Navigating the Pachner Graph: Algorithms for Searching and Sampling Triangulations

Daniel Bruwel

An essay submitted in partial fulfillment of
the requirements for the degree of
B.Sc. (Honours)

Pure Mathematics
University of Sydney

November 2025

CONTENTS

Introduction

Decomposing a surface or higher-dimensional manifold into discrete units, such as triangles and tetrahedra, is a standard technique used across a range of disciplines. This process, known as triangulation, transforms the analytic problem of studying a continuous space into a combinatorial one, thereby opening it up to a range of computational methods. Applications of triangulation are widespread. In physics, they appear in the Regge calculus of general relativity [**?**] and causal dynamical triangulations of quantum gravity [**?**], where spacetime is discretised for simulations. Engineering disciplines use meshes - a form of triangulation - to approximate structures like wings or bridges for finite element analysis [**?**]. Triangulations are also fundamental in computer graphics for rendering objects [**?**] and in data science for topological data analysis [**?**]. Within mathematics, they are used to compute topological invariants like homology groups [**?**], study combinatorial Ricci flows in differential geometry [**?**], and have been instrumental in graph theory, for instance, in the proof of the four-colour theorem [**?**]. A key challenge is that triangulations of a given space are not unique, and their suitability depends on the specific application. For instance, algebraic topology often seeks triangulations with the minimum number of faces [**?**], whereas computer graphics prioritises triangulations that best approximate the original surface geometry. Consequently, significant research has focused on understanding the space of all possible triangulations for a given manifold and developing techniques to generate optimal ones.

While there are various criteria for what constitutes an optimal triangulation, this report primarily explores topological and combinatorial properties. Specifically, we seek single-vertex triangulations of the 3-sphere that correspond to interesting or complex knots. Although the techniques developed can be extended to other notions of optimality, we focus on topological and combinatorial properties because they are fundamental to the triangulation itself. These properties do not depend on specific geometric embeddings, such as the angles or sizes of the component simplices. This approach narrows the focus to abstract triangulations - which are discrete and countable - improving computational manageability and ensuring that the results are general and dependent only on the triangulation's intrinsic structure, not on extrinsic geometric factors. Additionally, there are numerous interesting open problems in low dimensional topology that can be addressed by such techniques [**?**].

The primary objective of this report is to implement, analyse, and compare a range of computational techniques for searching the space of triangulations. We investigate classical search and sampling methods, namely Markov Chain Monte Carlo (MCMC), greedy search, and simulated

annealing. While greedy search algorithms have been a central tool in this field, MCMC is a more recent technique [**?**]. We further adapt the MCMC framework to implement simulated annealing. Additionally, we lay the groundwork for a new approach using transformer-based models to explore the space. This opens the door for reinforcement learning, a field with a growing repertoire of successes in complex problems including playing Go [**?**], discovering faster sorting algorithms [**?**], and writing mathematical proofs [**?, ?**].

The report is structured as follows:

- Chapter 1 introduces the necessary preliminary concepts, including manifolds, knot theory, and an overview of relevant statistical and machine learning techniques.
- Chapter 2 provides a detailed outline and discussion of the specific algorithms implemented, including justifications for our design choices.
- Chapter 3 presents a description and analysis of a series of numerical experiments conducted on five different search problems.
- Chapter 4 concludes with a discussion of our findings and outlines directions for future research.

CHAPTER 1

Preliminaries

1.1. Mathematical Preliminaries

1.1.1. Manifolds. We begin by recalling the definition of a "manifold" and introducing a "piecewise linear manifold". Piecewise linear manifolds are fundamentally related to triangulations - the central objects we will be studying.

A d-dimensional manifold, or d-manifold, is a topological space that generalises the notion of a surface by being locally homeomorphic to Euclidean space, $\mathbb{R}^d$. More precisely, we recall the following definitions.

Definition 1.1 (Second Countable)**.** *A topological space $(\mathcal{M}, \tau_{\mathcal{M}})$ is **Second Countable** if the topology $\tau_{\mathcal{M}}$ of $\mathcal{M}$ has a countable base.*

Definition 1.2 (Hausdorff)**.** *A topological space $(\mathcal{M}, \tau_{\mathcal{M}})$ is **Hausdorff** if for any two distinct points $x, y \in \mathcal{M}$, there exist neighbourhoods U of x and V of y such that $U \cap V = \varnothing$.*

With these, we can formally define a manifold.

Definition 1.3 (d-dimensional manifold)**.** *A d-**dimensional manifold** is a Hausdorff, second-countable topological space $\mathcal{M}$ where for every point $p \in \mathcal{M}$ there is a neighbourhood $U(p)$ of p that is homeomorphic to an open subset of Euclidean space $\mathbb{R}^d$.*

Remark 1.4. The condition of having a neighbourhood homeomorphic to an open subset of $\mathbb{R}^d$ is equivalent to stating that each point is either isolated (if $d = 0$) or has a neighbourhood homeomorphic to all of $\mathbb{R}^d$.

Remark 1.5. We will refer to a d-manifold simply as a manifold. In other contexts, this might be called a *topological manifold* to distinguish it from objects with additional structure (c.f. *differentiable manifold, Riemannian manifold, etc.*).

Common examples of manifolds include the circle S^1, the sphere S^2, the torus T^2, and Euclidean space $\mathbb{R}^d$. Some non-examples include the disk D^2, because its boundary points have no neighbourhood homeomorphic to an open subset of $\mathbb{R}^2$ - though this is a "manifold with boundary"; the figure-eight curve, because at the crossing point no neighbourhood is homeomorphic to an open subset of $\mathbb{R}$; and the line with two origins, because it is not Hausdorff.

In the study of triangulations, we are interested in an additional structure known as a *piecewise-linear structure*, or *PL-structure*. Intuitively, a PL-structure allows a manifold to be realised as a

collection of "flat pieces" joined together linearly. For instance, S^1 admits a PL-structure that can be realised as the boundary of a triangle or a square. Two PL-manifolds (manifolds with a PL-structure) are *PL-homeomorphic* if they can be transformed into each other via a homeomorphism that is itself piecewise-linear. A formal treatment of PL-structures in terms of charts and atlases can be found in introductory texts on geometric topology (e.g., "Introduction to piecewise-linear topology" [**?**]).

A key result for this work, due to Radó [**?**] and Moise [**?**], establishes an equivalence between topological and piecewise-linear structures in low dimensions.

Theorem 1.6 (Hauptvermutung, simplified)**.** *For dimensions $d \leq 3$, every manifold admits a PL-structure that is unique up to PL-homeomorphism.*

Remark 1.7. This theorem does not hold for dimensions four or more [**?**].

As this work focuses on manifolds of dimension 3, we will often treat a 3-manifold and its unique PL-structure as equivalent.

1.1.2. Triangulations. We now provide a light introduction to triangulations, though some familiarity with the concept is expected and can be found for example in Hatcher's "Algebraic Topology" [**?**].

We begin by defining a *simplex*, the fundamental building blocks of triangulations.

Definition 1.8 (Simplex)**.** *A d-simplex Δ is the d-dimensional convex hull of $d+1$ affinely independent vertices.*

A 0-simplex is a point, a 1-simplex is a line segment, a 2-simplex is a triangle, and a 3-simplex is a tetrahedron. For any $i < d$, any subset of $i+1$ vertices of a d-simplex forms an i-dimensional simplex that we call an i-**dimensional face** of the original simplex.For for $i < d$, any subset of $(i+1)$ points of an $d-$simplex forms another simplex that we call an $i-$**dimensional face**. From this we can define the boundary of a simplex:

Definition 1.9 (Boundary of a Simplex)**.** *For a d-simplex Δ, the boundary, denoted $\partial\Delta$, is the union of all its $(d-1)$-dimensional faces.*

We can take simplices and glue them together along their faces to form more complex geometric structures.

Definition 1.10 (Face Gluing)**.** *For two d-simplices, Δ_1 and Δ_2, and two of their respective i-dimensional faces, $F_1 \subseteq \Delta_1$ and $F_2 \subseteq \Delta_2$, a **face gluing** is the space $\Delta_1 \sqcup \Delta_2/\sim$ obtained by identifying the faces via a homeomorphism, or gluing map, between F_1 and F_2.*

Typically, the gluing map is defined by a bijection between the vertices of F_1 and F_2, which is then extended linearly.

Definition 1.11 (PL-Sphere)**.** *The PL 0-sphere is a pair of isolated points. For integers $d > 0$, the PL d-sphere is a PL-manifold that is PL-homeomorphic to the boundary of a $(d+1)$-simplex.*

Definition 1.12 (PL-Triangulation)**.** *A **PL-triangulation** T is a topological space constructed from a finite collection of d-simplices $\{\Delta_1, ..., \Delta_k\}$, called **facets**, by gluing their $(d-1)$-dimensional faces together, subject to the following conditions:*

a) If a face is identified with itself as a result of the gluings, the identification map must be the identity.

*b) For any vertex v, the **link** of v (the complex formed by all simplices $\sigma \in T$ such that v is not a vertex of σ but whose vertices, together with v, form a simplex in T) must be a PL $(d-1)$-sphere.*

This is a PL-triangulation of a manifold $\mathcal{M}$ if there is a PL-homeomorphism from T to $\mathcal{M}$.

Remark 1.13. We can view PL-manifolds (manifolds with PL-structure) and PL-triangulations as the same thing in some sense. That being that for every PL-manifold there is a PL-triangulation that is compatible with it, and every PL-triangulation is a PL-manifold.

1.1.3. Knot Theory. This section introduces fundamental concepts from knot theory, building upon standard results from algebraic topology. For a comprehensive background reference, see Hatcher's "Algebraic Topology" [**?**]. We are particularly interested in understanding the "Alexander polynomial", as this is involved in many of the search problem we are examining.

Definition 1.14 (Ambient Isotopy)**.** *For a pair of manifolds N and M and embeddings g, h of N into M, an **ambient isotopy** from g to h is a continuous map $F : M \times [0,1] \to M$ such that for each $t \in [0,1]$, the map $F_t : M \to M$ is a homeomorphism, F_0 is the identity map, and $F_1 \circ g = h$.*

Definition 1.15 (Knot)**.** *A **knot** is a smooth embedding of the circle S^1 into the 3-sphere S^3.*

Remark 1.16. We often refer to a knot as both the embedding function and its image in S^3. When we say "a knot K", we typically mean the image of the embedding.

Definition 1.17 (Knot Equivalence)**.** *Two knots are **equivalent** if there is an ambient isotopy between their respective embedding functions.*

Definition 1.18 (Knot Invariant)**.** *A **knot invariant** is a quantity or property assigned to a knot K that is the same for all knots equivalent to K.*

Remark 1.19. A knot invariant is not necessarily unique; non-equivalent knots may share the same invariant.

Definition 1.20 (Knot Complement)**.** *For a knot K, its **tubular neighbourhood** $\nu(K)$ is a small, closed, 3-dimensional region surrounding K that is homeomorphic to the solid torus $S^1 \times D^2$. The **knot complement** is the space $X_K = S^3 \setminus \nu(K)$.*

The boundary of the knot complement, ∂X_K, is a torus, $\partial X_K \cong T^2$. A central knot invariant for this project is the Alexander polynomial, the definition of which requires the following concepts.

Definition 1.21 (Infinite Cyclic Cover)**.** *For the knot complement X_K, a cover $\tilde{X}_K$ of X_K is an **infinite cyclic cover** if the group of deck transformations $\mathrm{Deck}(\tilde{X}_K/X_K)$ is isomorphic to the infinite cyclic group $\mathbb{Z}$.*

Every knot complement has a unique infinite cyclic cover. This follows from the fact that its first homology group is infinite cyclic, $H_1(X_K;\mathbb{Z}) \cong \mathbb{Z}$ (a result of Alexander duality), and from the classification theorem for covering spaces.

Definition 1.22 (Alexander Module)**.** *For a knot K, consider the infinite cyclic cover $\tilde{X}_K$ of its complement. The deck transformation group of this space is isomorphic to $\mathbb{Z}$ and is generated by some transformation t. We form the ring of Laurent polynomials $\Lambda = \mathbb{Z}[t, t^{-1}]$. The **Alexander module** is the first homology group $H_1(\tilde{X}_K;\mathbb{Z})$, viewed as a module over the ring Λ.*

Definition 1.23 (Elementary Ideal)**.** *For a module M over a unique factorisation domain R with a presentation of n generators g_i and m relations $r_j = \sum_{i=1}^{n} a_{ji} g_i = 0$, we construct the $m \times n$ presentation matrix $A = (a_{ji})$. The k-th **elementary ideal** is the ideal $E_k(M)$ generated by all $(n-k) \times (n-k)$ minors of A.*

Theorem 1.24. *The elementary ideals are independent of the chosen presentation of the module.*

Definition 1.25 (Alexander Polynomial)**.** *The **Alexander Polynomial**, $\Delta_K(t)$, is the generator of the first elementary ideal of the Alexander module. This ideal is principal, and its generator is unique up to multiplication by a unit in Λ, i.e., a factor of $\pm t^k$ for some integer k.*

Theorem 1.26. *The Alexander polynomial is a knot invariant.*

The proof that this is a well-defined knot invariant is originally due to J. W. Alexander [**?**].

1.1.4. Isomorphism Signatures. We say that two PL-triangulations are *combinatorially isomorphic* if they are identical up to a relabelling of their faces. Since the initial labelling of a triangulation's components is arbitrary, a canonical representation is necessary for unique identification and comparison. While specific implementations vary, the general algorithm to produce a canonical labelling is as follows.

a) For each facet, calculate an initial, labelling-independent invariant (e.g., the number of distinct facets glued to its boundary faces).
b) Partition the facets into bins based on the value of this invariant. This initial partitioning is canonical.
c) For each facet, generate a new identifier based on the current partition of its neighbours (i.e., which bins its adjacent facets belong to). Use these new identifiers to refine the partition; if facets within the same bin now have different identifiers, the bin is split.

d) Repeat the refinement process until the partition stabilises. If this process does not result in each bin containing a single, unique facet, an arbitrary but deterministic choice is made to break a tie and continue the algorithm.
e) If any arbitrary choices were required, the entire process is repeated for each possible choice, generating a set of potential canonical labellings. The final isomorphism signature string is derived from each, and the lexicographically smallest string is chosen as the definitive canonical representation.

