Chop-SAT in Non-Euclidean Geometry

Thatcher Geary
University of Utah

UUCS-24-008

School of Computing
University of Utah
Salt Lake City, UT 84112 USA

4 May 2024

Abstract

This study explores methods of applying Non-Euclidean Geometry to the Boolean Satisfiability Problem (SAT). When presented with a knowledge base in Conjunctive Normal Form (CNF) with n atoms, it can be represented as an n-dimensional hypercube, where each corner corresponds to a unique combination of the logical truth assignments to the atoms. A geometric approach to solving SAT, CHOP-SAT [10] performs cuts on the hypercube's corners, with each chop arising from a conjunct in the CNF sentence. This process eliminates non-solution points and preserves only those corners that satisfy the CNF sentence within the feasible region in Euclidean Geometry. The SAT problem is solvable if corners within the feasible region of the hypercube are detected following the cuts. These corners signify the existence of a solution within the given constraints. The PoincarÂ´e disk is a model within Non-Euclidean Geometry that is represented as a unit disk in Euclidean geometry, but which has an associated metric which makes the boundary of the disk infinitely distant from the center. The corners of a hypercube superscribed about the unit disk can be projected onto the unit disk's boundary. Since these solution points are the only points at infinite distance from the center of the unit disk, the hope is that there will be a low-cost computational method to find them. The specific goal of this study is to investigate whether performing CHOP-SAT in this Non- Euclidean Geometry representation can yield an efficient algorithm for solving the Boolean Satisfiability Problem (SAT).

CHOP-SAT IN NON-EUCLIDEAN GEOMETRY

by

Thatcher Geary

A Senior Honors Thesis Submitted to the Faculty of
The University of Utah
In Partial Fulfillment of the Requirements for the

Honors Degree in Bachelor of Science

In

Computer Science

Approved:

Thomas C. Henderson	Mary Hall
Thomas C. Henderson Thesis Faculty Supervisor	Mary Hall Director, School of Computing
Thomas C. Henderson	
Thomas C. Henderson Departmental Honors Liaison	Monisha Pasupathi, PhD Dean, Honors College

May 2024

ABSTRACT

This study explores methods of applying Non-Euclidean Geometry to the Boolean Satisfiability Problem (SAT). When presented with a knowledge base in Conjunctive Normal Form (CNF) with n atoms, it can be represented as an n-dimensional hypercube, where each corner corresponds to a unique combination of the logical truth assignments to the atoms. A geometric approach to solving SAT, CHOP-SAT [10] performs cuts on the hypercube's corners, with each chop arising from a conjunct in the CNF sentence. This process eliminates non-solution points and preserves only those corners that satisfy the CNF sentence within the feasible region in Euclidean Geometry. The SAT problem is solvable if corners within the feasible region of the hypercube are detected following the cuts. These corners signify the existence of a solution within the given constraints. The Poincaré disk is a model within Non-Euclidean Geometry that is represented as a unit disk in Euclidean geometry, but which has an associated metric which makes the boundary of the disk infinitely distant from the center. The corners of a hypercube superscribed about the unit disk can be projected onto the unit disk's boundary. Since these solution points are the only points at infinite distance from the center of the unit disk, the hope is that there will be a low-cost computational method to find them. The specific goal of this study is to investigate whether performing CHOP-SAT in this Non- Euclidean Geometry representation can yield an efficient algorithm for solving the Boolean Satisfiability Problem (SAT).

For my parents, who have always supported me.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

The Boolean Satisfiability Problem (SAT) is a fundamental problem within computer science and mathematical logic. Given a Boolean formula, which is a logical expression constructed of boolean variables, conjunctions (AND), disjunctions (OR), and negations (NOT) [14], the goal of SAT is to determine if there exists such a truth assignment to the logical variables to the make the entire expression true. The SAT problem is classified as NP-complete. A problem is NP-complete if it is both the class of NP (problems which have a polynomial run time on a deterministic Turing machine for checking if a proposed solution is correct) and NP-hard (problems at least as hard in NP, where there are currently no polynomial time solutions). A problem that has a polynomial time solution on a deterministic Turing machine is classified as P. The P vs NP problem revolves around a question of whether P, the class of problems that can be solved in polynomial time is equivalent to NP, the class of problems where a solution can be verified in polynomial time. Since every NP-complete problem can be reduced to some form of the SAT, solving SAT in polynomial time would imply that every NP problem can be solved in polynomial time, proving that P = NP. Proving P = NP would hold significant implications for the field of computer science. The goal of this research is to investigate the application of Non-Euclidean Geometry to the SAT problem, particularly to CHOP-SAT, aiming to discover a polynomial time solution.

