swap of two, non-necessarily consecutive blocks. When the circular chromosome is fused back to the original chromosome in a reversed order, then the overlap graph of the so-obtained permutation will not contain an all-white non-trivial component. In this thesis, the generalized block-interchanges are called block-interchanges for short.

The following two results are corollaries of the Hannenhalli-Pevzner theorem [45].

Corollary 122. *For two hurdle-free co-tailed unichromosomal genomes* Π *and* Γ*, there is a DCJ scenario of size* $d_{DCJ}(\Pi, \Gamma)$ *which contains only reversals.*

Corollary 123. *In any DCJ scenario between two co-tailed unichromosomal genomes* Π *and* Γ *of size* $d_{DCJ}(\Pi, \Gamma)$*, there is no reversal involving gene extremities of an unoriented component (of the graph of desire and reality of* $\Gamma^{-1}\Pi$*).*

The number of different DCJ scenarios between two hurdle-free co-tailed unichromosomal genomes Π and Γ can be easily computed [70], while computing the number of DCJ scenarios containing only reversals, as well as only reversals and block-interchanges, are open problems.

For two scenarios S_1 and S_2, we define $d(S_1, S_2)$ as the smallest integer d such that there exists k verifying:

$$\text{For all } i \notin [k, k+d-1], \quad S_1(i) = S_2(i).$$

In other words, $d(S_1, S_2) \leq d$ if it is possible to replace d consecutive DCJs of S_1, by d other DCJs to obtain S_2.

10.1 Transforming DCJ scenarios to reversal scenarios with small perturbations

In this section, we prove that small perturbations are sufficient to transform any DCJ scenario to a reversal scenario in such a way that the energy function of the scenarios are monotonously decreasing. With other words, if we define a topology of DCJ scenarios such that two scenarios are neighbours if small perturbations are sufficient to transform one another, then all local optima in this topological space are global ones.

Theorem 124. *For two hurdle-free co-tailed unichromosomal genomes* Π *and* Γ*, let* S_1 *be a DCJ scenario transforming* Π *into* Γ*. There exists a finite sequence* $S_1 S_2 \dots S_k$ *of DCJ scenarios, such that*

- *for all* i, $d(S_i, S_{i+1}) \leq 3$*;*
- *for all* i, $c(S_i) \geq c(S_{i+1})$*;*
- S_k *is a reversal scenario.*

Proof. Let $S_1 = \rho_1 \dots \rho_k$ be a scenario between unichromosomal co-tailed hurdle-free genomes Π and Γ such that $c(S_1) > 0$. We prove that there always exists a finite sequence $S_1 S_2 \dots S_l$ such that $c(S_l) < c(S_1)$ and for all i, $d(S_i, S_{i+1}) \leq 3$ and $c(S_i) \geq c(S_{i+1})$. This proves the theorem.

A DCJ scenario that is not a reversal scenario contains fissions and fusions. If in a DCJ scenario every fission is immediately followed by a fusion, then we say it is a reversal/block-interchange scenario. The first step of our proof shows that any DCJ scenario can be transformed into a scenario of this type. The second step of the proof shows that this can further be transformed into a reversal scenario.

Case 1. Not all fission is immediately followed by a fusion.
Let ρ_p be the first (with minimum p) such fission in S_1. So $\Pi_{p-1}^{S_1}$ is a unichromosomal genome. The DCJ ρ_p fissions the unique chromosome of $\Pi_{p-1}^{S_1}$ into two chromosomes C_1 and C_2. Let G_1 (resp. G_2) be the set of gene extremities in C_1 (resp. C_2). Now let ρ_q be the first DCJ after ρ_p in S_1 which involves gene extremities both from G_1 and G_2. It exists as Γ is unichromosomal, and by hypothesis $q > p + 1$.

By hypothesis on ρ_q, ρ_{q-1} involves either only gene extremities from G_1 or only gene extremities from G_2. Suppose w.l.o.g. that it is G_1. If ρ_q and ρ_{q-1} commute, let $\rho'_{q-1} = \rho_q$ and $\rho'_q = \rho_{q-1}$. If they do not commute, let ρ_{q-1} be $(a, b|c, d)$ and let ρ_q be $(a, c|e, f)$, with a, b, c, d being gene extremities of G_1 and e, f being gene extremities from G_2. Now let $\rho'_{q-1} = (a, b|e, f)$ and $\rho'_q = (c, d|f, b))$. In both cases the scenario

$$S_2 = \rho_1 \dots \rho_{q-2} \rho'_{q-1} \rho'_q \rho_{q+1} \dots \rho_n$$

is composed of valid DCJ operations, we trivially have $d(S_1, S_2) \leq 2$ and we also have $c(S_2) \leq c(S_1)$ because

- For all $i \neq q - 1$, $\Pi_i^{S_1} = \Pi_i^{S_2}$;

- Clearly $|circ(\Pi) - circ(\Pi/\rho)| \leq 1$ for any genome Π and DCJ ρ, so $circ(\Pi_{q-1}^{S_1}) \geq circ(\Pi_{q-2}^{S_1}) - 1$;

- The DCJ ρ'_{q-1} is a chromosome fusion so $circ(\Pi_{q-1}^{S_2}) = circ(\Pi_{q-2}^{S_2}) - 1$, yielding $circ(\Pi_{q-1}^{S_1}) \geq circ(\Pi_{q-1}^{S_2})$.

Now in S_2, ρ'_{q-1} is the first DCJ after ρ_p which involves gene extremities both from G_1 and G_2. Applying this transformation again to S_2 decreases the index q of the first DCJ after ρ_p which involves gene extremities both from G_1 and G_2. We may apply the same transformation until $q = p + 1$, which means that $\rho_p\rho_{p+1}$ is a block-interchange.

Applying the same transformation to every ρ_p fission which is the first DCJ of a non-block-interchange type eventually gives a reversal/block-interchange scenario.

Case 2. The scenario S_1 is a reversal/block-interchange scenario.

Note that in that case the number of circular chromosomes in a scenario — its energy — is also the number of block-interchanges. So the goal will be to get progressively rid of all block-interchanges and arrive at a reversal scenario.

First, if for a block-interchange $\rho_p\rho_{p+1}$ (both operations are DCJs), the pemutation $(\Pi_{p+1}^S)^{-1}\Pi_{p-1}^S$ have an oriented component, then $\rho_p\rho_{p+1}$ can easily be replaced by two reversals, as a direct consequence of Corollary 122. Doing this yields a scenario which is distant from the initial one of at most two, and the number of circular chromosomes is deceased by one, proving the theorem.

So we may assume that all block interchanges $\rho_p\rho_{p+1}$ are *unoriented*, which means the graph of desire and reality of $(\Pi_{p+1}^S)^{-1}\Pi_{p-1}^S$ and has only unoriented components.

Let now ρ_r and ρ_b be a reversal and a block interchange in the scenario S. We note Π^- the genome before the application of the first event among ρ_r and ρ_b, and Π^+ the genome after the application of the last one. We say that ρ_r and ρ_b *cross* if the extremities on which ρ_r andρ_b act are in the same component of the overlap graph of $(\Pi^+)^{-1}\Pi^-$. It is easy to see that if they are consecutive and do not cross, then they commute and swapping their position yields a scenario which is distant of 3 from the original one, and has the same score (the reversal stays a reversal, the block-interchange stays a block interchange).

Choose now ρ_r and ρ_b which cross, and such that there are as few DCJs as possible between them in S. These exist because else the reversals and block-interchange would act on different components, which is not possible because there is no unoriented component in $\Gamma^{-1}\Pi$.

From now we assume that ρ_r is before ρ_b, but a symmetric reasoning yields the proof for the opposite case.

Suppose there are rearrangements between ρ_r and ρ_b. If among those rearrangements, there is a reversal which is immediately before a block interchange, then by hypothesis on ρ_r and ρ_b, they do not cross. So it is possible to swap them without changing the score of the scenario, nor the crossing properties of any pair or reversal and block-interchange. Iteratively applying this allows to assume that all reversals occur after all block-interchanges between ρ_r and ρ_b.

Now ρ_r occurs before a block-interchange, and if it is not ρ_b, then they don't cross. So it is possible to swap them. This has the effect of applying a block-interchange to Π^-. As this block-interchange does not cross any reversal applied before ρ_b, it cannot change the crossing properties of reversals and block-interchanges, so it is possible to repeatedly apply this procedure while there are block-interchanges between ρ_r and ρ_b, there are only reversals left. In the same way, as ρ_b occurs after a reversal, if it is not ρ_r then they don't cross and it is possible to swap them without changing the crossing properties of reversals and block-interchanges. After repeating this procedure there are no rearrangement anymore between ρ_r and ρ_b and they are immediately consecutive. The following observation yields the theorem in that case.

In a scenario S, let ρ_r and ρ_b be respectively a reversal and a block-interchange that are consecutive (in any order) and crossing. Then it is possible to replace them by three reversals in S.

Indeed, by Corollary 123 the component of Π^- and Π^+ containing the gene extremities involved in ρ_r and ρ_b is is oriented since there is a scenario with a reversal transforming Π^- into Π^+. So by Result 122 there are three reversals transforming Π^- into Π^+, which proves the result. □

10.2 Parallel tempering

The Parallel Tempering is a Markov chain Monte Carlo technique in which parallel Markov chains are given. The i^{th} Markov chain converges to the

distribution

$$\pi_i(R(\Pi,\Gamma)) \propto e^{-\frac{c(R(\Pi,\Gamma))}{T_i}} \tag{10.3}$$

$R(\Pi,\Gamma)$ is a most parsimonious DCJ scenario between Π and Γ, and T_i is the hypothetical temperature of the chain. It is easy to see that swapping the states of two consecutive chain with acceptance probability

$$\min\left\{1, \frac{e^{-\frac{c(R_i)}{T_{i+1}}} \times e^{-\frac{c(R_{i+1})}{T_i}}}{e^{-\frac{c(R_i)}{T_i}} \times e^{-\frac{c(R_{i+1})}{T_{i+1}}}}\right\} \tag{10.4}$$

does not change the equilibrium distribution of the states [39].

In this section, we prove the following theorem, showing that with a non-negligible probability, the MCMC can diversify efficiently in the solution space of reversal scenarios:

Theorem 125. *For any pair of hurdle-free, co-tailed, linear genomes* Π *and* Γ *with* n *genes,* $k = O(n^3 \log(n))$ *parallel chains sampling from most parsimonious DCJ scenarios following target distributions given by Equation 10.3 with the following properties:*

- *The temperature of the* 1^{st} *chain is infinite,*
- *The swapping probability between any two consecutive chains given by Equation 10.4 is at least* $\frac{1}{2}$*,*
- *The probability of a reversal scenario in the target distribution of the* k^{th} *chain is at least* $\frac{1}{2}$*.*

The first property provides that all most parsimonious DCJ scenarios are equally probable in the target distribution of the 1^{st} chain. We proved that it is easy to sample from this distribution with Markov chains, see Chapter 6, and if the genomes are co-tailed, exact sampling is also possible [70]. The second property provides that the information change between the parallel chains is not negligible. The third property provides that a few samples from the k^{th} chain is sufficient to get reversal scenarios. Although these together do not prove fast mixing of our method, it is definitely takes us closer to a final proof.

The theorem is proved using the following lemmas.

Lemma 126. *For any most parsimonious DCJ scenario S between Π and Γ,*

$$c(S) \leq \frac{n(n-2)}{4} \tag{10.5}$$

Proof. Since Π and Γ are linear, $circ(\Pi) = circ(\Gamma) = 0$. $|circ(\Pi') - circ(\Pi/\rho)| \leq 1$ for any genome Π', thus $c(S)$ is maximal, if the number of circles increases till the middle of the path, and then dicreases. Since the maximum length of the path is $n-1$, Equation 10.5 immediately holds. □

Lemma 127. *The number of most parsimonious DCJ scenarios between Π and Γ is at most $(4n^2-n)^{n-1}$.*

Proof. n genes have $2n$ extremities, forming at most $2n$ telomeres and adjacencies. There are at most 2 DCJs acting on a given pair of telomeres and/or adjacencies, having an upper bound of $2\binom{2n}{2}$ DCJs acting on a pair of telomers/adjacencies. Above these, there are fissions involving one adjacency. The number of them is at most n, thus the number of DCJs applicable for a genome with n genes cannot be more than $4n^2 - n$. The length of a most parsimonious DCJ scenario is at most $n-1$, thus the number of most parsimonious DCJ scenario is a most $(4n^2-n)^{n-1}$. □

The following lemma sets the largest temperature we need.

Lemma 128. *If the inverse of the temperature of the Markov chain is greater than $(n-1)\log(4n^2-n)$, then the probability of the reversal scenarios is at least $\frac{1}{2}$ in the target distribution.*

Proof. $c(S)$ is at least 1 for any non-reversal path, thus the probabilty of any non-reversal path is at least $(4n^2-n)^{n-1}$ times smaller than that of a reversal path. Since there are less than $(4n^2-n)^{n-1}$ times more non-reversal paths than reversal paths, the probability of the reversal paths in the target distribution is at least $\frac{1}{2}$. □

The following lemma tells what difference between the temperature of neighbour chains is necessary for a swapping probability greater or equal than $\frac{1}{2}$.

Lemma 129. *If the difference between the inverse temperatures is $\frac{4\log 2}{n(n-2)}$, then the swapping probability given by Equation 10.4 is at least $\frac{1}{2}$.*

Proof. Let Δc denote the difference between $c(R_i)$ and $c(R_{i+1})$ and let ΔT denote $\frac{1}{T_{i+1}} - \frac{1}{T_i}$. Equation 10.4 can be simplified as

$$\min\left\{1, e^{-\Delta c \Delta T}\right\} \tag{10.6}$$

Since Δc is at most $\frac{n(n-2)}{4}$, the swapping probability is at least $\frac{1}{2}$. □

Proof of Theorem 125. We set the temperature of the first chain to infinite, as prescribed, and the temperature of the $i+1^{st}$ chain as

$$T_{i+1} := \frac{1}{\frac{4\log 2}{n(n-2)} + \frac{1}{T_i}} \tag{10.7}$$

The swapping probability between two chains will be at least $\frac{1}{2}$, based on Lemma 129. The chain with index $\left\lceil \frac{n(n-1)(n-2)\log(4n^2-n)}{4\log 2} + 1 \right\rceil$ will have temperature at most $\frac{1}{(n-1)\log(4n^2-n)}$, and thus, the probability of the reversal scenarios in its target distribution is at least $\frac{1}{2}$. □

Chapter 11

Gibbs sampling of optimal SCJ labelings on arbitrary binary trees

The number of most parsimonious scenarios on evolutionary trees under the SCJ model is known to be computationally intractable. Miklós and Smith proved that it is #P-complete and Miklós, Tannier and Kiss proved that it does not have an FPRAS approximation assuming RP $\neq$ NP (see Chapter 8). On the other hand, the number of most parsimonious labeling of evolutionary trees has an unknown computational complexity. One optimal labeling can be found by applying the Fitch algorithm [34] on each adjacency, and choosing the absence of the adjacency at the root when the Fitch algorithm says that both the presence and absence of the adjacency give the minimum number of necessary SCJ mutations for that particular adjacency. Feijão and Meidanis [32] proved that the so-obtained genomes will always be valid. It is known that the Fitch algorithm cannot find all most parsimonious solutions for a particular character. The Sankoff-Russeau algorithm [78] is a dynamic programming algorithm that is capable to find all optimal solutions for a particular character, in case of SCJ model, for an adjacency. However, it is easy to show that solutions might be in conflict, as conflicting adjacencies might be assigned to a genome labeling an internal node (making the genome and thus the solution invalid), therefore, the solution space of optimal labelings is only a subset of the set that the Sankoff-Russeau algorithm gives. It is known when there is no constraint among the characters, the number of optimal labelings is in FP, and the number of most parsimonious

scenarios is not in FPRAS assuming that RP $\neq$ NP even for constraint-free characters [126]. Therefore the computational intractability of counting the number of most parsimonious scenarios on binary trees under the SCJ model by no means implies that the counting of most parsimonious labeling would be a hard computational problem. On the other hand, the constraints among the adjacencies make the counting problem more complicated than the constraint-free version. It is unclear if this particular counting problem is in FP or #P-complete and should it be in #P-complete whether or not it has an FPRAS approximation. In this chapter, we give a Gibbs sampler exploring the solution space of the most parsimonious labeling that seems to be rapidly mixing on some real life data. However, these examples can give only experimental evidence of rapid mixing only suggesting that the problem might have an FPRAS approximation.

This was a joint work with Heather Smith. The original work has been published in BMC Bioinformatics, 16(Suppl 14): S6., `https://doi.org/10.1186/1471-2105-16-S14-S6`. The author of the thesis suggested the research topic, constructed the example in Figure 11.1 **d)** and developed the main concept on conflicting adjacencies. All lemmas and theorems were proved jointly.

11.1 Gibbs sampling of most parsimonious labeling of evolutionary trees under the SCJ model

The Gibbs sampling is a special version of Markov chain Monte Carlo, when the multivariate target distribution is hard to sample from, however, the conditional distribution of each variable is easy to sample [38]. This is exactly the case for the most parsimonious labelings of an evolutionary tree under the SCJ model, as we show below.

11.1.1 Description of the Gibbs sampler

Let a rooted binary tree, $T(V, E)$ be given, together with a function f mapping genomes under the SCJ model to the leaves of the tree, L. We assume that all genomes appearing as an image for some leaf has the same labels for their edges. Let $\mathcal{A}$ represent the set of all adjacencies in $\cup_{v \in L} f(v)$. Let an

arbitrary indexing on $\mathcal{A}$ be given, then each genome G can be represented as a 0-1 vector $\mathbf{x}$ where x_i is 1 if and only if $a_i \in \mathcal{A}$ is in G. A 0-1 vector of length $|\mathcal{A}|$, $\mathbf{x}$, is called valid if for all pairs of coordinates satisfying $x_i = x_j = 1$, adjacencies a_i and a_j do not share an extremity. Each valid vector represents a valid genome.

Genomes labeling the vertices of T are represented by such 0-1 vectors, and the Gibbs sampler works on these representations. The target distribution is the uniform distribution of the possible most parsimonious labelings. Consider any most parsimonious labeling as a set of vectors representing the genomes labeling the tree T. Choose one coordinate, i, then Gibbs sampling is to sample uniformly from all possible most parsimonious labelings that have the same coordinates than the current labeling except coordinate i, which might be the same or might be different.

Formally, given a most parsimonious labeling of the internal nodes, a Gibbs sampling step is the following:

1. Draw a random coordinate i uniformly from $1, 2, \ldots |\mathcal{A}|$.

2. Consider the ith coordinates of the vector representations of the genomes labeling the leaves, and on these 0-1 characters, do the Sankoff-Russeau dynamic programming algorithm, described below. For each leaf l, assign the value $s(l,k) = 0$ if k is the character assigned to l and $s(l,k) = \infty$ otherwise.

   For each vertex v with children u_1 and u_2, the recursion is

$$\begin{aligned} s(v,0) &= \min\{s(u_1,0), s(u_1,1)+1\} + \\ &\quad \min\{s(u_2,0), s(u_2,1)+1\} \qquad (11.1) \\ s(v,1) &= \min\{s(u_1,0)+1, s(u_1,1)\} + \\ &\quad \min\{s(u_2,0)+1, s(u_2,1)\} \qquad (11.2) \end{aligned}$$

3. Create a directed metagraph M, whose vertices are $s(v,0)$ for each vertex v of the tree, and also those $s(v,1)$ for which writing 1 into the ith coordinate of the vector representing the genome labeling vertex v still a valid vector. Draw a directed edge from $s(u,k)$ to $s(v,k')$ if $s(u,k)$ gives the minimum for $s(v,k')$ in Equations (11.1) and (11.2). See also Fig. 11.1. **c)** and **d)**.

4. Do an enumeration dynamic programming on M. Let $m(w) = 1$ if $w = s(l,k)$, $k \in \{0,1\}$ and l is a leaf. For other nodes, do the following.

Let $w = s(v, k)$, $k \in \{0, 1\}$, and let the two children of v in the tree T be u_1 and u_2. Let $\mathcal{U}_1$ denote the set of in-neighbors of w that are $s(u_1, k)$, $k \in \{0, 1\}$ and $\mathcal{U}_2$ denote the set of in-neighbors of w that are $s(u_2, k)$, $k \in \{0, 1\}$. Then

$$m(w) = \left(\sum_{z_1 \in \mathcal{U}_1} m(z_1) \right) \times \left(\sum_{z_2 \in \mathcal{U}_2} m(z_2) \right) \tag{11.3}$$

$m(w)$ is called the weight of w.

5. If there is only one vertex in the metagraph M that is $s(root, k)$, $k \in \{0, 1\}$, choose that one at the root. Otherwise, choose randomly from the two vertices following the distribution proportional to their weights. For the chosen vertex w, $m(w)$ is not 0, therefore it has at least 1 in-neighbor from both $\mathcal{U}_1$ and $\mathcal{U}_2$. From both in-neighbor sets, choose a random vertex from the distribution proportional to their weight, or the only one if only one vertex is in a set. Propagate down this process along the tree, thus one vertex from M is selected for each vertex of the tree T. Update the ith coordinates of the vectors according to the selected meta-graph vertices: if $w = s(v, k)$ was selected for vertex v then write k into the ith coordinate of the vector representing the genome labeling vertex v.

It is well-known that the number of most parsimonious labelings by one character can be calculated by Equation (11.3) [115], and when some of the solutions should be excluded due to some constraints, they simply should be omitted from the calculations. This is how the metagraph M was constructed. It is also a folklore that following the distribution proportional to the weights calculated in a recursion leads to the uniform distribution over the cases that the recursion calculates, and the uniform distribution is the one what we would like to sample from in the Gibbs sampling.

11.1.2 Irreducibility of the Gibbs sampler

The Gibbs sampler, as a Markov chain, will converge to the prescribed distribution if the Markov chain is irreducible, that is, any most parsimonious labeling can be transformed into any another by a finite number of Gibbs sampling steps. Due to the constraints on the coordinates, it is far not trivial.

