Radboud University Nijmegen

Faculty of Science

Simulating Quantum Gravity

Using Triangles and Trees

Supervisor:
dr. Timothy Budd

Author:
T.B.H. Gerstel

Second reader:
prof. dr. Renate Loll

July 2023

Abstract

Dynamical Triangulations is an approach for discretizing the Euclidean quantum gravity path integral using regular simplices. On the 2-sphere such models have been shown to lead to a family of universality classes with a consistent topology. However, in three (or more) dimensions such a correspondence is currently out of reach. An attempt to remedy this has been proposed by T. Budd and L. Lionni in 2022 [4] in the form of the triple trees model. This model puts a heavy restriction on the possible configurations by requiring a specific decomposition of the geometry into three trees. In this work, the model is investigated numerically using a Markov Chain Monte Carlo simulation written in Rust. In particular, the phase transition is analysed qualitatively and quantitatively. Furthermore, a peculiar distance scaling in the low κ phase is presented.

Philosophy is written in this grand book — I mean the Universe — which stands continually open to our gaze, but it cannot be understood unless one first learns to comprehend the language and interpret the characters in which it is written. It is written in the language of mathematics, and its characters are triangles, circles, and other geometrical figures, without which it is humanly impossible to understand a single word of it; without these, one is wandering around in a dark labyrinth.

- Galileo Galilei, *Il Saggiatore*

Contents

Chapter 1

Introduction

1.1 Euclidean Quantum Gravity

One of the major challenges in contemporary theoretical physics is defining a predictive theory of quantum gravity. Our modern understanding of classical gravity in the form of general relativity was introduced by Albert Einstein in 1915. General relativity states that gravity is in fact a geometric effect, linking the presence of matter with the curvature of spacetime. For simplicity, we shall restrict ourselves to geometries with Euclidean signature.

The dynamics of the Euclidean version of Einstein's theory in d dimensions can be very compactly summarized in the Euclidean Einstein-Hilbert action

$$S[g_{ab}] = \frac{1}{16\pi G} \int_{\mathcal{M}} \mathrm{d}^d x \sqrt{g(x)}[-R(x) + 2\Lambda], \tag{1.1}$$

where G is Newton's gravitational constant and Λ is the cosmological constant. The integration is over the d-dimensional Riemannian manifold $\mathcal{M}$ with metric $g_{ab}(x)$, and it has a measure related to the determinant of the metric $g(x)$. Finally, $R(x)$ is the Ricci scalar curvature at $x \in \mathcal{M}$. As with any classical action, the equations of motion can be found using the stationary-action principle.

The usual approach to quantizing any classical theory described by an action is to embed said action in a path integral. In the case of Euclidean gravity this results in

$$Z = \int \mathcal{D}g_{ab} \frac{e^{-S[g_{ab}]}}{|\mathrm{Diff}\,\mathcal{M}|}, \tag{1.2}$$

where the integration is over all geometries[1]. To prevent overcounting due to invariance under coordinate transformations we divide by the volume of the diffeomorphism group Diff $\mathcal{M}$. By considering the Euclidean version of general relativity, the path integral can also be interpreted as a statistical partition function. In particular, the action $S[g_{ab}]$ determines the Boltzmann weight $e^{-S[g_{ab}]}$.

[1]More precisely, the integration is over equivalence classes of metrics $[g_{ab}]$ related by diffeomorphisms.

However, making quantitative statements derived from (1.2) is still a major open problem. It is not straightforward at all to assign a well-defined probability space and measure consistent with (1.2). The current goal of Euclidean quantum gravity, in this sense, is therefore to assign a well-defined statistical interpretation to (1.2) as a partition function on the space of all geometries.

The standard method to make sense of functional integrals such as (1.2) is to employ perturbative methods. In particular, one tries to quantize fluctuations around some classical solution of the field equations. However, it turns out that the quantum field theory corresponding to (1.2) is perturbatively non-renormalizable. This implies that we cannot make predictions by probing the model using perturbative methods.

Dynamical Triangulations In order to study (1.2) non-perturbatively we can introduce a discretization of geometries. In particular, we can discretize a manifold $\mathcal{M}$ by gluing a set of d-dimensional polyhedra together. One can vary either the shape of the polyhedra or their connectivity (or both) to obtain different geometries.

In the Dynamical Triangulations (DT) approach, we keep the shape of all polyhedra fixed, while varying their connectivity. This has the advantage of resulting in discrete degrees of freedom, turning (1.2) into a summation. As for the shape of the polyhedra, we make the simplest choice by only considering regular simplices with physical side length a. By regular simplices we mean, in order of increasing dimension: vertices, edges, triangles, tetrahedra and higher-dimensional generalizations of these. All in all, the discretized partition function will be of the form

$$Z = \sum_{T \in \mathcal{T}} \frac{e^{-S[T]}}{C[T]}, \tag{1.3}$$

where the summation is over the space of all possible gluing relations $\mathcal{T}$. Here $C[T]$ is a symmetry factor to correct for overcounting.

Since DT is a discretization of (1.2), we hope to find a continuum limit relating DT back to a continuum model. Ideally, we would like to use such a continuum model as a potential definition of (1.2). Taking a continuum limit means that we take the physical length of the building blocks $a \to 0$ and increase the number of building blocks $N \to \infty$, while keeping the physical volume fixed. Of course there are infinite ways in which such a limit can be taken, but only a few lead to convergent results.

Our approach to taking this limit is to look for a convergence of metric spaces in line with the so-called Gromov-Hausdorff definition, as used in [7]. In particular, we consider the possibility of a power law rescaling of geodesic distances. This type of rescaling is only possible if the model has a well-defined, finite Hausdorff dimension d_H. The Hausdorff dimension describes how the volumes of geodesic balls grow as a function of their radius. The relationship between d_H and a continuum limit is further elaborated in section 2.3.

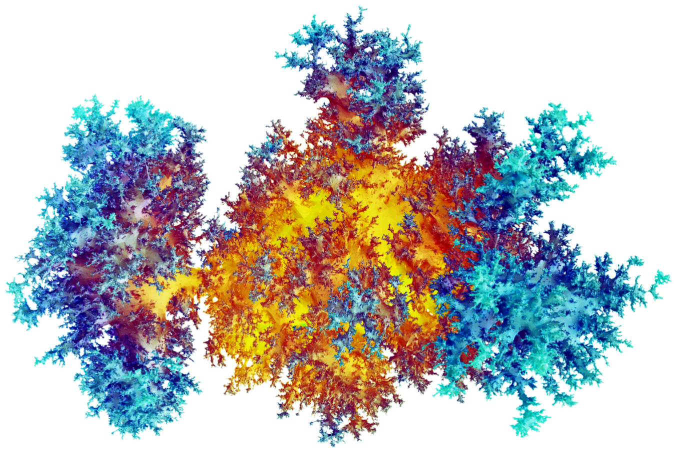

Figure 1.1: Visualization of a triangulation in $d = 2$ with $N = 8 \cdot 10^6$ triangles. Adapted from Benedikt Stufler's gallery.

1.2 Current Models

Two Dimensions We start our tour of DT models in $d = 2$ dimensions, since a lot is already known about these models mathematically. When considering the action 1.1 for $d = 2$ we will see that by the Gauß-Bonnet theorem the integrated curvature is constant,

$$\int_{\mathcal{M}} \mathrm{d}^2 x \sqrt{g(x)} R(x) = 4\pi\chi(\mathcal{M}), \tag{1.4}$$

where $\chi(\mathcal{M})$ is the so-called Euler characteristic of the manifold $\mathcal{M}$. It depends only on the topology of $\mathcal{M}$, so in this context it is constant and therefore irrelevant to the dynamics. Thus, for $d = 2$, only the integrated volume contributes to the dynamics. We can therefore write the action as

$$S[T] = \lambda N_2[T], \tag{1.5}$$

where λ is related to the continuum cosmological constant Λ from (1.1). Here $N_d[T]$ denotes the number of d-simplices contained in T. In this case $N_2[T]$ denotes the number of triangles in T. For $d = 2$ the model is so simple that at fixed volume (i.e. number of triangles) the action is constant, resulting in an equal probability for all triangulations of the given volume, modulo symmetry factors. An example of such a triangulation is shown in figure 1.1.

It has been shown [7] that this model has a continuum limit, known as the Brownian sphere. Many properties of the Brownian sphere are known analytically. In particular, it has a Hausdorff dimension $d_H = 4$. It provides an example for how a metric structure can be obtained from a DT model. A major caveat here is that the resulting manifold turns out to be fractal in nature.

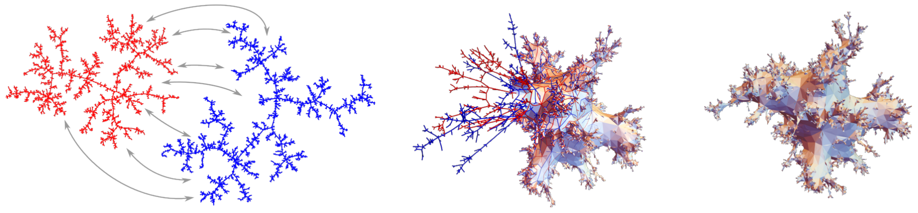
Figure 1.2: Visualization of the mating of trees procedure. Left: two trees including a gluing relation. Middle: the trees are being glued together. Right: the trees have been glued into a manifold. Adapted from the 3rd EPS Conference on Gravitation talk “Quantum Gravity and Random Geometry” by Timothy Budd.

Alternatives to pure DT can be constructed by introducing extra information (i.e. decorations) on the triangulations. A whole zoo of models has been investigated here, including spanning tree decorations, Ising models and scalar fields, to name a few. It turns out that many of these models belong to a 1-parameter family of universality classes known as Liouville quantum gravity. A very peculiar property of these geometries is that they can be constructed by interlacing two continuum random trees with each other [6]. This is highly non-trivial, since it allows us to construct a simple manifold from topologically extremely complicated objects, i.e. trees. A visualization of this procedure is shown in figure 1.2.

Three Dimensions For $d = 3$, the DT action takes the form [2]

$$S[T] = \lambda N_3[T] - \kappa N_0[T], \tag{1.6}$$

where κ is related to the inverse of Newton’s gravitational constant G from (1.1). At fixed volume the model still has a free parameter, namely κ. It turns out [2] [8] that various phases emerge, depending on the value of κ.

For $\kappa < \kappa_c$ (where κ_c is the critical value at which the phase transition happens) the model is in the so-called crumpled phase. In this phase there is generally one (or a few) vertices with very high degree, indicating a singular behaviour. In particular, it appears [12] to have a Hausdorff dimension $d_H = \infty$. Because of this it seems unlikely that a continuum limit to a metric space can be obtained. That is, not using methods analogous to the $d = 2$ case, as presented in [7].

On the other hand, for $\kappa > \kappa_c$ the model is in a tree-like phase, commonly referred to as the “branched polymer” phase. The universality class of this phase seems [8] to be the continuum random tree of Aldous [1]. Hence, this phase also does not correspond to physical quantum gravity.

Finally, there is the possibility of new behaviour emerging at the phase transition. According to statistical mechanics, we would expect to find crit-

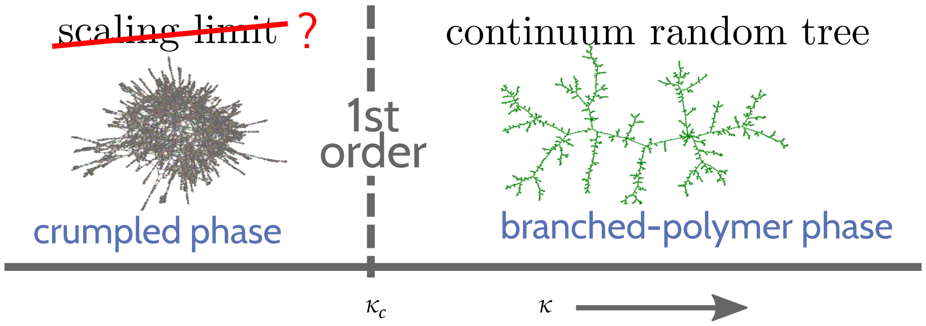

Figure 1.3: An overview of the two phases in $d = 3$ DT. Adapted from the 3rd EPS Conference on Gravitation talk “Quantum Gravity and Random Geometry” by Timothy Budd.

ical behaviour if the transition were to be continuous. This behaviour includes a correlation length ξ that diverges as a power law near the transition. This divergence could then be used to redefine the physical distance a such that the physical correlation length ξa converges. We could then take $a \to 0$ and $N \to \infty$ while retaining a finite physical distance scale. Unfortunately there is strong evidence [9] that the phase transition in standard $d = 3$ DT is discontinuous. An overview of the phase space for $d = 3$ DT is shown in figure 1.3

1.3 The Triple Trees Model

In 2022, Budd and Lionni [4] introduced an alternative model to pure DT in $d = 3$. The defining feature of this model is that triangulations are constructed from three different trees, hence the name triple trees. This construction can in some sense be viewed as reverse-engineering an analogue to the mating of trees construction presented in [6], extending the idea from $d = 2$ to $d = 3$.

Since the model is by construction decomposable into three trees, it shares some useful properties that usually come with trees. Most notably exponential bounds on the canonical partition function have been established, something that has at present not been done for pure DT. This is also an indication of just how restrictive the model is on the geometry. Since this restriction is so extreme, it seems plausible that this model could contain new behaviour not seen in pure DT. Additionally, better control over enumerations is expected to go hand in hand with a greater ability to apply analytical methods.