The specific canonical labelling algorithm used in this work is that implemented in the 3-manifold software Regina [**?**], which includes details on the initial invariant and various optimisations. In the worst-case, this process takes $\mathcal{O}((d+1)! \cdot n^2)$ time, where d is the dimension and n is the number of facets. For a 3-dimensional triangulation, the $(d+1)! = 24$ term is a small constant, making the algorithm efficient in practice.

Once a canonical labelling is found, an "isomorphism signature" is calculated. For a 3-dimensional triangulation, the signature must encode the number of tetrahedra and, for each face of each tetrahedron, the identity of the tetrahedron it is glued to and the orientation of the gluing. The destination tetrahedron and the vertex permutation uniquely determine the destination face, so it does not need to be stored explicitly. To store this information efficiently as a string, the following procedure is used:

a) For each face of each tetrahedron, an integer is calculated as $v_i = (24 \cdot \text{destination\_id}) + \text{permutation\_id}$. The `destination_id` is the canonical index of the target tetrahedron, and the `permutation_id` is an integer from 0 to 5 representing one of the $4! = 24$ possible vertex mappings for the gluing. This value will be less than $24n$ for a triangulation with n facets.
b) A sequence $(v_0, v_1, \ldots, v_{4n-1})$ is created by iterating through the tetrahedra in their canonical order, and for each tetrahedron, iterating through its four faces in a fixed local order.
c) These values are combined into a single large integer: $I = v_0 + v_1(24n) + v_2(24n)^2 + \cdots + v_{4n-1}(24n)^{4n-1}$.
d) The integer I is converted to a base-64 string for compact storage.
e) The number of tetrahedra, n, is converted to a string and prepended to the string representation of I.

This process yields a unique string that serves as the combinatorial isomorphism signature for any given 3-manifold triangulation, within practical limits on the number of facets n. The algorithm used in practice is more general, allowing, for example, for boundary faces. Details can be found in Burton [**?**].

1.1.5. Pachner Moves and the Pachner Graph. A Pachner move is a local transformation that modifies a PL-triangulation without changing the underlying topology of the manifold. In two

dimensions, the simplest example is an "edge flip", where two adjacent triangles are replaced by two different triangles that span the same quadrilateral region.

This concept can be generalised to higher dimensions.

Definition 1.27 (Complementary Triangulation)**.** *Let Δ be a $(d+1)$-simplex and let A be a connected subcomplex of its boundary $\partial\Delta$. The **complementary triangulation** of A is the subcomplex $B = \partial\Delta \setminus A$.*

Definition 1.28 (Pachner Move)**.** *For a d-dimensional PL-manifold, a **Pachner move** is a local replacement of a subcomplex. It involves identifying a collection of facets that is combinatorially equivalent to a subcomplex $A \subset \partial\Delta^{d+1}$ and replacing it with the corresponding complementary triangulation $B = \partial\Delta^{d+1} \setminus A$.*

The following theorem, due to Pachner [**?**], establishes the fundamental importance of these moves.

Theorem 1.29. *Two triangulations, T and T', represent the same PL-manifold if and only if one can be transformed into the other through a finite sequence of Pachner moves.*

For triangulations of 3-manifolds, there are two fundamental pairs of inverse Pachner moves:

- The **(2,3)-move** and its inverse, the **(3,2)-move**. The (2,3)-move acts on two tetrahedra that share a common triangular face. These two tetrahedra are removed and replaced by three tetrahedra that share a common edge connecting the two vertices opposite the original shared face. The (3,2)-move is the reverse of this process.
- The **(1,4)-move** and its inverse, the **(4,1)-move**. The (1,4)-move acts on a single tetrahedron, which is subdivided into four smaller tetrahedra by introducing a new vertex in its interior. The (4,1)-move is the reverse, removing an interior vertex of degree four and replacing the four surrounding tetrahedra with a single one.

Since any two triangulations of a specific PL-manifold can be reached from one another through Pachner moves, we can define a graph structure on the space of triangulations.

Definition 1.30 (Pachner Graph)**.** *For a given PL-manifold $\mathcal{M}$, the **Pachner graph** is the graph (V, E) where the set of vertices V is the set of all combinatorial isomorphism classes of triangulations of $\mathcal{M}$, and an edge $(v_i, v_j) \in E$ exists if and only if the triangulations v_i and v_j are related by a single Pachner move.*

By Theorem 1.29, the Pachner graph is connected for any PL-manifold. The graph is, however, infinite. We can stratify the vertices of the Pachner graph into "levels", where each level contains all triangulations with a fixed number of facets.

Theorem 1.31. *The number of vertices in the Pachner graph (i.e., the number of non-isomorphic triangulations) grows super-exponentially with its level.*

Notice that the (2,3) and (3,2) moves preserve the number of vertices in the triangulation, while the (1,4) and (4,1) moves change it. We are often interested in "single-vertex" triangulations, which can be explored using only the vertex-preserving moves. A key result for this subclass, due to Matveev [**?**], is that this restricted graph is still connected for the 3-sphere.

Theorem 1.32. *Any single-vertex triangulation of the 3-sphere S^3 with at least two tetrahedra can be reached from any other such triangulation through a sequence of only (2,3) and (3,2) moves.*

We can analogously define the Pachner graph for 1-vertex triangulations of S^3, which is also a connected graph. It is currently an open question whether the number of vertices in this restricted graph grows exponentially or super-exponentially with its level.

1.2. Machine Learning and Computational Preliminaries

1.2.1. Computational Complexity. An understanding of computational complexity is crucial for motivating the search and learning-based approaches used in this thesis, as many fundamental problems in topology are computationally intractable.

Definition 1.33 (Time Complexity)**.** *For an algorithm with input size n, the **time complexity** is the asymptotic worst-case number of elementary operations the algorithm performs, written in "Big-O" notation as $\mathcal{O}(g(n))$.*

Definition 1.34 (Space Complexity)**.** *For an algorithm with input size n, the **space complexity** is the asymptotic worst-case amount of memory the algorithm requires, written in "Big-O" notation as $\mathcal{O}(g(n))$.*

Definition 1.35 (Decision Problem)**.** *A **decision problem** is a problem that can be answered with either "yes" or "no".*

Definition 1.36 (Decidable)**.** *A decision problem is **decidable** if there exists an algorithm that is guaranteed to provide the correct answer for any input in a finite amount of time. Conversely, the problem is **undecidable** if no such algorithm exists.*

Remark 1.37. Many computational problems can be rephrased as decision problems. For example, the optimisation problem "find the shortest path between two nodes" can be converted into the decision problem "does a path of length at most k exist between these two nodes?".

Definition 1.38 (Complexity Class)**.** *A **complexity class** is a set of problems solvable under a given model of computation within a specified resource bound. For the standard Turing machine model, some major complexity classes are:*

a) P*: Solvable in polynomial time.*
b) NP*: A "yes" solution can be verified in polynomial time.*
c) $\mathrm{co}\,NP$*: A "no" solution can be verified in polynomial time.*

d) $PSPACE$*: Solvable using a polynomial amount of memory.*

e) $EXPTIME$*: Solvable in exponential time.*

Many other complexity classes exist for various computational models and resource constraints.

Remark 1.39. The distinction between NP and $\mathrm{co}\,NP$ is important. For instance, determining if a graph has a Hamiltonian path (a path visiting each node exactly once) is in NP, as a proposed path can be easily verified. However, its complement asks to determining that no such path exists. For this to be in $\mathrm{co}\,NP$ would imply that we could provide a "certificate" that we could use to verify that no such path exists in P time. It is widely believed that no such certificate exists, and that $NP \neq \mathrm{co}\,NP$. On the other hand $P = \mathrm{co}\,P$ as we can simply solve the problem and check if the answer is yes or no.

Remark 1.40. The definitions of NL and NP provided here are based on the concept of a verifiable "certificate" for a solution. The formal definitions use non-deterministic Turing machines, but these two formulations can be shown to be equivalent.

Theorem 1.41. *The following inclusions hold for the major complexity classes:*

$$P \subseteq NP \subseteq PSPACE \subseteq EXPTIME$$

Remark 1.42. It is unknown whether any of these inclusions are proper. The question of whether $P = NP$ is the most famous unsolved problem in computer science.

Definition 1.43 (Hard Problem)**.** *For a complexity class C, a problem P_1 is C-**hard** if every problem $P_2 \in C$ can be reduced to P_1 by an algorithm that is efficient relative to the complexity of C.*

Remark 1.44. For complexity classes like NP and $PSPACE$, an "efficient" reduction is one that runs in polynomial time.

Definition 1.45 (Complete Problem)**.** *For a complexity class C, a problem P is C-**complete** if it is C-hard and $P \in C$.*

In practice, problems in P are considered computationally tractable. In contrast, all known algorithms for NP-complete problems require resources that grow exponentially with the input size, quickly becoming computationally infeasible. If a problem is proven to be NP-complete or harder, it implies that efficient, exact solutions are unlikely to exist, and we must often resort to heuristics or approximation algorithms.

The following theorems classify the computational difficulty of the core geometric problems addressed in this thesis, motivating our use of heuristic search methods.

Theorem 1.46. *Finding the shortest path between two triangulations on the Pachner graph is at least NP-complete.*

The subproblem of finding the shortest distance for a restricted set of 2-manifolds was shown to be NP-complete, so the general problem must be at least as hard [**?**].

Theorem 1.47 (Adian-Rabin)**.** *Determining if a given group presentation defines the trivial group is undecidable.*

For a proof, see Lyndon and Schupp chapter IV section 4 [**?**].

Theorem 1.48. *Recognising if a given 3-dimensional triangulation is homeomorphic to the 3-sphere, S^3, is in $NP \cap \text{co}\,NP$.*

The proof of this is due to Schleimer [**?**] and Zetner [**?**]. This result implies that both a "yes" (it is S^3) and a "no" (it is not S^3) answer have efficiently verifiable proofs. Problems in this class are not believed to be NP-complete.

Theorem 1.49. *Determining if a given knot is the unknot is in $NP \cap \text{co}\,NP$.*

1.2.2. Markov Chain Monte Carlo. We now examine Markov chain Monte Carlo (MCMC), a general class of algorithms used to generate samples from a probability distribution. The fundamental requirements are that: for any point in our space, we must be able to randomly generate a new point with a known probability, and we must know the target probability density function (at least up to a normalisation constant).

We will begin with an abstract examination of MCMC in terms of measure theory. This formal approach provides the necessary framework to rigorously understand the algorithm's. While this theoretical underpinning is important, our practical applications will only directly use the results summarised in Definition 1.56 onwards.

Definition 1.50 (Markov Chain)**.** *A Markov Chain is a sequence of random variables $x_0, x_1, x_2, \ldots$ indexed by "time", where the sequence possesses the Markov Property - that is, the distribution of x_{t+1} depends only on the value of x_t.*

Monte Carlo methods are a broad class of algorithms that use random sampling to obtain numerical results, often to approximate complex integrals or probability distributions. *Markov Chain Monte Carlo* (MCMC) is a specific class of these methods for sampling from a probability distribution by constructing a Markov Chain whose equilibrium distribution is the desired one.

Definition 1.51 (Markov Transition Kernel)**.** *Let $(\mathcal{X}, \mathcal{F})$ be a measurable space. A Markov Transition Kernel is a function $P : \mathcal{X} \times \mathcal{F} \to [0, 1]$ such that:*

a) for any $x \in \mathcal{X}$, the function $A \mapsto P(x, A)$ is a probability measure on $(\mathcal{X}, \mathcal{F})$;
b) for any $A \in \mathcal{F}$, the function $x \mapsto P(x, A)$ is a measurable function.

The Markov Transition Kernel describes the evolution of the system's probability distribution. If at time t the system is distributed according to a measure ν_t, then the distribution at time $t + 1$ is

given by

$$\nu_{t+1}(A) = (\nu_t P)(A) = \int_{\mathcal{X}} P(x, A) d\nu_t(x). \tag{1.52}$$

Definition 1.53 (Target Distribution)**.** *A probability distribution π is the **target distribution** for a Markov Transition Kernel P if the following two conditions hold:*

a) ***Stationarity****: $\pi P = \pi$.*

b) ***Ergodicity****: For π-almost all starting points x_0, the distribution of the chain's state after n steps converges to π in total variation norm,*

$$\lim_{n\to\infty} \|P^n(x_0, \cdot) - \pi(\cdot)\|_{TV} = 0.$$

Remark 1.54. A statement is true for "π-almost all x_0" if the set of x_0 for which it is false has measure zero under π.

Directly proving stationarity and ergodicity can be difficult. Instead, a set of sufficient conditions is often used.

Theorem 1.55. *For a Markov transition kernel P, if the following conditions hold for a distribution π, then π is the unique target distribution for P:*

a) ***π-irreducibility****: For any set $A \in \mathcal{F}$ with $\pi(A) > 0$, there exists some n such that $P^n(x, A) > 0$ for all $x \in \mathcal{X}$. (This ensures the chain can reach any part of the state space from any starting point).*

b) ***Aperiodicity****: The chain is not trapped in deterministic cycles.*

c) ***Recurrence****: The chain is guaranteed to return to any given region of the space.*

d) ***Reversibility (Detailed Balance)****: For any two measurable sets $A, B \in \mathcal{F}$, we have*

$$\int_A P(x, B) d\pi(x) = \int_B P(x, A) d\pi(x).$$

(This is a stronger condition than stationarity that is often easier to satisfy).

By the Ergodic Theorem for Markov chains, for any integrable function f of the state (an observable), the sample average converges to the true expectation under the target distribution: $S_n = \frac{1}{n}\sum_i^n f(x_i) \to \mathbb{E}_\pi[f]$. The error of this approximation typically decreases at a rate of $\mathcal{O}(1/\sqrt{n})$.