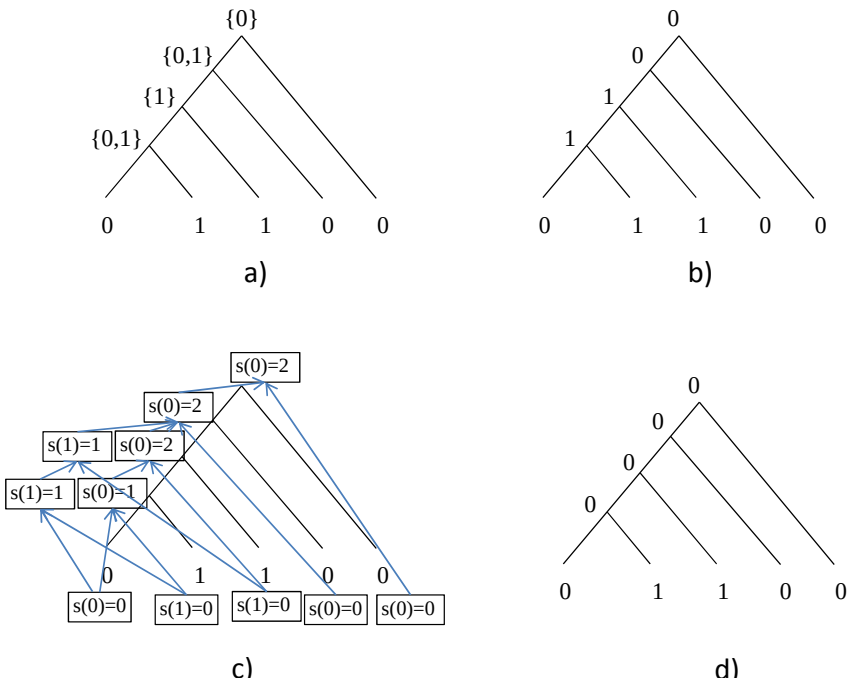


Figure 11.1: A rooted binary tree with two most parsimonious labelings of internal nodes. **a)** The B functions of the Fitch algorithm calculated in the bottom-up phase. **b)** The (canonical) Fitch solution. **c)** The values calculated in the Sankoff-Russeau algorithm and the edges in the metagraph M (see text for details). Due to sake of readability, only those values are indicated that contribute in estimating the number of most parsimonious solutions. Also, vertices of the tree are not indicated, i.e. $s(k)$ are written instead of $s(v, k)$. From positioning, it should be obvious which s value belongs to which vertex. **d)** The most parsimonious solution that can be obtained only by the Sankoff-Russeau algorithm and not by the Fitch algorithm.

Below we prove irreducibility by proving that any most parsimonious labeling can be transformed to a canonical labeling, the one was described by Feijaõ and Meidanis [32]. Below we define formally this most parsimonious labeling. First, we recall the Fitch algorithm.

Definition 130. *The* Fitch algorithm *[34] is a greedy algorithm for finding a most parsimonious labeling of a tree, given a rooted binary tree, and the leaves of the tree is labeled by characters from some finite set. It has two phases (see also Fig. 11.1* **a)** *and* **b)***).*

1. *(Bottom-up phase) for each leaf v assign a set $B(v) = \{c\}$ where c labels v. Then for each internal node v with children u_1 and u_2*

$$B(v) = \begin{cases} B(u_1) \cap B(u_2), & \text{if } B(u_1) \cap B(u_2) \text{ is not empty} \\ B(u_1) \cup B(u_2), & \text{otherwise} \end{cases} \tag{11.4}$$

2. *(Top down phase) Choose any member from $B(root)$ that labels the root. This is denoted by $F(root)$. Then propagate down characters labeling internal nodes on the tree using the following recursion, where v is the parent of u*

$$F(u) = \begin{cases} F(v) \cap B(u), & \text{if } F(v) \cap B(u) \text{is not empty} \\ \text{any member from } B(u), & \text{otherwise} \end{cases} \tag{11.5}$$

Although Equation (11.5) might be ambiguous for alphabets with size larger than 2, for 0-1 alphabet, there is no ambiguity. Ambiguity for 0-1 alphabet might happen only at the root when $B(root) = \{0, 1\}$.

Definition 131. *Given a rooted binary tree, $T(V, E)$ and genomes labeling the leaves of the tree. Assume that each genome is represented as a 0-1 vector indicating which adjacency can be found in the genome, as described above. Then the canonical solution for the most parsimonious labeling of the tree under the* SCJ *model is given by applying the Fitch algorithm for each position of the representing vectors, and choosing 0 at the root whenever $B(root) = \{0, 1\}$. The so-obtained values are the coordinates of the vectors representing the genomes labeling the internal nodes of the tree.*

Feijaõ and Meidanis proved that the so-obtained vectors are always valid, thus they indeed give a most parsimonious labeling of the internal nodes [32]. Below we show that any solution to the most parsimonious labeling of the internal nodes under the SCJ model (which might be a solution that cannot be obtained by the Fitch algorithm just by the Sankoff-Russeau algorithm, see for example, Fig. 11.1. **d)**) can be transformed into the canonical solution by a finite series of Gibbs sampling steps. First we have to prove a lemma regarding the values calculated in the Fitch algorithm and the Sankoff-Russeau algorithm.

Lemma 132. *Assume an arbitrary rooted binary tree, leaves are labelled by 0s and 1s. Then for any internal node v, $B(v) = \{0, 1\}$ if and only if $s(v, 0) = s(v, 1)$.*

Proof. The $\Rightarrow$ direction was proved in [126]. The $\Leftarrow$ direction is proved by strong induction on h, the height of v. We prove the equivalent form $B(v) \neq \{0, 1\} \implies s(v, 0) \neq s(v, 1)$. When $h = 0$, v is a leaf, and the statement is true as $s(v, 0) \neq s(v, 1)$ and $B(v) \neq \{0, 1\}$.

For any node $h \geq 1$, assume that the statement holds for any node with height $k < h$. If $B(v) \neq \{0, 1\}$ then either $B(v) = \{0\}$ or $B(v) = \{1\}$. The two cases are symmetric, so we might assume that $B(v) = \{0\}$, the proof for the other case is symmetric.

If $B(v) = \{0\}$ and u_1 and u_2 are the children of v, then either $B(u_1) = B(u_2) = \{0\}$ or $B(u_1) = \{0\}$, $B(u_2) = \{0, 1\}$ or $B(u_1) = \{0, 1\}$, $B(u_2) = \{0\}$.

If $B(u_1) = B(u_2) = \{0\}$, then by the induction, $s(u_1, 0) \neq s(u_1, 1)$, and since the Fitch algorithm gives a most parsimonious solution, $s(u_1, 0) < s(u_1, 1)$. Similarly for the other node, $s(u_2, 0) < s(u_2, 1)$. Then $s(v, 0) < s(v, 1)$, according to Equations (11.1) and (11.2).

If for one of the children, the B function takes $\{0, 1\}$, then for that node u, $s(u, 0) = s(u, 1)$. For the sibling node u', $s(u', 0) < s(u', 1)$, and it is easy to check (by considering Equations (11.1) and (11.2)) that $s(v, 0) < s(v, 1)$. □

Lemma 133. *Given a most parsimonious labeling $\mathcal{L}$ of a tree $T(V, E)$ under the* SCJ *model. Assume that the genomes are given in a binary vector representation as described above. Let v be the minimum height node for which some adjacency α, $B_\alpha(v) = \{0\}$, however, α is presented in the genome labeling v ($B_\alpha(v)$ is the set that the Fitch algorithm calculates for the vertex v when the algorithm is applied to the presence/absence of adjacency α). Change the current labeling in the following way. Remove α from the genome*

labeling the node v and propagate down the presence-absence of adjacency α below the subtree rooted in v according to the Fitch algorithm as v was the root of the tree. Then the so obtained new labeling $\mathcal{L}'$

a) *contains valid genomes*

b) *also a most parsimonious labeling.*

Proof. Changing any presence to absence cannot turn a valid genome into invalid. The only case when the genome might become invalid is when an absence is turned into presence (a possible example for this is on Fig. 1. **d)** and **b)**, **d)** is a Sankoff-Russeau solution, **b)** is the canonical Fitch solution). This might be the case when

- for some node u below v, $B_\alpha(u) = \{1\}$ or
- on connected parts C of the tree where for all nodes, $u \in C$, $B_\alpha(u) = \{0, 1\}$, except for the root of C, r, for which $B_\alpha(r) = \{1\}$.

If $B_\alpha(u) = \{1\}$ then for all adjacencies β being in conflict with α, $B_\beta(u) = \{0\}$ (Lemma 6.1. in [32]). But then β must be absent in the genome labeling u otherwise it would contradict to the minimum height of v.

For any connected part of the tree, C with the above described property, we prove that for any adjacency β being in conflict with α, β is absent in the genomes labeling the vertices of C. For the root r, it holds as $B_\alpha(u) = \{1\}$, thus $B_\beta(u) = \{0\}$. For any node $u \in C$, for whose parent w, we showed that β is absent in the genome labeling w, we show that β also absent in the genome labeling u. If $B_\beta(u) = \{0\}$, then β is absent in the genome labeling u due to the minimal height of v. If $B_\beta = \{0, 1\}$, then $s_\beta(u, 0) = s_\beta(u, 1)$. Then in a most parsimonious labeling, it cannot be the case that β is absent in the genome labeling w but is presented in the genome labeling u. Indeed, such a labeling would have a parsimony score 1 for the edge (u, v), and a cost $s_\beta(u, 1)$ below the subtree rooted in u. On the other hand, if we change the labeling at the node u that β is absent in the genome labeling u, and on the subtree below u, we can change the presence/absence of β to get a parsimony score $s_\beta(u, 0)$. Then the parsimony score regarding β for the edge (u, v) is 0, hence this new labeling has a smaller total cost on the tree compared to the current one, a contradiction. By induction, on the whole connected part C, β is absent in the genomes labeling the vertices of C.

We proved that the new labeling $\mathcal{L}'$ contains valid genomes. We are going to prove that it is also a most parsimonious labeling. Since $B_\alpha(v) = \{0\}$, it follows that $s_\alpha(v, 0) < s_\alpha(u, 1)$. Hence, in the old labeling $\mathcal{L}$, the parsimony score regarding α on the subtree rooted in v was greater than in the modified labeling. On the edge connecting v to its parent, the new score might be 1, the old score might be 0, and then here the parsimony score might increase by 1, however, this loss cannot be greater than the gain we obtained on the subtree rooted at v. (And if the old labeling $\mathcal{L}$ was most parsimonious it turns out that the old parsimony score regarding α on edge connecting v to its parent was 0.) □

Since the number of adjacencies as well as the height of the tree is finite, in a finite number of steps, any labeling can be transformed into a labeling such that for all vertices v and all adjacencies α, $B_\alpha(v) = \{0\}$ indicates that adjacency α is absent in the genome labeling v. Next, we consider transforming such labelings.

Lemma 134. *Given a most parsimonious labeling $\mathcal{L}$ of a tree $T(V, E)$ under the* SCJ *model. Assume that the genomes are given in a binary vector representation as described above. Furthermore, assume that for all vertices w and all adjacency α, $B_\alpha(w) = \{0\}$ indicates that adjacency α is absent in the genome labeling w.*

Let v be the minimum height node for which some adjacency α, $B_\alpha(v) = \{1\}$, however, α is absent in the genome labeling v. Change the current labeling in the following way. Add α to the genome labeling the node v and propagate down the presence-absence of adjacency α below the subtree rooted in v according to the Fitch algorithm as v was the root of the tree. Then the so obtained new labeling $\mathcal{L}'$

a) *contains valid genomes*

b) *also a most parsimonious labeling.*

Proof. The proof of validity in Lemma 134. is exactly the same than the proof of Lemma 133. except each reasoning "if $B_\beta(u) = \{0\}$, then β is absent in the genome labeling u due to the minimal height of v" should be replaced to "if $B_\beta(u) = \{0\}$, then β is absent in the genome labeling u due to the given conditions".

Proving that the new labeling $\mathcal{L}$' is also most parsimonious is exactly the same than the proof of Lemma 133. just 0 and 1 should be switched. □

Hence any most parsimonious labeling can be transformed to a most parsimonious labeling such that for each node v and each adjacency α, $B_\alpha(v) = \{0\}$ indicates the absence of α in the genome labeling v, and $B_\alpha(v) = \{1\}$ indicates the presence of α in the genome labeling v. Furthermore, each transformation is a possible Gibbs sampling step, since one coordinate is changed from a most parsimonious labeling to another most parsimonious, valid labeling. During these transformations, when the labeling was changed below a vertex v, for which $B_\alpha(v) \neq \{0, 1\}$ for some α, the new labeling is the canonical Fitch labeling. What about the subtrees below vertices v and adjacencies α, for which $B_\alpha(v) = \{0\}$ and the adjacency α was absent in the initial labeling or $B_\alpha(v) = \{1\}$, and the adjacency α was presented in the initial labeling? The following lemma claims that for such subtrees, the initial labeling was already the Fitch labeling.

Lemma 135. *Assume that in a most parsimonious labeling, $B_\alpha(u) = \{0, 1\}$ and α is presented (respectively, absent) in the genome labeling the parent of u. Then α is presented (respectively, absent) in the genome labeling u.*

Proof. Assume that the presence/absence of α in u and its parent is different. Then the parsimony score on the edge connecting u to its parent is 1. However, $s_\alpha(u, 0) = s_\alpha(u, 1)$, hence switching the presence/absence of α is possible without changing the parsimony score on the subtree rooted at u (changing the presence/absence of α in genomes labeling vertices below u might be needed). On the other hand, the parsimony score on the edge connecting u to its neighbor could decrease by 1, a contradiction to the assumption that we start with a most parsimonious labeling. □

The consequence of the Lemma 135. is that we can transform by finite series of Gibbs sampling steps any most parsimonious labeling to a labeling $\mathcal{L}'$ such that for all vertices u and all adjacencies α, for which $B_\alpha(u) \neq \{0, 1\}$ or a vertex v above u (v is not necessarily the parent of u, it might be arbitrary higher node above u) exists such that $B_\alpha(v) \neq \{0, 1\}$, the genome labeling u is the Fitch canonical solution regarding adjacency α. These labelings are almost in the Fitch canonical solutions, except for connected parts C containing the root of the tree on which for some α, $B_\alpha(v) = \{0, 1\}$, $\forall v \in C$. The next lemma claims that they can be transformed into the Fitch canonical solution.

Lemma 136. *Given a most parsimonious labeling $\mathcal{L}$ of a tree $T(V, E)$ under the* SCJ *model. Assume that the genomes are given in a binary vector*

representation as described above. Furthermore, assume that for all vertices w and all adjacency α, $B_\alpha(w) = \{0\}$ indicates that adjacency α is absent in the genome labeling w and $B_\alpha(w) = \{1\}$ indicates that the adjacency is presented in the genome.

Consider any adjacency α, and let C denote the connected subset C containing the root for which $B_\alpha(v) = \{0, 1\}$, $\forall v \in C$. (C might be the empty set.) Change the current labeling $\mathcal{L}$ such that in the new labeling $\mathcal{L}'$ adjacency α be absent in each genome labeling any vertex $v \in C$, and do not change the labeling otherwise. Then the new labeling

a) *is a valid labeling*

b) *is also a most parsimonious labeling*

Proof. Changing the presence to absence cannot make an invalid genome, therefore proving the validity is trivial.

For any vertex v, $B(v) = \{0, 1\}$, the B function for both children is also $\{0, 1\}$ or for one of the children, it is $\{1\}$ and for the other child, it is $\{0\}$. Extend C to C' such that we add to C all cherry motifs (pair of children) for which one of the children, the B_α function is $\{1\}$ and for the other child, it is $\{0\}$. We know from the condition that α is presented in the genome labeling one of the children and is absent in the genome labeling the other child. If we do not change the current labeling at the leaves of C', there are two possible most parsimonious labelings regarding adjacency α: i) α is presented in all genomes labeling the internal nodes, ii) α is absent in all genomes labeling the internal nodes. This later is what $\mathcal{L}'$ contains. □

We are ready to prove the main lemma.

Lemma 137. *Given a most parsimonious labeling $\mathcal{L}$ of a tree $T(V, E)$ under the* SCJ *model. Then $\mathcal{L}$ can be transformed into the canonical Fitch solution by finite series of Gibbs sampling steps.*

Proof. In the first phase, while there is a vertex v and adjacency α such that $B_\alpha(v) = \{0\}$, however α is presented in the genome labeling vertex v, find the α and v with the minimal height, and do the Gibbs sampling indicated in Lemma 133.

After the first phase, in the second phase, while there is a vertex v and adjacency α such that $B_\alpha(v) = \{1\}$, however, α is absent in the genome

labeling v, find the α and v with the minimal height, and do the Gibbs sampling indicated in Lemma 134.

After the second phase, in the third phase, while there is an adjacency α, for which the connected part C containing the root with the property that $\forall v \in C$, $B_\alpha(v) = \{0, 1\}$ is not the empty set, and α is presented in any of the genomes labeling any of the vertices $v \in C$, choose one of the such adjacencies, and remove from all genomes labeling the vertices in v. Since it yields a most parsimonious labeling, it is also a Gibbs sampling step.

After the third phase, the labeling is the Fitch canonical labeling. □

The main lemma directly leads to the following theorem.

Theorem 138. *Any most parsimonious labeling of a tree under the* SCJ *model can be transformed into any another most parsimonious labeling by finite series of Gibbs sampling steps.*

Proof. A most parsimonious labeling $\mathcal{L}_1$ can be transformed into the canonical labeling $\mathcal{L}_c$ and also labeling $\mathcal{L}_2$ can be transformed into $\mathcal{L}_c$ by Gibbs sampling steps. Note that the inverse of a Gibbs sampling step is also a Gibbs sampling step, thus $\mathcal{L}_1$ can be transformed into $\mathcal{L}_2$ by first transforming $\mathcal{L}_1$ into $\mathcal{L}_c$ then transforming $\mathcal{L}_C$ into $\mathcal{L}_2$ by the invers transformation that moves $\mathcal{L}_2$ into $\mathcal{L}_c$. □

Chapter 12

Markov kernels with large perturbations and large acceptance ratios

In Chapter 7, we showed that large perturbations might cause small acceptance ratios in the Metropolis-Hastings algorithm causing torpid mixing of the resulting Markov chain. Here we give examples of Markov kernels that perturb a non-constant part of the current state, however, when they are plugged into a Metropolis-Hastings algorithm, the inverse of the acceptance ratio is still upper bounded by a polynomial. The first such a Markov kernel we are aware of was given by Jacobson and Matthew [51] for sampling Latin squares. It is easy to see that for any prime number $p > 3$, any transition kernel is not irreducible if it perturbs at most two rows of a Latin square. On the other hand, perturbing at most 3 rows is sufficient for irreducibility. However, the number of entries in these three rows are not bounded.

Latin squares can be considered as factorizations of the complete bipartite graphs into 1-factors. With other words, edge colorings of the complete bipartite graph. Here we present two generalizations. In a joint work with two BSM students, Kathleen Zhu and Mark Aksen, we gave an irreducible Markov chain on the half-regular factorizations of complete bipartite graphs. In that Markov chain, at most 3 factors are perturbed in a single step. Although, there is no restriction on how many edges are affected, the inverse of the accpeptance ratio is still upper bounded by a polynomial. The original work has been published in Discrete Applied Mathematics, 230:21-33, DOI: `https://doi.org/10.1016/j.dam.2017.06.003`. The author of this thesis

suggested the definition of exeedance number, developed the Markov chain Monte Carlo algorithm, proved the theorem on it and contributed to the proofs of all other lemmas and theorems.

In a joint work with a BSM student, Carina Hong, we gave an irreducible Markov chain of edge colorings of bipartite graphs. In each step, at most three colors are altered. Although there is no restriction how many edges change color, the inverse of the accpetance ratio is still upper bounded by a polynomial. The original work is under review at the moment. Carina Hong proved Lemma 156 in case of regular bipartite graphs, and the author of this thesis extended the proof for arbitrary bipartite graphs. The author of this thesis developed the Markov chain Monte Carlo method and proved the theorems on it.

12.1 Half-regular factorizations of the complete bipartite graph

12.1.1 Preliminaries

In this section, we will work with realizations of half-regular degree matrices, defined below.

Definition 139. *A* bipartite degree sequence $D = ((d_1, d_2, \dots d_n), (f_1, f_2, \dots f_m))$ *is a pair of sequences of non-negative integers. A bipartite degree sequence is* graphic *if there exists a simple bipartite graph G whose degrees correspond exactly to D. We say that G is a* realization *of D.*

Definition 140. *A* bipartite degree matrix $\mathcal{M} = (D, F)$ *is a pair of $k \times n$ and $k \times m$ matrices of non-negative integers. A bipartite degree matrix is graphic if there exists an edge colored simple bipartite graph $G(U, V, E)$ such that for all color c_i and for all $u_j \in U$, the number of edges of u_j with color c_i is $d_{i,j}$ and for all $v_l \in V$, the number of edges of v_l with color c_i is $f_{i,l}$. Such graph is called a* realization *of $\mathcal{M}$. A bipartite degree matrix is* half-regular *if for all $i \leq k$ and $j, l \leq n$, $d_{i,j} = d_{i,l}$.*

The rows of $\mathcal{M}$ are bipartite degree sequences that are also called factor*s and the edge colored realization of $\mathcal{M}$ is also called an* $\mathcal{M}$-factorization.

We consider two problems. One is the existence problem which asks if there is a realization of a given half-regular degree matrix. The other is the

sampling problem, which considers the set of all realizations of a half-regular degree matrix and asks how to sample uniformly a realization from this set. Markov Chain Monte Carlo (MCMC) methods are generally applicable for such problems, and we are also going to introduce an MCMC for sampling realizations of half-regular degree sequences.

12.1.2 The existence problem

Definition 141. *Let $G(V, E)$ be a multigraph, and for each $u, v \in V$, let $f(u, v)$ be*

$$f(u,v) := \begin{cases} 0 & \text{if } (u,v) \notin E \\ m(u,v) - 1 & \text{otherwise} \end{cases} \tag{12.1}$$

where $m(u, v)$ denotes the multiplicity of edge (u, v). The exceedance number *of graph G is defined as*

$$ex(G) := \sum_{u,v \in V} f(u,v) \tag{12.2}$$

Clearly, the exceedance number is 0 iff G is a simple graph.

Theorem 142. *Let*

$$\begin{aligned} \mathcal{M} = \{ & \{d_{1,1} = d_{1,2} = \ldots = d_{1,n}\}, \quad \{f_{1,1}, f_{1,2}, \ldots f_{1,m}\} \\ & \{d_{2,1} = d_{2,2} = \ldots = d_{2,n}\}, \quad \{f_{2,1}, f_{2,2}, \ldots f_{2,m}\} \\ & \qquad \vdots \\ & \{d_{k,1} = d_{k,2} = \ldots = d_{k,n}\}, \quad \{f_{k,1}, f_{k,2}, \ldots f_{k,m}\}\} \end{aligned} \tag{12.3}$$

be a half-regular bipartite degree matrix. The bipartite complete graph $K_{n,m}$ has an $\mathcal{M}$-factorization iff

1. $\forall i,\ n d_{i,1} = \sum_{j=1}^{m} f_{i,j}$
2. $\sum_{i=1}^{k} d_{i,1} = m$
3. $\forall j,\ \sum_{i=1}^{k} f_{i,j} = n.$

Proof. $\Rightarrow$ If $K_{n,m}$ has an $\mathcal{M}$-factorization, then clearly all factors are graphic and the degrees sum to the degrees of the complete bipartite graph. Conditions 2 and 3 explicitly state that the sum of the degrees in the factors sum

up to the degree of the complete bipartite graph. Condition 1 states that in each factor, the sums of the degrees in the two vertex classes are the same, which is a necessary condition for a graphic degree sequence.