This Work As the triple trees model is relatively new it still leaves much to be explored. In particular, the phase transition (if it still exists in triple trees) may show interesting behaviour. If the transition turns out to be continuous, it could potentially host a new universality class. Thus, we focus our efforts on trying to answer the following questions:

1. Can we find evidence for a phase transition in the triple trees model?
2. What are the properties of these phases, and do they differ from pure DT?
3. Assuming there is a phase transition, what can we say about its (dis)continuity?

We will investigate these questions numerically using a Markov Chain Monte Carlo simulation written in the Rust programming language.

In chapter 2 we will define the triple trees model in more detail, and we will elaborate on how we define the expectation values that can be extracted from such a statistical system. How these expectation values are extracted in practice is elaborated in chapter 3. In particular, we elaborate on how we can design a Markov chain that we can use to sample configurations from our configuration space. Then, in chapter 4 we give an overview of which types of observables we can measure on our triangulations and what these can tell us about possible phase transitions. Additionally, we briefly discuss the error analysis employed on the data itself. The results of this analysis are presented in chapter 5. Finally, we interpret these results in chapter 6.

Chapter 2

Defining the Triple Trees Model

As mentioned in the introduction, DT can be interpreted as a statistical model. As such, its dynamics can be captured in a partition function Z. This partition function is given by (1.3), and we restate it here:

$$Z = \sum_{T \in \mathcal{T}} \frac{e^{-S[T]}}{C[T]}, \tag{2.1}$$

where the summation is over all triangulations T contained in some configuration space $\mathcal{T}$. Each triangulation T has a Boltzmann weight (or euclidean action in the context of quantum gravity) $S[T]$, which for $d = 3$ has already been presented in (1.6). Additionally, a symmetry factor $C[T]$ is included to compensate for overcounting. Note that the model is uniquely determined by specifying $\mathcal{T}$ and $S[T] : \mathcal{T} \to \mathbb{R}$.

2.1 Labels and Symmetry

As mentioned, the symmetry factor $C[T]$ is present in (2.1) in order to compensate for overcounting. It is equal to the order of the automorphism group of T

$$C[T] = |\mathrm{Aut}\, T|. \tag{2.2}$$

For abstract triangulations T, as considered here, $|\mathrm{Aut}\, T|$ depends on the symmetry of T. This results in $C[T]$ having a rather complicated structure, and it is therefore difficult to work with both analytically and numerically. However, if we instead consider *labelled* triangulations $T_\ell \in \mathcal{T}_\ell$, there are no (non-trivial) automorphisms. Instead, each permutation of labels occurs as a separate term in the summation. If a triangulation T has a set of labels $\ell[T]$, there are in principle $|\ell[T]|!$ different permutations of these labels. However, due to automorphisms some of these permutations are equivalent, thus reducing the actual number of distinct labellings. This means that an unlabelled triangulation $T \in \mathcal{T}$ will correspond to

$$\frac{|\ell[T]|!}{|\mathrm{Aut}\, T|} \tag{2.3}$$

labelled triangulations $T_\ell \in \mathcal{T}_\ell$. Hence, we can rewrite (2.1) to

$$Z = \sum_{T_\ell \in \mathcal{T}_\ell} \frac{e^{-S[T_\ell]}}{|\ell[T_\ell]|!}. \tag{2.4}$$

Note that we are agnostic over what exactly the labels refer to. For instance, a label could be assigned to each vertex ($|\ell[T]| = N_0[T]$), or to each half-edge ($|\ell[T]| = 12N_3[T]$), or both ($|\ell[T]| = N_0[T] + 12N_3[T]$). As long as the labels completely break the symmetry the argument is the same. Note that for the remainder of this work, we will only be considering labelled triangulations. Therefore, the subscript ℓ shall be suppressed from this point onward.

2.2 Configuration Space

We have defined a Boltzmann weight $e^{-S[T]}$ using the action in (1.6). However, in order to completely specify our model we still need to define a configuration space $\mathcal{T}$. The biggest restriction here is that we do not consider changes in topology. In particular, we only consider spherical topology, since S^3 is the simplest choice which is compact and without boundaries. Besides fixing the topology of our triangulation, we can also impose restrictions relating to the degeneracy of its underlying simplices, further restricting $\mathcal{T}$. Finally, we can introduce extra structures (such as trees) on top of the triangulation. This generally has the effect of enlarging $\mathcal{T}$, although putting restrictions on these structures can also reduce the size of $\mathcal{T}$.

Subsimplices Configuration space restrictions in dynamical triangulations are more straightforwardly explained by first introducing the concept of subsimplices. In this context a p-simplex P is a subsimplex of a q-simplex Q (with $p < q$) if and only if $P \subset \partial Q$. Conversely, Q is a supersimplex of P. A sub- or supersimplex is *direct* if and only if $q = p + 1$. For example, a triangle is a direct supersimplex of any one of its edges. It is also a supersimplex of any one of its vertices, but in these cases it is not a direct supersimplex. In general, a simplex could have duplicates among its subsimplices. Additionally, two distinct simplices may have exactly the same subsimplices.

Degeneracy Categories As already suggested, there are various types of degeneracies that can occur in a triangulation. Broadly speaking, there are two main categories of degeneracies:

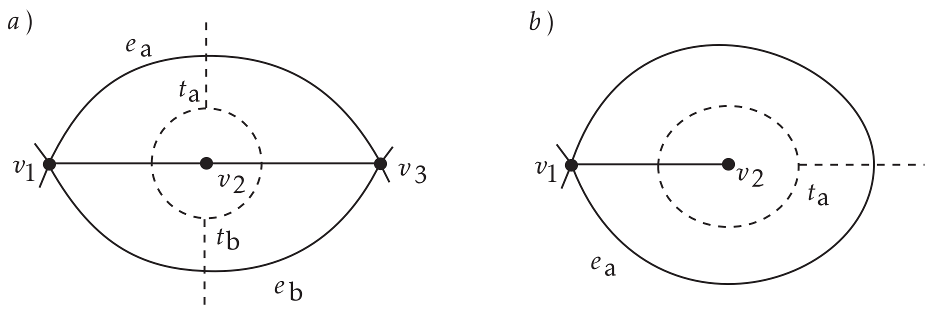

Figure 2.1: Two examples of degenerate $d = 2$ triangulations. The dashed lines indicate the corresponding dual graphs. In (*a*) the triangles t_a and t_b have all three vertices v_1, v_2 and v_3 in common (a self-energy diagram in the dual graph); this is a degeneracy of the first category with $(p, q) = (0, 2)$. In (*b*) the triangle t_a contains same vertex v_1 twice (corresponds to a tadpole in the dual graph); this is a degeneracy of the second category with $(p, q) = (0, 2)$. Adapted from [12].

1. Two distinct q-simplices share exactly the same set of p-dimensional subsimplices.
2. A q-simplex has duplicates among its p-dimensional subsimplices.

Here p and q should be specified in order to specify a particular type of degeneracy. A visualization of these degeneracies is shown in figure 2.1. Triangulations without degeneracies are commonly formally described by so-called simplicial complexes. However, these rely on the use of sets, so duplicate simplices as they occur in a degeneracy of the first type are not representable. Furthermore, in a simplicial complex simplices are identified based on their subsimplices. This implies that also degeneracies of the second category cannot be represented.

In the model studied in this work, we require that each edge connects two distinct vertices. This excludes all degeneracies in the second category. In particular, this restriction implies that all vertices of each triangle and tetrahedron are distinct. This is because their edges form complete graphs on their vertices. Hence, duplicates in these vertices would result in an edge connecting a vertex to itself. Building on this, we see that duplicate edges (within triangles or tetrahedra) and duplicate triangles (within tetrahedra) are also excluded. These would require the identification of two (or more) vertices in their parent triangle or tetrahedron.

We do still allow degeneracies of the first category. For example, two vertices can be connected by multiple edges, and triangles can share more than one edge. Choosing exactly this restriction is convenient for simulation purposes, as will be discussed in section 3.2.

Three Trees Now we consider the unique defining feature of the triple trees model: the three trees. We shall denote the trees by b_v, b_d and b_m, for vertex, dual and middle tree, respectively[1]. Suppose that $\Sigma[T]$ is the set of all simplices of $T \in \mathcal{T}$. Then b_v, b_d and b_m must partition $\Sigma[T]$. That is, each simplex can only be part of one of the trees, and all simplices must be contained in some tree. Furthermore, we consider two simplices within the same tree to be linked if and only if one is a direct subsimplex of the other. Hence, simplices can only be connected if their dimensionality differs by exactly one.

But what exactly are these trees? As a start, b_v is a spanning tree of the underlying graph. This means that all vertices must be contained in b_v. Additionally, since b_v is a tree it consists of a single connected component, and it contains no loops. To match our aforementioned requirements, we consider the edges that connect vertices to also be part of the tree. That is, part of the edges in $\Sigma[T]$ are contained in b_v. Similarly, b_d is a spanning tree of the dual graph. It therefore contains all tetrahedra, as well as part of the triangles of $\Sigma[T]$.

What remains of $\Sigma[T]$ is a subset of all triangles and edges of T, specifically $b_m = \Sigma[T] \setminus (b_v \cup b_d)$. To clarify again, we consider a triangle and an edge to be connected if the edge is a subsimplex of the triangle. In this sense, we can define a "middle" graph b_m on the remaining triangles and edges. Note that b_m can only be a tree if it contains a single connected component, and if it contains no cycles.

In summary, adding the spanning trees b_v and b_d, and requiring the remaining graph b_m to be a tree, will result in the partition function

$$Z = \sum_{T \in \mathcal{T}} \sum_{b_v \in \mathcal{B}_v[T]} \sum_{b_d \in \mathcal{B}_d[T]} \theta(C[b_m] = 1)\, \theta(L[b_m] = 0)\, \frac{e^{-S[T]}}{|\ell[T]|!}, \tag{2.5}$$

where $\mathcal{B}_v[T]$ and $\mathcal{B}_d[T]$ denote the set of possible spanning trees on T and spanning trees on the dual graph of T, respectively. Furthermore, $C[b_m]$ and $L[b_m]$ denote the number of connected components and the number of cycles in b_m. An alternative way to view this is to redefine the configuration space to

$$\mathcal{T}_{\text{triple trees}} = \{(T, b_v, b_d) \mid T \in \mathcal{T}, b_v \in \mathcal{B}_v[T], b_d \in \mathcal{B}_d[T], C[b_m] = 1, L[b_m] = 0\}. \tag{2.6}$$

An example of how such a triangulation can be constructed is shown in figure 2.2.

Now we make a crucial observation: there are many configurations T for which b_m cannot be a tree, even considering all possible choices of b_v and b_d. This has a major effect on the configuration space $\mathcal{T}$. Furthermore, adding decorations (such as trees) will also have a combinatorial effect on $\mathcal{T}$. To make this effect more explicit we rewrite (2.5) to

$$Z = \sum_{T \in \mathcal{T}} \omega[T] \frac{e^{-S[T]}}{|\ell[T]|!}, \tag{2.7}$$

where $\omega[T]$ can be interpreted as a weight. Thus, we can also view the triple trees model as a modification to DT where we assign a (possibly vanishing)

[1]Note that we use the symbol b to denote a tree, from the Dutch "boom". The logical choice t could be confused with T for triangulation.

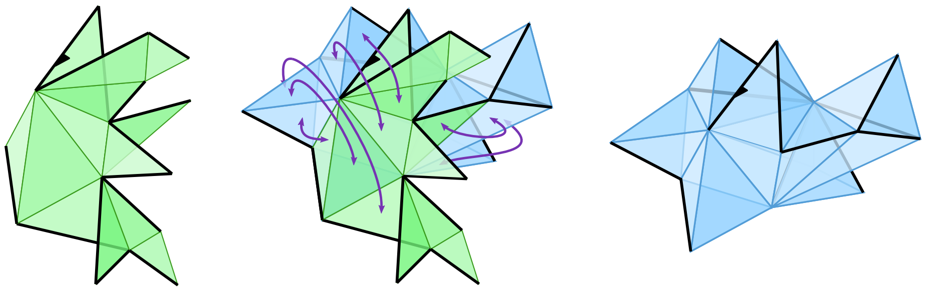

Figure 2.2: Visual overview of the construction of a triple trees geometry. Left: the middle tree b_m (in green) and the vertex tree b_v (in black). Right: the dual tree b_d (in blue) and the vertex tree b_v (in black). Center: the middle tree b_m is used to define a gluing on the dual tree b_d. Adapted from [4].

weight $\omega[T]$ to each $T \in \mathcal{T}$. In the case of the triple trees model we have

$$\omega[T] = \sum_{b_v \in \mathcal{B}_v[T]} \sum_{b_d \in \mathcal{B}_d[T]} \theta(C[b_m] = 1)\, \theta(L[b_m] = 0), \tag{2.8}$$

which is significantly different from the DT model where $\omega[T] = 1$. From this point onward, we shall drop the subscript "triple trees" from the symbol $\mathcal{T}_{\text{triple trees}}$. In other words, the symbol $\mathcal{T}$ will now directly refer to (2.6).

2.3 Continuum Limit

At its core, DT is (an attempt at) an approach for regularizing the quantum gravity path integral as presented in (1.2). The corresponding regularization parameter is the physical edge length a. Hence, the goal is to eventually go back to the continuum picture by taking an appropriate limit. In the case of DT, this means taking the physical edge length a to zero and the number of building blocks N to infinity, while keeping the physical volume fixed. Before we can do this we first need to be able to study the properties of the triangulations in our ensemble as a function of N.