CHAPTER 2

BACKGROUND

2.1 Boolean Satisfiability Problem (SAT)

The Boolean Satisfiability Problem (SAT) is a task of determining whether a given Well-Formed Formula(WFF) is satisfiable. A Well Formed Formula is defined as follows:
Given a set of variables, $A = \{a_i | i \in \mathbf{N}\}$ where a_i takes on the value of $\{1, 0\}$.

1. $a_i \in A$ is a WFF.
2. If a WFF follows a negation symbol $\neg$, such that $\neg$(WFF), it remains a WFF.
3. Any expression of the form $(\alpha \vee \beta)$, where α, $\beta \in$ WFF.
4. Any expression of the form $(\alpha \wedge \beta)$, where α, $\beta \in$ WFF.
5. Any finite application of these rules results in a WFF.

Additionally, a literal, L can be defined as either a propositional variable or the negation of a propositional variable $L = a_i$ or $L = \neg a_i$. The objective of SAT is to find an assignment of truth values (*true* or *false*) to the atoms such that the expression, constructed from the operators $\{\neg, \vee, \wedge\}$, evaluates to *true*.

Any WFF can be converted to Conjunctive Normal Form (CNF), a form where the expression is represented as a conjunction (AND) of clauses, each of which is a disjunction (OR) of literals. This transformation process was proved by Stephen Cook in his paper "The Complexity of Theorem-Proving Procedures" published in 1971 [5]. Cook showed that any propositional logic formula, such as WFF, can be converted to a CNF form in polynomial time.

Any problem categorized in P is one that can be solved in polynomial time on a deterministic Turing machine. In contrast, a problem is categorized in NP if it has a polynomial time verifier [14]. The crucial link between the two classifications of problems

lies in NP-complete problems. NP-complete problems comprise a set of problems that are part of both NP and a special subset within NP, where any other problem can be reduced to them in polynomial time. Among all NP-complete problems, SAT stands out as the quintessential example. Due to the simplicity of SAT's logical variable structure, it has become the most frequently used problem for reductions from other NP-complete problems. Discovering a polynomial time solution for SAT would imply the equality of P and NP. This would have significant implications for the field of computer science, in both practical and theoretical applications.

Although polynomial time SAT solvers have not been proven to exist, modern SAT solvers such as Glucose, Lingeling, MapleCOMSPS, CaDiCaL, and MiniSat are highly practical and widely used in various applications. These SAT solvers are able to solve SAT's with KB's of millions of clauses or more by using a form of Conflict-Driven Clause Learning (CDCL) algorithm. These algorithms follow a method of incomplete local search, where the algorithm focuses on local modifications to the total assignments. This approach is called the Davis-Putnam-Logemann-Loveland (DPLL) algorithm, which underpins the CDCL model.

The DPLL algorithm employs a backtracking search strategy, systematically selecting values for variables at each step to explore various scenarios. After branching, if a conflict is detected, the algorithm will backtrack, undoing the branching until the conflict is resolved. The CDCL algorithm will operate in series of phrases: simplification, decision, and learning [3].

In the simplification phase, the algorithm extends the assignment by inferring new truth values based on the current state of the formula. Next, in the decision phase, the algorithm strategically selects an unassigned variable and assigns it a truth value,(*true* or *false*) usually with some sort of heuristic to guide the selection. After repeating these two phases, the assignment in the branch will either meet the conditions of the formula or not. If the assignment does not result in any contradictions in the clauses, the algorithm terminates the search. However, if conflicts arise, meaning a clause was falsified, the algorithm transitions into the learning phase. The learning phase analyzes the conflict in a clause and modifies the assignment to resolve it. If the conflict cannot be resolved, the algorithm backtracks to previous assignments and explores alternative paths by going

back to the simplify and decision phase. If no logical assignments can be found to satisfy the clause, the CNF sentence is deemed unsatisfiable.

Modern CDCL SAT solvers employ additional techniques to accelerate the algorithm. These procedures include extending additional clauses to the sentence from conflicts encountered during the backtrack search, having heuristics for restarting the backtrack search, exploiting the structure of conflicts during clause learning and more [13].

Unlike these CDCL SAT solvers which employ a more brute force approach, CHOP-SAT is not intended to compete on efficiency, but rather represents a theoretical attempt to find a polynomial time solution to the SAT problem.