$\Leftarrow$ Conditions 2 and 3 also say implicitly that for each i, $d_{i,1} \leq m$ and for each i, j, $f_{i,j} \leq n$. Together with condition 1, they are sufficient conditions that a half-regular bipartite degree sequence be graphic. Namely, the theorem says if all factors are graphic and the sums of the degrees are the degrees of the complete bipartite graph, then $K_{n,m}$ has such a factorization.

For each i, let G_i be a realization of the degree sequence $(d_{i,1}, \ldots d_{i,n}), (f_{i,1}, \ldots f_{i,m})$. Let $G = \cup_{i=1}^{k} G_i$. Color the edges of G such that an edge is colored by color c_i if it comes from the realization G_i. G might be a multigraph, however, for each color c_i and each pair of vertices u, v, there can be at most one edge between u and v with color c_i. If $ex(G) = 0$, then G is a simple graph, and due to the conditions, it is $K_{n,m}$, thus we found an $\mathcal{M}$-factorization of $K_{n,m}$.

Assume that $ex(G) > 0$. Then there is a pair of vertices (u, v) such that there is more than one edge between u and v. Fix one such pair (u, v), and let V_0 denote the set $\{v\}$. Since the degree of v in G is n, there must be a u' such that there is no edge between u' and v. Let C_0 denote the set of colors that appear as edge colors between u and v. Since for each color c_i, u and u' have the same number of edges with color c_i, there must be at least one vertex v' such that there are more edges with colors from C_0 between u' and v' than the edges with colors from C_0 between u and v'. Let V' denote the set of vertices for which this condition holds. There are three possibilities (see also Figure 12.1):

1. There is a vertex $v' \in V'$ such that there are more than one edge between u' and v'.

2. There is a vertex $v' \in V'$ such that there is only a single edge between u' and v'. However, there is no edge between u and v'.

3. For each vertex $v' \in V'$, there is only a single edge between u' and v', and there is at least one edge between u and v'.

If case 1 holds, then let c_{i_0} be a color in C_0 such that there is a c_{i_0}-colored edge between u' and v' and there is no c_{i_0}-colored edge between u and v'. Remove the edge between u and v and also between u' and v' and add a c_{i_0} colored edge between u' and v and also between u and v'. Note that G_{i_0},

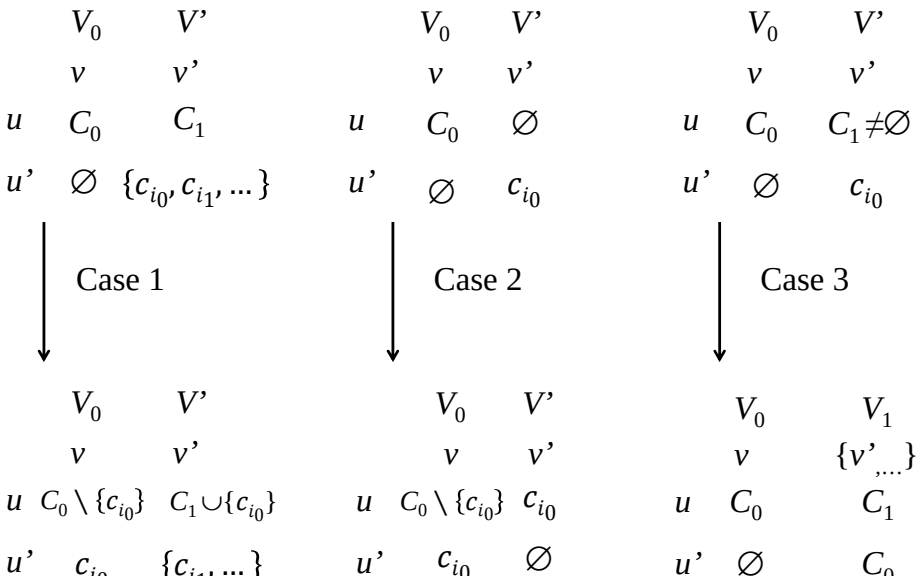


Figure 12.1: The three possible cases when there is more than one edge between vertices u and v. See text for details.

the graph with color c_{i_0} remains a simple graph with the prescribed degree sequence. $f(u, v)$ decreases by 1. $f(u', v)$ remains 0, since the edge with color c_{i_0} is the only edge between them. $f(u', v')$ decreases by 1. $f(u, v')$ might increase by 1 if there was already at least one edge between them before the change, but otherwise it remains the same. In total, $ex(G)$ is decreased at least by 1.

If case 2 holds, then do the same edge changing as in case 1. This causes a decrease in $f(u, v)$ by one, and no change in $f(u', v)$, $f(u, v')$ and $f(u', v')$. In total, $ex(G)$ decreases by 1.

In case 3, select a subset V_1 from V' such that the colors of edges between u' and the vertices in V_1 are exactly the set C_0. This is possible, since for each vertex $v' \in V'$, there is only one edge between u' and v'. Furthermore, the union of colors of edges between u' and V' must contain a subset C_0. Let C_1 denote the (possibly multi)set of colors that appear as edge colors between vertex u and the set of vertices V_1. The multiset of colors between u and $V_0 \cup V_1$ is $C_0 \uplus C_1$, where $\uplus$ denote the multiset union, while the set of colors between u' and $V_0 \cup V_1$ is C_0. Therefore, there is at least one vertex $v' \notin V_0 \cup V_1$ such that the number of edges with colors from C_1 between u'

and v' is more than the number of edges with colors from C_1 between u and v'. Let V' denote the set of vertices with this property. If case 3 holds, then we can select a subset V_2 from V' such that the multiset of colors between vertex u' and V_2 is exactly C_1. Then the multiset of colors between u and $V_0 \cup V_1 \cup V_2$ is $C_0 \uplus C_1 \uplus C_2$, while the multiset of colors between u' and $V_0 \cup V_1 \cup V_2$ is $C_0 \uplus C_1$.

Therefore, while case 3 holds, we can select the subset of vertices $v' \notin \cup_{i=0}^{j-1} V_i$ such that the number of edges with colors from C_{j-1} between u' and v' is more than the number of edges with colors from C_{j-1} between u and v', and from this set, we can select an appropriate subset V_j. Since there are finite number of vertices in V_j, for some j, case 1 or 2 must hold (see also Figure 12.2). Let v' be the vertex for which case 1 or 2 holds, and let $c_{i_{j-1}}$ be the color such that there is an edge with color $c_{i_{j-1}}$ between v' and u' and there is no such edge between u and v'. Remove the edge between u' and v' and add it between u and v'. Let $v_{j-1} \in V_{j-1}$ be a vertex such that there is an edge between u and v_{j-1} with color $c_{i_{j-1}}$. Remove that edge and add it between u' and v_{j-1}. Let the color between u' and v_{j-1} be $c_{i_{j-2}}$. Remove it and add it between u and v_{j-1}. (Note that due to the definition of V' and due to case 3 held in the $j-1$st iteration, before the change, there was no $c_{i_{j-2}}$ colored edge between u and v_{j-1}. Thus after the change, all edges between u and v_{j-1} have different colors.) Iterate this process; in the lth iteration, let v_{j-l} be a vertex in V_{j-l} such that there is an edge with color $c_{i_{j-l}}$ between u and v_{j-l}. Remove that edge and add it between u' and v_{j-l}, and remove the edge with color $c_{i_{j-l-1}}$ between u' and v_{j-l} and add it between u and v_{j-l}. Finally, remove the edge with color c_{i_0} between u and v and add it between u' and v. Let this new graph be G'. Then in G', $f(u,v)$ decreases by 1, while $f(u',v)$ remains the same. For each $i = 1, \ldots j-1$, neither $f(u,v_i)$ nor $f(u',v_i)$ is changed since the total number of edges between them is not changed. Finally, $f(u,v') + f(u',v')$ is not increased. Altogether, $ex(G')$ is at least 1 less than $ex(G)$, see Figure 12.2. It is easy to verify that all vertices participating in the modification of G, the same colored edges were removed and added. Therefore, G' is still the union of realizations of the prescribed degree sequences.

Since the exceedance number is a non-negative finite integer, after a finite number of steps, the exceedance number will be 0, thus we get $K_{n,m}$ as an $\mathcal{M}$-factorization. □

It is clear that one of the colors might encode "non-edge", namely, we

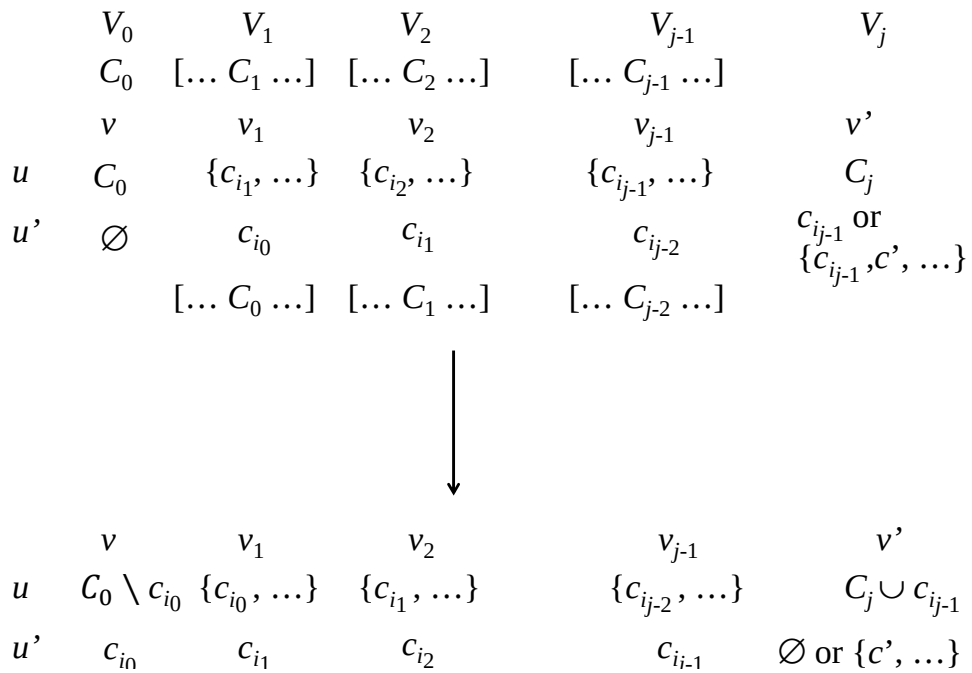


Figure 12.2: An illustration of Case 1 or Case 2 after j iterations, where $[\ldots C_l \ldots]$ indicates that the (possibly multi)set of colors between u and V_l is C_l and the (possibly multi)set of colors between u' and V_l is C_{l-1}, for all $l = 1, \ldots j-1$. From the set of vertices V_l, a vertex v_l is selected such that there is one edge between u' and v_l with color $c_{i_{l-1}}$ and there is at least one edge between u and v_l including an edge with color c_{i_l}. See text for details.

can delete those edges and get a realization of a half-regular degree matrix obtained by deleting one row from $\mathcal{M}$. Therefore, the following theorem also holds.

Theorem 143. *Let*

$$\begin{aligned}\mathcal{M} = \{&\{d_{1,1} = d_{1,2} = \ldots = d_{1,n}\}, \quad \{f_{1,1}, f_{1,2}, \ldots f_{1,m}\}\\ &\{d_{2,1} = d_{2,2} = \ldots = d_{2,n}\}, \quad \{f_{2,1}, f_{2,2}, \ldots f_{2,m}\}\\ &\qquad\qquad\qquad\vdots\\ &\{d_{k,1} = d_{k,2} = \ldots = d_{k,n}\}, \quad \{f_{k,1}, f_{k,2}, \ldots f_{k,m}\}\}\end{aligned} \tag{12.4}$$

be a half-regular bipartite degree matrix. Then $\mathcal{M}$ has a realization iff

1. $\forall i$, $nd_{i,1} = \sum_{j=1}^{m} f_{i,j}$
2. $\sum_{i=1}^{k} d_{i,1} \leq m$
3. $\forall j$, $\sum_{i=1}^{k} f_{i,j} \leq n$.

Proof. It is clear that equality in condition 2 holds iff equality in condition 3 holds for all j. In case of equality, the statement follows from Theorem 142. In case of inequality, let

$$d_{k+1,1} = d_{k+1,2} = \ldots d_{k+1,n} = m - \sum_{i=1}^{k} d_{i,1}$$

and let

$$f_{k+1,i} = n - \sum_{j=1}^{k} f_{j,i} \quad \forall i = 1, 2, \ldots m.$$

Extend $\mathcal{M}$ with this $k+1$st row, and this matrix will satisfy the conditions of Theorem 142. Find a realization of this extended matrix, and delete the edges with the $(k+1)$st color, thus obtaining a realization of $\mathcal{M}$. □

In the following subsection, we consider the solution space of realizations of half-regular degree matrices. Since there is no difference between realizations of half-regular degree matrices and $\mathcal{M}$-factorizations of the complete bipartite graph $K_{n,m}$, we will consider the latter. Namely, we consider non-edges as $(k+1)$st colors when non-edges exist.

12.1.3 The connectivity problem

In this subsection, we give necessary and sufficient perturbations to transform any realization of a half-regular bipartite degree matrix, $\mathcal{M}$, into another realization of $\mathcal{M}$. First, we extend the space of graphs on which perturbations are applied. We show how to transform a realization of $\mathcal{M}$ into another realization in this extended space, then we show how to transform realizations into each other, while remaining in the space of realizations. The concept is very similar to the concept applied in [51], but different perturbations are necessary to temporarily extend the space of graphs. First, we introduce a necessary definition, the $(+c_1 - c_2)$ deficiency. We will extend the space of graphs that have at most 3 vertices with deficiency. We show how to transform G_1, a realization of $\mathcal{M}$ into another realization G_2, via graphs having at most 3 vertices with deficiency. In doing so, we first arrange the colored edges of a vertex $u_0 \in U$ so that they agree with realization G_2. Then we fix these edges and reduce the problem to a similar, smaller problem. Note that U is the regular class, and if we fix (technically: remove) the vertex u_0 together with its edges, the remaining graph is still half-regular. Finally, we prove how to transform realizations of $\mathcal{M}$ into each other remaining in the space of realizations of $\mathcal{M}$.

First, we define deficiency.

Definition 144. *Let $\mathcal{M}$ be a bipartite degree matrix, and let G be an edge colored bipartite graph. We say that a vertex u_j in G has a $(+c_{i_1} - c_{i_2})$-deficiency w.r.t. $\mathcal{M}$ if the number of its c_{i_1} colored edges is $d_{i_1,j} + 1$, the number of its c_{i_2} colored edges is $d_{i_2,j} - 1$ and for all other $i \neq i_1, i_2$, the number of c_i colored edges of u_j is $d_{i,j}$.*

Then we define an auxiliary graph that will be used repeatedly.

Definition 145. *Let $G(U, V, E)$ be an edge colored bipartite graph in which the edges are colored with colors $c_1, c_2, \ldots c_k$, and let u and u' be two vertices in U. Then the directed, edge labeled multigraph $K(G, u, u')$ is defined in the following way. The vertices of K are the colors $c_1, c_2, \ldots c_k$, and for each $v \in V$, there is an edge going from c_i to c_j, where c_i is the color of the edge between u and v and c_j is the color of the edge between u' and v. Such an edge is labeled with v.*

The next two lemmas show how to handle graphs with a small amount of deficiency.

Lemma 146. *Let G be such an edge colored simple graph of $K_{n,m}$ which is almost a factorization of a half-regular bipartite degree matrix $\mathcal{M} = (D, F)$ in the following sense:*

1. *For each color c_i and vertex v_j, the number of edges with color c_i is $f_{i,j}$.*
2. *For each color c_i and vertex u_j the number of edges with color c_i is $d_{i,j}$ except for two colors c_{i_1} and c_{i_2} and two vertices u_{j_1} and u_{j_2}, where u_{j_1} has a $(+c_{i_1} - c_{i_2})$-deficiency and u_{j_2} has a $(+c_{i_2} - c_{i_1})$-deficiency.*

Then there exists a perturbation of G that affects only edges of u_{j_1} and u_{j_2} and transforms G into a realization of $\mathcal{M}$.

Proof. Consider $K(G, u_{j_1}, u_{j_2})$. For each vertex c_i, $i \neq i_1, i_2$ of the graph $K(G, u_{j_1}, u_{j_2})$, the number of incoming and outgoing edges are the same, while c_{i_1} has two more outgoing edges than incoming and c_{i_2} has two more incoming edges than outgoing. Therefore, there is a trail from c_{i_1} to c_{i_2} due to the pigeonhole principle. For each edge e_h labeled by v_h along the trail, swap the corresponding edges in G, namely, the edge between u_{j_1} and v_h and the edge between u_{j_2} and v_h. This transforms G into a realization of $\mathcal{M}$. Indeed, u_{j_1} will have one less edge with color c_{i_1} and one more edge with color c_{i_2} while the effect on u_{j_2} is the opposite. The number of edges with other colors are not affected, since in the trail, the number of incoming and outgoing edges are the same for all colors not c_{i_1} and not c_{i_2}, and swapping the edges in G is equivalent with inverting the direction of the corresponding edges in $K(G, u_{j_1}, u_{j_2})$. □

Lemma 147. *Let G be such an edge colored simple graph of $K_{n,m}$ which is almost a factorization of a half-regular bipartite degree matrix $\mathcal{M} = (D, F)$ in the following sense:*

1. *For each color c_i and vertex v_j, the number of edges with color c_i is $f_{i,j}$.*
2. *For each color c_i and vertex u_j, the number of edges with color c_i is $d_{i,j}$ except for three colors c_{i_1}, c_{i_2} and c_{i_3} and three vertices u_{j_1}, u_{j_2} and u_{j_3}, for which u_{j_1} has $(+c_{i_1} - c_{i_2})$-deficiency, u_{j_2} has $(+c_{i_2} - c_{i_3})$-deficiency and u_{j_3} has $(+c_{i_3} - c_{i_1})$-deficiency.*

Then there exists a perturbation of G that affects only edges of u_{i_2} and u_{i_3} and transforms G into an edge colored simple graph of $K_{n,m}$ satisfying conditions 1 and 2 in Lemma 146.

Proof. Consider the graph $K(G, u_{j_3}, u_{j_2})$. For each vertex c_i, $i \neq i_1, i_2, i_3$ of the graph $K(G, u_{j_3}, u_{j_2})$, the number of incoming and outgoing edges are the same, while c_{i_3} has two more outgoing edges than incoming and c_{i_1} and c_{i_2} has one-one more incoming edges than outgoing. Therefore there is a trail from vertex c_{i_3} to c_{i_2} (and also to c_{i_1}) due to the pigeonhole principle. Take a trail from c_{i_3} to c_{i_1}, and for each edge along the trail, swap the corresponding edges in G, namely, the edge between u_{j_2} and v_h and the edge between u_{j_3} and v_h. This transforms G into a graph satisfying conditions 1 and 2 in Lemma 146. Indeed, u_3 will have one less edge with color c_{i_3} and one more edge with color c_{i_1}, namely, the transformation cancels its deficiency. Vertex u_2 will have one more edge with color c_{i_3} and one less edge with color c_{i_1}, therefore its $(+c_{i_2} - c_{i_3})$ deficiency becomes a $(+c_{i_2} - c_{i_1})$ deficiency. The number of edges with other colors are not affected, since in the trail, the number of incoming and outgoing edges are the same for all colors not c_{i_1} and not c_{i_3}, and swapping the edges in G is equivalent with inverting the direction of the corresponding edges in $K(G, u_{j_3}, u_{j_2})$. □

The following is the key lemma in transforming a realization into another realization. As we mentioned above, the strategy is to transform a realization G_1 into an intermediate realization H, such that the colors of edges of a vertex u_0 in the regular class agrees with the colors of the edges of u_0 in the target realization G_2. We have to permute the edges, and the basic ingredient of a permutation is a cyclic permutation. The following lemma shows how to perturb a realization along a cyclic permutation.

Lemma 148. *Let G_1 and G_2 be two realizations of the same half-regular bipartite degree matrix $\mathcal{M}$. Let V' be a subset of vertices such that for some $u \in U$, each possible colors appears at most once on the edges between u and V' in G_1, furthermore, there exists a cyclic permutation π on V' such that for all $v \in V'$, the color between u and v in G_1 is the color between u and $\pi(v)$ in G_2. Then there exists a sequence of colored graphs $G_1 = H_0, H_1, \ldots, H_l$ with the following properties*

1. *For all $i = 1, \ldots, l-1$, H_i is a colored graph satisfying either the properties 1 and 2 in Lemma 146 or the properties 1 and 2 in Lemma 147.*

2. *For all $i = 0, \ldots, l-1$, a perturbation exists that transforms H_i into H_{i+1} and perturbs only the edges of two vertices in U.*

3. *H_l is a realization of $\mathcal{M}$ such that for all $v' \in V'$, the color of the edge between u and v' is the color between u and v' in G_2, and for all $v \in V \setminus V'$, the color between u and v is the color between u and v in G_1.*

Proof. Let $(v_{i_1}, v_{i_2}, \dots v_{i_r})$ denote the cyclic permutation, and for all $l = 1, 2, \dots r$, let c_{j_l} be the color of the edge between u and v_{i_l} in G_1. Since G_2 is a realization of $\mathcal{M}$, there is a u' such that the color between u' and v_{i_1} is c_{j_r}. Indeed, the color between u and v_{i_1} in G_2 is c_{j_r} (the permutation π moves v_{i_r} to v_{i_1}). Thus, v_{i_1} has an edge with color c_{j_r}. Swap the edges between u and v_{i_1} and between u' and v_{i_1}. This will be H_1, which satisfies the conditions 1 and 2 in Lemma 146. Indeed, u has a $(+c_{j_r} - c_{j_1})$ deficiency, while u' has a $(+c_{j_1} - c_{j_r})$ deficiency. Clearly, H_0 and H_1 differ only on edges of u and u', furthermore, all edges of u has the same color than in G_1 except the edge between u and v_{i_1}.