Volume Partitioning In order to investigate triangulations at a given volume, we must partition the configuration space $\mathcal{T}$ into components, each corresponding to a specific volume:

$$\mathcal{T} = \bigcup_{N=0}^{\infty} \mathcal{T}_N, \tag{2.9}$$

where we have defined

$$\mathcal{T}_N = \{T \in \mathcal{T} \mid N_d[T] = N\} \tag{2.10}$$

to be the set of all triangulations of volume N. Applying this partitioning to the partition function in (2.4) results in

$$Z = \sum_{N=0}^{\infty} \sum_{T \in \mathcal{T}_N} \frac{e^{-S[T]}}{|\ell[T]|!} = \sum_{N=0}^{\infty} Z_N, \tag{2.11}$$

where in our case $S[T]$ is the DT action in (1.6), and we have defined

$$Z_N = \sum_{T \in \mathcal{T}_N} \frac{e^{-S[T]}}{|\ell[T]|!}. \tag{2.12}$$

Rescaling Expectation values of observables on these fixed-volume configuration spaces $\mathcal{T}_N$ are given by

$$\langle \mathcal{O} \rangle_N = \frac{1}{Z_N} \sum_{T \in \mathcal{T}_N} \mathcal{O}[T] \frac{e^{-S[T]}}{|\ell[T]|!} \tag{2.13}$$

Suppose that such an expectation value grows as a power law in the limit of large volume, i.e.

$$\langle \mathcal{O} \rangle_N \overset{N \to \infty}{\sim} N^{\nu}, \tag{2.14}$$

where ν is some scaling exponent specific to the observable. In such cases we can *rescale* the observables as

$$\langle \mathcal{O} \rangle_N \mapsto \frac{\langle \mathcal{O} \rangle_N}{N^{\nu}}, \tag{2.15}$$

such that the rescaled observable has a well-defined limit for $N \to \infty$. Thus, we look for observables that asymptotically scale as a power law.

Hausdorff Dimension As mentioned in section 1.1 we need a well-defined Hausdorff dimension d_H in order to apply our recipe for a continuum limit. Starting from a point $x \in T$ on a random triangulation T, we grow a geodesic ball $B_{x \in T}(r)$ of radius r. After averaging over x and T, we hope to find a power law dependence on r of the volume

$$\langle |B_{x \in T}(r)| \rangle_{x,T} \sim r^{d_H}, \tag{2.16}$$

where the absolute value denotes the volume of the ball. To avoid discretization and finite-size effects we only look at the range where $1 \ll r \ll N$. If d_H is well-defined and finite, we can say that distances scale as $\xi \sim N^{1/d_H}$. If we then set the physical edge length

$$a = N^{-1/d_H}, \tag{2.17}$$

we will see that physical distances of the form $a\xi$ should approach constant values in the continuum limit.

Chapter 3

Simulation Design

In the previous chapter we have seen that it is interesting to compute expectation values of the form

$$\langle \mathcal{O} \rangle_N = \sum_{T \in \mathcal{T}_N} \mathcal{O}[T] \frac{e^{-S[T]}}{|\ell[T]|!}, \tag{3.1}$$

where $\mathcal{T}_N$ is the volume-fixed configuration space of (2.10), applied to the configuration space $\mathcal{T}$ of triple trees from (2.6). The Boltzmann weight $e^{-S[T]}$ is determined by the action of DT in $d = 3$ given by (1.6). In this chapter we will discuss the methods by which we obtain such expectation values.

3.1 Markov Chain Monte Carlo

Monte Carlo Directly computing expressions of the form of (3.1) does not seem feasible. There is no clear way to parameterize $\mathcal{T}_N$, and finite expansions quickly become extremely unwieldy, since $|\mathcal{T}_N|$ grows exponentially as a function of N. Instead, we can approximate these expectation values by taking n random samples from the distribution defined by $\mathcal{T}_N$ and $S[T]$, and computing the sample mean. In particular, to compute the expectation value of some observable $\mathcal{O}$ we independently sample n triangulations $T_i \in \mathcal{T}_N$ ($i = 1, \ldots, n$) distributed according to the Boltzmann weights $S[T]$. The expectation value $\langle \mathcal{O} \rangle_N$ can then be approximated with

$$\langle \mathcal{O} \rangle_N \approx \frac{1}{n} \sum_{i=1}^{n} \mathcal{O}[T_i]. \tag{3.2}$$

The r.h.s. of (3.2) has expectation value $\langle \mathcal{O} \rangle_N$ and its standard deviation is proportional to $\frac{1}{\sqrt{n}}$. Therefore, for $n \to \infty$ it will approach $\langle \mathcal{O} \rangle_N$ almost surely.

Markov Chain A practical question remains: How can we sample triangulations T_i from $\mathcal{T}_N$? There does not seem to be a straightforward and systematic way to pull a T_i out of a hat (i.e. to directly sample $T_i \in \mathcal{T}$). Instead, we use a Markov chain to generate the samples. We start with some initial configuration $T_0 \in \mathcal{T}_N$. Then we iteratively apply a specific, non-deterministic update rule to transition from T_t to T_{t+1}. This update rule should satisfy two criteria:

- It should be able to explore the entire configuration space $\mathcal{T}_N$, at least in principle. This is called ergodicity.
- It should result in the desired distribution. This can be achieved by satisfying detailed balance, as in (3.3).

We will see that the update rules used in this work are local. This means that T_t and T_{t+1} will be very similar, and thus they cannot be considered as independent samples. However, the idea is that after many of these updates the samples will be effectively independent. Still, we should check whether this actually the case. This will be investigated in 4.3.

3.2 A Markov Chain for DT

Before attempting to construct an ergodic update rule for the triple trees model we shall consider the case of dynamical triangulations without decorations. It has been argued by Thorleifsson [12] that for a model in three dimensions with non-degeneracy restrictions as described in section 2.2 it is sufficient to use two types of so-called (p, q) moves (and their inverses). In such a move, a group of p tetrahedra is replaced by a group of q tetrahedra.

In particular the moves used in [12] are the $(1, 4)$ and $(2, 3)$ moves, with inverse moves $(4, 1)$ and $(3, 2)$. Executing a $(1, 4)$ move means inserting a vertex of order four into a tetrahedron, and a $(2, 3)$ move corresponds to replacing a triangle by an edge. A visual representation of these moves is shown in figure 3.1.

However, in this work we use a $(0, 2)$ move instead of the $(1, 4)$ move. This move replaces a triangle by a complex consisting of two tetrahedra sharing three triangles. Note that the boundary of such a complex is a two-sided triangle. We use this move because the $(4, 1)$ move requires a vertex of order four, which is relatively rare. It turns out that this substitution results in an equivalent set of moves. This can be shown by the fact that a $(1, 4)$ move can be emulated by performing a $(0, 2)$ move on some triangle followed by a $(2, 3)$ move on that triangle. The inverse move $(4, 1)$ can be constructed accordingly.

One thing we still need to consider is whether we accidentally create degenerate manifolds. The only restriction we have imposed is to exclude edges with duplicate vertices. Edges can only be created in the $(0, 2)$ and $(2, 3)$ moves. For the $(0, 2)$ move we create three new edges connecting an already existing vertex to a newly inserted one. So in this case we cannot create degenerate edges and the constraint is automatically satisfied. Only in the case of a $(2, 3)$ move we

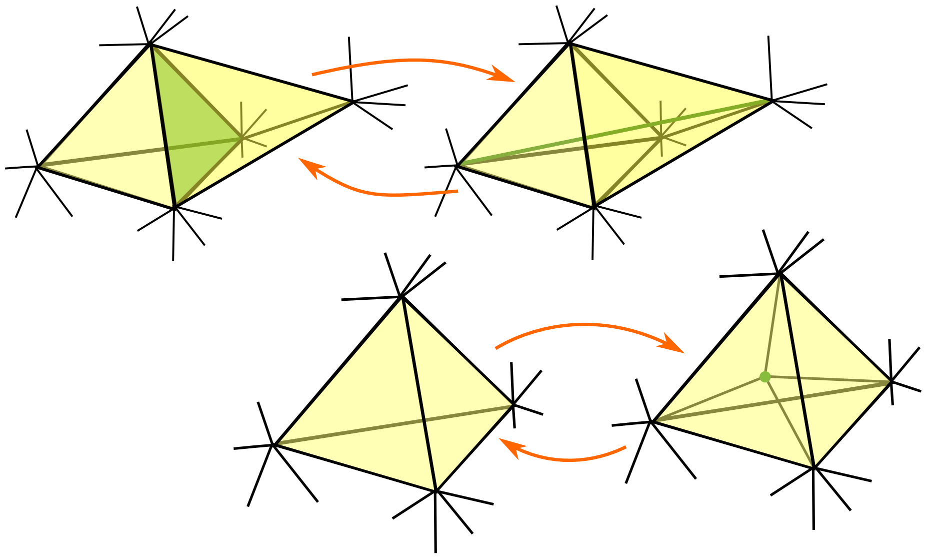

Figure 3.1: Visual representation of $(2, 3)$ move (top) and $(1, 4)$ move (bottom), including inverse moves. Regular moves are shown going from left to right, while inverse moves go from right to left. Adapted from the lecture series "Monte Carlo methods in Dynamical Triangulations" by Timothy Budd.

need to check whether the two vertices at the tips of the complex are distinct. If not, the move is rejected. According to Thorleifsson [12] this does not affect the ergodicity of the moves.

From this point onward we shall assign names to these moves. We will refer to the $(0, 2)$ and $(2, 0)$ moves collectively as *shard* moves, since they involve inserting or removing a *shard* (i.e. a complex of two tetrahedra sharing three triangles). Similarly, we will refer to the $(2, 3)$ and $(3, 2)$ moves as *flip* moves. This is because a triangle is *flipped* into an edge and vice versa. A visual representation of these moves is shown in figure 3.2.

In order to execute shard and flip moves we need to be able to determine whether T has the desired local geometry near a given half-edge. The (0, 2) and (2, 3) moves can always be performed on an arbitrary triangle in T. Note that each triangle has exactly six half-edges corresponding to it. For the (3, 2) move we require an edge of order three, again corresponding to exactly six half-edges. Finally, the (2, 0) move requires two tetrahedra that share three of their triangles, i.e. a shard. The shard has a boundary of two triangles. This once again corresponds to exactly six half-edges when considering only half-edges on the interior of the shard.

Note that these moves will change the volume $N = N_3[T]$ of the triangulation, while we ideally wish to investigate a fixed-volume configuration space as defined in (2.10). This issue will be addressed in section 3.5, but it will not affect the methods discussed here.

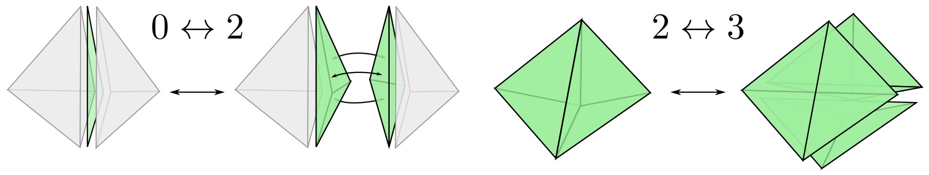

Figure 3.2: Visualization of the *shard* move (left) and *flip* move (right), including inverse moves. Regular moves correspond to arrows pointing to the right, while inverse moves correspond to arrows pointing to the left. Adapted from the Tensor Journal Club talk “A family of triangulated 3-spheres constructed from trees” by Timothy Budd.

3.3 Extension to Triple Trees

Keeping the Forest Alive A logical next step is to try and apply the shard and flip moves on the triple trees model. However, one needs to take care in handling the trees, as various simplices will be created, destroyed and rearranged during these moves. Somehow the end result should still contain exactly the three trees as described in section 2.2. The approach we take is to modify the trees only locally, while imposing that their connectivity inside this local region remains the same. This ensures that no loops or disconnected components are created.

This approach does have an impact on ergodicity, unfortunately. It turns out for some given tree configurations there are some shard and flip moves that are no longer possible due to this constraint. That is, there are cases for which there is no valid tree configuration that preserves local connectivity after performing the move. If this is the case, we are forced to reject the entire move. Hence, at this stage it is far from obvious whether the update rule is still ergodic.

Moving Trees One attempt to regain ergodicity (or at least reduce autocorrelations) is to introduce moves that act solely on the trees (and not on the geometry). The idea is that in cases where a shard or flip move is not possible, we can massage the trees in such a way that they no longer block that specific shard or flip move. However, we know that the triple trees model restricts the geometry itself, so this is not always possible. In fact, currently it is not even clear whether all geometries that *are* allowed are reachable in this manner.

Still, this will not stop us from trying to define an update rule on the trees. We will consider two types of moves: one for b_v, and one for b_d. These moves will each modify both their corresponding tree, and b_m. The third tree remains untouched.