2.2 CHOP-SAT

One notable observation regarding the SAT problem and logical assignments of n unique logical variables is the exponential growth in the number of possible assignments. With n unique logical variables $\{a_1, a_2, ..., a_n\}$, there are 2^n distinct truth assignments that can be made. This exponential increase can be geometrically visualized by constructing an n-dimensional hypercube. In an n-dimensional hypercube, each vertex represents a unique combination of truth values for the logical variables. Thus, the hypercube contains exactly 2^n vertices in n-dimensions, corresponding to each possible truth assignment of the atoms.

A geometric approach to solving SAT problems were first explored by Gomory [8], aiming to discover integer solutions to linear programs by introducing the Cutting-plane method. In this method, for each disjunction within a knowledge base (KB), a linear inequality can be formed by summing x_i for atoms in the clause and $(1 - x_i)$ for negated atoms in the clause and setting the expression to be greater than or equal to one. If a non-integer solution is found, then the non-integer solution is separated (cut) from the integer solutions. Chvatal expanded upon the method by proving supporting theorems for bounded polytopes and applying these results to solve various combinatorial problems [4]. It is important to note that integer programming is in NP. The CHOP-SAT method was developed independently from Gomory and Chvatal's work and fundamentally explores different geometric insights.

CHOP-SAT is a geometric approach to solving the SAT problem by leveraging the geometric equivalence described above. The algorithm proceeds as follows : Given a

Knowledge Base (KB) with n atoms represented by a CNF sentence with m conjuncts:

$$S = C_1 \wedge C_2 \wedge C_3 \wedge \ldots \wedge C_m$$

$$C_i = L_{i_1} \vee L_{i_2} \vee L_{i_3} \vee \ldots \vee L_{i_k}$$

Notice that the negation of C_i describes the assignments that do not satisfy the CNF sentence.

$$\neg C_i = \neg(L_{i_1} \vee L_{i_2} \vee L_{i_3} \vee \ldots \vee L_{i_k})$$

$$= \neg L_{i_1} \wedge \neg L_{i_2} \wedge \neg L_{i_3} \wedge \ldots \wedge \neg L_{i_k}$$

Each negation of the conjunct serves as a cut, eliminating corners corresponding to variable assignments that fail to satisfy the sentence. Given n atoms, each conjunct will have a certain subset of these atoms, i_k. For each atom not present in the conjunct, there are 2^{n-i_k} combinations of truth assignments that are possible. All these combinations represents the truth assignments that do not satisfy the CNF sentence. Thus each conjunct will cut 2^{n-i_k} vertices on the n-dimensional hypercube.

In CHOP-SAT, these cuts are applied iteratively for each conjunct, effectively pruning corners that do not fulfill the SAT condition. If after the cuts, the hypercube still retains corners, a satisfying assignment for the CNF sentence exists. Conversely, if no corners persist, no truth value assignment can satisfy the CNF sentence. Determining the existence of remaining corners in the feasible region within polynomial time is a pivotal aspect of the problem.

2.3 Linear Programming

Linear programming is a mathematical optimization technique used to find the best outcome in a model that is subject to linear constraints. It is defined as follows:

$$\begin{aligned} &\text{minimize} \quad f^T x \\ &\text{subject to} \quad \begin{cases} Ax \leq b, \\ A_{\text{eq}} x = b_{\text{eq}}, \\ \text{lb} \leq x \leq \text{ub}. \end{cases} \end{aligned}$$

where $f, x, b, b_{\text{eq}}, \text{lb},$ and ub are vectors, and A and A_{eq} are matrices.

By setting up the constraints of the feasible region using linear programming, we can effectively identify the extremes of the region. This process involves representing the hypercube and the cuts as constraints in the linear programming formulation. By seeking the

minimum of x in both positive and negative directions for each dimension ($2n$ directions in total), it may possible to detect the uncut corners.

To achieve this, let x represent a point within the feasible region, and set up the constraints $Ax \leq b$ derived from the cuts. The linear programming problem is then formulated to minimize $f^T x$ subject to these constraints.

The concept can be further elaborated by visualizing the hypercube and the cuts within it. Each cut delineates a facet of the hypercube, and the feasible region is the intersection of these facets. The objective is to find a point x that lies within the feasible region, representing an uncut corner. By adjusting the constraints through linear programming, the objective is to converge towards an uncut corner efficiently in polynomial time.