Definition 1.56 (Metropolis-Hastings Algorithm)**.** *Given a function $f(x)$ that is proportional to the target probability density, $f(x) \propto \pi(x)$, and a proposal distribution $g(x'|x)$, the Metropolis-Hastings algorithm generates a sequence of samples as follows: Initialise x_0. For each step $t = 0, 1, 2, \ldots$:*

a) Propose a new state x' according to $g(x'|x_t)$.

b) Compute the acceptance ratio $\alpha = \frac{f(x')}{f(x_t)}$.

c) Sample u from a uniform distribution on $[0, 1]$.

d) If $\alpha > u$, accept the new state by setting $x_{t+1} = x'$. Otherwise, reject and set $x_{t+1} = x_t$.

Theorem 1.57. *If the proposal function g is symmetric (i.e., $g(x|y) = g(y|x)$), the Markov chain generated by the Metropolis-Hastings algorithm has π as its target distribution.*

Remark 1.58. If the proposal distribution is not symmetric, the acceptance ratio must be modified by the Hastings ratio, $\alpha_H = \alpha \cdot \frac{g(x_t|x')}{g(x'|x_t)}$, to maintain detailed balance.

1.2.2.1. *Statistics in MCMC.* For any finite number of samples n, the empirical distribution may not have converged to π. This is particularly common if π is multi-modal, as the chain can become "stuck" in one mode. Consequently, statistical diagnostics are used to assess convergence, typically by running multiple independent chains.

Definition 1.59 (Variance Components)**.** *Let there be J chains, each of length L, with samples $x_1^j, \dots, x_L^j$ for the j-th chain.*

*a) **Between-chain variance**: $B = \frac{L}{J-1} \sum_{j=1}^{J} (\bar{x}_j - \bar{x}_*)^2$, where $\bar{x}_j$ is the mean of chain j and $\bar{x}_*$ is the global mean. This measures the variance between the means of each chain.*

*b) **Within-chain variance**: $W = \frac{1}{J} \sum_{j=1}^{J} s_j^2$, where s_j^2 is the sample variance of chain j. This is the average of the individual chain variances.*

From these components, we can estimate the total variance of the underlying distribution.

Theorem 1.60 (Gelman-Rubin Variance Estimate)**.** *An estimate for the total variance of the target distribution, accounting for both within- and between-chain variation, is given by*

$$\hat{V} = \frac{L-1}{L} W + \frac{1}{L} B.$$

This leads to a practical convergence diagnostic.

Definition 1.61 (Gelman-Rubin Statistic)**.** *The Gelman-Rubin statistic, or potential scale reduction factor, is*

$$\hat{R} = \sqrt{\frac{\hat{V}}{W}}.$$

If the chains have converged, they are all sampling from the same distribution, so the within-chain variance should approximate the total variance, yielding $\hat{R} \approx 1$. A value of $\hat{R} > 1$ indicates that the between-chain variance is still large compared to the within-chain variance, suggesting the chains have not yet fully explored the state space. A common heuristic is to require $\hat{R} < 1.01$ to indicate convergence [**?**].

1.2.3. Simulated Annealing. For a measurable space $(\mathcal{X}, \mathcal{F}, \mu)$, we may have a measurable function $E : \mathcal{X} \to \mathbb{R}$, often called an "energy" or "cost" function that we wish to minimise.

Simulated annealing is a probabilistic optimisation metaheuristic inspired by the annealing process in metallurgy. The core idea is to sample from the temperature-dependent Gibbs distribution, $\pi_T(x) \propto e^{-E(x)/T}$, while gradually adapting the temperature parameter T.

Simulated annealing is an MCMC method where the target distribution, π_T, changes over time. When using the Metropolis-Hastings algorithm with a symmetric proposal, the acceptance ratio for a move from state x to x' at temperature T is

$$\alpha = \frac{\pi_T(x')}{\pi_T(x)} = \frac{e^{-E(x')/T}}{e^{-E(x)/T}} = e^{-(E(x')-E(x))/T}.$$

This creates a trade-off between exploration and exploitation. At high temperatures ($T \gg 1$), the acceptance probability for energetically unfavourable moves (where $E(x') > E(x)$) is high, allowing the chain to traverse the state space freely and escape local minima. As the temperature is lowered ($T \to 0$), the probability of accepting such "uphill" moves vanishes, causing the algorithm to behave like a greedy search that settles into the nearest energy minimum.

The performance of simulated annealing is highly dependent on the choice of a "cooling schedule" - the rate at which the temperature T is decreased. While theoretically a sufficiently slow cooling schedule guarantees convergence to the global minimum, in practice this may require an infeasibly long runtime, and the method is used as a heuristic to find good, though not necessarily optimal, solutions.

1.3. Deep Learning

1.3.1. Transformers. Similar to the section on MCMC, we will begin with an abstract examination of transformers to provide a rigorous framework to understand the algorithm. Practically we will only be using the results from definition 1.69 onwards.

Definition 1.62 (Row Operator)**.** *For a field F and a function $s : F^d \to F^d$, define the **row operator** $\mathcal{R}_s : M_d(F) \to M_d(F)$ as the application of s to each row of a matrix in $M_d(F)$. That is,*

$$\mathcal{R}_s\left((r_1, r_2, \ldots, r_d)^T\right) = (s(r_1), s(r_2), \ldots, s(r_d))^T.$$

Definition 1.63 (Abstract Transformer Head)**.** *Let $(v_1, v_2, \ldots, v_d)$ be a collection of vectors in a vector space V over a field F. Let $\dot{V}$ be the space V equipped with a bilinear form $B : V \times V \to F$. Let $G : F^d \otimes V \to M_d(F)$ be the "Gram operator" which computes the matrix of pairwise interactions under B. Let $L : V \to W$ be a linear operator and $s : F^d \to F^d$ be a function.*

*An **abstract transformer head** is a function $H : F^d \otimes V \to F^d \otimes W$ defined by*

$$H(x) = (\mathcal{R}_s(G(x)) \otimes L)(x). \tag{1.64}$$

Definition 1.65 (Abstract Multi-Head Attention)**.** *Let H_i be an abstract transformer head from $F^d \otimes V_i \to F^d \otimes W_i$ for $i = 1, \ldots, q$, where each V_i is the space V equipped with a potentially unique bilinear form. Let $O : \bigoplus_{i=1}^q W_i \to U$ be a linear operator. The **multi-head attention** is an*

operator $A : F^d \otimes V \to F^d \otimes U$ *defined as*

$$A(x) = (I \otimes O)\left(\bigoplus_i H_i(x)\right). \tag{1.66}$$

Definition 1.67 (Abstract Feedforward Network)**.** *For a vector space* U *over a field* F*, an* ***abstract feedforward network*** *is an operator* $N : U \to W$ *defined by* $N(x) = L_2(a(L_1(x)))$*, where* $L_1 : U \to V$ *and* $L_2 : V \to W$ *are linear operators and* $a : V \to V$ *is a non-linear function.*

Definition 1.68 (Abstract Transformer Block)**.** *An* ***abstract transformer block*** $T : F^d \otimes V \to F^d \otimes V$ *is an operator defined by the sequence of operations:*

$$y_1 = x + A(x)$$

$$T(x) = y_1 + (I \otimes N)(y_1)$$

While these abstract formulations are theoretically rich, in practice we work over $F = \mathbb{R}$ with finite-dimensional vector spaces. This allows us to represent linear transformations as matrix multiplications and bilinear forms as $B(v_1, v_2) = v_1^T M v_2$ for some matrix M. The abstract components correspond to the standard implementation as follows: the Gram operator $G(x)$ is realised by projecting the input vectors into "query" and "key" spaces before computing their dot products; the row-wise function $\mathcal{R}_s$ becomes the softmax function; and the linear map L is a projection into a "value" space. The non-linear functions are fixed or have a small number of learnable parameters. This gives a finite set of scalar parameters θ that defines the transformer block. Additionally, components for numerical stability and regularisation are introduced.

Definition 1.69 (Standard Transformer Head)**.** *For a collection of vectors* $(v_1, \ldots, v_d)$*, where* $v_i \in \mathbb{R}^{d_e}$*, we form the matrix* $X \in \mathbb{R}^{d \times d_e}$ *by stacking these vectors as rows. A single attention head is calculated as:*

$$h(X) = \mathrm{softmax}\left(\frac{(XW^Q)(XW^K)^T}{\sqrt{d_k}} + M\right)(XW^V), \tag{1.70}$$

where the softmax is applied row-wise. The matrices $W^Q, W^K \in \mathbb{R}^{d_e \times d_k}$ *and* $W^V \in \mathbb{R}^{d_e \times d_v}$ *are learnable weight matrices for the "query", "key", and "value" projections. The term* $1/\sqrt{d_k}$ *is a scaling factor for gradient stability.* $M \in M_d(\mathbb{R} \cup \{-\infty\})$ *is an optional mask, often used in auto-regressive models to prevent positions from attending to subsequent positions.*

Definition 1.71 (Standard Multi-Head Attention)**.** *For an input* X *and* q *attention heads* $h_1, \ldots, h_q$*, multi-head attention is defined as:*

$$MHA(X) = \mathrm{Concat}(h_1(X), \ldots, h_q(X))W^O, \tag{1.72}$$

where Concat *joins the output matrices of each head along their feature dimension, and* W^O *is a final learnable output projection matrix.*

Definition 1.73 (Standard Feedforward Network)**.** *For $x \in \mathbb{R}^n$, a standard two-layer feedforward network is $FFN(x) = \sigma_{act}(xW_1 + b_1)W_2 + b_2$, where W_i are weight matrices, b_i are bias vectors, and σ_{act} is a non-linear activation function (e.g., ReLU).*

Definition 1.74 (Layer Normalisation)**.** *For a vector $x \in \mathbb{R}^d$, let μ_x and σ_x be its mean and standard deviation. The layer normalisation of x is*

$$LN(x)_i = \gamma_i \left(\frac{x_i - \mu_x}{\sqrt{\sigma_x^2 + \epsilon}} \right) + \beta_i, \tag{1.75}$$

where $\gamma, \beta \in \mathbb{R}^d$ are learnable scale and shift parameters, and ϵ is a small constant for numerical stability.

Definition 1.76 (Dropout)**.** *For a rate $r \in [0, 1]$, the **dropout** operator $DO(X, r)$ randomly sets a fraction r of the entries in the matrix or vector X to zero during training. This is a form of regularisation to prevent overfitting.*

Definition 1.77 (Standard Transformer Block)**.** *A standard transformer block applies these components in sequence. Given an input X, it computes:*

$$X' = X + MHA(LN(X)) \qquad \textit{(Multi-head attention with residual connection)}$$
$$X_{out} = X' + FFN(LN(X')) \qquad \textit{(Feedforward network with residual connection)}$$

Dropout is typically applied after the MHA and FFN steps during training.

Because these transformer blocks map from $\mathbb{R}^{d \times d_e}$ to $\mathbb{R}^{d \times d_e}$, they can be chained together to form a deep model. For many tasks, the input is a sequence of discrete tokens rather than vectors.

Definition 1.78 (Token Embedding)**.** *For a vocabulary of size V, a token is represented by a one-hot vector $v \in \{0, 1\}^V$. The **token embedding** of v is $h = vE$, where $E \in \mathbb{R}^{V \times d_e}$ is a learnable embedding matrix. This projects the sparse, high-dimensional token vector into a dense, lower-dimensional space.*

The transformer block is permutation-invariant, so to incorporate sequential information, we use:

Definition 1.79 (Positional Encoding)**.** *For a sequence of embedded vectors $(h_0, h_1, \ldots, h_{d-1})$, a **positional encoding** is added to each vector. This is typically done by adding a vector P_i from a pre-defined or learned matrix $P \in \mathbb{R}^{D_{max} \times d_e}$ to the input vector h_i, where i is the position in the sequence. Thus, the final input to the first transformer block is $h'_i = h_i + P_i$.*

A prototypical example is the GPT-2 model. A sequence of words is converted to a sequence of token embeddings, to which positional encodings are added. This input is then processed by a series of transformer blocks. The output from the final block is passed through a final layer normalisation

and then a linear projection back to the vocabulary dimension. At each position i, this yields a vector of logits, which is passed through a softmax function to produce a probability distribution over the entire vocabulary for the next word in the sequence, w_{i+1}. The model is trained using a cross-entropy loss objective, comparing the predicted distribution at each position with the actual next word in the training data.

1.3.2. Gradient Descent. Given a neural network $N(x \mid \theta)$ with parameters $\theta \in \mathbb{R}^d$, the goal of training is typically to find a parameter set $\hat{\theta}$ that minimises a loss function $\mathcal{L}(N(x, \theta), y)$ averaged over a set of training data $\{(x_i, y_i)\}$. Since the network N is a complex, non-linear function of θ, this objective function cannot be minimised analytically. Instead, iterative optimisation methods like gradient descent are used.

The fundamental principle of gradient descent is to update the parameters in the opposite direction of the gradient of the loss function, $\theta_{i+1} = \theta_i - \eta \nabla_\theta \mathcal{L}(\theta_i)$, where η is the learning rate. More formally, each step can be viewed as solving a local linear approximation of the loss function, regularised by a penalty on the step size. A single update step is the solution to:

$$\theta_{i+1} = \arg\min_\theta \left(\langle g, (\theta - \theta_i) \rangle + \frac{\lambda}{2} \|\theta - \theta_i\|^2 \right), \tag{1.80}$$

where $g = \nabla_\theta \mathcal{L}(\theta_i)$ is the gradient and the second term penalises large deviations from the current point θ_i.

Computing the gradient over the entire training dataset is computationally expensive. Therefore, in practice, the gradient is estimated using a small, random "mini-batch" of data. This approach is known as stochastic gradient descent (SGD). Because each mini-batch provides only a stochastic estimate of the true gradient, the updates can be noisy. To stabilise this process, many algorithms incorporate a momentum term, which uses an exponentially weighted moving average of past gradients.