First we consider the problem of a Markov chain for uniform spanning trees on a fixed graph. We can construct an ergodic move for this system by following the algorithm in [10]. We start by uniformly selecting an edge a that is not part of the spanning tree. Adding a to the spanning tree will create a loop. We uniformly select an edge $b \neq a$ from this loop and remove it from the spanning

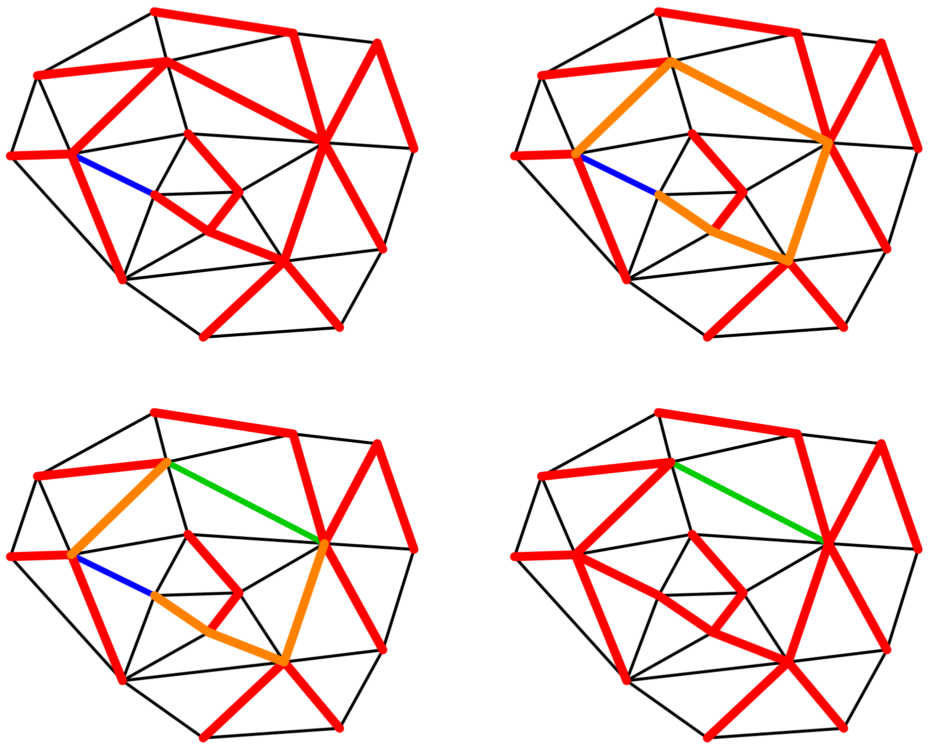
Figure 3.3: Overview of a spanning tree move as described in [10]. Top left: uniformly select an edge a (blue) which is not in the spanning tree (red). Top right: consider the loop (orange) that would be created after including a in the spanning tree. Bottom left: choose an edge $b \neq a$ (green) which is part of this loop, and remove it from the spanning tree. Bottom right: add a to the spanning tree.

tree, thus breaking the loop. A visualization of this procedure is shown in figure 3.3. This procedure can be applied on either the underlying graph (modifying b_v) or on the dual graph (modifying b_d).

Note that this move as it stands can destroy b_m. Whenever an edge is added to b_v, it is by definition removed from b_m, and vice versa. In order to prevent b_m from being cut, we impose that a must be a leaf of b_m before being added to b_v. Additionally, only one of the triangles around b can be in b_m, otherwise a loop would be created in b_m. That is, b will be a leaf of b_m after the move. Similar conditions must hold for the b_d move. We can modify the algorithm presented in [10] to accommodate for these conditions. However, due to these constraints it is not obvious whether the moves are still ergodic.

Ergodicity As we have seen, we are still stuck with two uncertainties regarding the ergodicity of the update rule:

- Are the flip and shard moves that are still allowed in triple trees ergodic on subset of geometries (disregarding trees) allowed by triple trees?
- Can, for a fixed geometry, all valid tree decorations be obtained using only the two types of spanning tree moves described above?

Answering "Yes" to both of these questions would prove ergodicity of the update rule used in this work. However, they are not necessary conditions, and it seems plausible that one or both statements are false, while still satisfying ergodicity on the full set of triple trees configurations. Hence, they are merely mentioned to highlight the difficulties in proving ergodicity.

Another approach is to consider inductive methods. For example, a proof could also consist of showing that starting from an arbitrary triple trees configuration one can always reduce the volume by executing a finite sequence of moves, until reaching a minimal configuration. As the moves are reversible, one can also reach any configuration starting from the minimal one. Therefore, one can always go from one configuration to another by going via the minimal configuration. Unfortunately, as of present there is no clear path for a proof in this direction either.

3.4 Detailed Balance

If the update rule satisfies so-called detailed balance, in addition to ergodicity, it will result in the correct distribution. Suppose that the configuration at Markov chain time t is $T_i \in \mathcal{T}$, and that the configuration at time $t+1$ is $T_j \in \mathcal{T}$. Then the transition probability $P_{ij} = p(T_i \to T_j)$ should satisfy

$$\pi_i P_{ij} \overset{!}{=} \pi_j P_{ji}, \tag{3.3}$$

where $\pi_i = p(T_i)$ is the desired distribution as defined by (2.4). If the move is rejected we have $i = j$ (i.e. there is no transition) and thus we automatically arrive at $P_{ij} = P_{ji}$, as required by (3.3). For some action $S[T]$ we have

$$\pi_i \propto \frac{1}{|\ell[T_i]|!} e^{-S[T_i]}. \tag{3.4}$$

For the tree moves we simply have $\pi_i = \pi_j$, since the volume and vertex count do not change. This immediately leads us to $P_{ij} = P_{ji}$, i.e. a tree move and its inverse should be equally likely. Given the tree move described above this is always satisfied[1], so we need not consider the tree moves any further.

However, the shard and flip moves are somewhat more involved. Without loss of generality, we can assume that $N_3[T_i] < N_3[T_j]$. That is, the transition $T_i \to T_j$ increases N_3. If this is not the case, we simply consider the inverse move

[1]The selection probability for a tree move is $P_{ij} = \frac{1}{N_2(L_{ij}-1)}$, where L_{ij} is the length of the loop that is created by inserting an edge. A move and its inverse correspond to the same loop, so $L_{ij} = L_{ji}$. Since N_2 does not change this implies $P_{ij} = P_{ji}$.

by relabelling i and j. We adopt the notational shortcut $\Delta N_d = N_d[T_j] - N_d[T_i]$ for $d = 0, 3$. Combining this with (3.3) results in

$$\frac{P^+_{ij}(N_3)}{P^-_{ji}(N_3 + \Delta N_3)} \overset{!}{=} \frac{|\ell[T_i]|!}{|\ell[T_j]|!} e^{-\Delta S}, \tag{3.5}$$

where $\Delta S = S[T_j] - S[T_i]$. Our goal is then to construct an update rule such that P_{ij} satisfies (3.5).

Breakdown of probabilities Now we consider a breakdown of the transition probabilities by investigating which steps are taken in performing shard or flip moves. Since the steps are very similar, we consider them simultaneously:

1. Choose according to some probability $s^\pm(N_3)$ whether the next move will increase (+) or decrease (−) the volume N_3.
2. Uniformly sample $\gamma \in \Gamma^\pm[T_i]$ a site from the set of possible sites $\Gamma^\pm[T_i]$.
3. If the particular move is not possible at γ, reject the move.
4. Choose a new local configuration of the trees according to some transition matrix F_{ij}.
5. Accept and perform the move according to a Metropolis-Hastings acceptance probability $A^\pm(N_3)$, otherwise reject it.
6. Uniformly relabel all half-edges after the move is performed[2].

In summary, the probability that a particular move occurs is given by

$$P^\pm_{ij}(N_3) = \frac{s^\pm(N_3) A^\pm(N_3) F_{ij}}{|\Gamma^\pm[T_i]| \, |\ell[T_j]|!}. \tag{3.6}$$

We can choose a site for performing a shard or flip move by uniformly choosing one of $12N_3$ half-edges. We have seen in section 3.2 that for all moves six half-edges will correspond to the same site γ. Thus, effectively the number of unique sites is

$$\left|\Gamma^\pm[T]\right| = \frac{12N_3[T]}{6} \propto N_3[T]. \tag{3.7}$$

Furthermore, we choose a symmetric tree transition matrix such that $F_{ij} = F_{ji}$. The exact form of the transition matrix is described below in more detail.

[2]This step is not actually executed in practice. All operations performed on the triangulation are designed such that they are independent of labels. Hence, relabelling is redundant.

Reducing Rejections Plugging all the above into (3.5) and rearranging results in the requirement

$$\frac{s^+(N_3)A^+(N_3)}{s^-(N_3+\Delta N_3)A^-(N_3+\Delta N_3)} \overset{!}{=} \frac{N_3}{N_3+\Delta N_3}e^{-\Delta S} \equiv r(N_3), \tag{3.8}$$

where we have defined $r(N_3)$ for convenience. There is still some amount of freedom in choosing $s^\pm(N_3)$ and $A^\pm(N_3)$, so we try to choose them in a particular way as to reduce rejections as much as possible. To prevent rejections due to $s^\pm(N_3)$, we impose the constraint

$$s^+(N_3) + s^-(N_3) \overset{!}{=} 1. \tag{3.9}$$

Ideally, we would like to choose $s^\pm(N_3)$ such that $A^\pm(N_3) = 1$. However, due to the combination of (3.8) and (3.9) this is not always possible[3]. Instead, we will choose

$$s^+(N_3) = \frac{1}{z(N_3)}\min\{1, r(N_3)\}, \tag{3.10}$$

$$s^-(N_3) = \frac{1}{z(N_3)}\min\left\{1, \frac{1}{r(N_3-\Delta N_3)}\right\}, \tag{3.11}$$

where we have introduced the normalization

$$z(N_3) = \min\{1, r(N_3)\} + \min\left\{1, \frac{1}{r(N_3-\Delta N_3)}\right\}, \tag{3.12}$$

which ensures that (3.9) is satisfied. In order to satisfy (3.8) we choose the Metropolis-Hastings acceptance probabilities

$$A^\pm(N_3) = \min\left\{1, \frac{z(N_3)}{z(N_3 \pm \Delta N_3)}\right\}. \tag{3.13}$$

Note that if $r(N_3)$ varies mildly, then $A^\pm(N_3)$ is close to unity. This is the case if $\Delta N_3 \ll N_3$ and if $\Delta S \ll 1$. Since we are interested in large N_3, the first condition is reasonably satisfied. The second condition depends on the particular action that is used.

Transition Matrix As mentioned above, we wish to choose the tree transition matrix such that $F_{ij} = F_{ji}$. Recall that i and j each denote the tree configurations on the meta graph before and after the move, respectively. In order to better understand the structure of F_{ij}, we consider the transition graph. This graph encodes all valid transitions, where nodes represent valid configurations, and

[3]There is a minimal volume $N_3^{\min}$ beyond which the triangulation cannot shrink, meaning $s^-(N_3^{\min}) = 0$. By (3.9) this leads to $s^+(N_3^{\min}) = 1$. Requiring $A^\pm(N_3) = 1$ in (3.8) then imposes $s^-(N_3^{\min}+\Delta N_3) = \frac{1}{r(N_3^{\min})}$. For a certain range of κ this will result in probabilities larger than unity. Hence, imposing both (3.9) and $A^\pm(N_3) = 1$ is not possible in general.

edges correspond to valid transitions. This means that $F_{ij} = 0$ if there is no edge between i and j in the transition graph. In order to ensure the triple tree constraint, we disallow edges between nodes for which the local tree connectivity changes. This means the graph will contain many disconnected components, each corresponding to a particular local tree connectivity. There are two types of configurations (initial and final states), and the transition must be between an initial and a final state (in any direction). This means the transition graph must be bipartite, where its parts represent initial and final states. In fact, since equivalence of tree connectivity is a transitive property, each component is a complete bipartite graph.

For the following discussion, we only consider one of these components at a time. We denote the number of initial and final states in this component by n and m, respectively. Without loss of generality we can impose $n \geq m \geq 2$. By symmetry, we see that each allowed transition has an equal probability, in both directions. This means that F_{ij} depends on a single free parameter p, denoting the transition probability for any non-identity transition. This means that $F_{ij} = p$ if $i \neq j$ and if $i \rightarrow j$ is allowed. The values for $i = j$ are chosen such that probability is conserved.

We can tune p in order to optimize the mixing of the Markov chain. In particular, we choose to minimize the absolute value of the second-largest eigenvalue of a Markov chain on a complete bipartite graph. According to [3] this, for a complete bipartite graph, corresponds to

$$p^* = \min\left\{\frac{1}{n}, \frac{2}{n+2m}\right\}. \tag{3.14}$$

Uniformly choosing a final state results in a selection probability of $\frac{1}{m}$. In order to match (3.14) we introduce an acceptance probability $A = mp^*$. In conclusion, this procedure provides an efficient method for choosing a local configuration of the trees.

There are only a finite number of local configurations possible for each type of move. Hence, to save on computation time we compute the entire matrix F_{ij} only once at the start of the simulation. By doing this it becomes very easy to find the final states to which the initial state can transition. Note that also the number of initial and final configurations corresponding to the same local connectivity is easily obtained. These values are required to determine the acceptance probability in (3.14).

3.5 Controlling the Volume

Quadratic Potential Term We have seen in section 3.2 that the volume N of the triangulations will fluctuate during the Markov chain. In fact, depending on our choice of λ the volume will either keep growing indefinitely, or the geometry will collapse and fluctuate near zero. Additionally, we have seen in section 2.3 that we would like to investigate expectation values as a function of volume N. In order to investigate a particular volume N, we should only consider measurements matching this volume, throwing away everything else. However, considering the aforementioned behaviour this will be unreasonably inefficient.

To come up with a solution to this, first notice that we can add an arbitrary function of $N_3[T]$ to the action without changing $\langle \mathcal{O} \rangle_N$. Suppose our new action is of the form

$$S'[T] = S[T] + V(N_3[T]), \tag{3.15}$$

where we have added a potential term $V(N_3[T])$ to the action. The fixed-volume expectation values do not change:

$$\begin{aligned} \langle \mathcal{O} \rangle'_N &= \frac{1}{Z'_N} \sum_{T \in \mathcal{T}_N} \mathcal{O}[T] \frac{e^{-S'[T]}}{|\ell[T]|!} && (3.16) \\ &= \frac{1}{Z_N} \sum_{T \in \mathcal{T}_N} \mathcal{O}[T] \frac{e^{-S[T]}}{|\ell[T]|!} && (3.17) \\ &= \langle \mathcal{O} \rangle_N . && (3.18) \end{aligned}$$

This is because the potential term can be absorbed into the normalization:

$$Z'_N = e^{-V(N)} Z_N. \tag{3.19}$$

Thus, as long as we only consider samples of the correct volume, adding a potential $V(N)$ to the action will not change the expectation values $\langle \mathcal{O} \rangle_N$.