Assume that some H_t is achieved for which conditions 1 and 2 in Lemma 146 are satisfied with u having $(+c_{j_r} - c_{j_s})$ deficiency and with some u' having $(+c_{j_s} - c_{j_r})$ deficiency and for all vertices $v_{i_1}, v_{i_2}, \dots, v_{i_s}$, the edges between u and these vertices have a color as in G_2. Furthermore, all other edges between u and $v \neq v_{i_1}, v_{i_2}, \dots, v_{i_s}$ did not change color. The color between u and $v_{i_{s+1}}$ in G_2 is c_{j_s}, therefore, there is a u" such that the color between u" and $v_{i_{s+1}}$ is c_{j_s} in H_t. Swap the edges between u and $v_{i_{s+1}}$ and between u" and $v_{i_{s+1}}$. This will be the graph H_{t+1}. Clearly, H_t and H_{t+1} differ only on edges of u and u". If $u' = u$", then u has $(+c_{j_r} - c_{j_{s+1}})$ deficiency, and u' has $(+c_{j_{s+1}} - c_{j_r})$ deficiency, hence H_{t+1} satisfies conditions 1 and 2 in Lemma 146. We can rename H_{t+1} to H_t and iterate the chain of transformations.

If $u' \neq u$", then u has $(+c_{j_r} - c_{j_{s+1}})$ deficiency, u" has $(+c_{j_{s+1}} - c_{j_s})$ deficiency and u' has $(+c_{j_s} - c_{j_r})$ deficiency. Thus, the conditions 1 and 2 in Lemma 147 hold. Due to Lemma 147, there exists a perturbation that affects only edges on u' and u" and transforms H_{t+1} to an H_{t+2} for which conditions 1 and 2 in Lemma 146 hold with u having $(+c_{j_r} - c_{j_{s+1}})$ deficiency and u" having $(+c_{j_{s+1}} - c_{j_r})$ deficiency. We can rename H_{t+2} to H_t and iterate the chain of transformations.

With this series of transformations, we can reach H_t which satisfies conditions 1 and 2 in Lemma 146 with u having $(+c_{j_r} - c_{j_{r-1}})$ deficiency and with some u' having $(+c_{j_{r-1}} - c_{j_r})$ deficiency, furthermore, all vertices $v_{i_1}, v_{i_2}, \dots v_{i_{r-1}}$, the edges between u and these vertices have a color as in G_2, while all other edges of u did not change color. There is a u" such that the color of the edge

between u" and v_{i_r} is $c_{j_{r-1}}$. H_t is transformed into H_{t+1} by swapping the edges between u and v_{i_r} and between u" and v_{j_r}. If $u' = u$", then this transformation leads to a realization of $\mathcal{M}$, and we are ready, namely, $H_{t+1} = H_l$. Otherwise, H_{t+1} satisfies the conditions 1 and 2 in Lemma 146 with u' having $(+c_{j_{r-1}} - c_{j_r})$ deficiency and u" having $(+c_{j_r} - c_{j_{r-1}})$ deficiency. According to Lemma 146, H_{t+1} can be transformed into H_l with a perturbation affecting only edges on u' and u". □

Since H_l is also a realization of $\mathcal{M}$, it is desirable to have a series of transformation from G_1 to H_l such that all the intermediate graphs are realizations of $\mathcal{M}$. To do this, we need a slightly larger perturbation modifying the edges of three vertices in the regular vertex set, as stated in the following lemma.

Lemma 149. *Let G_1, G_2, V' and π be the same as in Lemma 148. Then there exists a sequence of colored graphs $G_1 = H'_0, H'_1, \ldots H'_{l'}$ with the following properties*

1. *For all $i = 1, \ldots l'$, H'_i is a realization of $\mathcal{M}$.*
2. *For all $i = 0, \ldots l' - 1$, a perturbation exists that transforms H'_i into H'_{i+1} and perturbs only the edges of at most three vertices in U.*
3. *$H'_{l'}$ is a realization of $\mathcal{M}$ such that for all $v' \in V'$, the color of the edge between u and v' is the color between u and v' in G_2, and for all $v \in V \setminus V'$, the color between u and v is the color between u and v in G_1.*

Proof. Consider the series of colored graphs $H_0, H_1, H_2, \ldots H_l$ obtained in Lemma 148. Find the largest t_1 satisfying $\forall 0 < t' < t_1$, $H_{t'}$ has two vertices with a deficiency. Due to the construction, for all t', these two vertices are the same, and G_1 and $H_{t'}$ differ only in edges of these two vertices, u and u'. If $t_1 = l$, then G_1 and H_l differ only on edges of two vertices of U, thus $G_1 = H'_0, H'_1 = H_l$ will suffice. Otherwise, H_{t_1} has deficiency on 3 vertices, u, u' and u". H_{t_1+1} has deficiency on 2 vertices, u and u", and G_1 differ from H_{t_1+1} on edges of three vertices, u, u' and u". Apply Lemma 146 on H_{t_1+1}, the so-obtained graph will be H'_1. H'_1 is a realization of $\mathcal{M}$ and differ from G_1 only on edges of 3 vertices in U.

Iterate this construction, find the largest t_j such that $\forall t_{j-1} < t' < t_j$, $H_{t'}$ has two vertices with deficiency. If $t_j = l$, then $l' = j$, and $H'_{l'} = H_l$.

Otherwise, H_{t_j} has 3 vertices with deficiency, and differ from H'_{j-1} only on the edges of these 3 vertices. H_{t_j+1} has two vertices with deficiency, on which Lemma 146 is applied to get H'_j. H'_j differs from H'_{j-1} only on edges of 3 vertices, since H'_{j-1} differs from $H_{t_{j-1}+1}$ on the edges of the same 2 vertices that have deficiency for all $H_{t'}$, $t_{j-1} < t' < t_j$.

After a finite number of iterations, $t_j = l$, and then $j = l'$, $H'_{l'} = H_l$, and this finishes the construction. □

Theorem 150. *Let G_1 and G_2 be realizations of the same half-regular bipartite degree matrix $\mathcal{M}$. Then there exists a sequence of colored graphs $G_1 = H_0, H_1, \dots H_l = G_2$ with the following properties.*

1. *For all $i = 0, \dots l$, H_i is a realization of $\mathcal{M}$.*
2. *For all $i = 0, \dots l-1$, a perturbation exists that transforms H_i into H_{i+1} and perturbs only the edges of at most three vertices in U.*

Proof. Consider a vertex u in the regular vertex class U, and define the following directed multigraph K. The vertices of K are the colors $c_1, c_2, \dots c_k$ and the edges are defined by the following way. For each vertex $v \in V$, there is an edge going from c_i to c_j, where c_i is the color of the edge between u and v in G_2 and c_j is the color of the edge between u and v in G_1 (if $c_i = c_j$, then this edge will be a loop). Label this edge with vertex v. Since G_1 and G_2 are realizations of the same degree matrix $\mathcal{M}$, K is Eulerian and thus, can be decomposed into directed cycles. If all cycles are trivial (loops), then for all vertices $v \in V$, the colors of the edges between u and v in G_1 and G_2 are the same. If there is a nontrivial cycle C, then each color appears in this cycle at most once, and the edges of a cycle define a cyclic permutation on a subset of vertices V. Let $(v_{i_1}, v_{i_2}, \dots v_{i_r})$ denote this cyclic permutation π. By the definition of K, the color between u and v in G_1 is the color between u and $\pi(v)$ in G_2, for all $v = v_{i_1}, v_{i_2}, \dots v_{i_r}$. Therefore, we can apply Lemma 149 to transform G_1 into $H'_{l'}$. Now if we construct the same directed multigraph K considering $H'_{l'}$, G_2 and the same vertex u, then cycle C becomes separated loops for all of its vertices, while other edges are not affected (recall the second half of condition 3 in Lemma 148, "*for all $v \in V \setminus V'$, the color between u and v is the color between u and v in G_1.*"). While there are non-trivial cycles in K, we can apply Lemma 149, and in a finite number of steps, G_1 is transformed into a realization H' such that for all vertices $v \in V$, the color of the edge between u and v in H' and G_2 are the

same. Furthermore, in all steps along the transformation, the intermediate graphs satisfy the two conditions of the lemma.

Fix the edges of u both in H' and G_2. Technically, this can be done by deleting vertex u and its corresponding edges from both H' and G_2. Then these reduced graphs will be realizations of the same half-regular degree matrix that can be obtained from $\mathcal{M} = (D, F)$ by deleting a column from the row-regular D (which thus remains row-regular) and modifying some values of F according to the edge colors of u. On these reduced graphs, we can consider a vertex u from the regular vertex class U, and do the same. Once there is one remaining vertex in U, the two realizations will be the same, and thus, we transformed G_1 into G_2. □

Theorem 150 says that perturbing the edges of at most three vertices in the regular vertex class is sufficient to connect the space of realizations of a half-regular degree matrix $\mathcal{M}$, namely, a finite series of such perturbations is sufficient to transform any realization into another such that all intermediate perturbed graphs are also realizations of $\mathcal{M}$. A natural question to ask is if such perturbations are also necessary. The answer is in the affirmative. Latin squares can be considered as 1-factorizations of the complete bipartite graph $K_{n,n}$. The corresponding degree matrix $\mathcal{M} = (D, F)$ satisfies the definition of half-regular degree matrices. Indeed, both D and F are $n \times n$, all-1 matrices. It is well-known that even some 5×5 Latin squares cannot be transformed into each other via Latin squares if only two rows are perturbed in a step, and the same claim holds for any $p \times p$ Latin squares, where p is prime and $p \geq 5$ [51].

12.1.4 Markov Chain Monte Carlo for sampling realizations of a half-regular degree matrix

In this subsection, we give a Markov Chain Monte Carlo (MCMC) method for sampling realizations of a half-regular degree matrix $\mathcal{M}$. Theorem 150 says that the perturbations that change the edges of at most 3 vertices of the regular vertex class are irreducible on the realizations of half-regular degree matrices. A naïve approach would make a random perturbation affecting the edges of 3 vertices, and would accept this random perturbation if it is a realization of $\mathcal{M}$. Unfortunately, the probability that such random perturbation would generate a realization of $\mathcal{M}$ would tend to 0 exponentially quickly with the size of $\mathcal{M}$, making the MCMC approach very inefficient. Indeed, there

were an exponential waiting time for an acceptance event, that is, when the random perturbation generates a realization of $\mathcal{M}$.

Even if the random perturbation generates a realization of $\mathcal{M}$ with probability 1, the Metropolis-Hastings algorithm applying such random perturbation might generate a torpidly mixing Markov chain if the acceptance ratios are small. An example when this is the case can be found in [119] (see also Chapter 7).

In this subsection, we design a transition kernel that generates such perturbations and its transition probabilities satisfy the property that the inverse of the acceptance ratio bounded by a polynomial function of the size of $\mathcal{M}$. To do this, we first have to generate random circuits and trails in auxiliary graphs that we define below.

Definition 151. *Let $K(G, u, u')$ be the directed, edge labeled multigraph of the edge colored bipartite graph G as defined in Definition 145. Let c_s and c_e be two vertices of K such that for any vertex $c \neq c_e$ of K, the number of outgoing edges of c is greater or equal than the number of its incoming edges, and c_s has more outgoing edges than incoming edges if $c_s \neq c_e$. We define the following function $f(K, c_s, c_e)$ that generates a random trail from c_s to c_e when $c_s \neq c_e$ and a random circuit when $c_s = c_e$.*

1. *Select uniformly a random outgoing edge e of c_s which is not a loop. Let the sequence of labels $P := v$, where v is the label of e, namely, P be a sequence containing one label. Let c be the endpoint of e.*

2. *While $c \neq c_e$, select uniformly a random outgoing edge e of c which is not a loop and does not appear in P. Extend P with the label of e, and set c to the endpoint of e.*

Note that due to the pigeonhole principle, this random procedure will eventually arrive at c_e, since for any vertex $c \neq c_e$ reached by the algorithm, there exists at least one outgoing edge of c which is not a loop and is not in the sequence P.

The probability of a trail $\mathcal{T} = c_s, c_1, c_2, \ldots c_r, c_e$ or circuit $\mathcal{C} = (c_s, c_1, c_2, \ldots c_r)$ generated by $f(K, c_s, c_e)$ consists of the product of the inverses of the available edges going out from $c_s, c_1, c_2, \ldots c_r$. These probabilities will be denoted by $P(\mathcal{C})$ and $P(\mathcal{T})$.

We are going to perturb the graphs along the circuits and trails defined below.

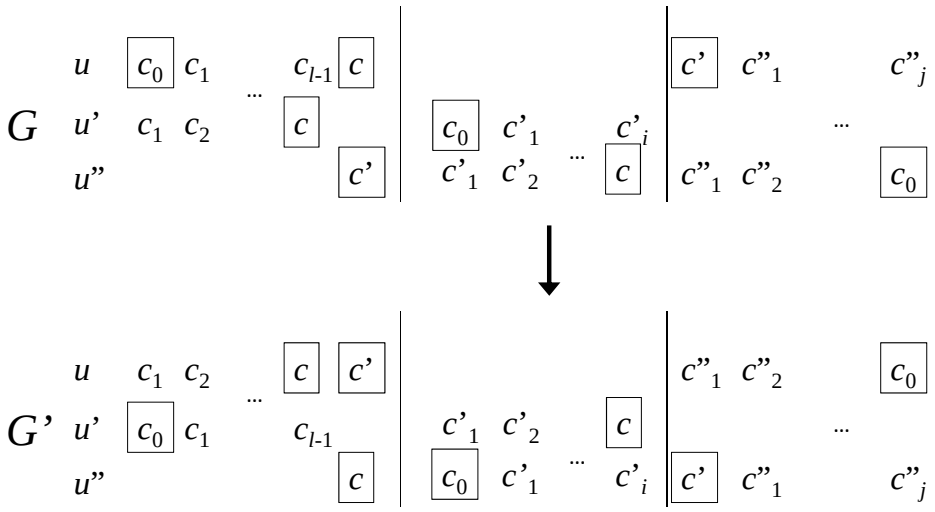


Figure 12.3: This figure shows how Algorithm 1, case II.(c)-(e) perturbs G into a G'. Vertices u, u' and u" are the vertices whose edges are perturbed. Pairs of colors are swapped in three trails (see also Definition 151), furthermore a pair of colors c and c' are swapped. Some colors are boxed in order to help understand that the perturbation yields a realization of $\mathcal{M}$, see also the proof of Theorem 153. The transformation swapping the columns until the first vertical line yields $\tilde{G}$, until the second vertical line yields $\bar{G}$. Note that for sake of readability, all color changes are indicated in separate columns, however, it is allowed that a column is used several times during the perturbation. Also note that due to readability, columns are in order as they follow each other in a trail, however, they might not be necessarily consecutive ones in G.

Definition 152. *Let $\mathcal{C}$ (resp., $\mathcal{T}$) be a random circuit (resp., random trail) generated by $f(K(G,u,u'),c_s,c_e)$. Then the transformation $G*(\mathcal{C},u,u')$ ($G*(\mathcal{T},u,u')$) swaps the colors of the edges between u and v and between u' and v for all edge labels v in $\mathcal{C}$ ($\mathcal{T}$).*

With these types of perturbations, we are going to define the random algorithm that generates a random perturbation.

Algorithm 1. *Let $G(U,V,E)$ be a realization of the half-regular degree matrix $\mathcal{M}$. Do the following random perturbation on G.*

I With probability $\frac{1}{2}$, do nothing. This is technically necessary for the Markov chain to be aperiodic. Any constant probability would suffice, $\frac{1}{2}$ is the standard choice for further technical reasons not detailed here (see, for example, [83]).

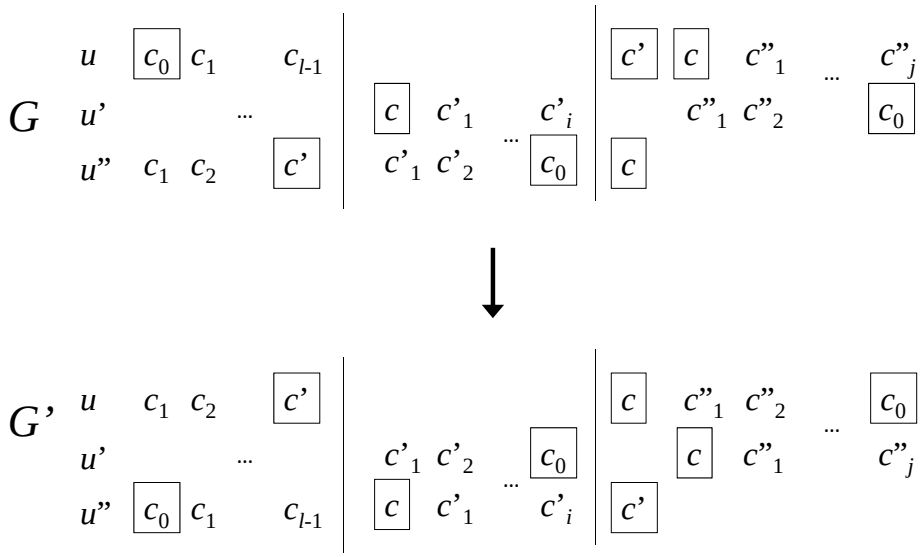


Figure 12.4: This figure shows how Algorithm 1, case III.(c)-(e) perturbes G into a G'. Vertices u, u' and u" are the vertices whose edges are perturbed. Couples of colors are swapped in three trails (see also Definition 151), furthermore a pair of colors c and c' are swapped. Some colors are boxed in order to help understand that the perturbation yields a realization of $\mathcal{M}$, see also the proof of Theorem 153. The transformation swapping the columns until the first vertical line yields $\bar{G}$, until the second vertical line yields $\tilde{G}$. Note that for sake of readability, all color changes are indicated in separate columns, however, it is allowed that a column is used several times during the perturbation. Also note that due to readability, columns are in order as they follow each other in a trail, however, they might not be necessarily consecutive ones in G.

II With probability $\frac{1}{4}$, do the following

(a) *Draw uniformly a random ordered pair of vertices u and u' from the vertex set U. Construct the auxiliary graph $K(G, u, u')$, draw uniformly a random color c_0 and draw a random circuit $\mathcal{C}$ using $f(K(G, u, u'), c_0, c_0)$.*

(b) *Chose a random integer l uniformly from $[1, m]$. If $l \geq |\mathcal{C}|$, then let G' be $G*(\mathcal{C}, u, u')$. G' is the perturbed graph, exit the algorithm.*

(c) *If $l < |\mathcal{C}|$, do the following (see also Figure 12.3). Let $e_1, e_2, \ldots$ denote the edges of circuit $\mathcal{C}$ starting with the edge going out from c_0. Let $\mathcal{T}$ be the trail with edges $e_1, e_2, \ldots e_l$. Perturb G to $G * (\mathcal{T}, u, u')$. Draw uniformly a u" from $U \backslash \{u, u'\}$ and also uniformly a vertex v from the subset of V satisfying the condition that the color of the edge between u and v in $G * (\mathcal{T}, u, u')$ is the last color in the trail $\mathcal{T}$. Let c' denote the color of the edge between u" and v . Swap the colors of the edges between u and v and between u" and v. Note that c' might equal to c, and in that case, swapping the colors has no effect. Denote the thusfar perturbed graph $\tilde{G}$.*

(d) *Construct $K(\tilde{G}, u', u")$, and draw a random trail $\mathcal{T}_1$ using function $f(K(\tilde{G}, u', u"), c_0, c)$. Let $\bar{G}$ be $\tilde{G} * (\mathcal{T}_1, u', u")$.*

(e) *Construct $K(\bar{G}, u, u")$, and draw a random trail $\mathcal{T}_2$ using function $f(K(\bar{G}, u, u"), c', c_0)$. Let G', the perturbed graph be $\bar{G}*(\mathcal{T}_2, u, u")$.*

III With probability $\frac{1}{4}$, do the following.

(a) *Draw uniformly a random ordered pair of vertices u and u" from the vertex set U. Construct $K(G, u, u")$, draw uniformly a random color c_0, and draw a random circuit $\mathcal{C}' = (c_0, c_1, \ldots)$ using $f(K(G, u, u"), c_0, c_0)$.*

(b) *Draw uniformly a random integer from $[1, m]$. If $l \geq |\mathcal{C}'|$, then let G' be $G * (\mathcal{C}', u, u")$. G' is the perturbed graph, exit the algorithm.*

(c) *If $l < |\mathcal{C}'|$, do the following (see also Figure 12.4). Let $\mathcal{T}'$ be the trail from c_0 to c_l, and let $\bar{G} = G * (\mathcal{T}', u, u")$, and let c' denote c_l. Draw uniformly a random vertex u' from $U \setminus \{u, u"\}$. Build $K(\bar{G}, u", u')$, and draw a random trail $\mathcal{T}_1'$ using $f(K(\bar{G}, u", u'), c_0, c')$. Let $\mathcal{T}_1'[i]$ denote the prefix of the trail $\mathcal{T}_1'$ containing only the first i edges of the trail. Let e_i denote the ith edge of trail $\mathcal{T}_1'$. Construct*

the set of vertices V' which contains all vertex v_i such that the label of e_i is v_i and in the graph $\bar{G} * (\mathcal{T}_1'[i], u", u')$, there is a vertex v such that the color of the edge between u and v is c', the color of the edge between $u"$ and v is the color of the edge between u' and v_i. Furthermore, c is the first occurrence in the trail, that is, $\mathcal{T}_1'[i-1]$ does not contain c.

(d) If V' is empty, then terminate the algorithm, let the perturbed graph be the original graph. Otherwise, draw a random vertex v from V'. Shorten $\mathcal{T}_1'$ such that the last edge have label v. Let $\mathcal{T}_1"$ denote this trail. Let $\tilde{G}$ be $\bar{G} * (\mathcal{T}_1", u", u')$. If $u"$ has a deficiency $(+c - c')$ for some c in $\tilde{G}$, then draw uniformly a vertex among the vertices $v"$ for which the color of the edge between u and $v"$ is c' and the color of the edge between $u"$ and $v"$ is c. Modify $\tilde{G}$ by swapping the colors of these two edges.

If $u"$ has no deficiency in $\tilde{G}$, then rename c' to c.

(e) Build $K(\tilde{G}, u, u')$, and draw a random trail $\mathcal{T}_2'$ using $f(K(\tilde{G}, u, u'), c, c_0)$. Let G', the perturbed graph be $\tilde{G} * (\mathcal{T}_2', u, u')$.

Theorem 153. *The random perturbation in Algorithm 1 has the following properties.*

1. *The generated G' is a realization of $\mathcal{M}$, thus this random perturbation is the transition kernel of a Markov chain on the realizations of $\mathcal{M}$.*
2. *The perturbations are irreducible on the realizations of $\mathcal{M}$*
3. *The perturbations form a reversible kernel and for any G and any G' generated from G,*

$$\frac{T(G|G')}{T(G'|G)} \geq \frac{2}{m^5} \tag{12.5}$$

where m is the number of vertices in V. Furthermore, if this reversible kernel is used in a Metropolis-Hastings algorithm, the expected waiting time (that is, how many Markov chain steps is necessary to leave the current state) at any state is bounded by $2m^5$.

Proof.

1. When $l \geq |\mathcal{C}|$ or $l \geq |\mathcal{C}'|$ (cases II.(b) and III.(b) in Algorithm 1), the constructed G' differs on edges of two vertices of U, u and u' (or u and

u"). Since the swapped colors are along a circuit in the auxiliary graph K, the effects of color swaps cancel each other, and we get a realization of $\mathcal{M}$.