The polynomial-time solvability of linear programming, was demonstrated by George Dantzig in the 1940s, suggesting that there exists an algorithmic approach to find these uncut corners within the feasible region in polynomial time [6].

2.4 Euclidean CHOP-SAT Detecting Satisfiablity

Appendix A contains the Euclidean CHOP-SAT algorithm, this section specifically focuses on the detection of corners following the application of CHOP-SAT. The development of an efficient polynomial-time method for corner detection holds significant implications, solving SAT in polynomial time and therefore proving P = NP. Given n unique atoms in the KB, the hypercube H_n will have 2^n corners. H_n has a range within [0, 1] for each axis. The feasible region represents the solution space of the KB, identified by the presence of any original hypercube corners within it. An observation can be made that every corner in H_n is $\frac{\sqrt{n}}{2}$ distant from its center, thus a feasible region produced by a KB with a satisfiable sentence must have a point that is $\frac{\sqrt{n}}{2}$ away from the center of H_n. A special convex polytope, denoted I_n, represents the largest feasible region for any unsatisfiable sentence, thus any feasible region produced from an unsatisfiable KB, using CHOP-SAT, is contained within I_n. The convex polytope is bounded above by a distance of $\frac{\sqrt{n-2}}{2}$ from the center. Hence, the KB sentence is satisfiable only if any point within the feasible region exceeds this distance from the center of H_n.

It is possible to determine the existence of a corner in the feasible region that is $\frac{\sqrt{n}}{2}$ away by using linear programming and projecting onto all hypercube diagonals but there are

$2^(n-1)$ of these, thus costing exponential time. It is clear that an alternative, more efficient algorithm is required. Two main methods have been devised to identify the feasible region that exceeds this upper bound: the Singleton Detection method through rotation, and the Maximum Volume Inscribed Ellipsoid method.

2.4.1 The Singleton Detection Method

An unsatisfiable feasible region is bounded above by a distance of $\frac{\sqrt{n-2}}{2}$ from the origin [10], then every rotation of said region about the center of H_n still has the same upper bound. The feasible region can be rotated and projected onto an axis using linear programming to identify any protruding points beyond the upper bound. This approach yielded promising outcomes for KB instances with $n \leq 10$. However, as the dimensionality increased, the projections were occluded by the other dimensions and using linear programming to detect the protruding points became very difficult.

2.4.2 The Maximum Volume Inscribed Ellipsoid Method

The Maximum Volume Inscribed Ellipsoid (MVE) within the convex feasible region may be used to determine the existence of a solution to CHOP-SAT. An ellipse, represented by E, is defined as the set of points $x \in \mathbb{R}^n$ satisfying $(x-c)^T A(x-c) \leq 1$, where A is a symmetric positive definite matrix and $c \in \mathbb{R}^n$ is the center of the ellipsoid. The MVE within a bounded full-dimensional convex body Ω can be computed efficiently, providing a tight ellipsoidal approximation of the region with maximum volume. This ellipsoid can be used to efficiently verify the satisfiability of new constraints, with one of its major semi-axes aligned with the most extreme length segment of the feasible region. If the maximum semi-axis of the MVE is less than $\sqrt{n-2}$, it may indicate unsatisfiability. While computing the minimum volume circumscribing ellipsoid is NP-hard, determining the maximum volume inscribed ellipsoid is possible in polynomial time. As dimensions increased, this method proved ineffective because the maximum volume results from a more compact ellipsoid rather than the one with the longest convex segment [11].

2.5 Non-Euclidean Geometry

As the dimension of the hypercube increases, the CHOP-SAT algorithm(s) encounters challenges in identifying corners within the feasible region in Euclidean geometry. A

different model of geometry can be used to interpret CHOP-SAT namely, Non-Euclidean Geometry.

In order to grasp the concept of non-Euclidean geometry, it is useful to establish a foundation in Euclidean geometry. Euclid in his famous book *Elements*, published approximately around 300 BCE, proposed five postulates for geometry [7]. In modern times, these are called axioms, a statement or proposition that is accepted as true without requiring proof. These axioms serve as the foundational building blocks upon which the rest of the mathematical or logical system is constructed. The set of postulates known as Euclid's postulates forms the foundation of Euclidean geometry, which has been widely used for centuries. These postulates are:

1. A straight line segment can be drawn joining any two points.

2. Any straight line segment can be extended indefinitely in a straight line.

3. Given any straight lines segment, a circle can be drawn having the segment as radius and one endpoint as center.

4. All Right Angles are congruent.

5. If two lines are drawn which intersect a third in such a way that the sum of the inner angles on one side is less than two Right Angles, then the two lines inevitably must intersect each other on that side if extended far enough. This postulate is equivalent to what is known as the Parallel Postulate.