To maximize the number of samples that match the desired volume, we should introduce a potential $V(N)$ for which we can control the location of its minimum. In particular, we choose a quadratic potential

$$V(N) = \frac{1}{2} \left(\frac{N - \bar{N}}{\sigma} \right)^2, \tag{3.20}$$

where $\bar{N}$ is the volume at which $V(N)$ is minimal, and σ will describe the width of the resulting peak in the volume distribution. In the actual simulation we will set σ at a reasonably low value such that the volume distribution is strongly peaked around the desired value. We will then keep all measurements, even those with a slightly wrong volume, at the cost of introducing a small measurement error. This will however yield drastically more measurements, thus decreasing the overall statistical error.

Critical Lambda Adding a potential term to the action does force a peak to appear in the volume histogram, however it may not necessarily occur at the desired $\bar{N}$. To see this, we have to consider the large N behaviour of the canonical partition function. We assume that it grows exponentially,

$$Z_N = \sum_{T \in \mathcal{T}_N} \frac{e^{\kappa N_0[T]}}{|\ell[T]|!} \overset{N \to \infty}{\sim} C(\kappa) e^{\lambda_c(\kappa) N}, \tag{3.21}$$

where $\lambda_c(\kappa)$ is the critical cosmological constant. Combining this with (2.11) results in

$$Z \overset{N\to\infty}{\sim} \sum_{N=0}^{\infty} \exp\left[(\lambda_c(\kappa) - \lambda)N - \frac{1}{2}\left(\frac{N-\bar{N}}{\sigma}\right)^2\right]. \tag{3.22}$$

Therefore, a peak will occur at

$$N_{\text{max}} = \bar{N} + \sigma^2(\lambda_c(\kappa) - \lambda). \tag{3.23}$$

In other words, the peak will shift away from $\bar{N}$, unless we choose $\lambda = \lambda_c(\kappa)$. Suppose that we run the simulation with a λ reasonably close to the critical value. Then we can use the observed offset of this peak to determine $\lambda_c(\kappa)$. Rearranging (3.23) results in

$$\lambda_c(\kappa) = \lambda + \frac{N_{\text{max}} - \bar{N}}{\sigma^2}. \tag{3.24}$$

Finally, we will use this newly computed λ_c to run the actual simulation.

3.6 Implementation

The model described above has been implemented as a computer simulation written in the Rust programming language. The actual code used in this work can be found in the **`src`** directory on the corresponding GitHub repository[4]. What follows is an overview of the core functionality of the simulation code.

Data Structures The core functionalities of the simulation are firstly creating triangulations using a Markov chain, and secondly measuring observables on these triangulations. The current state of the Markov chain is kept in a **`Triangulation`** structure which is modified in each step of the chain. Meanwhile, measurements are performed on the fly and are directly written to a file.

The main data structure keeping the current state of the triangulation is defined as follows:

```
struct Triangulation {
    vertices: Shelf<Vertex>,
    edges: Shelf<Edge>,
    triangles: Shelf<Triangle>,
    tets: Shelf<Tetrahedron>,
    forest: Shelf<Node<Simplex>>,
    tet_bag: Bag<Tetrahedron>,
    middle_edges: Bag<Edge>,
    middle_triangles: Bag<Triangle>,
}
```

[4]https://github.com/TomGerstel/dynamical-triangulations-3d

Here the **`Shelf`** and **`Bag`** types are custom collection data types. The **`Shelf`** is a data structure that can store objects with efficient access, deletion and insertion. It does all of this without moving objects around, thus ensuring the object labels remain valid. This means that gaps can occur when an object is removed, analogous to taking a book from a shelf. The **`Bag`** is a structure that keeps track of labels referring to objects in a **`Shelf`**. It can be used to uniformly sample from a (subset of) a **`Shelf`**. These two collection data types are described in more detail in appendix A.1.

Notice that the **`Triangulation`** contains a **`Shelf`** for each of the four types of simplices: **`Vertex`**, **`Edge`**, **`Triangle`** and **`Tetrahedron`**. Each of these contains a label referring to a **`Node<Simplex>`** contained in the forest **`Shelf`**. A collection of **`Node`** objects will collectively describe a link-cut tree (LCT). The LCT is an efficient data structure that can keep track of, modify and extract information from a collection of trees (i.e. a forest). In this case, we use an LCT of **`Simplex`** objects to keep track of the three trees. It is described in more detail in appendix A.2.

Finally, notice that all tetrahedra are stored in a **`Bag`**. This is done in order to be able to easily and uniformly sample a **`Tetrahedron`** (or more specifically, a **`Mouse`**, see the next paragraph) from the triangulation. This is a requirement for being able to perform the flip and shard moves or for calculating certain observables. Furthermore, we keep track of the **`Triangle`**s and **`Edge`**s in b_m using two **`Bag`**s. This is done in order to more easily sample **`Triangle`**s and **`Edge`**s for the tree moves.

Mice It would be convenient to have some kind of abstraction for navigating the geometry. In particular, we need to be able to identify certain local geometric structures on the fly so that we can determine whether it is possible to perform a particular move. Moreover, when we decide to make a move we need to be able to edit the local geometry. The approach we take is to keep track of a certain half-edge, which we dub a **`Mouse`**, from which we can always jump to another half-edge using one of three distinct transformations.

These three transformations are a ”next” transformation n and two ”adjacent” transformations a_2 and a_3. By applying n to the mouse we can move to the next half-edge within the same triangle. Furthermore, a_2 will move the mouse to the other half-edge within the same tetrahedron that is also part of the same edge. Finally, a_3 will move the mouse to the same position on the other side of the triangle, thus moving to a new tetrahedron. A visual overview of n, a_2 and a_3 is shown in figure 3.4.

Since a_3 connects adjacent tetrahedra, it can be used to store gluing relations. In particular, at each half-edge i in each **`Tetrahedron`** we store a mouse referring to $a_3(i)$. Because we are only using tetrahedra as building blocks this is all the information we need to reconstruct the geometry. Furthermore, each **`Vertex`**, **`Edge`** and **`Triangle`** contains a **`Mouse`** referring to one of the half-edges that is incident to the simplex. This is required for being able to go from the simplices to the geometry which they form a part of.

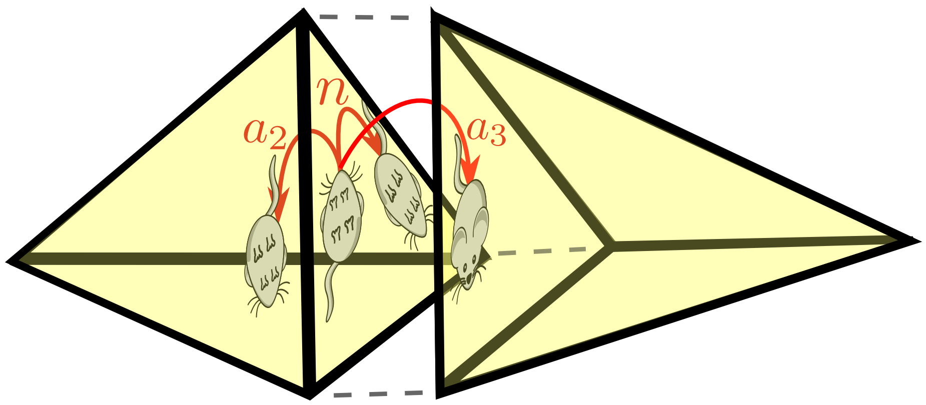

Figure 3.4: Visualization of the n, a_2 and a_3 transformations that can be applied to a mouse. Adapted from the lecture series ”Monte Carlo methods in Dynamical Triangulations” by Timothy Budd.

Chapter 4

Data Analysis

4.1 Observables

In previous chapters we have discussed how to compute expectation values of observables $\langle \mathcal{O} \rangle$. However, it is not yet clear what these observables actually are. What follows is an overview of various observables that can be studied in the triple trees model.

Simplex Counts The most straightforward class of observables we can consider is simply counting the number of simplices of each type. Since our model has $d = 3$ we can count four types of simplices on T. This results in the four observables $N_d[T]$, with $d = 0, 1, 2, 3$ for vertices, edges, triangles and tetrahedra, respectively. We shall omit the argument T for now.

However, not all of these numbers are independent. Each triangle is in-between two tetrahedra, and each tetrahedron has four triangles at its boundary. This gives us $N_2 = 2N_3$. Furthermore, since we are only considering S^3 topologies, the Euler characteristic is fixed, resulting in the constraint

$$\chi = N_0 - N_1 + N_2 - N_3 = 0. \tag{4.1}$$

Since we have four variables and two constraints we end up with two independent variables. We take these to be the volume $N = N_3$ and the vertex count N_0.

We could also look at how many simplices of each type are part of each of the three trees. To denote which type of tree we are considering we will include a superscript v, d or m, corresponding to the vertex, dual and middle trees, respectively. For instance, $N_2^{(m)}$ denotes the number of triangles in the middle tree. Since b_v and b_d are spanning trees, we know that $N_0^{(v)} = N_0$ and $N_3^{(d)} = N_3$. Furthermore, a tree always has exactly one more node than it has edges, resulting in $N_1^{(v)} = N_0 - 1$ and $N_2^{(d)} = N_3 - 1$. Finally, $N_1^{(m)}$ and $N_2^{(m)}$ can also be expressed in terms of N_0 and N_3 using the considerations above. An overview of all simplex counts is shown in table 4.1. Crucially, all simplex

	$d = 0$	$d = 1$	$d = 2$	$d = 3$
b_v	N_0	$N_0 - 1$	0	0
b_m	0	$N_3 + 1$	$N_3 + 1$	0
b_d	0	0	$N_3 - 1$	N_3
Total	N_0	$N_0 + N_3$	$2N_3$	N_3

Table 4.1: Overview of simplex counts for all three trees and their sum. All simplex counts can be expressed in terms of N_0 and N_3.

counts can be expressed in terms of only N_0 and N_3, meaning these are the only relevant observables in this category.

Distances Another very important class of observables, especially in the context of metric spaces, is distances. The following discussion applies to any type of graph, as long as there are no disconnected components. In particular, we can consider the underlying graph of T, the dual graph and any of the three trees. On a graph G, the distance $\mathrm{d}(x, y)$ between two nodes x and y is simply the number of edges in the shortest path connecting x to y.

Just looking at the distance between two nodes does not result in a coordinate independent observable. Instead, we can look at the distribution of distances between all pairs of nodes in some graph G. The quantity that describes this is the *distance profile* $\rho_G(r)$, defined as

$$\rho_G(r) = \frac{1}{N_G} \sum_{x \in G} \sum_{y \in G} \theta(\mathrm{d}(x, y) = r), \tag{4.2}$$

where N_G is the number of nodes in G. Of course, we wish to compute the expectation value of (4.2). However, actually calculating (4.2) is rather time-consuming, since it involves a double summation over all vertices in G. Instead, we choose to randomly sample a vertex $X \in G$ and only sum over vertex pairs containing X. Effectively we are then also averaging over all vertices. We can rewrite the expectation value of (4.2) to

$$\langle \rho(r) \rangle_G = \left\langle \sum_{y \in G} \theta(\mathrm{d}(X, y) = r) \right\rangle_{(G,X)}, \tag{4.3}$$

thus validating that the computed values actually correspond to the desired quantities. In practice N_G can fluctuate somewhat, but this does not affect the equality. This can be shown by first writing out the average over X, and then over G.

Simplex Degrees Another class of quantities we can investigate are simplex degrees. That is, the number of sub- or supersimplices of a certain type that are part of the given simplex σ or in which the given simplex σ is contained. These quantities may not be as physically relevant as distances, but they can provide

$\deg_p \sigma_q$	$q = 0$	$q = 1$	$q = 2$	$q = 3$
$p = 0$	1	2	3	4
$p = 1$	d_v	1	3	6
$p = 2$	$3(d_v - 2)$	d_e	1	4
$p = 3$	$2(d_v - 2)$	d_e	2	1

Table 4.2: Overview of all combinations of simplex degrees in $d = 3$.

useful insights on a potential phase transition. We denote the number of sub- or supersimplices of dimension p related to a simplex $\sigma_q \in \Sigma_q$ of dimension q by $\deg_p \sigma_q$.

If $p < q$ we are considering subsimplices, and the number of these is constant for any combination of p and q. If $p = q$ we are in a degenerate case and the degree is always one. Finally, for $p > q$ (i.e. when counting supersimplices) we find only two independent values. We refer to these as the vertex degree $d_v(\sigma_0)$ (the number of vertices connected to σ_0) and the edge degree $d_e(\sigma_1)$ (the number of triangles, or equivalently tetrahedra, connected to σ_1). Note these can take on any integer value such that $d_v \geq 3$ and $d_e \geq 2$, given the (non)-degeneracy of the configuration space considered in this work. An overview of all degree types is shown in table 4.2.