Different choices for the norm, the gradient calculation, and the use of momentum lead to different optimisers. Common examples include:

- *SGD*, which typically uses the Frobenius norm and the standard stochastic gradient.
- *Adam*, which uses exponential smoothing for both the gradient (first moment) and its squared values (second moment) to create an adaptive step size, which can be related to the $\ell_1 \to \ell_\infty$ induced norm [**?**].
- *Shampoo*, which uses pre-conditioning based on the spectral properties of the parameter matrices.
- *Natural Gradient Descent (NGD)*, which uses the Fisher information matrix to define a geometry-aware update direction.

1.3.3. Reinforcement Learning. In our context, reinforcement learning is a technique for optimising a parameterised probability distribution π_θ from which we can sample outputs y. Given a

reward function $R(y)$ that evaluates the quality of an output, the objective is to find parameters θ that maximise the expected reward:

$$J(\theta) = \mathbb{E}_{y\sim\pi_\theta}[R(y)]. \tag{1.81}$$

A direct approach to maximising this objective is to use a policy gradient method known as REINFORCE. We perform gradient ascent on $J(\theta)$ by updating the parameters using the gradient:

$$g = \nabla_\theta J(\theta) = \mathbb{E}_{y\sim\pi_\theta}[\nabla_\theta \ln(\pi_\theta(y))R(y)]. \tag{1.82}$$

This gradient can be estimated from a batch of samples and used in a gradient descent optimiser.

A problem with using the raw reward $R(y)$ is that it can lead to high variance in the gradient estimate and instability during training. For instance, if all actions in a batch receive a high positive reward, the algorithm will try to increase the probability of all of them, even if some were significantly better than others. To address this, the reward is often replaced by an **advantage function**, $A(y)$, which measures how much better a given sample's reward is compared to the expected reward: $A(y_i) = R(y_i) - \mathbb{E}_{y\sim\pi_\theta}[R(y)]$. The expected value, or "baseline", can be estimated by the average reward over the current batch of samples.

Remark 1.83. The variance of this simple baseline estimate can still be high. More advanced methods train a separate "critic" model to provide a learned estimate of the baseline. We do not explore this technique here.

A key challenge with policy gradient methods is that after each parameter update, the gradient estimate $\mathbb{E}_{y\sim\pi_\theta}[\dots]$ becomes invalid because the distribution has changed, requiring a new batch of samples to be generated. To improve sample efficiency, **importance sampling** allows a batch of samples drawn from an old policy $\pi_{\theta_{old}}$ to be reused for several update steps. This is achieved by re-weighting the advantage function with the importance ratio, $r(\theta) = \frac{\pi_\theta(y)}{\pi_{\theta_{old}}(y)}$:

$$\mathbb{E}_{y\sim\pi_\theta}[A(y)] = \mathbb{E}_{y\sim\pi_{\theta_{old}}}[r(\theta)A(y)]. \tag{1.84}$$

This allows us to perform several steps of gradient ascent on the same batch of samples. However, after a few updates, π_θ can diverge significantly from $\pi_{\theta_{old}}$, causing the importance ratio to become unstable and the gradient estimate unreliable. Proximal Policy Optimisation (PPO) addresses this by constraining the updates to a "trust region". This is implemented by clipping the importance ratio in the objective function. If the advantage $A(y)$ is positive, we cap the increase in probability to prevent excessively large updates. This leads to the PPO-Clip objective function:

$$L^{CLIP}(\theta) = \mathbb{E}_{y\sim\pi_{\theta_{old}}}[\min\left(r(\theta)A(y), \text{clip}(r(\theta), 1-\epsilon, 1+\epsilon)A(y)\right)]. \tag{1.85}$$

CHAPTER 2

Problem Statement and Approach

The central problem of this thesis is the optimisation of a real-valued function over the space of manifold triangulations. Formally, given the space of all triangulations of a manifold $\mathcal{M}$, denoted $\mathcal{T}(\mathcal{M})$, and a real-valued function $\mathcal{O} : \mathcal{T}(\mathcal{M}) \to \mathbb{R}$ - referred to as the *objective function* - our goal is to find a triangulation T^* that maximises this function, subject to a constraint on the triangulation's size:

$$T^* = \arg \max_{T \in \mathcal{T}(\mathcal{M}), \text{size}(T) \leq S_{\max}} \mathcal{O}(T). \tag{2.1}$$

As the space $\mathcal{T}(\mathcal{M})$ is combinatorially large and its global structure is not well understood, locating the global maximum via analytic or exhaustive search methods is computationally infeasible. Consequently, this work develops and compares heuristic search techniques designed to find triangulations T for which the value $\mathcal{O}(T)$ is a significant outlier relative to a baseline distribution of values over $\mathcal{T}(\mathcal{M})$.

While this formulation is general, our investigation focuses specifically on objective functions derived from topological and combinatorial properties. In particular, we seek to identify single-vertex triangulations of the 3-sphere, S^3, that contain knots with desirable characteristics, which will be elaborated upon in subsequent sections. To address this, we do the following

a) **Establishing a Baseline:** We first develop a method to quantify what constitutes a "high-value" triangulation. This is achieved by using MCMC sampling to approximate the baseline distribution of the objective function's values across the space of triangulations.

b) **Exploring Classical Optimisation:** We implement and evaluate two classical optimisation techniques: a direct ascent algorithm (an adaptation of greedy search) and simulated annealing. Both of these methods perform a local search on the Pachner graph and are thus limited in their ability to explore the search space globally.

c) **Developing a Generative Model:** We establish a deep learning framework capable of generating triangulations. This generative model serves as a foundation for applying advanced optimisation techniques from reinforcement learning. Furthermore, it enables a hybrid search strategy, where the model proposes diverse starting configurations for local search algorithms, allowing exploration of disparate regions of the Pachner graph.

CHAPTER 3

Search and Sampling Algorithms

3.1. Classical Techniques

3.1.1. Direct Ascent. Direct ascent is a probabilistic greedy search style algorithm designed to find high-value triangulations. For many objective functions, an initial exploration of small triangulations from a census suggests a correlation between the objective value and the size of the triangulation (i.e., the number of tetrahedra). This motivates a search strategy that iteratively increases the triangulation size while biasing the search towards regions of high objective value.

The algorithm begins with a small initial triangulation. At each step, it enumerates all neighbouring triangulations that are larger, and then probabilistically samples one of them, with the selection biased towards neighbours with higher objective scores. This process generates a path of increasing size through the Pachner graph. The entire process can be repeated to generate multiple independent paths, allowing for a broader exploration of the search space.

To implement this search, two components are required:

a) A method to enumerate all larger neighbours of a given triangulation T', which we denote as the set $N_+(T')$.
b) A mechanism to convert the objective function values $\{\mathcal{O}(T) \ : \ T \in N_+(T')\}$ into a selection probability distribution.

To address the first point, we exclusively use (2,3)-Pachner moves to increase the size of the triangulation. A (2,3)-move can be applied across any internal 2-face shared between two distinct tetrahedra. For a triangulation with N tetrahedra, there are $2N$ such internal faces. We can iterate through these faces to generate the set $N_+(T')$. The technical details for applying these moves are described by Altmann and Spreer [**?**].

Remark 3.1. The choice of (2,3)-moves is to maintain a fixed number of vertices. Vertex-increasing (1,4)-moves could also be considered for problems where the vertex count is not constrained.

For the second point, we convert the set of objective scores $\{\mathcal{O}_1, \ldots, \mathcal{O}_k\}$ into a probability distribution $\{p_1, \ldots, p_k\}$. This is a standard problem addressed by the softmax function:

$$p_i = \frac{e^{\beta \mathcal{O}_i}}{\sum_{j=1}^{k} e^{\beta \mathcal{O}_j}}. \tag{3.2}$$

Here, $\beta > 0$ is a parameter that controls the greediness of the selection. It is analogous to an inverse temperature from statistical physics. When $\beta \to \infty$, the algorithm becomes a deterministic greedy

search, always selecting the neighbour with the highest objective value. Conversely, as $\beta \to 0$, the selection becomes uniformly random.

With these components defined, the direct ascent algorithm is detailed in Algorithm 1. The algorithm describes a single run, which can be executed multiple times to explore different ascent paths.

Algorithm 1 Direct Ascent

Let $T_{start} \in \mathcal{T}(\mathcal{M})$ be the initial triangulation.
Let $\beta > 0$ be a fixed selection parameter.
Let $L \in \mathbb{N}$ be the desired chain length.
Let $\mathcal{O} : \mathcal{T}(\mathcal{M}) \to \mathbb{R}$ be the objective function.
Initialize $T_{current} \leftarrow T_{start}$ and $T_{best} \leftarrow T_{start}$.
for $n = 1$ to L **do**
  Generate the set of neighbours $N_+ = \{T_1, \ldots, T_k\}$ of $T_{current}$ by applying all possible (2,3)-moves.
  if N_+ is empty **then**
    break
  end if
  Compute scores $\mathcal{O}_i = \mathcal{O}(T_i)$ for each neighbour $T_i \in N_+$.
  Compute selection probabilities $p_i = \frac{e^{\beta \mathcal{O}_i}}{\sum_j e^{\beta \mathcal{O}_j}}$ for each neighbour.
  Sample T_{next} from N_+ according to the probabilities $\{p_i\}$.
  $T_{current} \leftarrow T_{next}$.
  if $\mathcal{O}(T_{current}) > \mathcal{O}(T_{best})$ **then**
    $T_{best} \leftarrow T_{current}$.
  end if
end for
return T_{best}.

3.1.2. Markov Chain Monte Carlo. For an objective function $\mathcal{O} : \mathcal{T}(\mathcal{M}) \to \mathbb{R}$, we are interested in determining the value of a "typical" triangulation. One way to do this is to adapt Markov chain Monte Carlo (MCMC) to the Pachner graph to generate a sample of the space, allowing us to analyse the statistical properties of $\mathcal{O}$ over this sample. The main issue is that the Pachner graph is infinite, so it is impossible to define a uniform distribution over the entire space. Furthermore, the number of triangulations for a given number of tetrahedra grows rapidly at an unknown rate. A uniform sample across all size levels would be difficult to justify and would be biased towards larger triangulations, simply because more of them exist.

Hence, we modify the problem to the goal of generating a sample that is uniform on a given level. We use a specific algorithm developed by Altmann and Spreer to achieve this [**?**]. The method ensures that the probability of sampling from larger size levels decreases sufficiently quickly to moderate the size of the triangulations sampled. The implemented algorithm is outlined in 2. The sampling procedure is proven to be uniform on each level. To obtain a final sample, the generated

chain is filtered by selecting only those triangulations of a specific size n, yielding a uniform sample of the space of triangulations of that size.

This algorithm represents a single chain of MCMC; typically, multiple chains are run to ensure convergence in accordance with Definition 1.61. The algorithm uses the alternate "i" notation for the Pachner moves where $i = 1$ is a (2-3) move and $i = 2$ is a (3-2) move.

Algorithm 2 MCMC for Triangulations

Let $T_1 \in \mathcal{T}(\mathcal{M})$ be the initial triangulation.
Let $\gamma = 1/k, k \in \mathbb{N}$
Let S be the number of samples
Let $\mathcal{O} : \mathcal{T}(\mathcal{M}) \to \mathbb{R}$ be some objective function.
for $s = 1$ to S **do**
  Sample $u \in U([0, 1])$ uniform
  if $u < e^{-\gamma n(T_s)}$ **then**
    Set $i := 1$, $m := 2n$
  else
    Set $i := 2$, $m := 2n - 2$
  end if
  Enumerate $i-$neighbours $T'_1, ..., T'_\ell$ of T_s
  Sample $v \in U([0, 1])$ uniform
  if $v < \ell/m$ **then**
    Set T_{s+1} to a random choice of $T'_1, ..., T'_\ell$
  else
    Set T_{s+1} to T_s
  end if
end for

3.1.3. Simulated Annealing. The uniform distribution generated by the Markov Chain Monte Carlo (MCMC) algorithm makes it a suitable proposal distribution for simulated annealing. We can therefore directly modify the MCMC algorithm by introducing an acceptance criterion. The process involves proposing a new triangulation and comparing its objective function value to that of the current triangulation. If the new value is better, the move is accepted. If it is worse, the move is accepted with a given probability; otherwise, the algorithm remains at the current triangulation. The probability of accepting a worse move is defined by the function:

$$p(o_p, o_c) = e^{-\beta(o_c - o_p)} \tag{3.3}$$

where o_p and o_c are the objective function values for the proposed and current triangulations, respectively, and $\beta = 1/T$ controls the degree of randomness. Unlike in direct ascent, the value of β varies at each iteration, s, according to a temperature schedule, denoted β_s.

The implemented standard simulated annealing algorithm is described in Algorithm 3.

The success of this algorithm depends on a good choice for β_s. If it is too high, the algorithm may converge prematurely to a local optimum. If it is too low, the search behaves like a random

walk and is no more effective than MCMC. Typically, to manage this trade-off a predefined schedule is used where β starts low and increases - this is ideal for searching for a global optima, and has strong convergence properties. However, because we are not looking for a global optima, but rather a set of proposed optima we use an adapted approach. we define a "target acceptance rate," which is the desired proportion of proposed moves to be accepted. A high acceptance rate encourages exploration, whereas a low rate can lead to stagnation.

During sampling, we track an exponential moving average estimate of the current acceptance rate according to:

$$r_s = (1 - \alpha)r_{s-1} + \alpha a_s \tag{3.4}$$

where $a_s = 1$ if the move was accepted and $a_s = 0$ otherwise. We then compare the moving average acceptance rate, r_s, to the target acceptance rate, r, and update β according to:

$$\beta_s = \beta_{s-1} e^{\lambda(r_s - r)} \tag{3.5}$$

Here, λ represents how aggressively to track the target, and α represents how quickly past acceptance rates are forgotten. This mechanism allows the algorithm to self-tune the β parameter, creating an adaptive simulated annealing method.