These postulates have formed the basis for Euclidean geometry. However, in the 19th century, mathematicians such as Nikolai Lobachevsky, János Bolyai, and Carl Friedrich Gauss developed geometrical systems without the Parallel Postulate. These systems, known as non-Euclidean geometries, primarily include two types of models: elliptical geometry, where lines do not have other lines parallel to them, and hyperbolic geometry, where given a line and a point not on that line, there are infinitely many lines through the point that do not intersect the given line.

2.5.1 The Upper Half-Plane Model

The upper half-plane model is a geometric model within hyperbolic geometry, where the upper half-plane $\mathbb{H}$ in the complex plane $\mathbb{C}$, is defined to be:

$\mathbb{H} = \{z \in \mathbb{C} \mid \text{Im}(z) > 0\}$

The hyperbolic lines in $\mathbb{H}$ can be represented using Euclidean geometry in $\mathbb{C}$. One is an intersection of $\mathbb{H}$ with a Euclidean line in $\mathbb{C}$ that is perpendicular to the real axis $\mathbb{R}$ in $\mathbb{C}$. The other is the upper half of a Euclidean circle, centered on the real axis $\mathbb{R}$. Examples of Hyperbolic lines in the upper half-plane are shown in Figure 2.1 below.

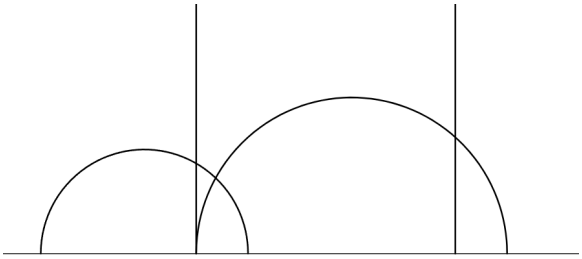


Figure 2.1: Hyperbolic lines in upper half-plane

In the upper half-plane model, shapes like triangles can be formed, similar to Euclidean geometry. Below are two hyperbolic triangles.

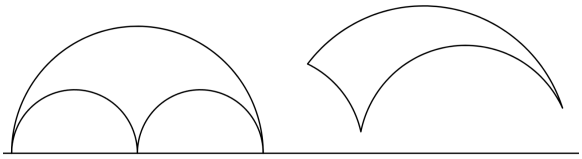


Figure 2.2: Hyperbolic triangles in upper half-plane

The upper half-plane can serve as a model for CHOP-SAT by projecting the points of the hypercube onto the upper half-plane $\mathbb{H}$ and applying the CHOP-SAT algorithm. However, there are certain challenges associated with this approach. Firstly, the corners that lie on the real line $\mathbb{R}$ do not belong to $\mathbb{H}$, which complicates the modeling process. Secondly, the presence of straight Euclidean lines in the upper half-plane model, similar to those in Euclidean geometry, may lead to analogous occlusion problems, especially as the

number of dimensions increases. To address these challenges, we explore an alternative model of Hyperbolic geometry known as the Poincaré disk model.

2.5.2 The Poincaré Disk Model

The Poincaré disk $\mathbb{D}$ is the open unit disk:

$\mathbb{D} = \{z \in \mathbb{C} \mid\mid z \mid < 1\}$

Hyperbolic lines within the Poincaré unit disk $\mathbb{D}$ can also be interpreted using Euclidean geometry in the complex plane $\mathbb{C}$. One representation involves a straight Euclidean line passing through the origin of the unit disk. Another representation entails a segment of a Euclidean circle in $\mathbb{R}$, intersecting the unit disk at a right angle. This segment corresponds to the intersection of an orthogonal circle within $\mathbb{D}$. A Few examples of lines in $\mathbb{D}$ are shown in Figure 2.3 below.