When $l < |\mathcal{C}|$ (case II.(c)-(e) in Algorithm 1), G is transformed into an intermediate graph $\tilde{G}$, which has deficiency on three vertices u, u' and u". Vertex u has $(+c' - c_0)$-deficiency, u' has $(+c_0 - c)$-deficiency, and u" has $(+c - c')$-deficiency (see also the boxed colors on Figure 12.3). The randomly generated trail $\mathcal{T}_1$ on $K(\tilde{G}, u'u")$ goes from c_0 to c. In Lemma 147, we proved that any such trail transforms $\tilde{G}$ into a graph satisfying the conditions of Lemma 146. The random trail $\mathcal{T}_2$ generated in $K(\bar{G}, u, u")$ is from c' to c_0, and Lemma 146 proves that swapping the edge colors along such trail transforms the graph into a realization of $\mathcal{M}$.

When $l < |\mathcal{C}'|$ (case III.(c)-(e) in Algorithm 1), G is transformed into a $\bar{G}$, in which u has a $(+c' - c_0)$-deficiency and u" has a $(+c_0 - c')$ deficiency (see also the boxed colors on Figure 12.4). Then $\bar{G}$ transformed into a $\tilde{G}$, in which two cases is possible. Either u" has a $(+c - c')$-deficiency and then u has a $(+c' - c_0)$-deficiency and u' has a $(+c_0 - c)$-deficiency or u" has no deficiency and then u has a $(+c' - c_0)$-deficiency and u' has a $(+c_0 - c')$-deficiency. In the former case, a pair of colors c and c' are swapped between u and u" causing u" has no deficiency and u has $(+c - c_0)$ deficiency, which are cancelled by the perturbation along the trail $\mathcal{T}_2'$, thus arriving to a realization of $\mathcal{M}$. In the later case, the perturbation along the trail $\mathcal{T}_2'$ directly leads to a realization of $\mathcal{M}$.

2. It is sufficient to show that the algorithm generates the perturbations appear in Theorem 150, which are actually the perturbations appear in Lemma 149. There are two types of perturbations in Lemma 149. The simpler one swaps the colors of edges along a cycle. This happens when $t_1 = l$ in the proof of Lemma 149 and also when $t_j = l$. Since a cycle is a circuit, and any circuit in the auxiliary graph K constructed in the algorithm can be generated with non-zero probability, this type of perturbation is in the transition kernel.

   The larger perturbation in Lemma 149 first swap the colors of edges of u and u' along a path, then swap the colors of two edges between u and v and u" and v for some v, then swap the edges of u' and u" along a path

to eliminate the deficiency of u", finally, swap the color of the edges u and u" along a path to eliminate their deficiencies, thus generate a realization. Since any path is a trail, the given algorithm can generate such perturbations when $l < |\mathcal{C}|$ (case II.(c)-(e) in Algorithm 1).

3. We show that for any perturbation of G to G', there exists a perturbation from G' to G, and we also compare the probabilities of the perturbations. Recall that the number of vertices in U is n, the number of vertices in V is m, and the number of colors is k; n, m and k will appear in the probabilities below several times.

   When the perturbation affects the edges of only two vertices, u and u' or u and u" (cases II.(b) or III.(b) in Algorithm 1), the algorithm first draws these vertices with probability $\frac{1}{n(n-1)}$, where $n = |U|$. Also a random color c_0 is drawn with probability $\frac{1}{k}$. Then a random circuit $\mathcal{C} = (c_0, c_1, c_2, \dots c_r)$ is generated with probability $P(\mathcal{C})$. Finally, a random $l \geq r$ is generated, this happens with $\frac{m-r-1}{m}$ probability, where $r = |\mathcal{C}|$ and the color of the edges indicated by the circuit are swapped. The so-obtained G' can be generated not only in this way, but an arbitrary such c_i can be generated as the starting point of the circuit which is visited only once in the circuit.

   To perturb back G' to G, we first have to select the same pair of vertices u and u', with the same, $\frac{1}{n(n-1)}$ probability. Then the random circuit $\mathcal{C}' = (c_0, c_r, c_{r-1}, \dots c_1)$ must be generated in the constructed auxiliary directed multigraph K'. Since for all color, the auxiliary graph K constructed from G has the same outgoing edges than the auxiliary graph K' constructed from G', $P(\mathcal{C}) = P(\mathcal{C}')$. Furthermore, this claim holds for all possible circuits having the possible starting colors c_i. Finally, generating the same l has the same probability, thus we can conclude that for this type of perturbation, $P(G'|G) = P(G|G')$, thus Equation 12.5 holds.

   When the perturbation affects the edges of three vertices, u, u' and u", then G is first perturbed to $\tilde{G}$, $\tilde{G}$ is perturbed to $\bar{G}$ and finally $\bar{G}$ is perturbed to G', or the third type of perturbation is applied in the algorithm where G first perturbed into $\bar{G}$ then $\tilde{G}$ and finally G' (case II.(c)-(e) and III.(c)-(e) in Algorithm 1). We show that these two types of perturbations are inverses of each other in the following sense. For any perturbation generated in case II.(c)-(e) in Algorithm 1, its

inverse can be generated in case III.(c)-(e) in Algorithm 1, and vice versa, for any perturbation generated in case III.(c)-(e), its inverse can be generated in case II.(c)-(e).

First we consider the perturbation generated by Algorithm 1 in case II.(c)-(e). In that case, first the ordered pair of vertices (u, u') is generated with probability $\frac{1}{n(n-1)}$ and a random color c_0 with probability $\frac{1}{k}$. A circuit $\mathcal{C} = (c_0, c_1, \dots c_l, \dots c_r)$ is generated together with a random number l with probability $\frac{1}{m}$. The generated number l must be smaller than $|\mathcal{C}|$, and thus, only the trail $\mathcal{T} = c_0, c_1, \dots c_l$ is important. The probability of the trail consists of the product of the inverses of the number of available outgoing edges. Let $P(\mathcal{T})$ denote this probability. The colors along the trail are swapped. Then random u" is generated from the appropriate set of vertices in $U \setminus \{u, u'\}$ with probability $\frac{1}{n-2}$ and also a vertex v is generated with probability $\frac{1}{d_{j,1}+1}$ where j is the index of color c_l. The colors $c(= c_l)$ and c' are swapped. Therefore, $P(\tilde{G}|G) = \frac{1}{n(n-1)(n-2)km(d_{j,1}+1)} P(\mathcal{T})$. Then a trail $\mathcal{T}_1$ is generated from c' to c_0 in the auxiliary graph $K(\tilde{G}, u'u")$, and $\tilde{G}$ is transformed to $\bar{G} = \tilde{G} * (\mathcal{T}_1, u', u")$ Thus, $P(\bar{G}|\tilde{G}) = P(\mathcal{T}_1)$. Finally, a random trail $\mathcal{T}_2$ from c' to c_0 is generated in $K(\bar{G}, u, u")$ with probability $P(\mathcal{T}_2)$, which is $P(G'|\bar{G})$. Since

$$P(G'|G) = P(\tilde{G}|G)P(\bar{G}|\tilde{G})P(G'|\bar{G})$$

we get that

$$P(G'|G) = \frac{1}{n(n-1)(n-2)km(d_{j,1}+1)} P(\mathcal{T})P(\mathcal{T}_1)P(\mathcal{T}_2). \qquad (12.6)$$

To transform back G' to G, first G' should be transformed back to $\bar{G}$, then $\bar{G}$ back to $\tilde{G}$ and finally $\tilde{G}$ back to G. This can be done in case III.(c)-(e) by first drawing the same u and u", then drawing a $\mathcal{T}'$ which is exactly the inverse of $\mathcal{T}_2$, then the same u' must be selected and the trail $\mathcal{T}_1$" must be the inverse of $\mathcal{T}_1$, the same edges with color c and c' must be swapped, and finally the trail $\mathcal{T}_2'$ must be the inverse of $\mathcal{T}$. First we show that these trails can be inverses of each other.

$\mathcal{T}$ is a shortening of a circuit with start and end vertex c_0. As such, it contains c_0 only once, but might contain c several times. $\mathcal{T}_2'$ is a trail

from c to c_0. Therefore, it can contain c several times, but contains c_0 only once. Thus, the inverse of any $\mathcal{T}$ might be a $\mathcal{T}_2'$ and vice versa.

$\mathcal{T}_1$ is a trail from c_0 to c. Therefore, it might contain c_0 several times, but contains c only once. $\mathcal{T}"_1$ is a shortening of a trail from c_0 to c'. It might contain c_0 several times, but can contain c only once, due to its definition (see case III.(c)-(d) of Algorithm 1). Hence, the inverse of any $\mathcal{T}_1$ can be a $\mathcal{T}"_1$, and vice versa.

$\mathcal{T}_2$ is a trail from color c' to c_0. It might hit c' several times, but only once c_0. $\mathcal{T}'$ is a shortening of a circuit with start and end vertex c_0. As such, it contains c_0 only once (as start and end vertex), but might contain c' several times. Thus the inverse of any $\mathcal{T}_2$ might be a $\mathcal{T}'$ and vice versa.

We are going to calculate the probability of a random perturbation in case III.(c)-(e). First, the ordered pair of vertices u and $u"$ should be selected with probability $\frac{1}{n(n-1)}$. The auxiliary directed multigraph $K(G', u, u")$ is constructed, a random c_0 is selected, and a random circuit $(c_0, c_1 \dots c', \dots c_{r'})$ in $K(G', u, u")$ is generated. The probability that the randomly generated l is exactly the index of the appropriate occurrence of c' in the trail $\mathcal{T}'$ is $\frac{1}{m}$. Then only the trail $\mathcal{T}' = c_0, c_1, \dots c'$ is interesting. Thus, transforming back G' to $\bar{G}$ has probability $\frac{1}{n(n-1)km} P(\mathcal{T}')$. $\mathcal{T}'$ in $K(G', u, u")$ is the inverse of the trail $\mathcal{T}_2$ in $K(\bar{G}, u, u")$ and $K(G', u, u")$ and $K(\bar{G}, u, u")$ differs in inverting the trail $\mathcal{T}_2$. Therefore $P(\mathcal{T}_2)$ and $P(\mathcal{T}')$ differ in the number of outgoing edges from c_0 at the begining of the trail $\mathcal{T}_2$ and the number of outgoing edges from c' at the beginning of the trail $\mathcal{T}'$. Since the number of outgoing edges might vary between 1 and m, the ratio of the two probabilities bounded by

$$\frac{1}{m} \leq \frac{P(\mathcal{T}')}{P(\mathcal{T}_2)} \leq m \tag{12.7}$$

After swapping the colors along the trail T', u' should be randomly generated, it has probability $\frac{1}{n-2}$. A random trail $\mathcal{T}_1'$ is generated using $f(K(\bar{G}, u', u"), c_0, c')$. A random v from the subset V' is generated in III.(d) of the algorithm. This has probability $\frac{1}{|V'|}$. Then the trail $\mathcal{T}_1'$ is shortened to $\mathcal{T}_1"$, and $\tilde{G}$ is obtained by transforming $\bar{G}$ along this trail.

It is not easy to calculate exactly the probability $P(\tilde{G}|\bar{G})$ since the set V' depends on the generated trail $\mathcal{T}_1'$. However, the size of V' cannot be greater than m and lower than 1, therefore the following inequality holds:

$$\frac{1}{m}P(\mathcal{T}_1\text{"}) \leq P(\tilde{G}|\bar{G}) \leq P(\mathcal{T}_1\text{"})$$

The trail $\mathcal{T}_1$" is the inverse trail $\mathcal{T}_1$, therefore, the ratio of their probabilities is also between $\frac{1}{m}$ and m.

Finally, the same edge colors c and c' must be swapped back, and $\tilde{G}$ must be transformed back to G along the trail $\mathcal{T}_2'$. The probability that the selected v" in III.(d) of the Algorithm 1 is a particular vertex depends on the number of vertices from which v" is selected. Therefore this probability is again between $\frac{1}{m}$ and 1. Altogether, the probability of the backproposal probability is bounded between

$$\begin{aligned}\frac{1}{n(n-1)(n-2)km^3}P(\mathcal{T}')P(\mathcal{T}_1\text{"})P(\mathcal{T}_2') \quad &\leq P(G|G') \leq \\ \leq \frac{1}{n(n-1)(n-2)km}P(\mathcal{T}')P(\mathcal{T}_1\text{"})P(\mathcal{T}_2') \qquad &(12.8)\end{aligned}$$

Comparing Equations 12.6 and 12.8, and also considering that the ratio of the probabilities of the trails and corresponding inverse trails are between $\frac{1}{m}$ and m, we get for the ratio of proposal and backproposal probabilities that

$$\frac{2}{m^5} \leq \frac{P(G|G')}{P(G'|G)} \leq m^4 \qquad (12.9)$$

Case II. is chosen with probability $\frac{1}{4}$. Given that case II. is selected, the probability that Algorithm 1 generates a realization being different from the current realization is 1. If edges of two vertices in U are perturbed, then the perturbation is accepted with probability 1. If edges of three vertices in U are perturbed, then the perturbation is accepted with probability

$$\frac{P(G|G')}{P(G'|G)} \geq \frac{2}{m^5}$$

according to Equation 12.9. Thus, the probability that the Markov chain defined by the Metropolis-Hastings algorithm does not remain in the same state is greater or equal than $\frac{1}{2m^5}$. The expected waiting time is upper bounded by the inverse of this probability, that is, $2m^5$.

□

12.2 Edge colorings of bipartite graphs

12.3 Preliminaries

First, we start with some basic definitions in graph theory.

Definition 154. *A* path *in a simple graph is a sequence of edges that joins a sequence of distinct vertices. A* walk *in a simple graph is a sequence of edges that joins a sequence of (non-necessarily distinct) vertices. Throughout this paper we consider only walks in which the edges are all distinct, though the vertices might be repeated. A* cycle *in a simple graph is a walk in which the only repeated vertices are the first and the last vertices.*

Observe that in any path, the edges are also distinct for the vertices are distinct.

12.3.1 Almost edge k-colorings

Next, we introduce the almost edge k-colorings and an important lemma on them. We define formally edge k-colorings and almost edge k-colorings:

Definition 155. *An* edge k-coloring *of a graph $G = (V, E)$ is a map $C : E \to \{c_1, c_2, \ldots, c_k\}$ such that there is no adjacent monochromatic edges.*

*In a map $C : E \to \{c_1, c_2, \ldots, c_k\}$, a vertex $v \in V$ has a $(+c - c')$-*deficiency *if there are exactly two c-edges incident to v, all other incident edges have different colors, and there is no c'-edge incident to v. A vertex is not deficient if all of its incident edges have different colors. We say that a vertex v has an* at most a $(+c - c')$-deficiency *if either v has a $(+c - c')$-deficiency or v is not deficient and has no incident edge with c' color.*

An almost edge k-coloring *of $G = (V, E)$ is a map $C : E \to \{c_1, c_2, \ldots, c_k\}$ such that at most two vertices have deficiency and all other vertices have no adjacent monochromatic edges.*

If $c \in \{c_1, c_2, \ldots, c_k\}$ and C is a coloring of G then $G\big|_{C,c}$ is the subgraph that contains only the edges colored by c.

Observe that a deficient vertex in an almost edge k-coloring might have both $(+c - c')$-deficiency and $(+c - c'')$-deficiency for some colors $c' \neq c''$ if the degree of the vertex is smaller than k. We also would like to highlight that any edge k-coloring is also an almost edge k-coloring

We show how to transform almost edge k-colorings to edge k-colorings.

Lemma 156. *Let C be an almost edge k-coloring of a bipartite graph G with one or two deficient vertices. Then C can be transformed to an edge k-coloring by flipping colors along at most two alternating paths with some colors c and c' (and possibly with colors c'' and c''', that is, the two alternating paths do not have to share any color).*

Proof. Let v_1 be a $(+c - c')$-deficient vertex, and let e and f be its incident edges with color c. Consider one of the longest alternating walks with colors c and c' that contains e. That is, a walk that contains the edge e, contains edges with alternating colors c and c', and cannot be extended at either of its ends. Assume that this walk revisits some vertex w. Then, the walk first arrives via an edge with color c_1 (such first arrival exists if $w \neq v_1$) and leaves w via an edge with color c_2. Since the graph is bipartite, the walk can arrive back to w only via an edge with color c_1. Observe that w cannot be v_1 since v_1 does not have an incident edge with color $c_1 = c'$. Furthermore, the walk must stop there, since it should leave via an edge with color c_2. This would mean that w has two incident edges with color c_1 and also two incident edges with c_2. However, such a vertex does not exist in an almost edge coloring. Still, w has a deficiency, which must be the other deficient vertex v_2.

If the walk does not return to any vertex, then it stops at a vertex u. The walk either arrives to u via an edge with color c, in which case u does not have an incident edge with color c', or it arrives via an edge with color c', in which case u does not have an incident edge with color c.

To conclude, the longest alternating walk we select with colors c and c' is either a path that ends in the aforementioned vertex u or is a walk that ends in the second visit of the other deficient vertex v_2 and no other vertex is revisited.

Now we flip the colors of the edges along the path between v_1 and u or between v_1 and the first visit of v_2. We claim that this perturbation eliminates

the deficiency of v_1 and does not create a new deficiency (although the type of deficiency of v_2 might be changed).

Indeed, the color of the edge e is changed from c to c', and this eliminates the $+c-c'$-deficiency of v_1. All intermediate vertices in the path have two edges in the path, one with color c and one with color c'. Flipping the colors of the edges does not create deficiencies on these vertices. If the last vertex is u, only one of its incident edges change color. However, this does not create a deficiency because the path is a longest one. Finally, assume the case when the last vertex in the path is v_2. Before the flip, v_2 has two incident edges with color c_1 and one incident edge with c_2. After flipping the colors, v_2 has two incident edges with color c_2 and one incident edge with color c_1. That is, the flip changes the $(+c_1-c")$-deficiency of v_2 to a $(+c_2-c")$-deficiency.

If there is one more deficient vertex remaining after the first flip, we do the same procedure with it. Observe that in this case the longest alternating walk must be a path because the almost edge coloring has exactly one deficient vertex. □

12.4 A Markov chain Monte Carlo on the edge k-colorings of a bipartite graph

Below we define a Markov chain on the edge k-colorings of a bipartite graph $G=(V,E)$. We are going to prove that this Markov chain is irreducible, its diameter is at most $2k|E|$, and when applied in a Metropolis-Hastings algorithm, the inverse of the acceptance ratio is bounded from above by a cubic function of $|V|$.

Definition 157. *Let G be a bipartite graph, and let $k \geq \Delta$, where Δ is the maximum degree of G. We define a Markov chain $M(G,k)$ on the edge k-colorings of G. Throughout this definition, whenever it is impossible to select a given object since there is none, then the algorithm does nothing. Let the current coloring be C. Then draw a random edge k-coloring in the following way.*

1. *With probability $\frac{1}{2}$, we do nothing (so the defined Markov chain is a Lazy Markov chain).*

2. *With probability $\frac{1}{4}$, we select two colors c and c' from the set of colors uniformly among the $\binom{k}{2}$ possible unordered pairs. Consider the sub-*

graph of G with colors c and c'; let it be denoted by H. Its non-trivial components are paths and cycles. Then we select one of the followings:

- *(a) With probability $\frac{1}{3}$, we do nothing.*
- *(b) With probability $\frac{1}{3}$, we uniformly select a subpath or a cycle of H from all paths and cycles in H. That is, a subpath might be selected from a cycle, too.*
- *(c) With probability $\frac{1}{3}$, we uniformly select two distinct subpaths of H from all possible pairs of subpaths such that each of them has exactly one end vertex which is also an end vertex in H.*

We flip the colors in the selected subpaths or possibly a cycle. If there are two deficient vertices, we select one of them uniformly, then we select one of the edges with the same color incident to the selected vertex uniformly and change its color from c to c' or c' to c. If there is only one deficient vertex with a deficiency $(+c-c')$, then with probability $\frac{1}{2}$, we select this deficient vertex, then we select one of the edges with the same color incident to it uniformly and change its color from c to c' or c' to c. We denote this subcase by (i) for reference (we will refer to these roman number indexes in the proof of Theorem 159). Otherwise, with probability $\frac{1}{2}$, we select uniformly a c'-edge from all the c'-edges, and change its color to c. We denote this subcase by (ii) for reference.

If there is no deficient vertex, then with probability $\frac{1}{2}$, we select either a c-edge or a c'-edge uniformly from all edges with color c and c', and change its color to the opposite (c' or c) (we denote this subcase by (iii) for reference) and with probability $\frac{1}{2}$, we do nothing. We eliminate all deficiencies by flipping the colors on appropriate alternating path, selecting uniformly from the two neighbor edges with the same color incident to the deficient vertices.

3. *With probability $\frac{1}{4}$ we select three colors uniformly from all possible $\binom{k}{3}$ unordered pairs, and select uniformly one from the three colors. Let the distinguished color be denoted by c'', and let the other two colors be denoted by c and c'. Consider the subgraph consisting of the colors c and c'; let it be denoted by H. Its non-trivial components are paths and cycles. Then we select one of the followings:*

   - *(a) With probability $\frac{1}{3}$, we do nothing.*

(b) *With probability* $\frac{1}{3}$*, we uniformly select a subpath of* H *from all subpaths such that at least one of its end vertices is not an end vertex in* H*. Furthermore, if both ends of the selected subpath is not an end vertex from the selected path, then the edges incident to the end vertices of the subpath must have different colors. These subpaths can also be selected from cycles.*

(c) *With probability* $\frac{1}{3}$*, we uniformly select two distinct subpaths of* H *from all possible pair of subpaths such that each of them has exactly one end vertex which is also an end vertex in* H*. Furthermore, the end vertices which are not end vertices in* H *must have incident edges with different colors in the subpaths.*

We flip the colors in the selected subpaths. If there are two deficient vertices, we select one of them uniformly, and denote it by v*. If there is only one deficient vertex, then with probability* $\frac{1}{2}$ *we select it and denote it by* v*, and with probability* $\frac{1}{2}$*, we select uniformly a non-deficient vertex among those which are incident to both a* c''*-edge and at least one of a* c*- or a* c'*-edge and denote it by* v*. If there is no deficient vertex, then we select uniformly a non-deficient vertex among those which are incident to both a* c''*-edge and at least one of a* c*- or a* c'*-edge and denote it by* v*.*

If there is a $(+c-c')$*-deficient vertex above* v*, and there is an alternating* c'*-*c'' *path* P *from* v *to that deficient vertex, then exclude* P *from finding a maximal path or cycle in the following procedure.*

If there is a $(+c-c')$*-deficient vertex above* v*, then we do the followings. If there is an alternating* c'*-*c'' *cycle on* v*, we select it. Otherwise, we select one of the maximal alternating* c'*-*c'' *paths containing* v*; if there are two of them, we select one of them uniformly. We flip the colors in this path or cycle. We denote this subcase by (iv) for reference.*

If there is no deficient vertex above v *(we do not require* v *be a deficient vertex), then we do the followings. If* v *is incident to both a* c *and a* c'*-edge, then we select one of the colors uniformly, and denote it by* $\tilde{c}$*. Otherwise, let the incident color be denoted by* $\tilde{c}$*. If there is an alternating* $\tilde{c}$*-*c'' *cycle on* v*, we select that cycle. Otherwise, we select one of the maximal alternating* $\tilde{c}$*-*c'' *paths containing* v*, if there are two of them, we select one of them uniformly. We flip the colors in this path or cycle. We denote this subcase by* (v) *for reference.*

We eliminate all deficiencies by flipping the colors on appropriate alternating path, selecting uniformly from the two neighbor edges with the same color incident to deficient vertices.