However, the values $d_v(\sigma_0)$ and $d_e(\sigma_1)$ only provide local information, as they depend on the particular simplices σ_0 and σ_1. As an alternative, we can look at the distribution of d_v and d_e for a given geometry T. In particular, for d_v we define

$$\beta_T^{(v)}(s) = \sum_{\sigma_0 \in \Sigma_0[T]} \theta(d_v(\sigma_0) = s), \tag{4.4}$$

where the summation is over the set of all vertices $\Sigma_0[T]$ in the configuration T. Analogously, we define $\beta_T^{(e)}(s)$ to be the distribution of edge degrees on a given T.

In addition to d_v and d_e we can also consider degrees within any one of the three trees. In the case of the spanning trees b_v and b_d, only the degrees of the vertices and tetrahedra, respectively, are variable. We denote these quantities by m_0 and m_3, respectively. For b_m, both the degrees of the edges and the triangles can vary. These are denoted by m_1 and m_2, respectively. In summary:

- m_0 = number of edges connected to a vertex in b_v.
- m_1 = number of triangles connected to an edge in b_m.
- m_2 = number of edges connected to a triangle in b_m.
- m_3 = number of triangles connected to a tetrahedron in b_d.

Note that $m_2 \in \{1, 2, 3\}$ and $m_3 \in \{1, 2, 3, 4\}$. Furthermore, m_1 and m_2 are only defined if the corresponding simplex is in b_m. Once again, we can define distributions of these degrees completely analogous to (4.4). We denote these distributions by $\mu_T^{(d)}(s)$, for $d = 0, 1, 2, 3$.

4.2 Phase Transitions

In $d = 3$ DT (and in the triple trees model) we have a free parameter κ that couples to the number of vertices $N_0[T]$. It is this parameter in which we look for phase transitions. We say that there is a phase transition if the partition function $Z_N(\kappa)$ of (2.12) is non-analytic at some κ_c in the limit $N \to \infty$. If this is the case, the critical cosmological constant

$$\lambda_c(\kappa) = \lim_{N\to\infty} \frac{\ln Z_N(\kappa)}{N}, \tag{4.5}$$

will also be non-analytic at κ_c. In particular, if $\lambda_c'(\kappa)$ is discontinuous (i.e. when $\lambda_c(\kappa)$ is non-differentiable) at κ_c, the transition is discontinuous. Otherwise, it is continuous.

Vertex Count The parameter κ couples directly to the number of vertices $N_0[T]$. Thus, when searching for a phase transition in κ it makes sense to first look at the behaviour of $\langle N_0\rangle_N$. In this case we can directly relate the first two moments of the observable to derivatives of the partition function (2.12), and therefore to derivatives of $\lambda_c(\kappa)$:

$$\langle N_0\rangle_N = \frac{\partial \ln Z_N}{\partial \kappa} \overset{N\to\infty}{\sim} N\lambda_c'(\kappa), \tag{4.6}$$

$$\left\langle N_0^2\right\rangle_N - \langle N_0\rangle_N^2 = \frac{\partial^2 \ln Z_N}{\partial \kappa^2} \overset{N\to\infty}{\sim} N\lambda_c''(\kappa). \tag{4.7}$$

Thus, seeing a discontinuity in

$$\lambda_c'(\kappa) = \lim_{N\to\infty} \frac{\langle N_0\rangle_N}{N}, \tag{4.8}$$

or a divergence in

$$\lambda_c''(\kappa) = \lim_{N\to\infty} \frac{\left\langle N_0^2\right\rangle_N - \langle N_0\rangle_N^2}{N}, \tag{4.9}$$

could be considered evidence in favour of a discontinuous transition.

Other Order Parameters However, at finite N evidence for phase transitions does not always appear in every observable. To have a better chance at catching a glimpse of the real order of the transition we must cast a bigger net by considering different observables. One caveat is then that other observables do not relate to derivatives of $\lambda_c(\kappa)$ as directly as N_0 does.

Therefore, we must use complementary methods to determine the order of the phase. It has been observed that near a phase transition the susceptibility (i.e. variance) of an observable can form a peak. The position and height of this peak depend on N. It was shown in [5] that, for a particular class of spin systems, the position of this peak changes linearly with the inverse of the system size. Thus, in the context of our model this would mean

$$\kappa_{\max} = \kappa_c + N^{-1}, \tag{4.10}$$

where the maxima occur at $\kappa = \kappa_{\max}$. Observing such a dependence could then be considered evidence for a discontinuous phase transition.

4.3 Error Analysis

All errors presented in this work are 1σ symmetric confidence intervals. Most errors in this work are estimated using the bootstrapping method. This method can be used to estimate a plethora of statistical estimators by resampling (with replacement) the original data. In this work it is mainly used to estimate the sample standard deviation σ_x. For n random samples x_i we use

$$\sigma_x = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(x_i - \bar{x})^2}, \tag{4.11}$$

where $\bar{x}$ is the sample mean

$$\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i. \tag{4.12}$$

Note that both σ_x and $\bar{x}$ are statistical estimators, meaning their exact values are random and depend on the particular sample.

Dealing with Correlations As briefly mentioned in section 3.1 we do not know a priori how correlated consecutive configurations T_i in the Markov chain are. In order to investigate the correlation quantitatively, we need an observable $\mathcal{O}[T]$. In particular, we look at the autocorrelation function

$$\rho(\tau) = \frac{\langle \mathcal{O}[T_i]\mathcal{O}[T_{i+\tau}]\rangle - \langle \mathcal{O}[T]\rangle^2}{\langle (\mathcal{O}[T])^2\rangle - \langle \mathcal{O}[T]\rangle^2}. \tag{4.13}$$

Generally speaking we observe that $\rho(\tau)$ follows an exponential decay:

$$\rho(\tau) = e^{-\tau/\tau_c}, \tag{4.14}$$

where τ_c is the correlation time. Note that τ_c can heavily depend on the type of observable used.

The error estimate of (4.11) assumes that all samples x_i are independent. However, if $\tau_c > 0$ we know that this is not the case. Due to correlations, we can say that we effectively have fewer independent (i.e. uncorrelated) samples. Thus, in order to correct for correlation effects we use the heuristic

$$\sigma_x' = \sqrt{\max\left\{1, \frac{\tau_c}{\tau_m}\right\}}\sigma_x, \tag{4.15}$$

where τ_m is the Markov chain time between taking measurements, in the same units as τ_c. This correction factor is motivated by the fact that the estimator of the standard error scales with the inverse square root of the number of samples. Thus, without this extra factor we would have underestimated the error.

Chapter 5

Results

5.1 Vertex Count

As mentioned in section 4.2 it is most natural to first investigate the expected vertex count $\langle N_0 \rangle$. This is because N_0 is the variable conjugate to our free parameter κ. In accordance with (4.8) we divide $\langle N_0 \rangle_N$ by N. The dependence of $\langle N_0 \rangle / N$ on κ is shown in figure 5.1. It is clear that, certainly for large volumes, this dependence quickly collapses to a sigmoid-like curve that interpolates between 0 and 0.5. Furthermore, the response of the vertex count to κ is as expected. Negative κ suppresses geometries with many vertices, while positive κ encourages these. We do not observe a clear discontinuity or divergence in this observable.

In order to further investigate a possible transition we consider the standard deviation σ of the vertex count, as shown in figure 5.2. We normalize with $\sqrt{N}$, in accordance with (4.9). It is clear that the distribution of N_0 increases in width somewhere near $\kappa = 0$. However, no discontinuous or diverging behaviour is observed for $N \to \infty$. Thus, in order to spot a potential phase transition we must look at other observables.

5.2 Maximal Vertex and Edge Degree

Another observable that could reveal signs of a phase transition is the maximal vertex degree $\max d_v$. More precisely, it is given by

$$(\max d_v)[T] = \max_{\sigma_0 \in \Sigma_0[T]} d_v(\sigma_0), \tag{5.1}$$

i.e. it takes on the value of the vertex with the highest coordination number. Its expectation value is shown in figure 5.3. Firstly, it is clear that for very positive κ effectively all vertices have a low degree. Conversely, for very negative κ there is typically at least one vertex with a very high degree. Also note the volume normalization in figure 5.3. Since we do not have a clear theoretical value for

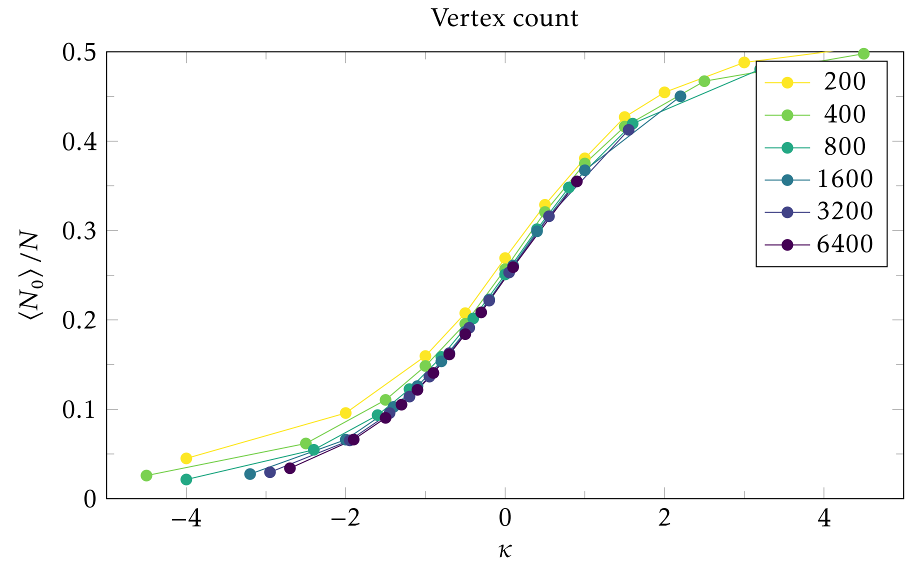

Figure 5.1: Expected vertex count $\langle N_0 \rangle$, normalized by N, as a function of κ.

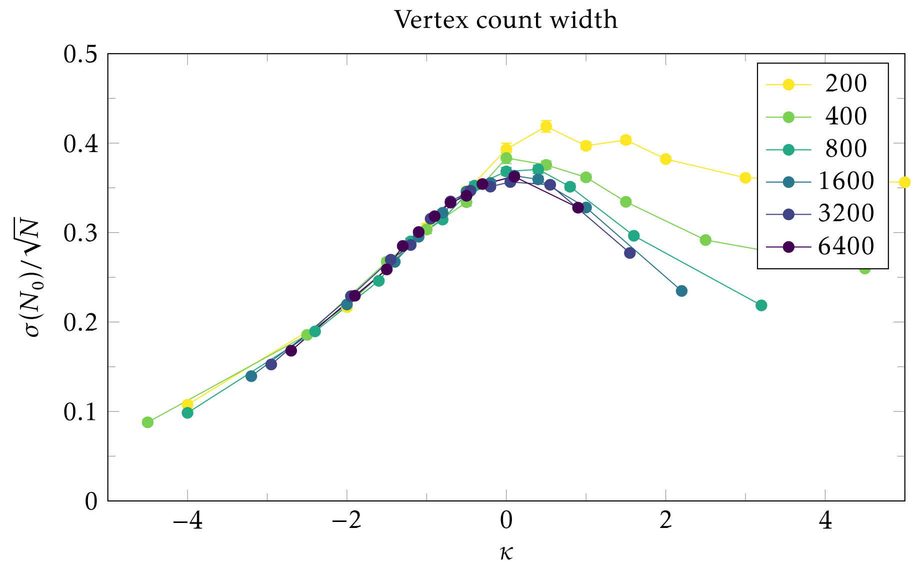

Figure 5.2: Vertex count distribution width as a function of κ.

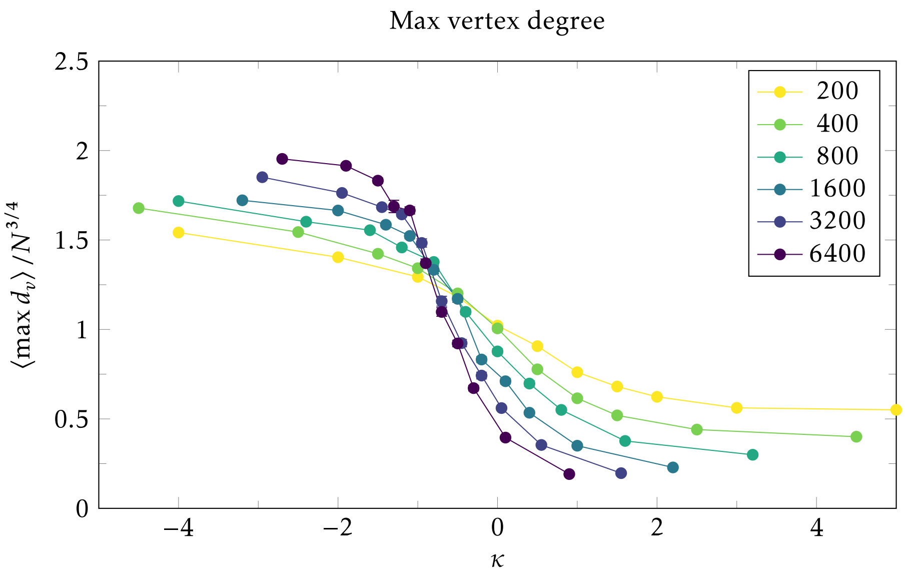

Figure 5.3: Expectation value of the maximal vertex degree as a function of κ.

the scaling exponent, we must set it by hand. The exponent 3/4 that is used seems to be too small for very negative κ and too large for very positive κ, at least judging by eye. This hints at the possibility of different scaling exponents in different regimes, i.e. different phases.