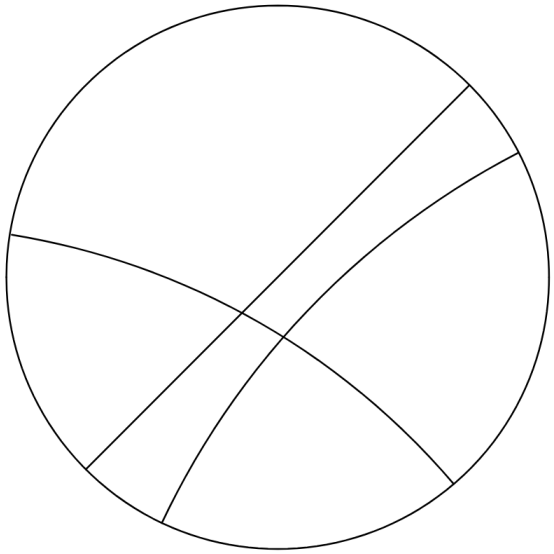


Figure 2.3: Hyperbolic lines in the Poincaré disk

One of the unique properties of the Poincaré disk is how distance is calculated. The hyperbolic length of a piecewise C^1 path $f : [a, b] \to \mathbb{D}$ is given by the integral

$$length_{\mathbb{D}}(f) = \int_f \frac{2}{1 - |z|^2} |dz|$$

The distance from the origin to the hyperbolic line segment between 0 and r for $0 < r < 1$ can be parameterized by $f : [0, r] \to \mathbb{D}$ given by $f(t) = t$. Since the image of f is the

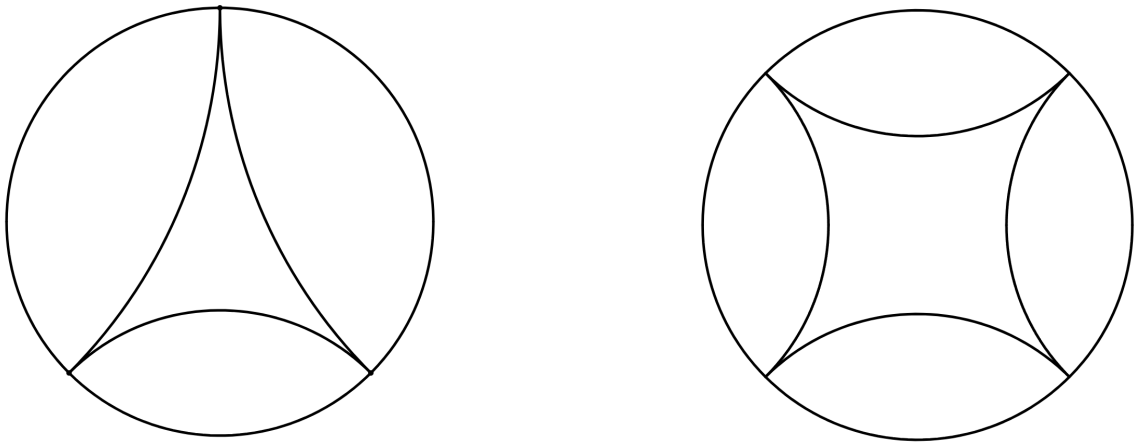
hyperbolic line segment in $\mathbb{D}$ that joins 0 and r, we get $d_{\mathbb{D}}(0,r) = \text{length}_{\mathbb{D}}(f)$. Thus for the distance, we get [1, pg.122]:

$$\begin{aligned} d_{\mathbb{D}}(0,r) &= \text{length}_{\mathbb{D}}(f) \\ &= \int_f \frac{2}{1-|z|^2}|dz| \\ &= \int_0^r \frac{2}{1-t^2}dt \\ &= \int_0^r \left(\frac{1}{1+t} + \frac{1}{1-t}\right)dt \\ &= \ln\left(\frac{1+r}{1-r}\right) \end{aligned}$$

Notice that as r approaches 1, the distance, $d_{\mathbb{D}}(0,r)$ tends to infinity.

The unit disk seems to be bounded from an external Euclidean perspective, but is unbounded and limitless from within $\mathbb{D}$. Thus, as points approach the boundary, their hyperbolic distance grows rapidly towards infinity. Utilizing this property, by projecting the corners of the hypercube onto the boundary of $\mathbb{D}$ (called ideal points, which corresponds to infinity) and subsequently applying CHOP-SAT, it becomes more feasible to identify the unchopped corners as they reside at infinity.

The Poincaré disk, like the upper half-plane, can also produce unique shapes that are not possible in Euclidean geometry. Below are examples of a hyperbolic triangle and hyperbolic square with the sum of their angles being 0.