Algorithm 3 Simulated Annealing

Let T_0 be some starting triangulation.
Let S be the number of samples.
Let M be some hash table memory.
Let $\mathcal{O} : \mathcal{T}(\mathcal{M}) \to \mathbb{R}$ be some objective function.
Let β_s be some temperature schedule.
for $s = 1$ to S **do**
    Propose T'_s from T_{s-1} by using $1-$step of the described MCMC algorithm.
    Retrieve $o_{s-1} = \mathcal{O}(T_{s-1})$ from M
    Check if $o_s = \mathcal{O}(T_s)$ is in M, if it is - retrieve it, if not - compute it and store it in M.
    Compute $\alpha = e^{-\beta_n(o_{s-1} - o_s)}$
    Generate some $p \sim U([0, 1])$
    if $p \leq \alpha$ **then**
        Accept $T_s = T'_s$
    else
        Let $T_s = T_{s-1}$
    end if
end for

3.2. Transformers

To sample triangulations independently, thereby avoiding the strong correlation inherent in MCMC methods, we implement a GPT-2 style autoregressive transformer trained to generate isomorphism signatures. The model processes each letter of an isomorphism signature as a distinct token. The vocabulary consists of all possible letters in a signature, plus three special tokens: `[BOS]` to mark the beginning of a sequence, `[EOS]` to mark the end, and `[PAD]` to pad sequences to a uniform length for efficient parallel processing in batches. The architecture used is described in Algorithm 5.

The transformer is trained for autoregressive generation. For a complete sequence such as `[BOS]`abc`[EOS]`, the model is given the input `[BOS]`abc and trained to predict the target sequence abc`[EOS]`. For each input token, the transformer outputs a vector of V logits, where V is the vocabulary size. These logits represent the unnormalized log-probabilities for the next token in the sequence. The probability p_i for each token i can be recovered from the logits $\{l_j \mid 0 \leq j < V\}$ using the softmax function:

$$p_i = \frac{e^{l_i}}{\sum_j e^{l_j}} \tag{3.6}$$

The model is trained to minimize the categorical cross-entropy loss, which for the correct token i simplifies to $-\log(p_i)$. This loss approaches zero as the predicted probability of the correct token approaches one, and infinity as it approaches zero.

The model parameters are optimized to minimize this loss function using the AdamW algorithm, described in Algorithm 4. Once the model is trained, a new isomorphism signature can be generated. The process begins with the `[BOS]` token. At each step, the current sequence is passed through the transformer to produce a logit vector for the next token. A token is then sampled from the probability distribution defined by these logits and appended to the sequence. This process is repeated until the `[EOS]` token is generated or a predefined maximum length is reached.

Algorithm 4 AdamW

Let $L : \theta \to \mathbb{R}$ be a loss function to minimise
Let $\beta_1, \beta_2, \eta, \lambda$ be fixed parameters
Let T be the number of steps
Initialise θ_0 randomly
for $t = 1$ to T **do**
  Compute $g_t = \nabla_\theta L(\theta_{t-1})$
  Compute $m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t$ and $v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2$
  Compute $\hat{m}_t = m_t / (1 - \beta_1^t)$ and $\hat{v}_t = v_t / (1 - \beta_2^t)$
  Update $\theta_t = \theta_{t-1} - \eta \left(\frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \varepsilon} + \lambda \theta_{t-1} \right)$
end for

Algorithm 5 IsoSig Transformer

Let V be the vocabulary size
Let $d_e \in \mathbb{N}$ the embedding dimension
Let $\ell \in \mathbb{N}$ be the number of layers
Let $h \in \mathbb{N}$ be the number of heads
Let $r \in [0, 1]$ be the dropout rate.
Let w be some partially complete isomorphism signature of size $l \geq 1$
Compute `tok_emb`$\in \mathbb{R}^{l \times d_e}$ as the token embedding of each letter in w
Compute `pos_emb`$\in \mathbb{R}^{l \times d_e}$
Compute $X =$`tok_emb`$+$`pos_emb`$\in \mathbb{R}^{l \times d_e}$
Compute $X = DO(X, r) \in \mathbb{R}^{l \times d_e}$
for $i \in [1, ..., \ell]$ **do**
    $X = TB(X, h, r) \in \mathbb{R}^{l \times d_e}$
end for
Compute $X = LN(X) \in \mathbb{R}^{l \times d_e}$
Return `logit`$= MX \in \mathbb{R}^{l \times V}$

CHAPTER 4

Numerical Experiments

We conduct a series of numerical experiments. The associated code can be found in the corresponding git repository [**?**].

4.1. Objective Functions

To test the effectiveness of our optimisation strategies, we consider five objective functions defined on the set of all single-vertex triangulations of the 3-sphere, denoted $\mathcal{T}_1(S^3)$.

4.1.1. Alexander Polynomial. For any triangulation $T \in \mathcal{T}_1(S^3)$, each edge e_α forms a closed loop. This loop can be viewed as a knot, K_α, embedded in S^3. The complexity of the Alexander polynomial, $\Delta_{K_\alpha}(t)$, is often related to the geometric complexity of the knot. Let the coefficient vector of $\Delta_{K_\alpha}(t)$ be $\vec{a} = [a_0, a_1, ..., a_n]$. This connection suggests three objective functions:

a) $\mathcal{O}_{deg} : T \mapsto \sum_{e_\alpha} \deg(\Delta_{K_\alpha}(t))$
b) $\mathcal{O}_{det} : T \mapsto \sum_{e_\alpha} |\Delta_{K_\alpha}(-1)|$
c) $\mathcal{O}_{norm} : T \mapsto \sum_{e_\alpha} \|\vec{a}\|^2$

The first objective, $\mathcal{O}_{deg}$, is motivated by the inequality $2g(K) \geq \deg(\Delta_K(t))$, where $g(K)$ is the Seifert genus of the knot, a measure of its complexity. The degree is the difference between the highest and lowest powers of t in the polynomial, a definition used because the Alexander polynomial is only unique up to multiplication by $\pm t^k$. The second objective, $\mathcal{O}_{det}$, uses the knot determinant, $|\Delta_K(-1)|$, which is a powerful invariant related to knot colorability. The third, $\mathcal{O}_{norm}$, is the squared Euclidean norm of the coefficient vector, providing a standard measure of the polynomial's overall magnitude.

4.1.2. Fundamental Group of the Knot Complement. For each edge e_α forming a knot K_α in $T \in \mathcal{T}_1(S^3)$, we can study the topology of its complement. This approach inspires our fourth objective function, which is based on the fundamental group $\pi_1(M_\alpha)$:

d) $\mathcal{O}_{gen} : T \mapsto \sum_{e_\alpha} \#_{gen}(\pi_1(M_\alpha))$ where $\#_{gen}$ is the number of generators in the presentation of the fundamental group.

Ideally, one would use the number of generators in a minimal presentation of the fundamental group. However, determining this is algorithmically undecidable. Therefore, we use the number of generators from the standard presentation returned by the software Regina [**?**] as a heuristic. A subsequent reduction process can be attempted on the resulting presentations to check if simplification is possible.

4.1.3. Edge Degree Variance. The previous four objective functions are topological, inspired by the knot structures within the triangulations. We also consider a fifth, combinatorial objective function related to the structure of the triangulation itself. This function measures the uniformity of the edge degrees, where the degree of an edge is the number of tetrahedra incident to it.

e) $\mathcal{O}_{var} : T \mapsto \mathrm{Var}_{e_\alpha}(\deg(e_\alpha))$

The choice of variance is justified as follows. For any triangulation in $\mathcal{T}_1(S^3)$, the number of vertices is fixed at $V = 1$. The Euler characteristic of the 3-sphere is $\chi(S^3) = V - E + F - N = 0$, where E, F, and N are the number of edges, faces, and tetrahedra, respectively. In a closed 3-manifold, each face is shared by two tetrahedra, so $F = 2N$. Substituting these into the Euler characteristic formula yields $1 - E + 2N - N = 0$, which implies $E = N + 1$. The sum of all edge degrees is $6N$, since each of the N tetrahedra has 6 edges. Therefore, the mean edge degree is $\frac{6N}{E} = \frac{6N}{N+1}$. Since the mean edge degree is constant for a fixed number of tetrahedra N, the variance of the edge degrees serves as a measure of how irregular the triangulation is. A low variance indicates a more uniform structure.

4.2. Markov Chain Monte Carlo

To establish a baseline, we used a Markov Chain Monte Carlo (MCMC) simulation to explore the distribution of each objective function over the Pachner graph. The algorithm, as described in section 3.1.2, samples uniformly from the set of triangulations with a number of tetrahedra close to a target value. We focused on the region around 30-tetrahedra triangulations, a size that is computationally intractable to enumerate exhaustively but has been shown to contain topologically interesting examples [**?**]. We ran seven parallel MCMC chains, collecting 10,000 samples from each chain with an interval of 100 steps between samples. For each of the resulting 70,000 samples, all five objective functions were calculated. This approach saves computational time and provides a unified dataset to assess correlations between the objective functions. Convergence of the chains was confirmed using the Gelman-Rubin statistic ($\hat{R}$) (Definition 1.61), with all values below the recommended threshold of 1.01 [**?**], as shown in Table 1. Figure 1 displays the correlation matrix for the objective functions across all samples. This represents the average correlation; relationships between the functions may differ significantly in specific regions of the state space, such as near their respective maxima.

From the total 70,000 samples generated, we filtered for those with exactly 30 tetrahedra. This yielded a subset of 8,250 triangulations, corresponding to a sampling efficiency of $8,250/70,000 \approx 11.8\%$. The following analyses are performed exclusively on this subset.

Metric	$\hat{R}$
$\mathcal{O}_{norm}$	1.003
$\mathcal{O}_{deg}$	1.004
$\mathcal{O}_{det}$	1.005
$\mathcal{O}_{var}$	1.006
$\mathcal{O}_{gen}$	1.009

TABLE 1. Convergence diagnostics for MCMC runs, displaying the Gelman-Rubin statistic ($\hat{R}$) for each run.

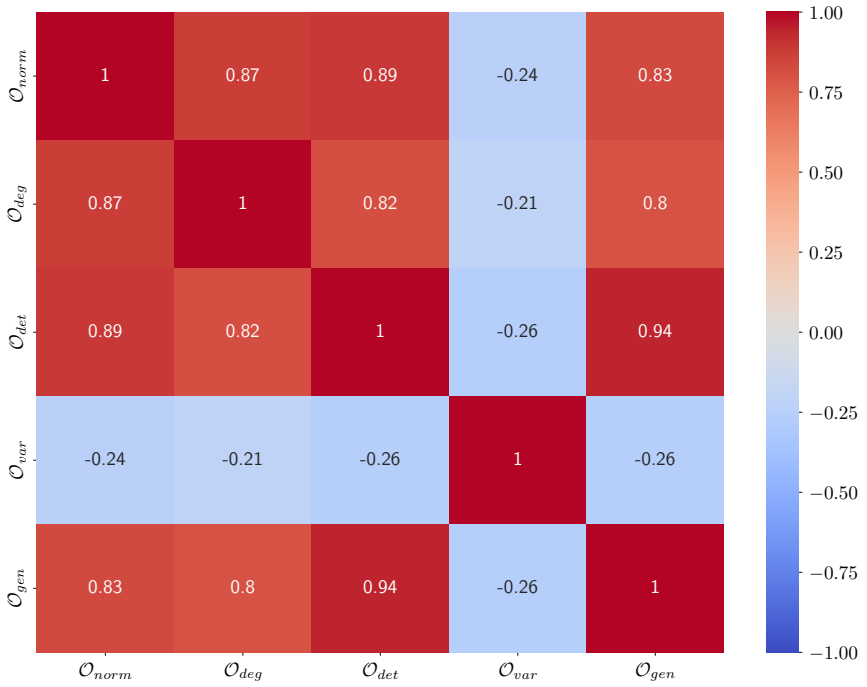
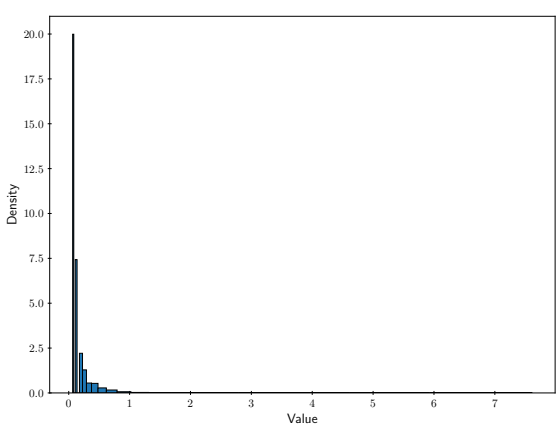


FIGURE 1. Correlation matrix heatmap illustrating the pairwise relationships between objective functions.

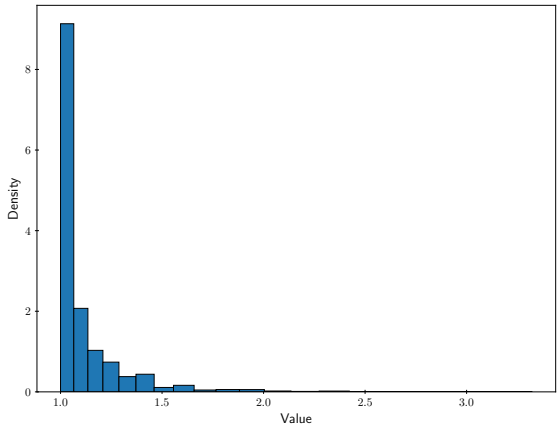
Figure 2 displays the distribution of scores for each objective function outlined in Section 4.1.

All objective functions except for $\mathcal{O}_{var}$ appear to exhibit power-law distributions or power-law tails. To verify this, we used maximum likelihood estimation (MLE) to fit a power-law distribution in two ways: first to the entire dataset, and second to the tail of the distribution beginning at an optimal lower bound, $x_{\min}$, determined using the method of Clauset, Shalizi, and Newman [**?**]. Figure 3 presents log-log histograms of the data overlaid with these fitted distributions.