The emergence of a phase transition becomes especially clear when looking at the width of the maximal vertex degree distribution, shown in figure 5.4. Here we clearly observe a peak taking shape, suggesting a divergence in the $N \to \infty$ limit. We can use the position of these peaks as a function of volume to quantify some properties of the phase transition. The peak positions are estimated numerically by finding the maximum of the quadratic function going through the highest data point and its two neighbours. We fit the peak coordinates (κ, N) of $\sigma(\max d_v)$ to the power law

$$\kappa - \kappa_c \propto N^{-\nu}, \tag{5.2}$$

where κ_c is the critical value at which the phase transition occurs in the $N \to \infty$ limit, and ν is a scaling exponent. The resulting curve and asymptote is shown in figure 5.5. The corresponding estimates for the fit parameters are

$$\kappa_c = -1.5 \pm 0.4, \quad \nu = 0.4 \pm 0.1. \tag{5.3}$$

We seem to be able to rule out $\nu = 1$ with a confidence level of at least 5σ. This would suggest that the transition is continuous, at least according to (4.10). However, this result should be taken with a grain of salt, as further discussed in section 6.1.

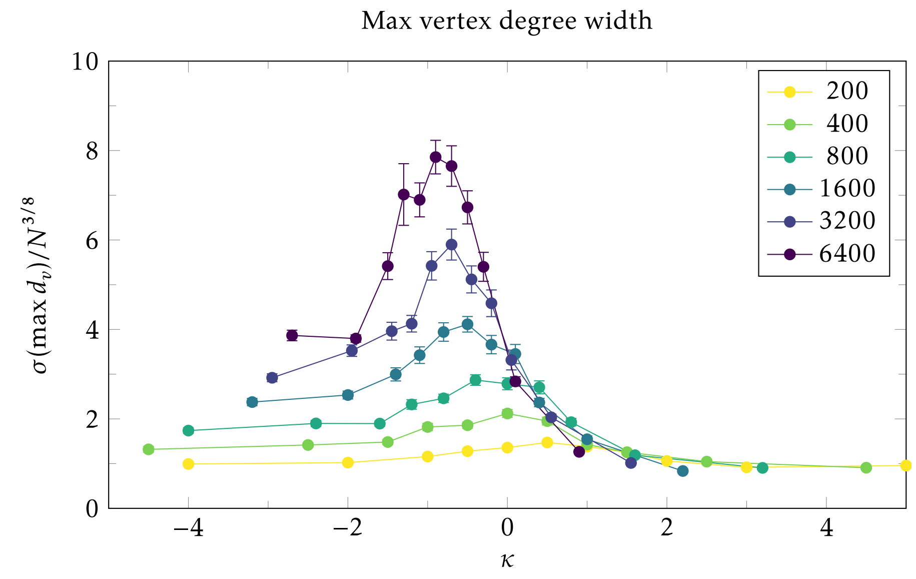

Figure 5.4: Max vertex degree distribution width as a function of κ.

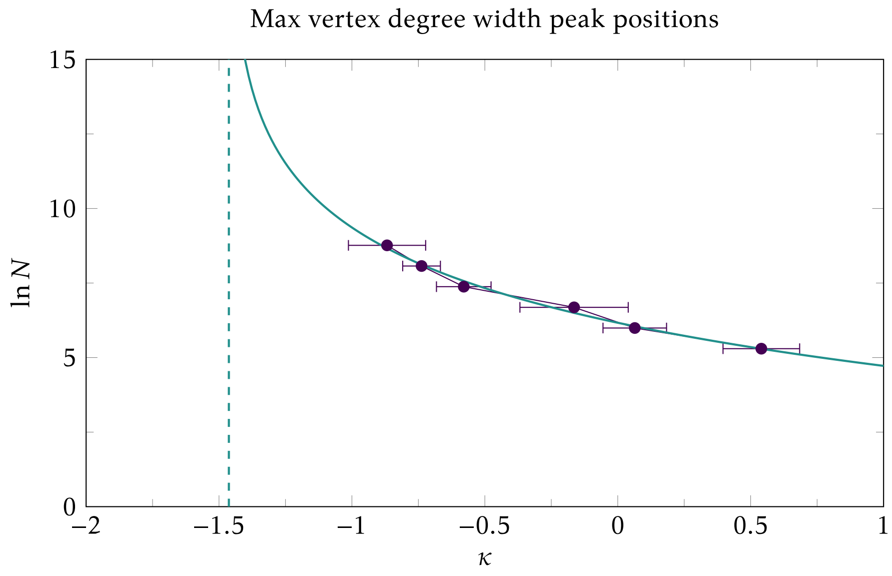

Figure 5.5: Peak positions $\kappa(N)$ of the maximal vertex degree width.

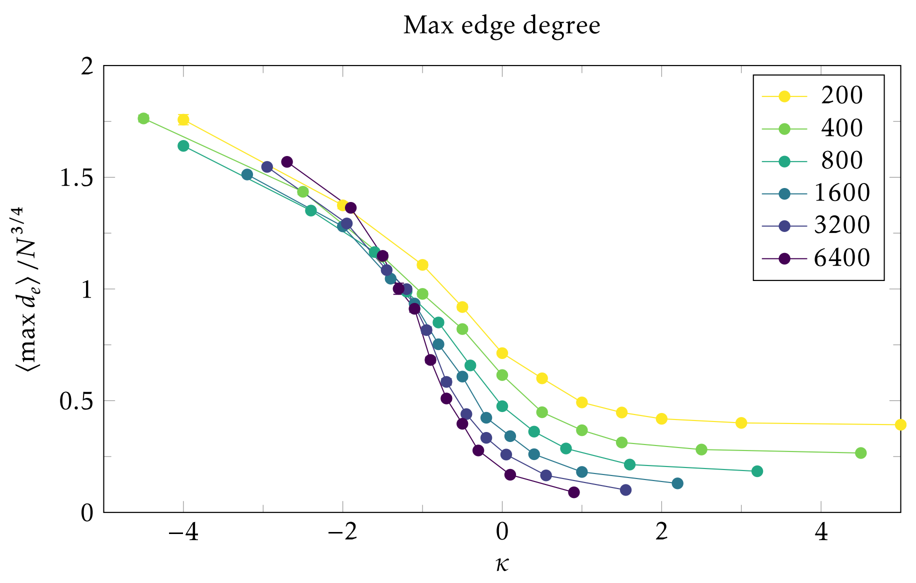

Figure 5.6: Expectation value of the maximal edge degree as a function of κ.

Max Edge Degree An alternative to the maximal vertex degree $\max d_v$ is the maximal edge degree $\max d_e$. If an edge has a very high degree, then it is reasonable to assume that its two vertices also have a high degree. Therefore, we expect $\max d_e$ to follow patterns similar to those of $\max d_v$. Similarly to (5.1) we define

$$(\max d_e)[T] = \max_{\sigma_1 \in \Sigma_1[T]} d_e(\sigma_1). \tag{5.4}$$

The dependence of $\max d_e$ on N and κ is shown in figure 5.6. As expected $\max d_e$ is large for $\kappa \ll \kappa_c$ and small for $\kappa \gg \kappa_c$. However, in contrast to the curves in figure 5.3 we do not observe a clear plateau emerging in the $\kappa \ll \kappa_c$ regime.

The behaviour of $\sigma(\max d_e)$ is shown in figure 5.7. In this case we can also see a peak emerging, though less clearly than for the vertex degree. Additionally, the background seems to be more complicated. Using exclusively this data set it seems difficult to perform a quantitative analysis such as a fit to (5.2), since we would only reasonably be able to extract 3 peak positions. Still, the data seems to be consistent with the value for κ_c as determined in (5.3).

5.3 Maximal Dual Distance Scaling

Finally, we look at the maximal dual distance. More precisely, this is the maximum distance on the dual graph between a given randomly sampled point X and any other point. More formally,

$$(\max \mathrm{d}_D)[T] = \max_{y \in D[T]} \mathrm{d}_{D[T]}(X, y), \tag{5.5}$$

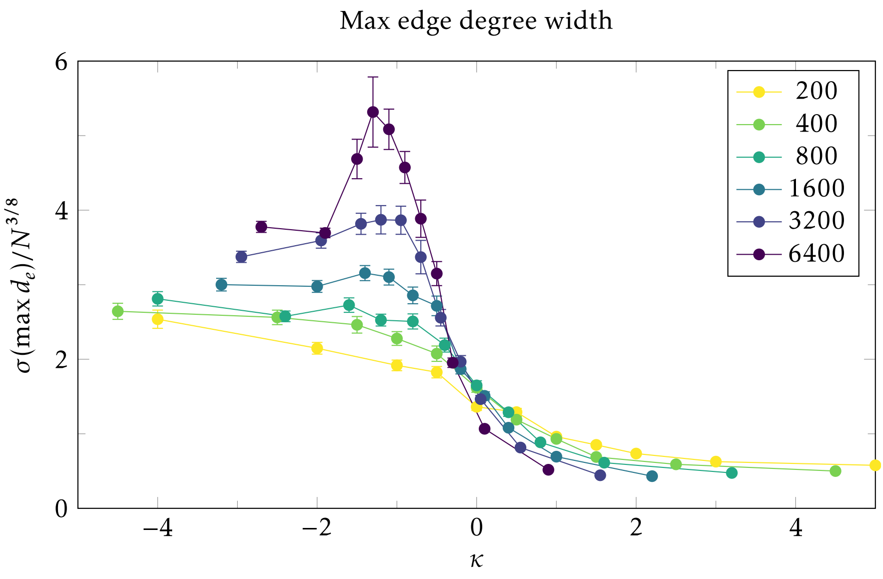

Figure 5.7: Max edge degree distribution width as a function of κ.

where $D[T]$ denotes the set of nodes in the dual graph of T (i.e., the tetrahedra). Note that since X is chosen randomly, expectation values are averaged over both T and X. The dependence on κ and N of $\langle \max \mathrm{d}_D \rangle$ is shown in figure 5.8. Again we use the normalization $N^{3/4}$. For $\kappa \ll \kappa_c$ the value of $\langle \max \mathrm{d}_D \rangle$ seems to grow faster than $N^{3/4}$, suggesting a scaling exponent larger than 3/4 in this regime.

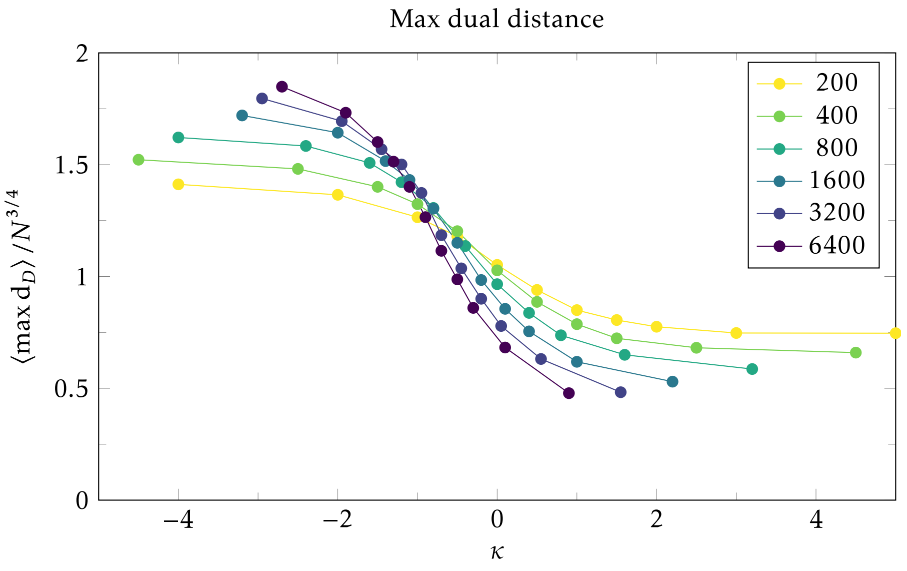

Figure 5.8: Maximal dual distance as a function of κ.

Chapter 6

Conclusions, Discussion and Outlook

6.1 Conclusions

We have numerically investigated the triple trees model using a Markov chain Monte Carlo simulation written in Rust. We have strong evidence for the existence of a phase transition. The $\kappa > \kappa_c$ phase exhibits behaviour similar to the corresponding tree-like ("branched polymer") phase in DT. However, the $\kappa < \kappa_c$ phase seems to show both similarities and differences with the so-called crumpled phase of regular DT. Two examples of configurations in these regimes are shown in figure 6.1.

In particular, we have seen in figure 5.2 that the width of the vertex count $\sigma(N_0)$ does not seem to express a diverging behaviour, as it does in DT [12]. That is, the phase transition seems to be less extreme in some sense. On the other hand, we do see a clear transition in the maximal vertex degree, and to a lesser extent in the maximal edge degree. So far, we were unable to find strong evidence for or against the continuity of the transition. However, this also means that a continuous transition is still within the realm of possibilities.

Among the observables we have considered, the vertex count N_0 must be the most elementary. Its response to variations in κ is as expected. However, from the data gathered in this work it does not show clear evidence for a transition. The transition is most clearly observed in the maximal vertex degree $\max d_v$, and to a lesser extent in the maximal edge degree $\max d_e$. In these observables a peak emerges in the width of their distributions, in the limit $N \to \infty$.

Using the peak positions of the width of $\max d_v$ we estimate the critical coupling to be $\kappa_c = -1.5 \pm 0.4$. Furthermore, we have found a finite-size scaling of $\nu = 0.4 \pm 0.1$. We seem to be able to exclude $\nu = 1$ with high confidence, thus suggesting the transition is continuous. However, this should be taken with a grain of salt as an exponent $\nu \approx 0.3$ has been obtained [12] for regular DT, where the transition is known to be discontinuous.