First we prove that $M(G,k)$ is irreducible and has a small diameter. That is, the perturbations presented in it are sufficient to transform any edge k-coloring C_1 to another edge k-coloring C_2 of a bipartite graph $G=(V,E)$ in at most $2k|E|$ steps. Our strategy is given below:

1. Fix an order of colors appearing in C_2: $c_1, c_2, \ldots, c_l$. We will have that C_1 is transformed via milestone realizations, i.e. edge colorings, $C_1 = K_0, K_1, K_2, \ldots, K_{l-1} = C_2$ such that for each $i = 1, 2, \ldots, l-2$ and each $1 \leq j \leq i$, $G\big|_{K_i,c_j} = G\big|_{C_2,c_j}$. Furthermore, when K_i is transformed into K_{i+1}, no edge with color $c_1, c_2, \ldots, c_i$ changes color. These K_i are called *large milestones*. Milestones are always edge colorings.

2. For each $i = 1, 2, \ldots, l-2$ we consider $H_i := G\big|_{K_{i-1},c_i} \bigoplus G\big|_{C_2,c_i}$ where $\bigoplus$ denotes the symmetric difference. The maximal degree in H_i is 2, hence H_i can be decomposed into isolated vertices, paths, and cycles. Let the non-trivial components of H_i be ordered and denoted by $N_1, N_2, \ldots, N_m$. The milestone K_{i-1} is transformed into K_i via milestones $K_{i-1} = L_0, L_1, \ldots, L_m = K_i$ such that for all j, L_j contains color c_i only in H_i, agrees with K_i in color c_i on components $N_1, N_2, \ldots, N_j$, and agrees with K_{i-1} in color c_i on components $N_{j+1}, N_{j+2}, \ldots, N_m$. These L_i are called *small milestones*. The last transition from K_{l-2} to $K_{l-1} = C_2$ is handled in a separate way.

3. For each $j = 1, 2, \ldots, m$, the transformation from L_{j-1} to L_j goes in the following way. In the description of transformations, we use almost edge colorings (see Definition 155), in which at most two vertices are incident to two edges with the same color. First we give a transformation via such almost edge colorings $A_1, A_2, \cdots$ using transformations $f_1, f_2, \cdots$. Then we show how to transform these almost edge colorings to edge colorings $X_1, X_2, \cdots$ using transformations $\varphi_1, \varphi_2, \cdots$. The transformation between edge colorings X_t and X_{t+1} is the composition of transformations $\varphi_{t+1} \circ f_{t+1} \circ \varphi_t^{-1}$ (see also Figures 12.5 and 12.6). Here the φ^{-1} transformations generate deficiencies, thus, creating almost edge colorings, and the φ transformations eliminate these deficiencies.

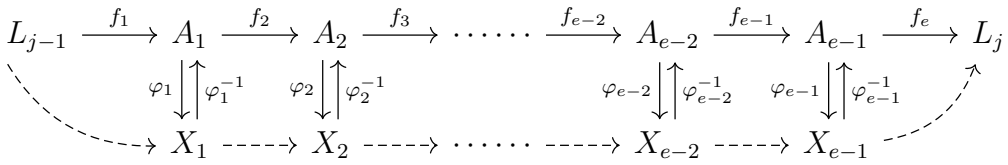


Figure 12.5: The transformation between two small milestones. See the text for details.

4. Transforming K_{l-2} to C_2 is done in the following way. Consider $H = G\big|_{K_{l-2},c_l} \bigoplus G\big|_{C_2,c_l}$. The non-trivial components of H are paths and cycles, each of which is colored alternately with colors c_{l-1} and non-c_{l-1} in K_{l-2}. For each component, we change the non-c_{l-1} colors to c_l and then flip the colors of the edges. This transforms K_{l-2} to C_2.

The precise description of the transformations from L_{j-1} to L_j is given by the following lemma and by Lemma 156.

Lemma 158. *Let L be an edge coloring of a bipartite graph G with $k \geq 3$ colors, and let N be a connected subgraph of G with maximal degree 2 and s edges. Assume that there exists a color c such that the colors of the edges in N are alternating between c and non-c. We further assume that for a path N, if any of its end edges have non-c color then N cannot be extended with a c-edge, and if any of the end edges have color c then that vertex has degree less than k. Then L can be transformed into an edge coloring L' via almost edge colorings in at most $2s$ steps such that $G\big|_{L,c} \bigoplus G\big|_{L',c} = N$ with the conditions below:*

1. *There exists a c' such that each intermediate almost edge coloring*
   (a) *either has no deficient vertex, that is, it is an edge-coloring or*
   (b) *has one deficient vertex with $(+c - c')$-deficiency or $(+c' - c)$-deficiency or*
   (c) *has two deficient vertices, one of them is $(+c - c')$ deficient, the other $(+c - c')$-deficient or $(+c' - c)$-deficient*
2. *At each step one of the following transformations is performed:*
   (a) *Changing the color of an edge from c to some c'. The edge is incident to a vertex that has a $(+c - c')$-deficiency.*

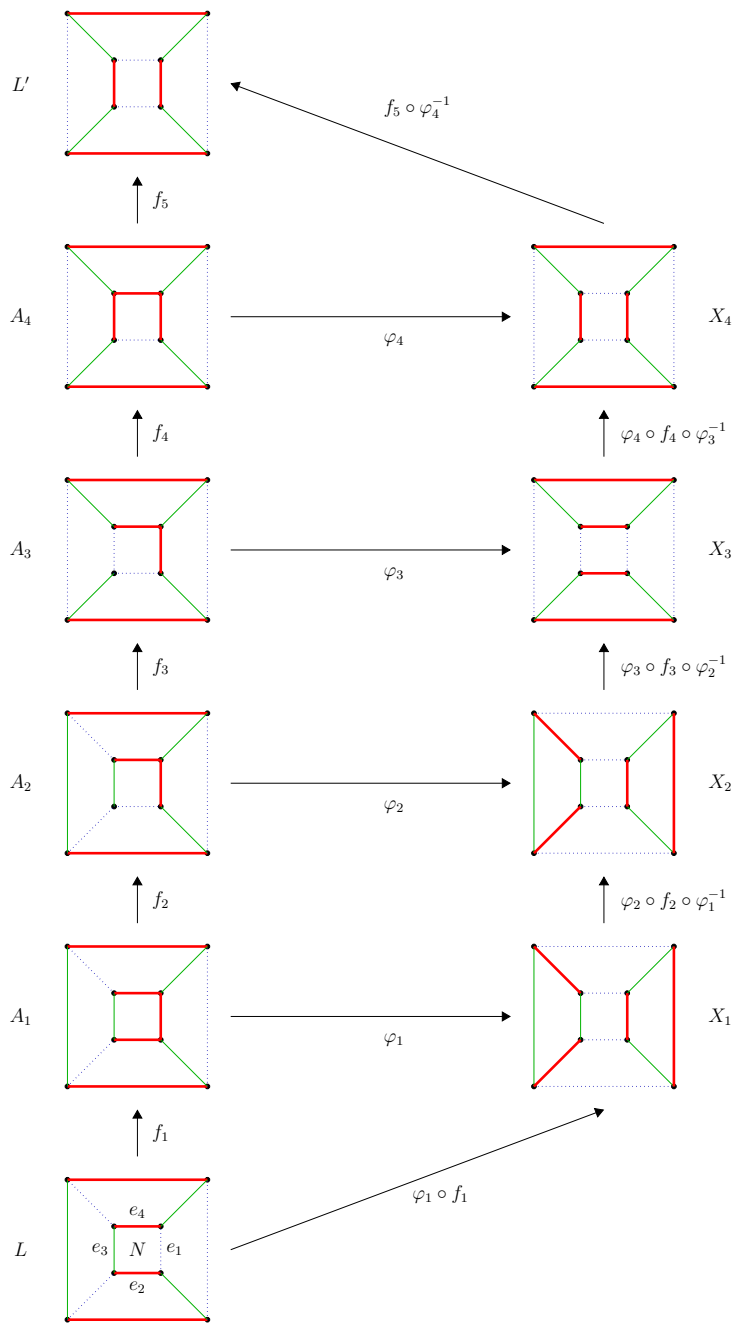


Figure 12.6: Transformation of the edge coloring L to the edge coloring L'. See the text for details.

(b) *Changing the color of an edge from some c' to c. The edge is incident to a vertex which is at most $(+c'-c)$-deficient.*

(c) *In a path or a cycle containing edges alternately colored by c' and c'' for some colors $c', c'' \neq c$, flipping the color of each edge from c' to c'' and vice versa. If there is a $(+c'-c)$-deficient vertex v, then the path or cycle contains v. If a path is changed it is maximal unless one of its end is a $(+c-c')$-deficient vertex. In that case, the path is maximal only at its non-deficient end. Its other end is either the $(+c'-c)$-deficient vertex if exists or a non-deficient vertex if a $(+c'-c)$-deficient vertex does not exist. In both cases, the path arrives to this vertex via a c''-edge.*

(d) *Colors not presented in $G\big|_{L,c} \bigoplus G\big|_{L',c}$ are never changed during the process.*

Before we prove Lemma 158, we give an example illustrated on the left hand side in Figure 12.6. In this example, an edge coloring L is transformed into an edge coloring L' via almost edge colorings in five steps. The component N is a 4-cycle of alternating red and non-red edges, labeled by e_1, e_2, e_3, and e_4. First the color of e_1 is changed from blue to red. Then the color of e_2 is changed from red to blue. Then colors of the edges in the alternating cycle of blue and green edges containing e_3 are flipped. Then the color of e_3 is changed from blue to red. Finally the color of e_4 is changed from red to blue.

Proof of Lemma 158. We have that N is either a path or a cycle. In both cases, the colors of its edges alternate between c and non-c in L. Order the vertices of N by traveling around a cycle or from one to the other end for a path. Start the walk along the cycle by a non-c-edge or from the end with a non-c-edge if such an end exists in the path. Otherwise start with a c-edge. Let the vertices along the walk be denoted by $v_1, v_2, \ldots, v_s, v_{s+1}$ with $v_{s+1} = v_1$ if N is a cycle. We prove the two cases (N is a cycle or a path) independently.

We first start with the case when N is a cycle. The first step is to flip the color of (v_1, v_2) from its non-c color c' to c. This makes each of v_1 and v_2 have a $(+c'-c)$-deficiency.

Now we are going to change the color of the edges along the walk with the operations described in the lemma and we prove inductively that the pre-

scribed properties of the almost edge colorings are maintained. The following cases are possible during the process:

1. The index i is even. We are going to change the color of the edge (v_i, v_{i+1}) from c to c'. In this case, the vertex v_1 has a $(+c - c')$-deficiency and v_i also has a $(+c - c')$-deficiency.

2. The index i is odd. We are going to change the color of the edge (v_i, v_{i+1}) from c' to c. In this case, the vertex v_1 has a $(+c - c')$-deficiency and v_i has at most a $(+c' - c)$-deficiency.

3. The index i is odd. We are going to change the color of (v_i, v_{i+1}) from c'' to some c' by flipping the colors along an alternating c', c'' path or cycle. In this case, the vertex v_1 has a $(+c - c')$-deficiency and v_i has at most $(+c' - c)$-deficiency.

In case 1, we flip the color of (v_i, v_{i+1}) from c to c'. This eliminates the deficiency of v_i. After the flip, there can be two possible cases: either v_{i+1} has a $(+c' - c)$-deficiency or v_{i+1} does not have a deficiency and also has no incident edge with color c. The former happens if v_{i+1} has an incident edge with color c' before the flip. The latter happens if v_{i+1} does not have an incident edge with color c'. That is, v_{i+1} has at most $(+c' - c)$-deficiency after the change. Index $i + 1$ is odd, so we arrive to one of the cases 2, 3, depending whether the color of the edge (v_{i+1}, v_{i+2}) is c' or c''.

In case 2, we flip the color of (v_i, v_{i+1}) from c' to c. As v_i has at most $(+c' - c)$-deficiency, this flip eliminates it, and v_i becomes a non-deficient vertex. Furthermore, v_{i+1} becomes $(+c - c')$-deficient. The index $i + 1$ is even, thus we arrive to case 1.

Case 3 happens if the color of (v_i, v_{i+1}) is some $c'' \neq c, c'$, where v_1 has a $(+c - c')$-deficiency. We have the following claims:

- An alternating c'-c'' walk starting from v_i via a c''-edge cannot reach v_1 for the graph is bipartite and v_1 does not have an incident edge with color c'.

- An alternating c'-c'' walk starting from v_i via a c'-edge can arrive to v_1 via a c''-edge. The walk must end in v_1 in this case as v_1 does not have an incident c'-edge.

- If v_1 has $(+c' - c)$-deficiency, there are two disjoint alternating c'-c'' walks starting from v_i via a c'-edge. Since they are disjoint, only at most one of them can arrive to v_1.

We do the followings. Consider the alternating c'-c'' walk starting from v_i with a c''-edge. If it is a cycle, let this cycle be denoted by X. Otherwise, the walk stops in a vertex u_1. Now if v_i has a $(+c' - c)$-deficiency, consider that alternating c'-c'' walk starting from v_i with a c'-edge that does not reach v_1. If there are two such walks, choose any of them. Let the end vertex of this walk be u_2. If v_i has no deficiency, consider an alternating c'-c'' walk starting from v_i via its c'-edge. (Observe that such edge exists, it is (v_{i-1}, v_i).) If it ends not in v_1, then let its end vertex be u_2, otherwise let u_2 be v_i. Let the alternating c'-c'' path from u_1 to u_2 be denoted by X. Now, whatever is X (path or cycle), we flip the color of the edges along X. It changes the color of (v_i, v_{i+1}) from c'' to c', and makes v_i at most $(c' - c)$-deficient. Thus, we arrive to case 2.

In the last step, we change the color of the edge (v_s, v_1) from c to c'. This eliminates the deficiency of both v_1 and v_s.

The transformation is simpler if N is a path since there is at most one deficient vertex during the process. The first step is to flip the color of (v_1, v_2) from c' to c or from c to some c' such that v_1 does not have an incident edge with color c' before the flip. Such color c' must exist due to the assumptions in the lemma. Also due to the assumptions, the vertex v_1 will not have a deficiency after the flip. The vertex v_2 becomes either $(+c - c')$-deficient or at most $(+c' - c)$-deficient. Then the following cases are possible during the process:

1. We are going to change the color of (v_i, v_{i+1}) from c to c'. In this case, the vertex v_i has a $(+c - c')$-deficiency.

2. We are going to change the color of (v_i, v_{i+1}) from c' to c. In this case, the vertex v_i has at most $(+c' - c)$-deficiency.

3. We are going to change the color of (v_i, v_{i+1}) from some c'' to c'. In this case, v_i has at most $(+c' - c)$-deficiency.

In case 1, flipping the color of (v_i, v_{i+1}) removes the deficiency of v_i and makes v_{i+1} at most $(c' - c)$-deficient, thus we arrive to case 2 or 3 depending on the color of (v_{i+1}, v_{i+2}) or v_{i+1} is the last vertex that we will handle separately.

In case 2, flipping the color of $(v_i, v_i + 1)$ makes v_i no longer deficient, and makes v_{i+1} $(+c - c')$-deficient, thus we arrive to case 1 or $v_{i+1} = v_{s+1}$, and then v_{i+1} is not deficient.

In case 3, we consider a maximal walk of alternating colors c' and c'' containing the edge (v_i, v_{i+1}) and containing only one edge incident to v_i with color c'. Since v_i might be the only deficient vertex (or it is at most $(c' - c)$-deficient), we can conclude that this walk is either a cycle or a path. We flip the colors along this path or cycle. It does not change the deficiency of v_i, however, it makes the color of (v_i, v_{i+1}) be c'. Now we arrived to case 2.

If the last edge (v_s, v_{s+1}) does not have color c, then it has color c' or applying the case 3 its color becomes c'. Then the last step is to change this color from c' to c. It makes v_s not deficient, and does not create a new deficiency for the path is maximal.

If the last edge (v_s, v_{s+1}) has color c, then after flipping its color from c to c', v_{s+1} might be $(+c' - c)$-deficient. However, since the degree of v_{s+1} is less than k (recall it is a condition of the lemma), v_{s+1} is also $(+c - c'')$-deficient for some $c'' \neq c$. Consider the longest alternating c'-c'' walk staring from v_{s+1} via its c'-edge. We flip the color of the edges along this path. It makes v_{s+1} be a non-deficient vertex.

Every second edge, unless it is the last one, can be transformed in one step, and every other edge can be transformed in at most two steps, furthermore, the first edge can be transformed in one step, and the last in at most two steps, hence the number of steps is indeed at most $2s$ steps. □

We are ready to state and prove the theorem on irreducibility and small diameter.

Theorem 159. *Let C_1 and C_2 be two edge k-colorings of the same bipartite graph $G = (V, E)$. Then C_1 can be transformed into C_2 in at most $2k|E|$ steps in the Markov chain $M(G, k)$.*

Proof. First we show that these transformations are sufficient, then we prove the upper bound on the number of necessary transformations.

Observe that from K_0 to K_{l-2}, any large milestone is also a small milestone. Therefore, it is enough to show that the listed perturbations are sufficient to transform one small milestone to the next small milestone and they are sufficient to transform K_{l-2} to K_{l-1}. Observe that the listed perturbations contain the case when the colors are flipped along a maximal alternating path or cycle. Indeed, in case (2)(b) in Definition 157, a maximal

subpath or a cycle can also be selected. These transformations are sufficient to transform K_{l-2} to K_{l-1}.

Any step between two small milestones is a transformation of the form $\varphi_{t+1} \circ f_{t+1} \circ \varphi_t^{-1}$ or $\varphi_1 \circ f_1$ or $f_e \circ \varphi_{e-1}^{-1}$, where φ is a transformation given in Lemma 156 and f is a transformation given in Lemma 158 (see also Figures 12.5 and 12.6.). A transformation φ corrects the deficiency of at most two vertices by flipping the edges along at most two maximal alternating paths. Therefore, its inverse creates at most two deficiencies along at most two alternating paths, and these deficiencies appear at the end of the path that could be extended. With some probability, no deficiency is created, thus providing the transformations of type $\varphi_1 \circ f_1$.

A transformation f achieves the followings:

(i) It changes the color of an edge which is adjacent to an edge with the same color and there is another deficient vertex. This happens when the component N described in the proof of Lemma 158 is a cycle, and the color of an edge is changed from c to c' or from c' to c, and the previous edge in the component N has also color c or c', or

(ii) It changes the color c' to c of an edge incident to an at most $(+c'-c)$-deficient, but otherwise not deficient vertex. Furthermore, there is a $(+c-c')$-deficient vertex in the almost edge coloring. This happens when the component N described in the proof of Lemma 158 is a cycle, or

(iii) It changes the color of an arbitrary edge if there is no deficient vertex. This happens in the first step of transforming the component N as described in the proof of Lemma 158 and when N is a path, and the edge whose color is changed is incident to an at most $(+c'-c)$-deficient but otherwise not deficient vertex, or

(iv) It flips the colors along an alternating c'-c'' path or cycle such that there is also a $(+c-c')$-deficient vertex in the almost edge coloring. This happens when the current edge e (as described in the proof of Lemma 158 has color c'' and its component N is a cycle, or

(v) It flips the colors along an alternating path or cycle and there is no additional $(+c-c')$-deficient vertex. This happens when the current edge e (as described in the proof of Lemma 158 has color c'', and its component N is a path.

These are exactly the five subcases given in Definition 157.

Finally, the transformation φ_{t+1} eliminates all the deficiencies, if they exist.

For each color, the number of edges in the components is at most $|E|$. That is, for each color, the number of necessary steps is at most $2|E|$. Altogether, $2k|E|$ is an upper bound on the number of steps necessary for transforming any edge coloring to any other one.

□

As an example, we show how to generate the transformations in Figure 12.6 by the transformations in Definition 157.

1. $\varphi_1 \circ f_1$: We select two colors, red and blue, then we do nothing (case (2)(a)). Since there is no deficient vertex, we can change the color of edge e_1 from blue to red. Then we eliminate all deficiencies by flipping the colors along a maximal alternating path between the two deficient vertices emerged. We can conclude that $P(\varphi_1 \circ f_1) = \frac{1}{4} \times \frac{1}{3} \times \frac{1}{3} \times \frac{1}{2} \times \frac{1}{8} \times \frac{1}{2}$. Indeed, $\frac{1}{4}$ is the probability of case (2), there are $\binom{3}{2} = 3$ possible pair of colors, $\frac{1}{3}$ is the probability of subcase (a) in case (2), $\frac{1}{2}$ is the probability of changing the color of an edge when there is no deficient vertex (referenced as *iii)*), and there are 8 edges with color blue or red, and finally, we select the path with which A_1 and X_1 differ with probability $\frac{1}{2}$.

2. $\varphi_2 \circ f_2 \circ \varphi_1^{-1}$: We select two colors, red and blue, then we select the subpath with which X_1 and A_1 differ (case (2)(b)). We flip the colors along this path. There are two deficient vertices incident to the edge e_1, and we select the vertex incident to both e_1 and e_2, and then we select the edge e_2 and flip its color from red to blue. Finally, we eliminate all deficiencies by flipping the colors along a maximal alternating path. We can conclude that $P(\varphi_2 \circ f_2 \circ \varphi_1^{-1}) = \frac{1}{4} \times \frac{1}{3} \times \frac{1}{3} \times \frac{1}{57} \times \frac{1}{4} \times \frac{1}{2}$. Indeed, $\frac{1}{4}$ is the probability of case (2), there are $\binom{3}{2} = 3$ possible pairs of colors, $\frac{1}{3}$ is the probability of subcase (b) in case (2), there are $\binom{8}{2} \times 2$ subpaths in the subgraph with colors blue and red and 1 cycle, thus we have to select one of the 57 possibilities, selecting the edge e_2 has probability $\frac{1}{4}$ (selecting uniformly one of the deficient vertices and uniformly one of the edges with repeated color), and finally, we select the path with which A_2 and X_2 differ with probability $\frac{1}{2}$.