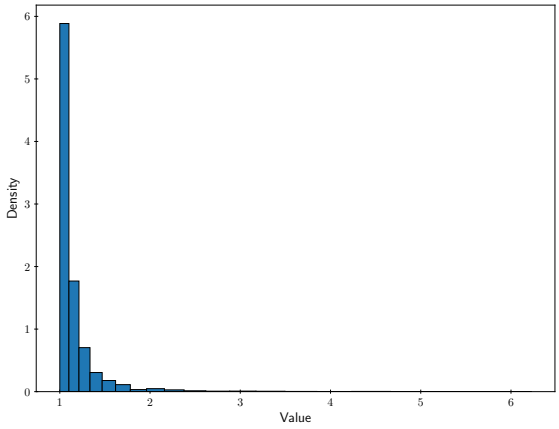
The distribution for $\mathcal{O}_{var}$ is positively skewed. While a chi-squared distribution might be expected for a variance measure, it provided a poor fit to the data. Instead, a log-normal distribution was found to model the data well, particularly its tail behaviour. This choice is justified empirically by the strong fit shown in the log-log plot in Figure 4.



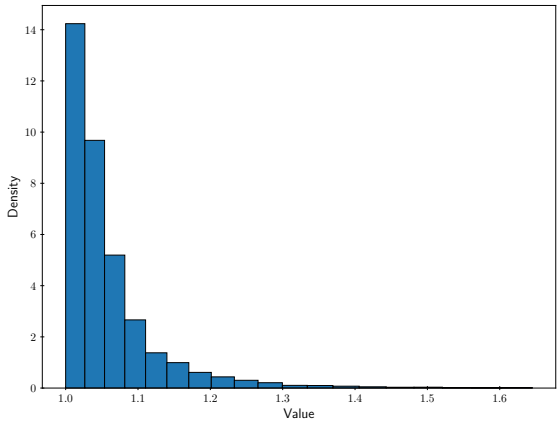
(A) $\mathcal{O}_{deg}$



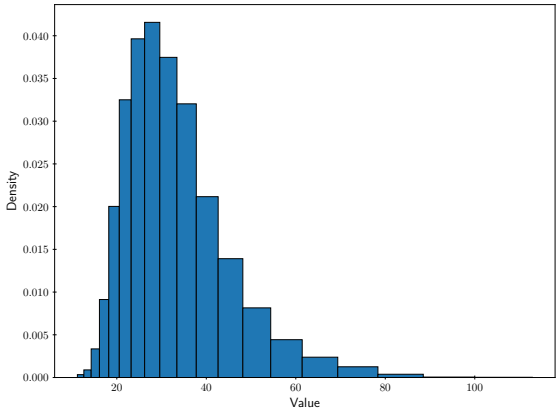
(B) $\mathcal{O}_{det}$



(C) $\mathcal{O}_{norm}$

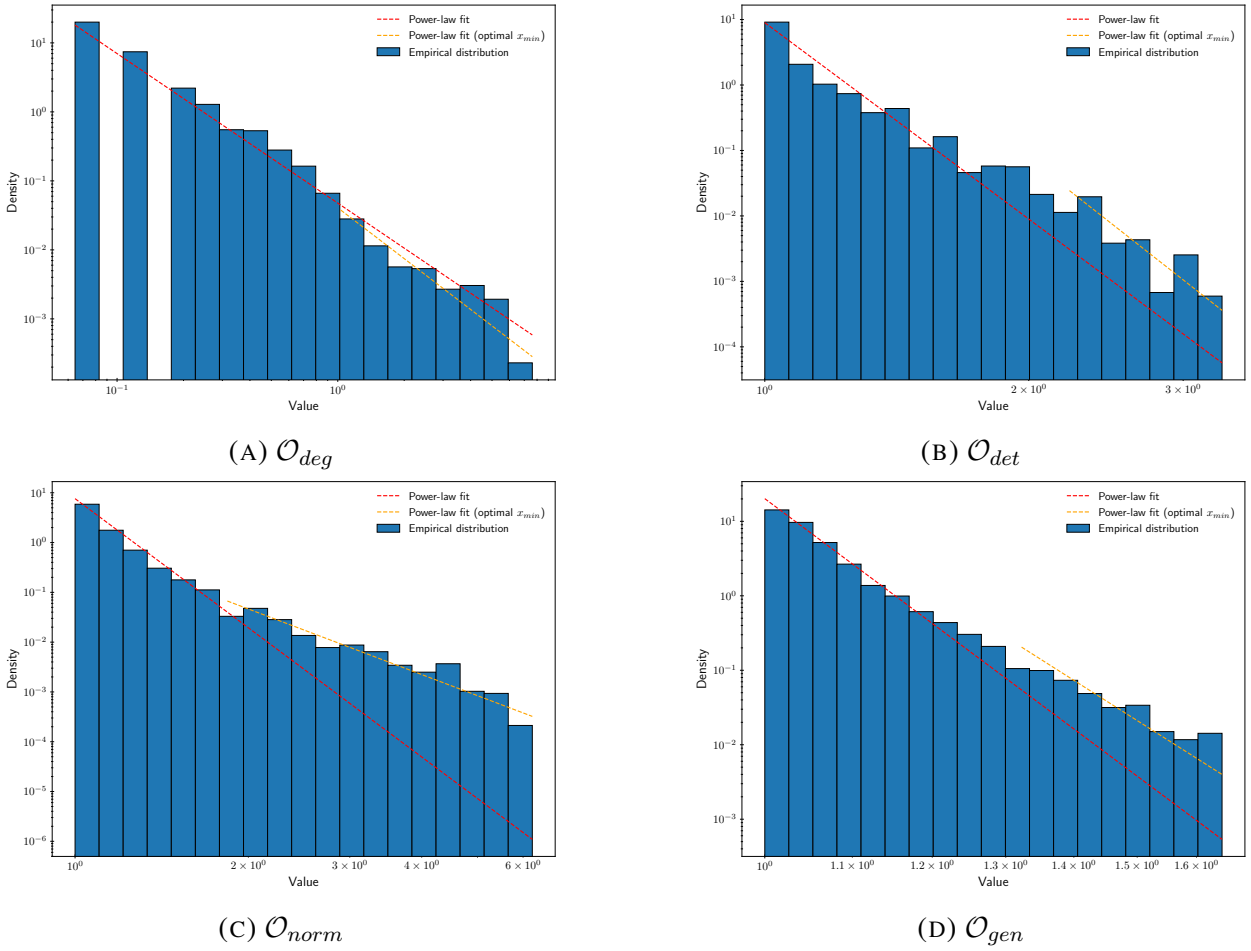


(D) $\mathcal{O}_{gen}$



(E) $\mathcal{O}_{var}$

FIGURE 2. Distributions of scores for each objective function, generated from MCMC samples at $\gamma = 1/10$.



(A) $\mathcal{O}_{deg}$

(B) $\mathcal{O}_{det}$

(C) $\mathcal{O}_{norm}$

(D) $\mathcal{O}_{gen}$

FIGURE 3. Log-log histogram of the score for each objective function under MCMC samples at $\gamma = 1/10$. The red line indicates the fitted power law distribution over the entire dataset; the orange line indicates the fitted power law distribution for $x > x_{\text{min}}$.

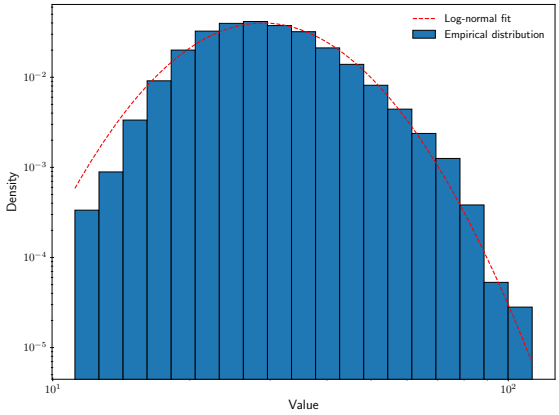


FIGURE 4. Log-log plot of the distribution for the O_{var} objective function, with the fitted log-normal distribution overlaid.

4.3. Classical Optimisation

4.3.1. Direct Ascent. We perform direct ascent as described in Section 3.1.1, starting from the two-tetrahedron, single-vertex seed triangulation `cMcabbgqs`. To determine an optimal temperature, we conducted 20 independent runs at five different temperatures ($\beta \in \{0.1, \sqrt{0.1}, 1, \sqrt{10}, 10\}$) and compared the maximum score achieved at each number of tetrahedra. An example for $\mathcal{O}_{deg}$ is shown in Figure 5. In all cases, a lower temperature (higher β) produced better results, indicating that a purely greedy ascent - where the neighbour with the highest score is accepted at each step - is the most effective strategy for this problem space.

Consequently, we performed a single greedy ascent for each objective function, proceeding for 28 steps to generate a sequence of triangulations up to 30 tetrahedra. The resulting optimisation paths are presented in Figure 6.

The ascent profile for each objective function is nearly deterministic and demonstrates monotonic growth. The score for $\mathcal{O}_{deg}$ shows accelerated growth, while $\mathcal{O}_{det}$, $\mathcal{O}_{var}$, and $\mathcal{O}_{norm}$ exhibit approximately linear growth. This trend suggests that higher scores may be achievable by simply continuing the ascent to triangulations with more tetrahedra. In contrast, the score for $\mathcal{O}_{gen}$ appears to plateau, suggesting this greedy approach may yield diminishing returns for this objective at larger sizes. Interestingly, the path for $\mathcal{O}_{var}$ displays a slight oscillating behaviour, where scores for triangulations with an even number of tetrahedra are consistently higher than those for triangulations with an odd number of tetrahedra, relative to the overall linear trend.

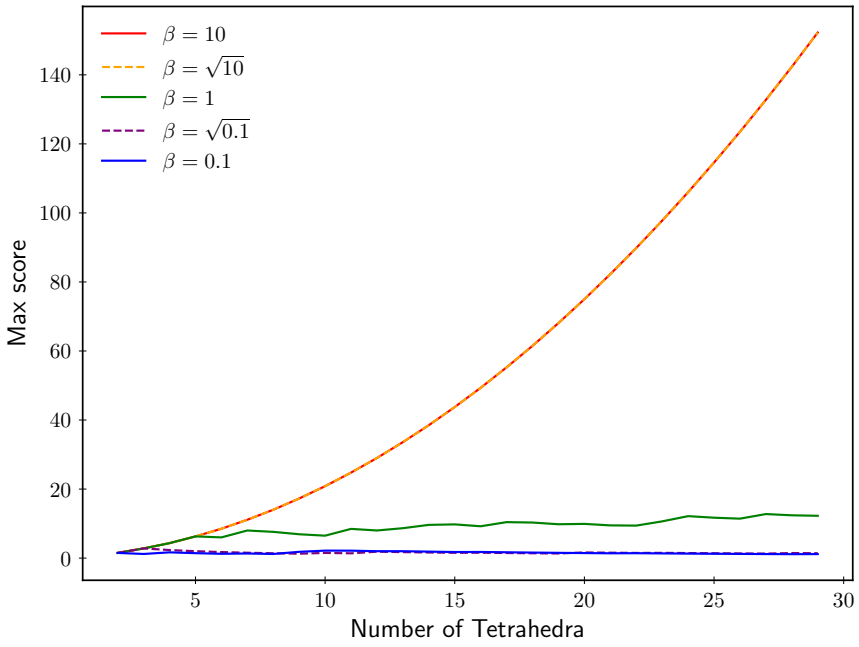
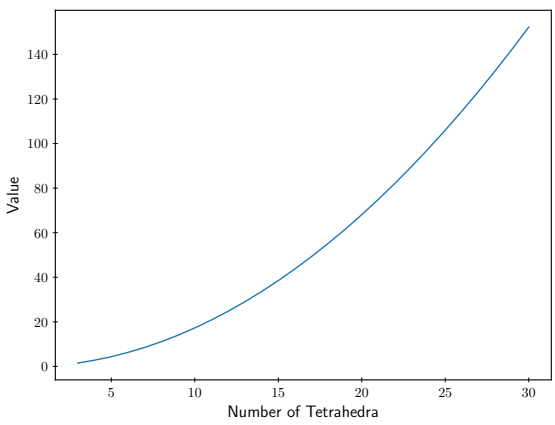
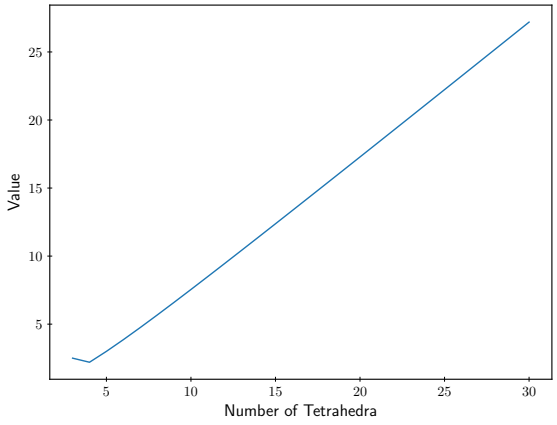


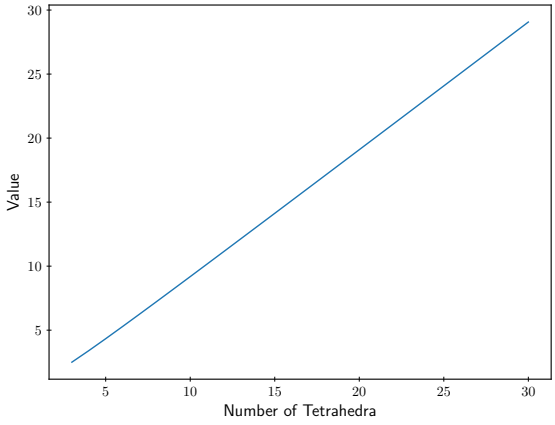
FIGURE 5. Direct Ascent paths for the O_{deg} objective function at different inverse temperatures ($\beta = 1/T$).