(a) Hyperbolic Triangle in the Poincaré disk

(b) Hyperbolic Square in the Poincaré disk

Figure 2.4: Shapes in Poincaré disk

CHAPTER 3

CHOP-SAT IN NON-EUCLIDEAN GEOMETRY

3.1 Hypersphere Construction

The motivation for using the Poincaré disk is to take advantage of the ideal points that reside at infinity. The corners of the n-dimensional hypercube are projected onto the surface of the n-dimensional hypersphere as unique ideal points that are infinite distance away from the center of the hypersphere (shown below in Figure 3.1).

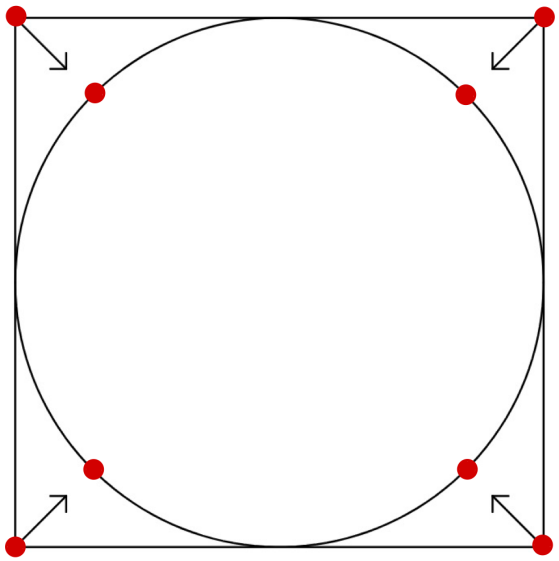


Figure 3.1: Projection of vertices from the hypercube onto the hypersphere in 2D.

In the Poincaré Disk representation, the feasible region is bounded only when no solution is present. The aim is to develop a low-complexity algorithm to identify points within the feasible region whose distance from the origin tends to infinity in Poincaré distance. Determining whether the feasible region is unbounded could provide valuable insights into discerning whether SAT has a solution.

3.2 Poincaré Disk Distance

Before establishing a representation of CHOP-SAT in the Poincaré Disk, we need to introduce a few fundamental geometric definitions. As discussed in Chapter 2, the Hyperbolic line l can be depicted as a segment of a unique Euclidean circle in $\mathbb{R}$, intersecting the unit disk at a right angle. Let p and q be two points located on the segment of the Euclidean circle, and denote the two ideal points a and b as the intersections of the unit disk and the circle, as illustrated in Figure 3.2 below.

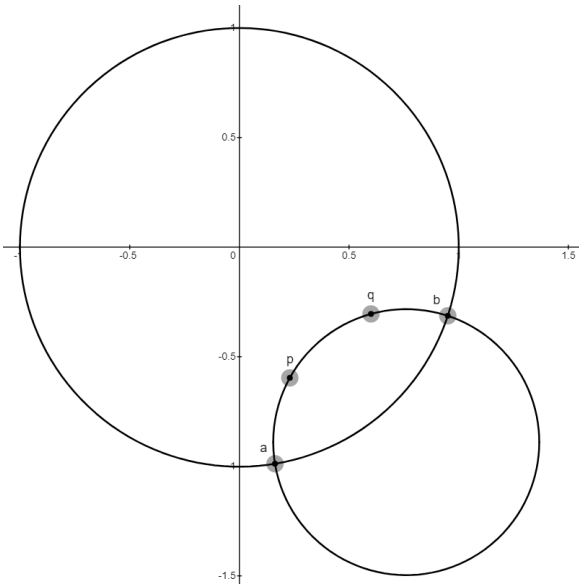


Figure 3.2: Representation of a hyperbolic line using a Euclidean circle.

The distance between the two points, p and q in the Poincaré disk defined in terms of Eucldiean distance is given by [2]:

$$d(p,q) = \ln\left(\frac{|ap| \cdot |qb|}{|aq| \cdot |pb|}\right)$$

where the vertical bars indicate Euclidean length of the line segment that connects the two points.