3. $\varphi_3 \circ f_3 \circ \varphi_2^{-1}$: We select three colors, red, blue and green, and we select green as the distinguished color c''. Then we select the path with which X_2 and A_2 differ, and swap the red and blue colors. We select the alternating blue-green cycle containing the edge e_3 and flip the green and blue colors in it. Then we eliminate the deficiencies by flipping the colors along the path with which A_3 and X_3 differ. We can conclude that $P(\varphi_3 \circ f_2 \circ \varphi_2^{-1}) = \frac{1}{4} \times \frac{1}{3} \times \frac{1}{3} \times \frac{1}{57} \times \frac{1}{4} \times \frac{1}{2}$. Indeed, the probability of case (3) is $\frac{1}{4}$, there is one way to select three colors, and there are three ways to select one of them distinguished, $\frac{1}{3}$ is the probability of subcase (b) in case (3), there are $\binom{8}{2} \times 2$ subpaths in the subgraph with colors blue and red and 1 cycle, thus we have to select one of the 57 possibilities, selecting the edge e_3 has probability $\frac{1}{4}$ (selecting uniformly one of the deficient vertices and uniformly one of the edges with repeated color), and finally, we select the path with which A_3 and X_3 differ with probability $\frac{1}{2}$.

4. $\varphi_4 \circ f_4 \circ \varphi_3^{-1}$: We select two colors, red and blue. Then we select the path with which X_3 and A_3 differ and flip the colors of the edges in it. Then we select e_3 and flip its color from blue to red. Finally, we eliminate all deficiencies by flipping the colors along a maximal alternating path. We can conclude that $P(\varphi_4 \circ f_4 \circ \varphi_3^{-1}) = \frac{1}{4} \times \frac{1}{3} \times \frac{1}{3} \times \frac{1}{26} \times \frac{1}{4} \times \frac{1}{2}$. Indeed, $\frac{1}{4}$ is the probability of case (2), there are $\binom{3}{2} = 3$ possible pair of colors, $\frac{1}{3}$ is the probability of subcase (b) in case (2), there are $\binom{4}{2} \times 2 \times 2 = 24$ subpaths in the subgraph with color blue and red and 2 cycles, thus we have to select one of the 26 possibilities, selecting the edge e_3 has probability $\frac{1}{4}$ (selecting uniformly one of the deficient vertices and uniformly one of the edges with repeated color), and finally, we select the path with which A_4 and X_4 differ with probability $\frac{1}{2}$.

5. $f_5 \circ \varphi_4^{-1}$: We select two colors, red and blue. Then we select the path with which X_4 and A_4 differ and flip the colors of the edges in it. Then we select e_4 and flip its color from red to blue. We can conclude that $P(f_5 \circ \varphi_4^{-1}) = \frac{1}{4} \times \frac{1}{3} \times \frac{1}{3} \times \frac{1}{26} \times \frac{1}{2}$. Indeed, $\frac{1}{4}$ is the probability of case (2), there are $\binom{3}{2} = 3$ possible pair of colors, $\frac{1}{3}$ is the probability of subcase (b) in case (2), there are $\binom{4}{2} \times 2 \times 2 = 24$ subpaths in the subgraph with color blue and red and 2 cycles, thus we have to select one of the 26 possibilities, selecting the edge e_4 has probability $\frac{1}{2}$ (selecting uniformly one of the deficient vertices and uniformly one of the edges

with repeated color has probability $\frac{1}{4}$, however, the edge e_4 can be selected in two different ways).

We can put $M(G,k)$ into a Metropolis-Hastings algorithm to design a Markov chain Monte Carlo converging to the uniform distribution of edge k-colorings of a bipartite graph. We can use the variant introduced by Lunter *et. al* [113], that is, when an edge coloring C_2 is proposed from C_1 in a way w, we obtain the inverse way w' transforming C_2 back to C_1, and use the probabilities $P(C_1, w'|C_2)$ and $P(C_2, w, |C_1)$ in the Metropolis-Hastings ratio. Here a way consists of a combination of operations φ_1^{-1}, f and φ_2 with the following meaning. The operation φ_1^{-1} perturbs the the coloring C_1 into an almost edge coloring X_1 as described in (2)(b-c) and (3)(b-c) in the Definition 157. Then comes the operation f which is either flipping an edge from c to c' or flipping the edges along a c'-c'' path or cycle or doing nothing. Thus, operation f transforms the almost edge coloring X_1 into another almost edge coloring X_2. Finally, the φ_2 operation corrects all deficiencies to get a new coloring C_2. The inverse way of $\varphi_2 \circ f \circ \varphi_1^{-1}$ is simply $\varphi_1 \circ f^{-1} \circ \varphi_2^{-1}$, which transforms C_2 first back to X_2, then to X_1 then to C_1. See also Figure 12.6. Furthermore, since the target distribution π is the uniform one, g can be set to any constant function, and is cancelled in the Metropolis-Hastings ratio, which simply becomes

$$\frac{P(C_1, w'|C_2)}{P(C_2, w|C_1)},$$

where the edge k-coloring C_2 is proposed from the edge k-coloring C_1 via the way w. Observe that the smallest acceptance probability is the minimum of the Metropolis-Hastings ratio, so the maximum of the inverse of the acceptance ratio is the maximum of the inverse of the Metropolis-Hastings ratio, and due to symmetry, it is simply the maximum of the Metropolis-Hastings ratio. Below we give upper bound for this maximum when $M(G,k)$ is used in the Metropolis-Hastings algorithm.

Since for all C_1 and C_2 and possible transformation ways w between them, $P(C_1, w'|C_2) \leq 1$, the maximum of the Metropolis-Hastings ratio is at most

$$\frac{1}{P(C_2, w|C_1)}.$$

This already would give a polynomial upper bound for the inverse of the acceptance ratio, however, with a more careful analysis, a better upper bound could be found.

If nothing is done in the Markov chain, then the Metropolis-Hastings ratio is 1. Otherwise, a step in the Markov chain does the followings:

1. It selects either 2 or 3 colors. Then based on the selected colors, it creates at most two deficient vertices. We denote this transformation by φ_1^{-1}.

2. It perturbs the current configuration. We call this perturbation f.

3. It eliminates all the deficiencies. We denote this transformation by φ_2.

Observe that in the reverse transformation, the same colors must be selected. Therefore, the probability of selecting the particular colors cancels in the Metropolis-Hastings ratio. Let $\tilde{P}(\varphi^{-1})$ denote the probability of a perturbation φ^{-1} omitting the probabilities of selecting a particular set of colors. The Metropolis-Hastings ratio simplifies to

$$\frac{\tilde{P}(\varphi_2^{-1})P(f^{-1})P(\varphi_1)}{\tilde{P}(\varphi_1^{-1})P(f)P(\varphi_2)}$$

We give individual upper bounds on $\frac{\tilde{P}(\varphi_2^{-1})}{P(\varphi_2)}$, $\frac{P(f^{-1})}{P(f)}$ and $\frac{P(\varphi_1)}{\tilde{P}(\varphi_1^{-1})}$, and their product is an upper bound on the Metropolis-Hastings ratio.

We claim that

$$\frac{\tilde{P}(\varphi_2^{-1})}{P(\varphi_2)} \leq \frac{1}{P(\varphi_2)} \leq 4.$$

Indeed, any probability is at most 1, thus the first inequality holds. The transformation φ_2 eliminates at most two deficiencies. It must select uniformly one of the edges with the same color incident to a deficient vertex, and it has probability $\frac{1}{2}$. Once the edge is selected, the maximal alternating path with the two prescribed colors is defined unequivocally, and thus with probability 1. Since an edge incident to a deficient vertex must be selected during the procedure φ_2 at most twice, the probability of φ_2 has a minimum $\frac{1}{4}$, and thus the second inequality.

Now we show that

$$\frac{P(\varphi_1)}{\tilde{P}(\varphi_1^{-1})} \leq \frac{1}{\tilde{P}(\varphi_1^{-1})} \leq 12\binom{|V|}{2}.$$

The reasoning for the first inequality is the same as above. The transformation φ_1^{-1} creates at most two deficient vertices by selecting at most two

alternating subpaths or a cycle in the subgraph H defined by the two colors and flipping the colors along these paths. When nothing is selected (subcases (2)(a) and (3)(a)), the inequality holds. If one path or a cycle is selected, then this option is chosen with probability $\frac{1}{3}$. After this, a subpath is selected uniformly. A subpath in a path is defined by its end vertices, and if it comes from a cycle, then also a direction must be selected. Therefore, there cannot be more than $2\binom{|V|}{2}$ subpaths coming from paths. Furthermore, the number of cycles is at most $|V| \leq \binom{|V|}{2}$. Thus, the probability of selecting one subpath or cycle is at least $\frac{1}{3\binom{|V|}{2}}$. If two subpaths are selected, then this option is chosen with $\frac{1}{3}$ probability. In this case, both of the subpaths must have an end vertex which is an end vertex also in H. Thus only the other end points must be selected, and there are 2 options for this. Therefore, the number of pairs of subpaths with the required properties is at most $4\binom{|V|}{2}$, so the probability of selecting one of them is at least $\frac{1}{4\binom{|V|}{2}}$. Therefore, it indeed holds that

$$\frac{1}{\tilde{P}(\varphi_1^{-1})} \leq \frac{1}{\frac{1}{3} \times \min\left\{\frac{1}{3\binom{|V|}{2}}, \frac{1}{4\binom{|V|}{2}}\right\}} = 12\binom{|V|}{2}.$$

Finally, we show that

$$\frac{P(f^{-1})}{P(f)} \leq \frac{1}{P(f)} \leq 4|V|.$$

The reasoning for the first inequality is the same as above. The transformation f does one of the followings:

1. When two colors are selected and there is a deficient vertex, it selects a deficient vertex, then it selects one of its edges causing deficiency, then it flips its color. It has probability at least $\frac{1}{4}$ ($\frac{1}{2}$ for selecting uniformly from two deficient vertices and $\frac{1}{2}$ for selecting uniformly from two edges).

2. When two colors are selected and there is no deficient vertex, then with probability $\frac{1}{2}$, it selects uniformly an edge with one of the two prescribed colors, and it flips its color. Observe that the number of edges in the subgraph induced by any two colors is at most the number of vertices in an edge coloring. Therefore, the probability of this transformation is at least $\frac{1}{2|V|}$.

3. When two colors are selected and there is no deficient vertices, it does nothing with probability $\frac{1}{2}$.

4. When three colors are selected then either a deficient vertex is selected with at least probability $\frac{1}{2}$ or a non-deficient vertex is selected with at least probability $\frac{1}{2|V|}$. If a deficient vertex is selected, then one of the edges causing deficiency is selected with probability $\frac{1}{2}$, then colors are flipped along an unequivocally defined alternating path (recall that a path P might be excluded as described in the Definition 157). If a non-deficient vertex is selected, then the colors are flipped along one of at most two paths. The path on which the colors are flipped is selected with at least probability $\frac{1}{2}$.

Therefore, $P(f)$ is indeed bounded from below by $\frac{1}{4|V|}$, thus its inverse is bounded from above by $4|V|$.

As discussed, an upper bound on the Metropolis-Hasting ratio is also an upper bound on the inverse of the Metropolis-Hastings ratio. Therefore, we have just proved the following theorem.

Theorem 160. *Let $M(G,k)$ be the Markov chain on the edge k-colorings on the bipartite graph $G=(V,E)$. Then the inverse of the acceptance ratio in the Metropolis-Hastings algorithm applied with the constant function g and with the Markov chain $M(G,k)$ is bounded from above by $96|V|^2(|V|-1)$.*

12.5 Latin rectangles

Definition 161. *A* Latin rectangle *is a $k\times n$ table such that each row is a permutation of $[n]$ and each column has no repeats. When $k=n$ we call it a* Latin square. *The* completion *of a $k\times n$ Latin rectangle R is an $n\times n$ Latin square such that its first k rows are R.*

Observation 162. *For any $(n-k)\times n$ Latin rectangle R, there is an k-regular bipartite graph G with n vertices in both vertex classes such that completions of R and edge k-colorings of G are in one-to-one correspondence.*

Proof. Let the two parts of vertices of G be $A_1, A_2, \ldots, A_n$ and $B_1, B_2, \ldots, B_n$ and define $A_iB_j \in E$ if and only if number i does not appear in the j-th

column of R. This gives us a desired k-regular graph for any given Latin rectangle R.

Indeed, it is clear that each B_i is of degree k. For A_i's, observe that they all have degree at least k. Otherwise, suppose that number i_0 is missed in less than k columns (i.e. it appears in at least $n-k+1$ columns). We know that there are only $n-k$ rows, so by the Pigeonhole principle there must be two i_0's in the same row, which contradicts our assumption that R is a Latin rectangle. Because $|E| = nk$ and the total degree of A_i's is nk, no A_i can have degree more than k, thus every vertex of G is of degree k and G is k-regular.

Now we give the bijection: if and only if number i is filled in the $(n-k+\ell, j)$-grid in R's completion, we color the edge A_iB_j using color c_ℓ in G. For one direction, the above discussion for G already shows the pre-image is a completion of R. Indeed, due to the rules of edge colorings, each number appears in each row exactly once. Furthermore, each column contains the necessary missing numbers for the completion. On the other hand, first of all, every B_j is adjacent to one c_ℓ-colored edge for all ℓ. Because each number appears k times in the last k rows of the table by the k-regularity of G each A_i also has exactly one edge for every color. Thus the image is an edge k-coloring. □

Remark 2. *These objects are also in one-to-one correspondence with half-regular factorizations of complete $(k+n)$-bipartite graphs with edge n-colorings, connecting our work to [102].*

We now give a simplified version of the Markov chain described in Definition 157 on the solution space of edge k-colorings of k-regular bipartite graphs, that is, Latin square completions of $(n-k) \times n$ Latin rectangles. In that, we utilize the following observation.

Observation 163. *Let C be an almost edge k-coloring of a k-regular bipartite graph with at least one deficient vertex. Then there are exactly two deficient vertices in C, furthermore, they are the endpoints of a maximal alternating path with two colors.*

Proof. Let the underlying bipartite graph be $G = (U, V, E)$. First we observe that due to the k-regularity and bipartite property, if there is a deficient vertex, then there must be at least two deficient vertices otherwise it would be impossible to have the same number of edges with each color in the two

vertex classes. By definition, there cannot be more than two deficient vertices in C. Furthermore, if one of the deficient vertices is $v \in V$ and has a $(+c - c')$-deficiency then either the other deficient vertex is $u \in U$ with a $(+c - c')$-deficiency or $v' \in V$ with a $(+c' - c)$-deficiency.

Let e be one of the edges with color c incident to v. Consider a alternating c-c' walk leaving v via the edge e. It cannot return to v as it could do it only via a c'-edge due to the bipartite property. It can only return to a deficient vertex. However, it can reach v' only via its c'-edge, and then the walk stops there, or it can reach u only via a c-edge and then the walk stops there. That is, this alternating walk is a path connecting the two deficient vertices. □

A consequence is that any φ operations always eliminates two deficient vertices by flipping the colors along an alternating path. This allows us to simplify the Markov chain in Definition 157 to get an irreducible Markov chain on the edge k-colorings of k-regular bipartite graphs. We also need the following observation.

Observation 164. *Let C be an almost edge k-coloring of a k-regular bipartite graph $G = (U, V, E)$ such that v_1 has a $(+c - c')$-deficiency and v_2 has a $(+c' - c)$-deficiency. Furthermore, assume that v_1 and v_2 are both in V. Let $c'' \notin \{c, c'\}$ be a third color. Then the longest alternating walk with colors c' and c'' that starts with color c'' at v_2 is a cycle.*

Proof. Since the graph is k-regular, each non-deficient vertex has exactly one c'-edge and one c''-edge. That is, the walk continues until it arrives at a deficient vertex or it travels back to v_2. Since v_1 and v_2 are in the same bipartite vertex class, the alternating walk could arrive at v_1 with a c'-edge, however, v_1 does not have an incident c'-edge. Therefore, the walk arrives back at v_2. □

Observations 163 and 164 help simplify the Markov chain. Indeed, regularity provides that

1. each component now is a cycle,

2. any almost edge coloring between two milestones have exactly two deficiencies

3. These two deficiencies can be corrected by a single alternating path

4. when the color of an edge must be changed from c'' to c', it always can be done by flipping the colors along an alternating c'-c'' cycle.

Definition 165. *Let G be a k-regular equi-bipartite graph. We define a Markov chain $M(G,k)$ on the edge k-colorings of G. Let the current coloring be C. Then draw a random edge k-coloring in the following way.*

1. *With probability $\frac{1}{2}$, we do nothing (so the defined Markov chain is a Lazy Markov chain).*

2. *With probability $\frac{1}{4}$, we select two colors c and c' from the set of colors uniformly among the $\binom{k}{2}$ possible unordered pairs. Consider the subgraph of G with colors c and c', let it be denoted by H. Note that H consists of disjoint alternating cycles. Then we select one of them:*

   (a) *With probability $\frac{1}{2}$, we do nothing.*

   (b) *With probability $\frac{1}{2}$, we uniformly select one of the possible connected subgraphs of H which is a subpath or a cycle.*

   We flip the colors in the selected subpath, or the cycle if one was selected. If there is any deficient vertex, then we select one of them uniformly, we select the edge with the same color incident to the selected vertex uniformly, and change its color from c to c' or c' to c. Otherwise, with probability $\frac{1}{2}$, we select either a c-edge or a c'-edge, uniformly from all edges with color c and c', and change its color to the opposite (c' or c), and with probability $\frac{1}{2}$, we do nothing. Eliminate all deficiencies by flipping the colors on the appropriate alternating path, selecting uniformly from the two neighbor edges with the same color incident to deficient vertices.

3. *With probability $\frac{1}{4}$ we select three colors uniformly from all possible $\binom{k}{3}$ unordered pairs, and select uniformly one from the three colors. Let the distinguished color be denoted by c'', and let the other two colors be denoted by c and c'. Consider the subgraph consisting of the colors c and c' and let it be denoted by H. Its non-trivial components are again cycles. Then we select one of the followings:*

   (a) *With probability $\frac{1}{2}$, we do nothing.*

   (b) *With probability $\frac{1}{2}$, we uniformly select a subpath of H with even length (even number of edges).*

We flip the colors in the selected subpath. The number of deficiencies must be even. If there are two deficient vertices, then we select one of them uniformly. If there is no deficient vertex, then we select uniformly a non-deficient vertex.

If a deficient vertex v is selected, let its repeated color be denoted by $\tilde{c}$. Consider the maximal alternating cycle with colors $\tilde{c}$ and c'' that contains v (from Observation 164 we know that this cycle exists). We flip the colors in this cycle.

If a non-deficient vertex is selected, let its incident edge with color c'' be denoted by e. We select uniformly from the incident edges with color c or c', and let it be denoted by f and its color be $\tilde{c}$. Take the maximal alternating cycle with colors c'' and $\tilde{c}$ that contains e and f. We flip the colors in this cycle.

We eliminate all deficiencies by flipping the colors on the appropriate alternating path, selecting uniformly from the two neighbor edges with the same color incident to deficient vertices.

We can state the following upper bounds on the diameter and the inverse of the acceptance ratio.

Theorem 166. *Let $M(G,k)$ be the Markov chain on the edge k-colorings of a k-regular bipartite graph $G=(V,E)$. Then the diameter of $M(G,k)$ is bounded from above by $3|E|$ and the inverse of the acceptance ratio in the Metropolis-Hastings algorithm with the uniform distribution is bounded from above by $48|V|(|V|-1)$.*

Proof. In each subgraph H_i and in each non-trivial component, half of the edges have color c_i in C_2. Indeed, observe that each non-trivial component is an alternating cycle with edges having color c_i in C_1 and C_2. Therefore, the total number of edges of the non-trivial components in all H_i's is at most $2|E|$. Each component is an even cycle of length s (and thus s is even), and can be transformed in at most $\frac{3}{2}s$ steps. See also the proof of Lemma 158. Therefore, $3|E|$ steps are sufficient to transform any edge coloring to any other one.

We can decompose the proposal and reverse proposal probabilities in the same way as we did in case of Theorem 160. Then the probabilities of selecting colors again cancel. We still have the bound

$$\frac{\tilde{P}(\varphi_2^{-1})}{P(\varphi_2)} \leq 4.$$

However, we can give a better upper bound on

$$\frac{P(\varphi_1)P(f^{-1})}{\tilde{P}(\varphi_1^{-1})P(f)} \leq \frac{1}{\tilde{P}(\varphi_1^{-1})P(f)} \leq 24\binom{|V|}{2}.$$

If nothing is selected (subcases (2)(a) and (3)(a)), the inequality holds. Since G is k-regular and its edges are colored with k colors, there are deficient vertices after the transformation φ_1^{-1} if and only if an alternating path is selected and the edges are flipped. The probability for that is at least $\frac{1}{2\times 3\binom{|V|}{2}}$. Indeed, $\frac{1}{2}$ is the probability for choosing the option to select a path or cycle, and there are at most $2\binom{|V|}{2}$ possible paths, since the two end vertices and a direction in the cycle define the alternating path unequivocally (recall that the two colors in the alternating path are fixed) and there are at most $|V| \leq \binom{|V|}{2}$ cycles. In such a case, there are two deficient vertices, one of them is selected uniformly and one of its edges with duplicated colors is selected uniformly, and the color of this edge is changed in a prescribed way. Note that this change is the transformation f. Therefore, in this case, $P(f) = \frac{1}{4}$. We get that

$$P(\varphi_1^{-1})P(f) \geq \frac{1}{24\binom{|V|}{2}} \tag{12.10}$$

If no path is selected, then the probability for that is $\frac{1}{2}$. Then a non-deficient vertex is selected uniformly, its probability is $\frac{1}{|V|}$. Then one of the edges incident to the selected vertex and having one of the two prescribed colors is selected uniformly. This has probability $\frac{1}{2}$. The color of the selected edge is changed in a prescribed way. Therefore, we get that

$$P(\varphi_1^{-1})P(f) \geq \frac{1}{4|V|} \tag{12.11}$$

Since $|V| \geq 2$, we have $4|V| \leq 24\binom{|V|}{2}$.

Putting the inequalities together, we obtain the claimed bound. □

12.6 Conclusion

We consider the solution space of edge colorings of any general bipartite graph and explicitly constructed an irreducible Markov chain $M(G, k)$ on the edge k-colorings of a bipartite graph G. We have shown that the diameter of

M is linearly bounded from above by the number of edges and also by the number of colors, and when we apply the Metropolis-Hastings algorithm to it so that the modified chain, $\tilde{M}(G,k)$, converges to the uniform distribution, the inverse of acceptance ratio is bounded from above by a cubic function of the number of vertices. A special case of our work provides a Markov chain Monte Carlo method to sample completions of Latin squares.