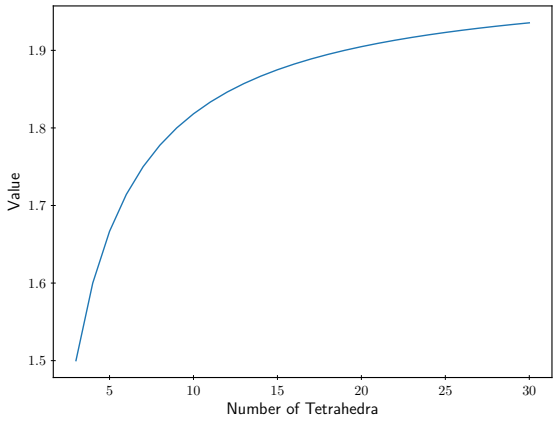
(A) $\mathcal{O}_{deg}$



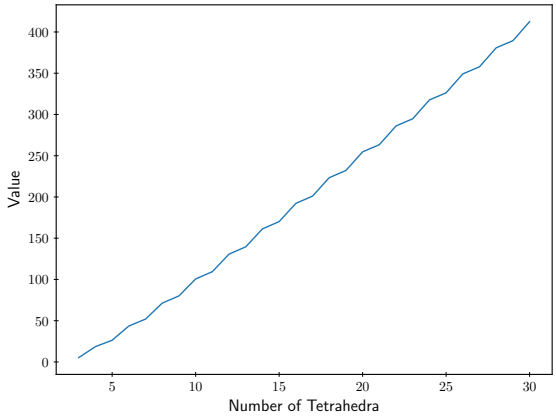
(B) $\mathcal{O}_{det}$



(C) $\mathcal{O}_{norm}$



(D) $\mathcal{O}_{gen}$



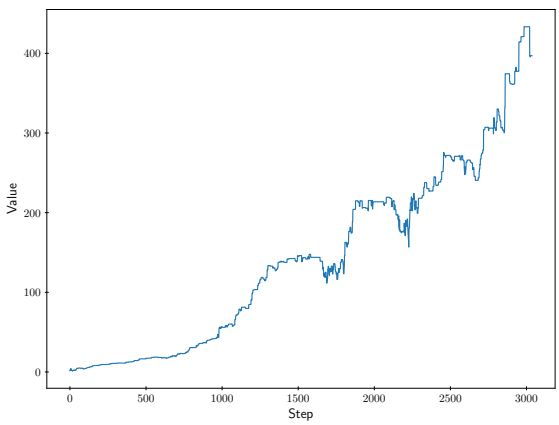
(E) $\mathcal{O}_{var}$

FIGURE 6. Direct Ascent paths, showing score versus iteration for each objective function.

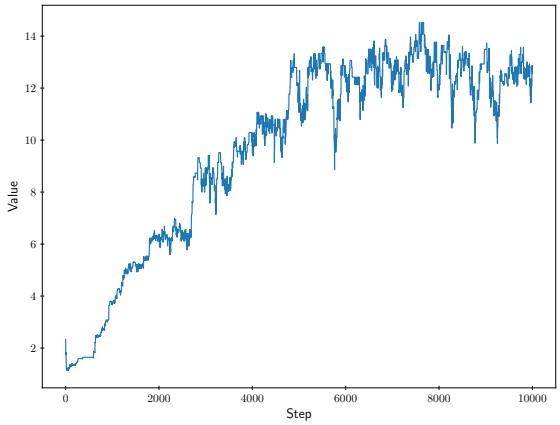
While direct ascent is simple to implement and performs predictably, it has significant limitations. The algorithm generates only a single high-scoring triangulation, and it is inefficient; generating the final 30-tetrahedron triangulation required exploring approximately 1,000 candidate triangulations along the path. Therefore, while this technique is valuable for finding a single exemplar, it is unsuitable for efficiently exploring a diverse set of high-scoring triangulations.

4.3.2. Simulated Annealing. We perform simulated annealing as described in Section 3.1.3. Each run was configured for $10,000$ iterations with an interval of 10 steps, a target acceptance rate of 20%, $\alpha = 0.01$, and $\lambda = 0.1$. The 20% acceptance rate was chosen heuristically based on preliminary experiments. To ensure a fair comparison with direct ascent, a hard constraint was imposed by setting the potential to $-\infty$ for any proposed triangulation with more than 30 tetrahedra. For objectives $\mathcal{O}_{deg}$ and $\mathcal{O}_{norm}$, the runs were terminated early (at 3,307 and 2,940 samples, respectively) because the time required to score a single triangulation became prohibitively long (> 10 seconds). The achieved acceptance rates for $\mathcal{O}_{det}$, $\mathcal{O}_{gen}$, and $\mathcal{O}_{var}$ were all within 1% of the target. The rates for $\mathcal{O}_{deg}$ and $\mathcal{O}_{norm}$ were lower, likely a consequence of the reduced number of samples from early termination. The optimisation paths and final acceptance rates for each objective function are shown in Figure 7.

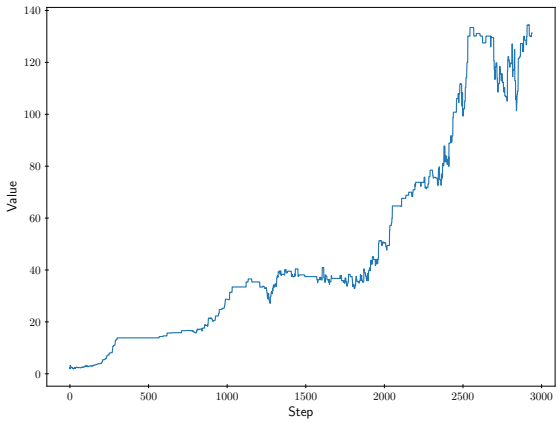
For $\mathcal{O}_{deg}$ and $\mathcal{O}_{norm}$, the objective function values were still increasing at the time of termination, suggesting higher scores could have been reached if the runs were allowed to complete. However, this trend is coupled with a practical limitation: as the algorithm finds more complex knots, the time required to compute the Alexander polynomial increases. This slowdown occurs because the calculation involves finding the symbolic determinant of a large matrix, which is a computationally intensive task. Future work could potentially accelerate this process by developing specialized algorithms to compute only the required properties (e.g., degree or norm) without the full symbolic polynomial, possibly leveraging the parallelism of GPUs. In contrast, the runs for $\mathcal{O}_{gen}$ and $\mathcal{O}_{var}$ appeared to plateau after approximately 2,000 iterations, with similar, though less conclusive, behaviour observed for $\mathcal{O}_{det}$.



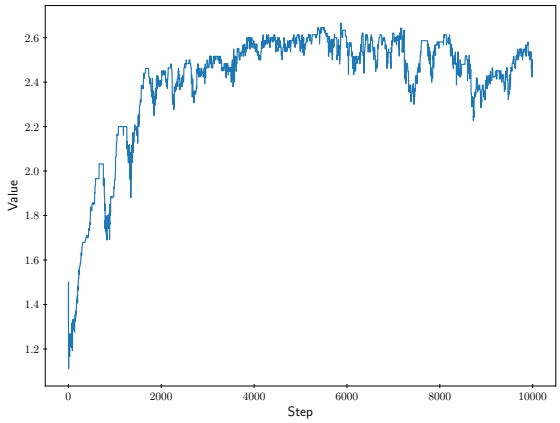
(A) $\mathcal{O}_{deg}$ acceptance rate: 17.78%



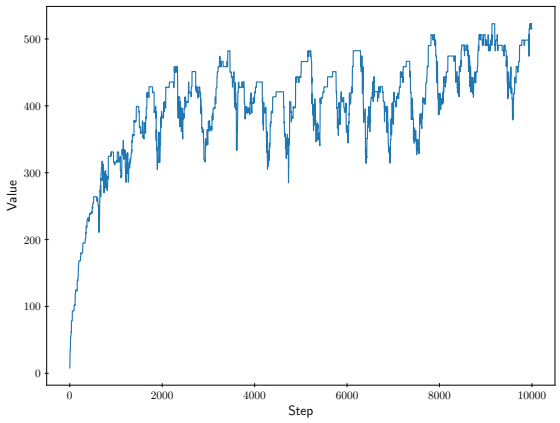
(B) $\mathcal{O}_{det}$ acceptance rate: 19.96%



(C) $\mathcal{O}_{norm}$ acceptance rate: 18.44%



(D) $\mathcal{O}_{gen}$ acceptance rate: 20.11%



(E) $\mathcal{O}_{var}$ acceptance rate: 19.44%

FIGURE 7. Trace plots for simulated annealing on each objective function for 10,000 iterations and a 20% acceptance rate.

4.4. Comparison of Classical Optimisation

The best results from each optimisation technique are compiled in Table 2. This table includes the **exceedance probability** (the probability of a random sample achieving a higher score) of the best sample, computed using the fitted power-law (for both the overall distribution and the tail) or log normal distributions from Section 4.2.

In all cases, the search techniques uncovered samples that were in the top 10^{-5} of the distribution. To achieve a comparably rare result with MCMC would require approximately 100,000 samples of 30-tetrahedra triangulations. Given our observed sampling efficiency of $\approx 12\%$, this would necessitate at least 1,000,000 MCMC steps. In contrast, simulated annealing used 10,000 steps and direct ascent required approximately 1,000 objective function evaluations, making these classical search techniques at least 100 times more efficient. In the best-case scenario ($\mathcal{O}_{det}$), the score was in the top 10^{-13} of the distribution, representing an efficiency gain of at least 10^{10} over MCMC.

With the exception of $\mathcal{O}_{det}$, simulated annealing performed better than greedy ascent. This improved performance came at the cost of requiring 2 to 5 times more objective function evaluations than direct ascent. However, the resulting improvement of 1-2 orders of magnitude suggests that simulated annealing is the superior choice when the primary goal is to maximise the objective function. The increased algorithmic complexity and slower runtime may be drawbacks if an absolute maximum is not required. Interestingly, $\mathcal{O}_{det}$ performed significantly better under direct ascent, achieving a score in the top 10^{-13} of the distribution. It is possible that the objective function landscape for $\mathcal{O}_{det}$ is sufficiently convex or has few local optima, making it particularly suited to a greedy strategy.

For the knot-based objective functions, we analyse the best triangulations found by either technique, identifying the properties of the edge corresponding to the most complex knot.

For $\mathcal{O}_{deg}$, the edge corresponding to the knot with the greatest Alexander polynomial degree had a degree of $1,626$. Its fundamental group was not of the form $\langle a, b \mid a^p b^{-q} \rangle$ and its Alexander polynomial was irreducible over $\mathbb{Q}$.

For $\mathcal{O}_{det}$, the edge with the greatest determinant had a value of 57. The Alexander polynomial factors to $(t^4{-}t^3{+}t^2{-}t{+}1)(t^{20}{-}t^{15}{+}t^{10}{-}t^5{+}1)$ and the fundamental group was $\langle a, b \mid a^{12}ba^{-13}b \rangle$. In total for this triangulation, 3 of the edges were unknots.

For $\mathcal{O}_{norm}$, the edge with the greatest norm had a value of 359. Its Alexander polynomial had a norm of 724 and its fundamental group had three generators. The polynomial's alternating $+1, -1$ coefficients are the same as a a torus knot. In total for this triangulation, 3 of the edges were unknots.

For $\mathcal{O}_{gen}$, the edge corresponding to the knot with the most complex fundamental group initially presented with 51 generators, which simplified to 3. Its Alexander polynomial had a span of

162 and featured alternating $+1, -1$ coefficients, again the same as a torus knot. In total for this triangulation, 4 of the edges were unknots.

For $\mathcal{O}_{var}$, the best triangulation had edges with degrees concentrated at $1, 2,$ or 3, but with one outlier edge of degree 131. For this triangulation, all edges were unknotted.

Obj. Function	Direct Ascent			Simulated Annealing		
	Score	Total Exceedance Prob.	Tail Exceedance Prob.	Score	Total Exceedance Prob.	Tail Exceedance Prob.
$\mathcal{O}_{deg}$	152.26	$7.19 \cdot 10^{-05}$	$1.21 \cdot 10^{-05}$	433.42	$2.12 \cdot 10^{-05}$	$2.66 \cdot 10^{-06}$
$\mathcal{O}_{det}$	27.19	$1.38 \cdot 10^{-13}$	$2.62 \cdot 10^{-13}$	13.84	$5.86 \cdot 10^{-11}$	$1.61 \cdot 10^{-10}$
$\mathcal{O}_{norm}$	29.06	$7.16 \cdot 10^{-12}$	$3.29 \cdot 10^{-06}$	152.26	$2.38 \cdot 10^{-17}$	$1.23 \cdot 10^{-08}$
$\mathcal{O}_{gen}$	1.94	$1.58 \cdot 10^{-06}$	$2.32 \cdot 10^{-05}$	2.65	$2.93 \cdot 10^{-09}$	$1.14 \cdot 10^{-07}$
$\mathcal{O}_{var}$	412.54	$3.06 \cdot 10^{-15}$		522.93	$7.52 \cdot 10^{-18}$	

TABLE 2. Comparison of maximum scores achieved by Direct Ascent and Simulated Annealing. Includes the exceedance probability for each score, calculated relative to the fitted bulk ('Total') and tail power-law ('Tail') distributions. For $\mathcal{O}_{gen}$, the exceedance probability is calculated relative to the fitted log-normal distribution.

	Best Triangulation (`IsoSig`)
$\mathcal{O}_{deg}$	`EMzMzMzMzMvAAMzMzMzMPabcdefghijklmnqpqrstuvwxyzABCDDjxxxxxxxxxxxxxlfxfxxxxxxxxxxxj`
$\mathcal{O}_{det}$	`ELLMMzMzMzMzMzMzMzMzQbeddfghijklmnopqrstuvwxyzABCDDDxxrfaxxxxxxxxxxxxxxxxxxxxxxxxxvc`
$\mathcal{O}_{norm}$	`EMzMzMzMzMvAAMzMzMzMPabcdefghijklmnqpqrstuvwxyzABCDDjxxxxxxxxxxxxxlfxfxxxxxxxxxxxj`
$\mathcal{O}_{gen}$	`ELwvMvQPLMzAPLLAPMzPQcbehgklimllpnqptrutwvzxyABDCBDDaacmvlpvlooashoqkmbuoageahamkcg`
$\mathcal{O}_{var}$	`EMzMzMzMzMzMvAAMzMzMPabcdefghijklmnopqsttuvwxyzABCDDvaaaaaaaaaaaaaaaaciqnaaaaaaaaav`

TABLE 3. Isomorphism signature of the optimal triangulation found using Direct Ascent.