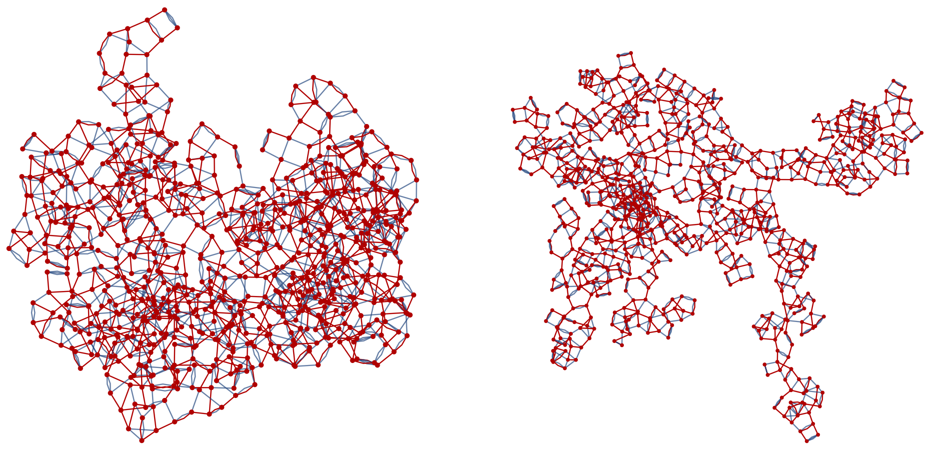

Figure 6.1: Dual graphs of two triangulations with $N = 800$ for different values of κ. The dual tree b_d is shown in red. Left: $\kappa = -4$; Right: $\kappa = 3.2$.

Finally, we see that several rescaling exponents (as used for normalization) seem to depend on κ. Taking $\nu = 1/2$ for the tree-like phase seems to be suitable for various observables considered in this work. In particular, for the maximal dual distance, but also for the maximal edge and vertex degrees. However, in the $\kappa < \kappa_c$ phase the proper rescaling exponent for the maximal dual distance seems to be between $\nu = 3/4$ and $\nu = 1$. This range is purely based on visual intuition, so no quantitative claims can be made here. The case $\nu = 1$ would result in a quite peculiar situation. In that scenario the "size" of the geometries would grow linearly with the number of building blocks. This suggests that the geometry might arrange itself into an effectively one-dimensional configuration, such as a line or perhaps a circle.

6.2 Discussion and Outlook

One way to extract more information out of the simulation is to either improve its efficiency or to let it run significantly longer. By gathering more data we can gain better statistics on the existing data, as well as extend the dataset to larger volumes. In particular the maximal edge degree $\max d_e$ could benefit from such an improvement. With more data, we could estimate κ_c and a finite-size scaling exponent to compare with the results obtained for $\max d_v$.

Another interesting change in perspective would be looking at observables as a function of N, while keeping κ fixed. With such a setup we could properly investigate properties that depend on the finite-size scaling of the model. In particular, we could make estimates for the Hausdorff dimension d_H in various regimes, e.g. deep into both phases and near the transition. In particular, this could illuminate the peculiar distance scaling that is observed for the $\kappa < \kappa_c$ phase, as there is a possibility that this is just a finite size artefact. Other scaling exponents can also be investigated as a function of κ. Furthermore, when such data is available we could investigate the cumulant presented in [5] to further investigate the order of the transition.

On the more theoretical side there is still the open question whether the Markov chain used in this work is actually ergodic. To reconcile this, we have compared vertex count measurements with the numerical model used in [4]. This resulted in an agreement within statistical error. We therefore have some empirical evidence to believe that our Markov chain is ergodic, but a formal proof would be ideal. Even still, it might be worth it to include new moves in the Markov chain, simply to decrease autocorrelations. In particular, creating or breaking up high-degree vertices could be beneficial in this sense.

Finally, we can consider possible extensions to the triple trees model. One option that is actively being worked on is to allow b_m to have loops and more than a single component. However, instead of just extending the configuration space we can assign weights to these objects. This would result in a three-dimensional phase space (excluding λ), thus allowing for a potentially more complicated phase structure.

Another option is to extend the model to four dimensions, by introducing a fourth tree. Such a model could consist of two spanning trees for the zero- and four-dimensional simplices, and two "middle" trees for the remaining simplices. Investigating such a model might as of present be somewhat premature, as there is still much to learn in three dimensions. In particular, finding a universality class with a three-dimensional topology is still an open problem. However, it is good to consider whether such a model even makes sense in four dimensions, since after all we do live in a four-dimensional spacetime.

Appendix A

Data Structures

A.1 Collections

In order to efficiently carry out the simulation we need two new data structures. Firstly, the **`Shelf`** will be used to store objects. Secondly the **`Bag`** will be used to sample objects from a **`Shelf`**

Shelf In order to explain the **`Shelf`** we must first establish what problem it solves. For the simulation we need some collection data structure to store all the simplex and tree node objects. It should meet the following requirements:

- Constant time insertion of an object,
- Constant time removal of an object when given its label,
- Constant time lookup of an object when given its label,
- Stable labels (i.e. labels do not change throughout the lifetime of an object).

A simple array together with a **`None`** value already meets most of our requirements. Removal can be done simply by replacing the corresponding position in the array with **`None`**. This is analogous to keeping books on a bookshelf: if we take a book away, a gap will be created. However, this makes inserting a new book non-trivial, since we first need to search for an empty spot. In order to counter this, we need to keep track of where the gaps are on the **`Shelf`**. We do this by writing an index value at every gap, each pointing to the next. We also store an auxiliary index outside the array pointing to the start of the trail. When inserting or removing objects we need to make sure that the trail is updated accordingly. This can all be done in constant time.

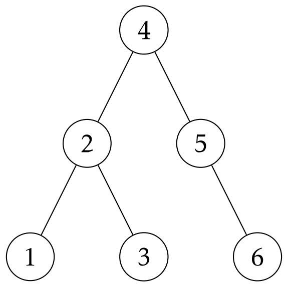

Figure A.1: Example representation of the sequence $1, 2, 3, 4, 5, 6$ as a binary tree.

Bag One practical issue with the **`Shelf`** is that we cannot easily sample from it uniformly. Additionally, in some cases we wish to keep track of a subset of objects on a **`Shelf`**, from which we would like to sample uniformly. The **`Bag`** structure solves this problem by storing a bijective map between (a subset of) n labels of **`Shelf`** objects and the integers from 0 to $n - 1$. Conceptually it rearranges (a subset of) objects on a **`Shelf`** as to eliminate any gaps, without actually rearranging anything in memory. This allows us to sample an object simply by uniformly choosing an integer $i \in [0, n-1]$. If we then follow the **`Bag`** to the proper label, we will find our object on the **`Shelf`**. Again, all necessary operations can be performed in constant time.

A.2 Link-Cut Tree

The link-cut tree is a data structure that can represent a dynamic forest (i.e. a set of trees). It can perform the **`link`**, **`cut`** and **`find_root`** operations in amortized $\mathcal{O}(\log N)$ time, where N is the number of nodes. The main idea of this data structure is that sequences are linked together to form trees, where each sequence is represented as a binary tree. The original version of the link-cut tree was proposed by Sleator and Tarjan in [11]. In this application, the link-cut tree is modified so that it can also perform the **`evert`**, **`depth`** and **`index_depth`** operations in amortized $\mathcal{O}(\log N)$ time. All these operations will be described in more detail below.

From Trees to Sequences to Trees Again First consider the simpler problem of efficiently representing a finite ordered sequence of nodes. One way to save an ordered sequence efficiently is using a binary tree. Given a binary tree, an ordering can be uniquely obtained by looking at the parent-child relations. We define that a left child always occurs earlier in the sequence than its parent, while a right child always occurs later in the sequence. For an example of this, see Figure A.1.

At the level of the binary tree, the most important operation is **`splay`**. If a binary tree data structure implements this operation, it is called a splay tree.

This operation moves a given node to the root of the binary tree, while preserving the ordering. It turns out that there is some freedom in the way we do this. The **splay** operation does this in a particular way, such that the number of vertical levels in the tree is minimized. That is, after many splays the maximum depth of the tree will be $\mathcal{O}(\log N)$. This is the main reason for why all operations can be performed in $\mathcal{O}(\log N)$ time.

In order to construct trees from a set of sequences, these sequences should be connected to each other in some way. If we can link the first node in sequence A to any node in another sequence B, we will create a branch. By repeating this process we can create any tree we like. This means that each sequence (except the root) must contain a pointer to the node to which it is attached.

A connection of the type described here is known as a path parent, whereas "standard" connections are referred to as binary tree parents. Each node must have either a path parent, a binary tree parent or it is the root of a tree. Each node can have at most one binary-tree children (The binary trees represent sequences, so a given node can only have one successor in the sequence.), while the number of path children is unlimited. It is possible to switch the binary-tree child of a given node by using the so-called **splice** method.

Basic Operations The operation at the heart of the link-cut tree is the **expose** method. With this method we can take a given node u and put it in the same binary tree as the root by using **splice** operations. Additionally, we make sure that u is the last in its sequence. Thus, we end up with a sequence starting from the root and ending at u, represented in a single binary tree. Finally, we put u at the root of that binary tree using the **splay** method. This means that u will be at the root of its binary tree, while the rest of the nodes are to its left. It can be shown [11] that the time cost of an **expose** operation is amortized $\mathcal{O}(\log N)$, where N is the number of nodes.

By moving a node into such a particular position we can make other operations easier. For example, to **cut** a node u we first **expose** it, and then we simply cut off its left child. To **link** two nodes we first both **expose** them, then link one to the other by connecting their binary trees. Finally, we can find the root of some node u by first exposing it. Then we keep walking down the binary tree through the left-side children, until reaching a node that does not have a left child. This node will be the root. All of these operations require one or two exposes, in addition to a few pointer reassignments. Thus, these operations are all performed in logarithmic time.

Extra operations Finally, we add some additional operations to our implementation of the link-cut tree. Both of these require the storage of information at each node. In particular, we store information using a delta representation. That is, at each node we only store the change in value relative to its parent node. Since the root node does not have a parent, we store the actual value there. Thus, to obtain the value at a node we must sum over all of its ancestors. By using this representation we can change the information at an entire branch of the tree by only changing one value.

One way in which this can be used is by assigning a "flipped-ness" to each node. If a node is flipped we must swap its left and right children when constructing a sequence from its binary tree. To store this information we save a delta-flip value at each node. Thus, changing the delta-flip value at a node u will reverse the sequence order of all nodes that are binary tree descendants of u. We can use this to change the root (i.e. **`evert`**) to a given node u: First we **`expose`** u, and then we change its delta-flip.

Due to the complex nature of the link-cut tree data structure, it is not a straightforward task to find the distance of a given node to the root. Therefore, we use the delta representation to save the depth of every node. Care needs to be taken to ensure that the delta-depth value is properly updated during all other operations. Additionally, we can use this depth information to efficiently (i.e. in logarithmic time) select the ith node from a sequence from the root to a particular node.

Bibliography

[1] D. Aldous. 'The Continuum Random Tree. I'. In: *The Annals of Probability* 19.1 (1991), pp. 1–28. DOI: 10.1214/aop/1176990534.

[2] J. Ambjørn et al. 'The vacuum in three-dimensional simplicial quantum gravity'. In: *Physics Letters B* 276.4 (1992), pp. 432–436. DOI: 10.1016/0370-2693(92)91663-T.

[3] S. Boyd, P. Diaconis and L. Xiao. 'Fastest Mixing Markov Chain on a Graph'. In: *SIAM Review* 46.4 (2004), pp. 667–689. DOI: 10.1137/S0036144503423264.

[4] T. Budd and L. Lionni. *A family of triangulated 3-spheres constructed from trees*. 2022. arXiv: 2203.16105.

[5] M.S.S. Challa, D.P. Landau and K. Binder. 'Finite-size effects at temperature-driven first-order transitions'. In: *Phys. Rev. B* 34 (3 1986), pp. 1841–1852. DOI: 10.1103/PhysRevB.34.1841.

[6] B. Duplantier, J. Miller and S. Sheffield. *Liouville quantum gravity as a mating of trees*. 2020. arXiv: 1409.7055.

[7] J.-F. Le Gall. 'Uniqueness and universality of the Brownian map'. In: *The Annals of Probability* 41.4 (2013), pp. 2880–2960. DOI: 10.1214/12-AOP792.

[8] H. Hagura, N. Tsuda and T. Yukawa. 'Phases and fractal structures of three-dimensional simplicial gravity'. In: *Physics Letters B* 418.3 (1998), pp. 273–283. DOI: 10.1016/S0370-2693(97)01320-8.

[9] T. Hotta, T. Izubuchi and J. Nishimura. 'Multicanonical simulation of 3D dynamical triangulation model and a new phase structure'. In: *Nuclear Physics B* 531.1 (1998), pp. 446–458. DOI: 10.1016/S0550-3213(98)00482-9.

[10] L.M.S. Russo, A. Teixeira and A. Francisco. 'Linking and Cutting Spanning Trees'. In: *Algorithms* 11.4 (2018), p. 53. DOI: 10.3390/a11040053.

[11] D.D. Sleator and R.E. Tarjan. 'A Data Structure for Dynamic Trees'. In: *Proceedings of the Thirteenth Annual ACM Symposium on Theory of Computing*. 1981, pp. 114–122. DOI: 10.1145/800076.802464.

[12] G. Thorleifsson. ‘Three-dimensional simplicial gravity and degenerate triangulations’. In: *Nuclear Physics B* 538.1-2 (1999), pp. 278–294. DOI: 10.1016/s0550-3213(98)00679-8.