Possible further work includes investigating the speed of convergence of the Markov chain $\tilde{M}(G,k)$, that is, whether it is rapidly mixing or not. The natural conjecture is that $\tilde{M}(G,k)$ is rapidly mixing, based on the proved properties on the diameter and the bound on the inverse of the acceptance ratio. The Markov chain $\tilde{M}(G,k)$ is similar to the Markov chains invented by Jacobson and Matthew [51] and Aksen *et al.* [102] (see section 12.1). Neither of these chains have been proved to be rapidly mixing, although rapid mixing is conjectured. The fact that the rapid mixing is a more than 25 year old open question in case of the Jacobson-Matthew Markov chain on Latin squares indicates that resolving these open questions might be extremely hard. A special case of the Markov chain introduced in this paper is the Markov chain on completions $(n-3) \times n$ Latin rectangles, or equivalently, on edge 3-colorings of 3-regular bipartite graphs. The structure of the solution space of edge 3-colorings of 3-regular bipartite graphs might be simpler than the structure of the Latin squares. Therefore, there is a hope that proving rapid mixing of the Markov chain on edge 3-colorings of 3-regular bipartite graphs might be easier.

Another possible direction is to extend the results of this section to bipartite multigraphs.

Bibliography

[1] Aho, A., Ullman, J., Hopcroft, J. (1974) The Design and Analysis of Computer Algorithms, Addison Wesley, Reading, Massachusetts.

[2] Agrawal, M., Kayal, N., Saxena, N. (2004) PRIMES is in P. Annals of Mathematics, 160(2):781–793.

[3] André, D. (1881) Mémoire sur les permutations alternées. Journal de mathématiques pures et appliquées, 7:167—184.

[4] Ajana, Y., Lefebvre, J.-F., Tillier, E.R.M., El-Mabrouk, N. (2002) Exploring the Set of All Minimal Sequences of Reversals - An Application to Test the Replication-Directed Reversal Hypothesis. *Lecture Notes in Computer Science*, vol. 2452, pp. 300–315.

[5] Aldous, D.J. (1982) Some inequalities for reversible Markov chains. *Journal of the London Mathematical Society*, vol. 2 num. 25, pp. 564–576.

[6] Babai, L. (2015) Graph Isomorphism in Quasipolynomial Time `arXiv:1512.03547`.

[7] Babai, L., Luks, E.M. (1983) Canonical labeling of graphs. In:Proceedings of the 15th Annual ACM Symposium on Theory of Computing, 171–183.

[8] Bader, D.A., Moret, B.M.E., Yan, M. (2001) A linear-time algorithm for computing inversion distance between signed permutations with an experimental study. *J. Comp. Biol.*, vol. 8, num. 5, pp 483–491.

[9] Baum, L.E., Egon, J.A. (1967) An inequality with applications to statistical estimation for probabilistic functions of a Markov process and to a model for ecology. Bulletin of the American Mathematical Society , 73:360–363.

[10] Baum, L.E., Petrie, T. (1966) Statistical Inference for Probabilistic Functions of Finite State Markov Chains. The Annals of Mathematical Statistics, 37(6):1554–1563.

[11] Baum, L.E., Sell, G.R. (1968) Growth functions for transformations on manifolds. Pacific Journal of Mathematics, 27(2):211–227.

[12] Bergeron, A. (2001) A very elementary presentation of the Hannenhalli-Pevzner theory. *Proceedings of CPM2001*, 106–117.

[13] Bergeron, A., Mixtacki, J., Stoye, J. (2006) A Unifying View of Genome Rearrangements. Lecture Notes in Computer Science, 4175:163–173.

[14] Braga, M.D.V., Stoye, J. (2010) The Solution Space of Sorting by DCJ. Journal of Computational Biology, 17(9):1145–1165.

[15] Brémoud, P. (1999) Markov Chains: Gibbs Fields, Monte Carlo Simulation, and Queues, Texts in Applied Mathematics, Springer, New York.

[16] Brightwell, G., Winkler, P. (1991) Counting linear extensions is #P-complete. In: Proceedings of the 23rd Annual ACM Symposium on Theory of Computing, 175–181.

[17] Caprara A. (2003) The reversal median problem. INFORMS Journal on Computing, 15:93–113.

[18] Chomsky, N. (1955) Transformational Analysis, PhD thesis, University of Pennsylvania

[19] Chomsky, N. (1959) On Certain Formal Properties of Grammars. Information and Control, 2:137–167.

[20] `http://www.claymath.org/sites/default/files/pvsnp.pdf`

[21] Cook, S. (1971) The Complexity of Theorem Proving Procedures. Proceedings of the 3rd Annual ACM Symposium on Theory of Computing, 151–158.

[22] Cooper, C., Dyer, M., Greenhill, C. (2007) Sampling regular graphs and a peer-to-peer network. Combinatorics, Probability and Computing, 16(4):557–593.

[23] Cormen, T.H., Leiserson, C.E., Rivest, R.L., Stein, C. (2009) Introduction to Algorithms, The MIT Press, Cambridge, Massachusetts.

[24] Darwin, C. (1859) On the Origin of Species by Means of Natural Selection, or the Preservation of Favoured Races in the Struggle for Life. John Murray, London.

[25] Diaconis, P., Saloff-Coste, L. (1993) Comparison theorems for reversible Markov Chains. The Annals of Applied Probability, 3(2):696–730.

[26] Diaconis, P., Stroock, D. (1991) Geometric bounds for eigenvalues of Markov chains. The Annals of Applied Probability, 1(1):36–61.

[27] Durbin, R., Eddy, S.R., Krogh, A., Mitchison, G. (1998) Biological Sequence Analysis: Probabilistic Models of Proteins and Nucleic Acids. Cambridge University Press, Cambridge.

[28] Durrett, R., Nielsen, R., York, T.L. (2004) Bayesian estimation of genomic distance. *Genetics*, vol. 166, pp. 621–629.

[29] Edmonds, J. (1965) Paths, trees, and flowers. Canadian Journal of Mathematics, 17:449–467.

[30] Erdős, P. (1932) Beweis eines Satzes von Tschebyschef. Acta Litt. Ac. Sci. Regige Univ. Hung. Fr.-Jos Sect. Sci. Math. 5:194–198.

[31] Erdős, P., Gallai, T. (1960) Graphs with vertices of prescribed degrees (in Hungarian) Matematikai Lapok, 11: 264–274.

[32] Feijão, P., Meidanis, J. (2011) SCJ: A Breakpoint-Like Distance that Simplifies Several Rearrangement Problems. IEEE/ACM Transactions on Computational Biology and Bioinformatics 8(5): 1318–1329.

[33] Felsenstein, J. (1980) Evolutionary trees from DNA sequences: a maximum likelihood approach. J Mol Evol., 17:368–376.

[34] Fitch, W M 1(971) Toward defining the course of evolution: minimum change for a specified tree topology. *Systematic Zoology* 20:406–416.

[35] Floyd, R. W. (1962). Algorithm 97: Shortest Path. Communications of the ACM. 5(6):345.

[36] Forney, G.D. (1973) The Viterbi algorithm. Proceedings of the IEEE, 61:268–278.

[37] Gale, D. (1957) A theorem on flows in networks. Pacific J. Math. 7 (2): 1073–1082.

[38] Geman, S, Geman, D. (1984) Stochastic relaxation, Gibbs distributions and the Bayesian restoration of images. IEEE Transactions on Pattern Analysis and Machine Intelligence 12:609–628.

[39] Geyer, C.J. (1991) Markov chain Monte Carlo maximum likelihood. In: Keramigas, E., Editor, Computing Science and Statistics: The 23rd Symposium on the Inference, Interface Foundation, Fairfax, pp. 156–163.

[40] Giegerich, R. (2000) A Systematic Approach to Dynamic Programming in Bioinformatics. Bioinformatics, 16:665–667.

[41] Giegerich, R. Steffen, P. (2002) Implementing algebraic dynamic programming in the functional and the imperative programming paradigm. In E.A. Boiten and B. Möller, editors, Mathematics of Program Construction, Lecture Notes in Computer Science, 2386:1–20.

[42] Guíñez, F., Matamala, F.M., Thomassé, S. (2011) Realizing disjoint degree sequences of span at most two: A tractable discrete tomography problem, Discrete Appl. Math. 159(1):23–30.

[43] Hakimi, S.L. (1962) On the realizability of a set of integers as degrees of the vertices of a simple graph. SIAM Appl. Math., 10:496–506.

[44] Hastings, W. K. (1970) Monte Carlo sampling methods using Markov chains and their applications. Biometrika, 57:97–109.

[45] Hannenhalli, S., Pevzner, P.A. (1999) Transforming cabbage into turnip: polynomial algorithm for sorting signed permutations by reversals. Journal of the ACM, 46(1):1–27.

[46] Hartman, T., Verbin, E. (2006) Matrix Tightness: A Linear-Algebraic Framework for Sorting by Transpositions. Proceedings of SPIRE'06, pp. 279–290.

[47] Havel, V. (1955) A remark on the existence of finite graphs. (in Czech), Časopis Pěst. Mat., 80:477–480.

[48] Helman, P., Rosenthal, A. (1985) A comprehensive model of dynamic programming. SIAM Journal on Algebraic Discrete Methods, 6(2):319–333.

[49] Hopcroft, J.E., Karp, R.M. (1983) An $n^{5/2}$ algorithm for maximum matchings in bipartite graphs. SIAM Journal on Computing, 2(4):225–231.

[50] Huelsenbeck, J.P., Ronquist, F. (2001) MRBAYES: Bayesian inference of phylogeny. Bioinformatics, 17:754–755.

[51] Jacobson, M.T., Matthews, P. (1996) Generating uniformly distributed random latin squares. J. Combin. Des., 4(6):404–437.

[52] Jerrum, M. (2003) Counting, Sampling and Integrating: Algorithms and Complexity. Lectures in Mathematics. ETH Zürich. Birkhäuser Verlag, Basel, Switzerland.

[53] Jerrum, M., Snir, M. (1982) Some Exact Complexity Results for Straight-Line Computations over Semirings. Journal of the ACM, 29(3):874–897.

[54] Jerrum, M., Valiant, L., Vazirani, V. (1986) Random generation of combinatorial structures from a uniform distribution. Theoretical Computer Science, 43:169–188.

[55] Kannan, R., Tetali, P., Vempala, S. (1999) Simple Markov-Chain Algorithms for Generating Bipartite Graphs and Tournaments, Random Structures Algorithms, 14(4):293–308.

[56] Karp, R.M. (1972) Reducibility among combinatorial problems. in: R. E. Miller and J. W. Thatcher (eds.) Complexity of Computer Computations. 85–103. Plenum, New York

[57] Karzanov, A., Kachiyan, L. (1991) On the conductance of order Markov chains. Order, 8:7–15.

[58] Larget, B., Simon, D.L., Kadane, B.J. (2002) Bayesian phylogenetic inference from animal mitochondrial genome arrangements. J. Roy. Stat. Soc. B., 64(4):681–695.

[59] Larget B, Simon DL, Kadane JB, Sweet D. (2005) A Bayesian analysis of metazoan mitochondrial genome arrangements. Mol. Biol. Evol., 22(3):486–495.

[60] Lawler, G.F., Sokal, A.D. (1988) Bounds on the L^2 spectrum for Markov chains and Markov processes: A generalization of Cheeger's inequality. Transactions of the American Mathematical Society, 309(2):557–580.

[61] Lovász, L, Plummer, MD (1986) Matching Theory. Amsterdam, Netherlands: North-Holland.

[62] Lyngsø, R., Zuker, M., Pedersen, C. (1999) Fast evaluation of internal loops in RNA secondary structure prediction. Bioinformatics, 15(6):440–445.

[63] Ma, J., Ratan, A., Raney, B.J., Suh, B.B., Miller, W., Haussler, D. (2008) The infinite sites model of genome evolution PNAS, 105(38):14254–14261.

[64] Martin, R., Randall, D. (2006) Disjoint decomposition of Markov chains and sampling circuits in Cayley graphs. Combinatorics, Probability and Computing, 15:411–448.

[65] McCaskill, J.S. (1990) The equilibrium partition function and base pair binding probabilities for RNA secondary structure. Biopolymers 29, 1105–1119.

[66] Metropolis, N., Rosenbluth, A.W., Rosenbluth, M.N., Teller, A.H., Teller, E. (1953) Equations of state calculations by fast computing machines. J. Chem. Phys., 21(6):1087–1091.

[67] Meyer, I.M., Durbin, R. (2002) Comparative ab initio prediction of gene structures using pair HMMs. Bioinformatics, 18(10):1309–1318.

[68] Needleman, S. B., Wunsch, C. D. (1970). A general method applicable to the search for similarities in the amino acid sequence of two proteins. Journal of Molecular Biology, 48(3):443–453.

[69] Nussinov, R., Jacobson, A.B. (1980) Fast algorithm for predicting the secondary structure of single-stranded RNA. Proceedings of the National Academy of Sciences of the United States of America, 77(11):6309–6313.

[70] Ouangraoua, A., Bergeron, A. (2010) Combinatorial Structure of Genome Rearrangements Scenarios. Journal of Computational Biology, 17(9):1129–1144.

[71] Papadimitriou, C.H. (1994) Computational Complexity, Addison-Wesley, Reading, Mass.

[72] Rabiner, R.L. (1969) A tutorial on hidden Markov models and selected applications in speech recognition. Proceedings of the IEEE, 77(2):257–286.

[73] Ronquist, F., Huelsenbeck, J.P. (2003) MrBayes 3: Bayesian phylogenetic inference under mixed models Bioinformatics, 19:1572–1574.

[74] Roy, B. (1959). Transitivité et connexité (in French). C. R. Acad. Sci. Paris 249: 216–218.

[75] Ryser, H. J. (1957) Combinatorial properties of matrices of zeros and ones. Can. J. Math. 9: 371–377.

[76] Sankoff, D. (1972) Matching sequences under deletion/insertion constraints. Proceedings of the National Academy of Sciences of the USA, 69(1):4–6.

[77] Sankoff, D., Kruskal, J., Mainville, S., Cedergren, R. (1983) Fast algorithms to determine RNA secondary structures containing multiple loops. In: Sankoff, D., Kruskal, J. (Eds.), Time Warps, String Edits, and Macromolecules: The Theory and Practice of Sequence Comparison. Addison-Wesley, Reading, MA, pp. 93–120.

[78] Sankoff, D, Rousseau, P (1975) Locating the vertices of a Steiner tree in an arbitrary metric space. Mathematical Programming, 9:240–246.

[79] Sellers, P.H. (1974) On the theory and computation of evolutionary distances. SIAM Journal on Applied Mathematics, 26(4):787–793.

[80] Simon, I. (1988) Recognizable sets with multiplicities in the tropical semiring. Mathematical Foundations of Computer Science 1988. Lecture Notes in Computer Science. 324:107–120.

[81] Simon, D., Larget, B. (2004) Bayesian Analysis to Describe Genomic Evolution by Rearrangement (BADGER), version 1.01 beta. Department of Mathematics and Computer Science, Duquesne University

[82] Sinclair, A.J. (1992) Improved bounds for mixing rates of Markov chains and multicommodity flow. Combinatorics, Probability and Computing, 1:351–370.

[83] Sinclair, A., Jerrum, M. (1989) Approximate counting, uniform generation and rapidly mixing Markov chains, Inform. and Comput., 82:93–133.

[84] Stratonovich, R.L. (1960) Conditional Markov Processes. Theory of Probability and its Applications, 5(2):156–178.

[85] Sturtevant, A.H., Novitski, E. (1941). The homologies of the chromosome elements in the genus Drosophila. Genetics 26: 517–541.

[86] Sturtevant, A.H., Tan, C.C. (1937). The comparative genetics of Drosophila pseudoobscura and Drosophila melanogaster. J. Genet. 34: 415–432.

[87] Tannier, E., Bergeron, A., Sagot, F-M. (2007) Advances on sorting by reversals. Discrete Applied Mathematics, 155(6-7):881–888.

[88] Tannier, E., Zheng, C., Sankoff, D. (2009) Multichromosomal median and halving problems under different genomic distances. BMC Bioinformatics, 10:120.

[89] Tarnas C, Hughey, R (1998) Reduced space hidden Markov model training. Bioinformatics, 1998, 14:401–406.

[90] Thorne, J.L., Kishino, H., Felsenstein, J. (1991) An evolutionary model for maximum likelihood alignment of DNA sequences. J. Mol. Evol., 33:114–124.

[91] Tinoco, I.J., Borer, P., Dengler, B., Levine, M., Uhlenbeck, O. (1973) Improved estimation of secondary structure in ribonucleic acids. Nat. New Biol. 246:40–41.

[92] Valiant, L.G. (1979) The Complexity of Computing the Permanent. Theoretical Computer Science, 8(3):189–201.

[93] Viterbi, A.J. (1967) Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. IEEE Transactions on Information Theory, 13(2):260–269.

[94] Wagner, R.A., Fischer, M.J. (1974) The string-to-string correction problem. Journal of the ACM, 21(1):168–173.

[95] Warshall, S. (1962). A theorem on Boolean matrices. Journal of the ACM. 9(1):11–12.

[96] Wheeler R, Hughey R (2000) Optimizing reduced-space sequence analysis. Bioinformatics,16(12):1082–1090.

[97] Wuchty, S., Fontana, W., Hofacker, I., Schuster, P. (1999) Complete suboptimal folding of RNA and the stability of secondary structures. Biopolymers 49, 145–165.

[98] Yancopoulos, S., Attie, O., Friedberg, R. (2005) Efficient sorting of genomic permutations by translocation, inversion and block interchange. Bioinformatics, 21(16):3340–3346.

[99] Yang, Z., Rannala, B. (1997) Bayesian phylogenetic inference using DNA sequences: a Markov chain Monte carlo method. Molecular Biology and Evolution. 14:717–724.

[100] York, T.L., Durrett, R., Nielsen, R. (2002) Bayesian estimation of inversions in the history of two chromosomes. J. Comp. Biol., 9:808–818.

[101] Zuckerkandl, E., Pauling, L. (1965) Molecules as documents of evolutionary history. J. Theor. Biol., 8(2):357–66.

Publications by the author of the thesis

[102] Aksen, M., **Miklós, I.**, Zhu, K. (2017) Half-regular factorizations of the complete bipartite graph. Discrete Applied Mathematics, 230:21–33.

[103] Bixby, E, Flint, T, **Miklós, I.**, (2016) Proving the Pressing Game Conjecture on Linear Graphs Involve, 9(1):41–56.

[104] Czabarka, É., Dutle, A., Erdős, P.L., **Miklós, I.** (2014) On Realizations of a Joint Degree Matrix. Discrete Applied Mathematics, 181(30):283–288.

[105] Darling, A., **Miklós, I.**, Ragan, M. (2008) Dynamics of genome rearrangement in bacterial populations. *PLoS Genetics*, vol. 4, num. 7., e1000128.

[106] Erdős, E.L., Greenhill, C., Mezei, T. R., **Miklós, I.**, Soltész, D., Soukup, L. (2022) The mixing time of the switch Markov chains: a unified approach, Eur. J. Comb., 99:103421.

[107] Erdős, E.L., Győri, E., Mezei, T. R., **Miklós, I.**, Soltész, D. (2021) Half-graphs, other non-stable degree sequences, and the switch Markov chain, Electronic Journal of Combinatorics, 28:3 #P3.7.

[108] Erdős, P.L., Kiss, Z.S., **Miklós, I.**, Soukup, L. (2015) Approximate Counting of Graphical Realizations. PLoS ONE, 10(7):e0131300.

[109] Erdős, E.L. Mezei, T., **Miklós, I.**, Soltész, D. (2018) Efficiently sampling the realizations of bounded, irregular degree sequences of bipartite and directed graphs. PLoS ONE, 13(8): e0201995.

[110] Erdős, P.L., **Miklós, I.**, Toroczkai, Z. (2015) A decomposition based proof for fast mixing of a Markov chain over balanced realizations of a joint degree matrix. SIAM Journal on Discrete Mathematics, 29:481–499.

[111] Erdős, P.L., **Miklós, I.**, Toroczkai, Z. (2018) New classes of degree sequences with fast mixing swap Markov chain sampling Combinatorics, Probability and Computing, 27(2):186–207.

[112] Hong, L., **Miklós, I.** (2022) A Markov chain on the solution space of edge-colorings of bipartite graphs. Discrete Applied Mathematics, accepted.

[113] Lunter, G.A., **Miklós, I.**, Drummond, A.J., Jensen, J.L., Hein, J.J. (2005) Bayesian Coestimation of Phylogeny and Sequence Alignment. BMC Bioinformatics, 6:83

[114] **Miklós, I.** (2003) MCMC Genome Rearrangement. Bioinformatics, 19:ii130–ii137.

[115] **Miklós, I.** (2019) Computational complexity of counting and sampling, CRC Press.

[116] **Miklós, I.**, Darling, A. (2009) Efficient sampling of parsimonious inversion histories with application to genome rearrangement in Yersinia Genome Biology and Evolution, 1(1):153–164.

[117] **Miklós, I.**, Erdős, P.L., Soukup, L. (2013) Towards random uniform sampling of bipartite graphs with given degree sequence, Electronic J. Comb., 20(1):#P16.

[118] **Miklós, I.**, Ittzés, P., Hein, J. (2005) ParIS genome rearrangement server. Bioinformatics, 21(6):817–820.

[119] **Miklós, I.**, Mélykúti, B., Swenson, K. (2010) The Metropolized Partial Importance Sampling MCMC mixes slowly on minimum reversal rearrangement paths ACM/IEEE Transactions on Computational Biology and Bioinformatics, 4(7):763–767.

[120] **Miklós, I.**, Meyer, I.M. (2005) A linear memory algorithm for Baum-Welch training. BMC Bioinformatics 6:231

[121] **Miklós, I.**, Meyer, I.M., Nagy, B. (2005) Moments of the Boltzmann distribution for RNA secondary structures Bul. Math. Biol., 67(5):1031-1047

[122] **Miklós, I.**, Smith, H. (2015) Sampling and counting genome rearrangement scenarios, BMC Bioinformatics, 16(Suppl 14): S6.

[123] **Miklós, I.**, Smith, H. (2019) The computational complexity of calculating partition functions of optimal medians with Hamming distance. Advances in Applied Mathematics, 102:18–82.

[124] **Miklós, I.**, Tannier, E. (2010) Bayesian sampling of genome rearrangement scenarios via Double Cut and Join. *Bioinformatics*, 26: 3012–3019.

[125] **Miklós, I.**, Tannier, E. (2012) Approximating the number of Double Cut-and-Join scenarios, Theoretical Computer Science 439:30–40.

[126] **Miklós, I.**, Tannier, E., Kiss, Z.S. (2014) On sampling SCJ rearrangement scenarios. Theoretical Computer Science, 552:93–98.

Subject Index