	Best Triangulation (`IsoSig`)
$\mathcal{O}_{deg}$	`ELvLLzvMvzMQAMLMQzQMQbdgfjpmtosmvruqptvwuzyxBCzACBDDafgfjxaovwrdsifjouitrwblqjfhroj`
$\mathcal{O}_{det}$	`ELLvMLAAzPwMzAPzvAMQQadeihhhljnplnqpturttxBBCADBAADDbabagjgxbadjbhohagkfauahoxrccjo`
$\mathcal{O}_{norm}$	`EvLPzvvzLAvzLPPQQMQQQdcdgklqsvoAptytAzBxxzCwyBBDzADDonnapdeaacqowdbauqadasckkjabvhd`
$\mathcal{O}_{gen}$	`ELMvLAAvMAQzLwPLAMMMPcbdgfiihnmloopnqtsvtxwyzyABACDDaaaavlsvaccfbxsaqranmglanxxnxxr`
$\mathcal{O}_{var}$	`EMLLLvzPPzwPLMQzAPzAQabddglnjokkqnouqswstuvwzyzBCDCDgagbaaagacvaggaccavgfbcafcoaaco`

TABLE 4. Isomorphism signature of the optimal triangulation found using Simulated Annealing.

4.5. Baseline Transformer Efficiency on Isomorphism Signatures

A dataset of single-vertex triangulations of S^3 was collected for sizes ranging from 6 to 12 tetrahedra, sourced from the census of all small triangulations of 3-manifolds [**?**] and MCMC sampling. A separate transformer model following the architecture described in Section 3.2 was trained for each tetrahedron count, all sharing identical parameters: an embedding dimension of 64, 6 layers, 8 attention heads, and a dropout rate of 0.1. Each model was initialised with the same random seed and trained for 50,000 iterations, where one iteration corresponds to a single batch of 32 triangulations randomly sampled from the training data. Training was done with AdamW (Algorithm 4) with $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\lambda = 0.01$. A fixed, random test set of 1,000 triangulations was used for validation. These parameters were chosen as a computationally reasonable baseline, although research suggests that larger models typically achieve better performance, particularly when a virtually unlimited amount of training data can be generated [**?**].

Figure 8 shows a log-log plot of the test loss versus the number of training iterations. The loss curves are still decreasing at 50,000 iterations with insufficient curvature to predict a plateau, suggesting that additional training would yield further improvements. Despite the limited training and small architecture, the models demonstrate effective generation capabilities.

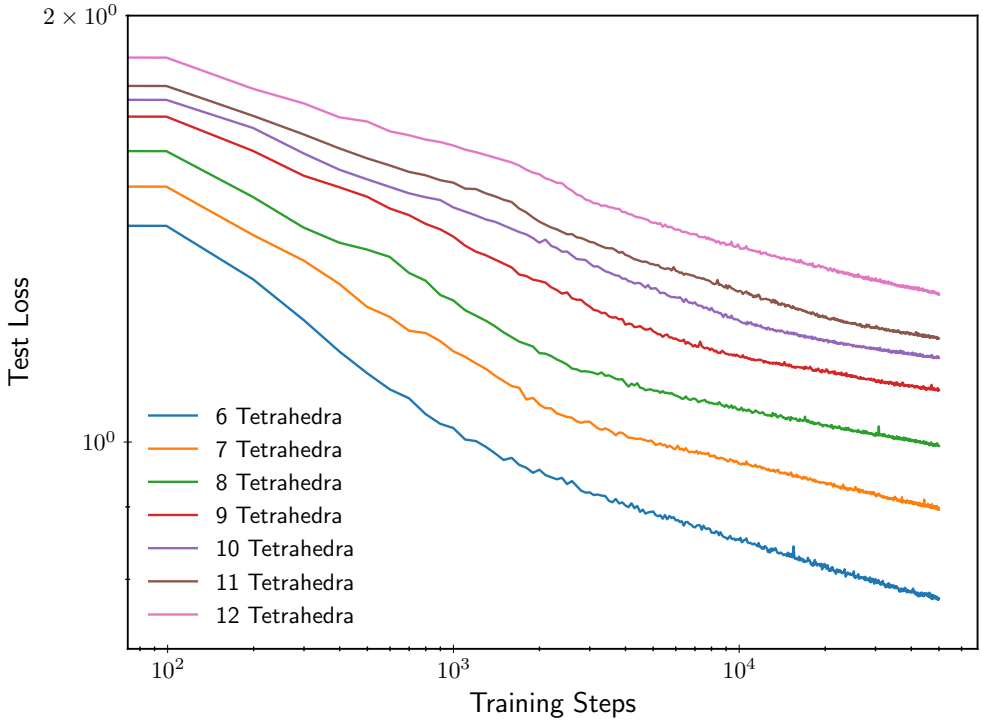


FIGURE 8. Log-log plot of the training loss versus iterations for the transformer models.

Figure 9 and Table 5 show the generation efficiency for the transformers. The success of a generated sequence is evaluated against four criteria:

- **Loads:** The generated string is correctly formatted and can be parsed by Regina.
- **Valid:** The string represents a valid combinatorial manifold without singularities.
- **Closed:** The manifold is compact and has no boundary.
- **Sphere:** The manifold is a 3-sphere, confirmed by a trivial fundamental group.

Of the strings that load successfully, nearly all represent valid triangulations. Of the valid triangulations that are closed, approximately 70% are 3-spheres. A key trend observed is that the fraction of valid manifolds that are also closed decreases as the number of tetrahedra increases.

# Tetrahedra	Loads	Valid	Closed	Sphere
6	0.195	0.195	0.115	0.086
7	0.08	0.08	0.039	0.0235
8	0.043	0.0425	0.02	0.0115
9	0.0185	0.0185	0.004	0.004
10	0.008	0.008	0.0025	0.001
11	0.0075	0.0075	< 0.001	< 0.001
12	0.001	< 0.001	< 0.001	< 0.001

TABLE 5. Generation efficiency of the transformer model as a function of the number of tetrahedra.

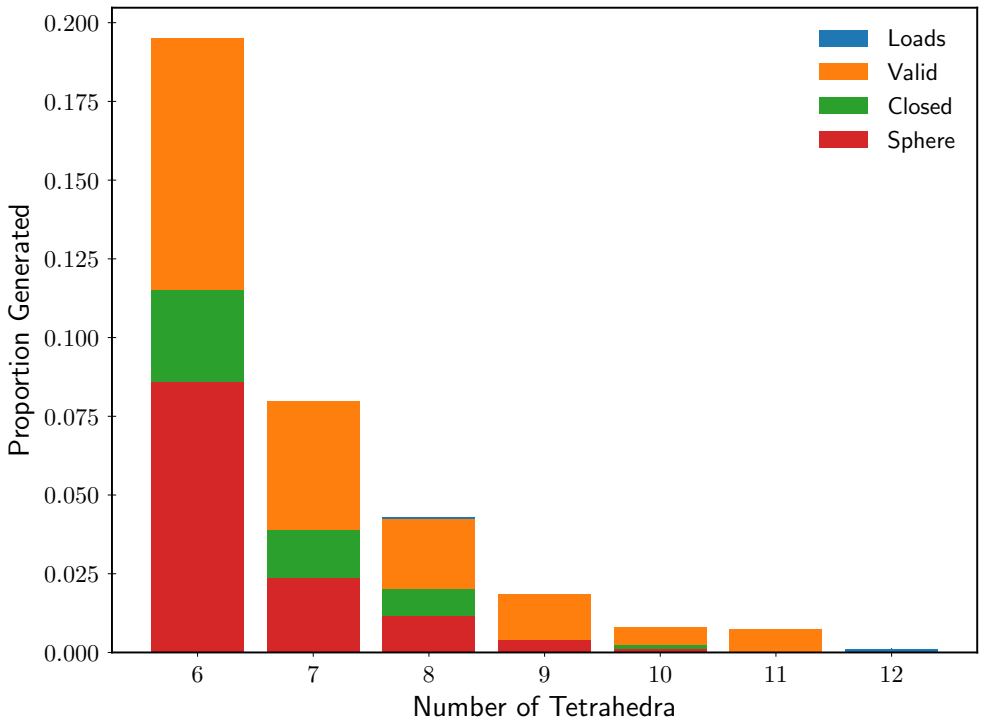


FIGURE 9. Generation efficiency of the transformer model as a function of the number of tetrahedra.

CHAPTER 5

Discussion and Conclusion

5.1. Discussion

This study examines single-vertex triangulations of the 3-sphere. We successfully implemented and compared classical heuristic search algorithms against a baseline established with MCMC. The primary result is that classical heuristics can find exceptional triangulations, often with knot-theoretic properties falling within the top 10^{-10}th of the distribution or rarer compared to the baseline distribution. The general success of simulated annealing over greedy ascent strongly suggests that the Pachner graph, as an objective function landscape, possesses a non-convex structure with numerous local optima. To navigate this landscape efficiently, a degree of randomness is necessary to escape these optima and discover globally superior solutions.

While these optimisation techniques are powerful, their success is intrinsically linked to the chosen objective function. During development, we investigated whether guided search could discover specific triangulations, such as one where all edges are knotted, an example of which was previously found by Burton using a targeted search [**?**]. The hypothesis was that a triangulation exhibiting high "knottedness" in general might be more likely to have this property. However, we found that our optimisation methods could produce triangulations with much higher objective function scores than Burton's example, yet without the desired all-knotted-edge property. This is not a failure of the optimisation technique but highlights a limitation in the objective function's definition; it did not sufficiently discriminate for the desired feature. If these search techniques are to be used to find specific topological structures, they must be paired with an objective function known to be uniquely optimised by a hypothetical solution. For instance, an objective function defined as "the number of unknotted edges" is tautologically minimised by the target triangulation. However, its landscape is too flat, with a very low gradient, rendering it ineffective for guiding a search.

The main limitations of the presented classical search techniques are twofold. Firstly, they are local search methods that generate correlated chains of samples. This can lead to extended periods of exploring regions with similar objective function values, thereby wasting computational resources without gaining new information. While we lay the groundwork for deep learning approaches that may combat this, further work is needed to validate their viability. Secondly, the objective functions become increasingly computationally intensive as the solutions become more optimal. This is somewhat unavoidable, but there is potential inefficiency in the current process. For

example, significant computational effort is spent determining the full symbolic Alexander polynomial, only for it to be reduced to a scalar value. Optimising this calculation could significantly reduce runtime and allow for greater parallelism on modern GPU hardware.

To address the sequential limitation of classical search, we investigated generative machine learning, specifically autoregressive transformers trained on isomorphism signatures. We have demonstrated a proof of concept with these models on small triangulations but observed diminishing success on larger ones. This is an expected result, and there is strong evidence that performance can be improved with longer training times and larger models. The necessity for increased training time is a bottleneck for developing models capable of generating large, complex triangulations. To make reinforcement learning a viable next step, the initial generative accuracy must exceed a minimum threshold, a condition currently met only for small triangulations.

5.2. Conclusion

This report has provided a range of approaches to navigating the Pachner graph. We have successfully demonstrated that classical search heuristics dominate uniform random sampling for uncovering exceptional triangulations with complex knot structures. The methods provide a practical, objective-function-agnostic approach to exploring the Pachner graph that is easy to implement. Furthermore, we have laid the groundwork for deep learning applications, successfully demonstrating a proof-of-concept generative transformer that can produce valid triangulations. While challenges remain with the efficiency of these generative techniques, there is evidence that their performance can be improved to the level required for reinforcement learning - a technique we believe holds great potential for success in this domain.

5.3. Future Work

The results and limitations discussed above suggest several avenues for future work, primarily centered on improving the generative model and applying reinforcement learning to determine if it can outperform simulated annealing.

5.3.1. Advancing the Generative Model. The immediate priority is to improve the generative efficiency of the model. Two paths forward are apparent.

Firstly, reinforcement learning can be used as a fine-tuning technique to encourage the generation of valid, closed, or spherical triangulations. Our results demonstrate that for small triangulations (6-8 tetrahedra), the generative efficiency is already in the 2-3% range, which is sufficient to receive positive reward signals within a reasonable batch size and makes this approach immediately viable.

Secondly, it is an empirically established result that transformer performance scales with model size, particularly the embedding dimension. This can be explained by two factors. Firstly, more parameters provide the model with greater capacity to fit the data. Secondly, and more pertinently,

the embedding layer can better learn the meaning of each token. With the current isomorphism signatures, the tokens do not contain any semantic meaning, but are rather an encoding strategy. As such, the model must "waste" capacity learning the syntax of the encoding itself. An alternative encoding strategy could be more effective. One possibility is "dehydrated isomorphism signatures," where each pair of letters consistently encodes specific geometric information and their position encodes gluing data. Such a representation is more aligned with the tasks transformers have been shown to excel on and could lead to faster training and higher accuracy for a given model size.

5.3.2. Alternative Architectures. If modifying the data representation and increasing compute prove insufficient, alternative neural network architectures could be explored.

Firstly, generating an isomorphism signature letter-by-letter is a fragile procedure. A single incorrect character can invalidate the entire sequence, and once such a mistake is made, it cannot be corrected. This compounding error suggests that diffusion-style models may be more suitable. These models operate on the entire sequence at once, allowing them to correct errors based on global context, which could significantly improve generation efficiency.

Secondly, a triangulation is fundamentally a graph. Therefore, Graph Neural Networks (GNNs), which are designed to operate on graph-structured data, may be a more natural fit than sequential models. A GNN might learn the underlying rules of a valid triangulation more efficiently, allowing smaller networks to be trained faster to achieve equivalent performance.

5.3.3. Reinforcement Learning for Topological Optimisation. Once generation efficiency is improved, the primary next step is to apply reinforcement learning to optimise the topological objective functions and compare the results against classical methods. The generative model would be trained using the objective functions from this report as reward signals. Using an algorithm such as Proximal Policy Optimisation (PPO), the model could be trained to directly generate novel triangulations that maximise knot-theoretic or combinatorial properties. This represents a powerful new approach, replacing heuristic search algorithms with models that can build an internal, learned representation of the geometric landscape.