In our pursuit of constructing an n-dimensional hypersphere, we are interested in deriving an alternative distance formula applicable to $\mathbb{R}^n$. Consider $u, v \in \mathbb{R}^n$ such that $|\mathbf{u}| < 1$ and $|\mathbf{v}| < 1$ (both u and v in $\mathbb{D}_n$) Then,

$$d(u,v) = \operatorname{acosh}\left(1 + \frac{2\|uv\|^2\|r\|^2}{(\|r\|^2 - \|u\|^2)(\|r\|^2 - \|v\|^2)}\right)$$

since $r = 1$ we have,

$$d(u,v) = \operatorname{acosh}\left(1 + \frac{2\|uv\|^2}{(\|1-\|u\|^2\|)(1-\|v\|^2)}\right)$$

$$= 2\operatorname{asinh}\sqrt{\left(\frac{\|uv\|^2}{(1-\|u\|^2)(1-\|v\|^2)}\right)}$$

$$= 2\ln\left(\frac{\|u-v\| + \sqrt{\|u\|^2\|v\|^2 - 2u\cdot v + 1}}{\sqrt{(1-\|u\|^2)(1-\|v\|^2)}}\right)$$

3.3 Orthogonal Hypersphere

To transition from an n-dimensional hypercube to an n-dimensional hypersphere, a transformation for the cuts is necessary. In Euclidean geometry, CHOP-SAT utilizes hyperplanes as cuts, each of which is $n-1$ dimensional. However, in the Poincaré disk, these $n-1$ dimensional hyperplane cuts would correspond to $n-1$ dimensional hyperspheres. Therefore, a representation for these hyperspheres must be established. One approach is to represent them using Euclidean hyperspheres that are orthogonal to the Poincaré disk.

Let $C_1, C_2 \in \mathbb{R}^2$ be the centers of two Euclidean Circles with radii r_1 and r_2, respectively. The angle between the two circles is given by:

$$\cos(\theta) = \frac{r_1^2 + r_2^2 - \|\mathbf{C_1} - \mathbf{C_2}\|^2}{2r_1r_2}$$

Given two points p and q in the Poincaré disk, the segment of a Euclidean circle with center, C_2, that represents the hyperbolic line between p and q, can be found by considering C_1 as the center of the Poincaré disk i.e, with C_1 at the origin with the radius $r_1 = 1$. Since two circles are orthogonal if $\theta = \frac{\pi}{2}$, let:

$$\cos(\frac{\pi}{2}) = \frac{r_1^2 + r_2^2 - \|\mathbf{C_1} - \mathbf{C_2}\|^2}{2r_1r_2}$$

Since $\cos(\frac{\pi}{2}) = 0$, $\mathrm{C}_1 = (0,0)$, and $r_1 = 1$ we get,

$$0 = \frac{1 + r_2^2 - \|\mathbf{C_2}\|^2}{2r_2}$$

$$0 = 1 + r_2^2 - \|\mathbf{C_2}\|^2$$

$$r_2^2 = \mathrm{C}_x^2 + \mathrm{C}_y^2 - 1$$

where C_x and C_y represent the x and y coordinates of the center, respectively. Since p and q are points on the circle centered at C_2, we get:

$$r_2^2 = (p_x - C_x)^2 + (p_y - C_y)^2$$
$$r^2 = p_x^2 - 2p_xC_x + C_x^2 + p_y^2 - 2p_yC_y + C_y^2$$

where p_x and p_y are the coordinates that represent x and y for p. Substituting the first equation into the second yields:

$$C_x^2 + C_y^2 - 1 = C_x^2 + C_y^2 - 2p_xC_x - 2p_yC_y + p_x^2 + p_y^2$$
$$= p_xC_x + p_yC_y = \frac{p_x^2 + p_y^2 + 1}{2}$$

Likewise for q we get:

$$= q_xC_x + q_yC_y = \frac{q_x^2 + q_y^2 + 1}{2}$$

This linear equation can be solved by:

$$A = \begin{pmatrix} p_x & p_y \\ q_x & q_y \end{pmatrix} b = \begin{pmatrix} \frac{p_x^2+p_y^2+1}{2} \\ \frac{q_x^2+q_y^2+1}{2} \end{pmatrix}$$
$$AC = b$$

Finally, the radius of the Euclidean circle, r can be found by:

$$r = \|C - p\|$$

If a Euclidean line can be drawn through p, q, and the origin, then the hyperbolic line between p and q is equivalent to the Euclidean line connecting them. This equivalence can also be represented as a Euclidean circle with an infinite radius. However, when constructing the cuts for CHOP-SAT in Non-Euclidean geometry, this scenario can be entirely avoided

3.4 Representing the Hypersphere

To represent the n-dimensional hypercube as an n-dimensional hypersphere, there must be a way to accurately project the vertices of the hypercube to the hypersphere. An n-dimensional hypercube has $2n$ faces that are $(n-1)$ dimensional hyperplanes. These bounding faces in the Poincaré disk would follow a hyperbolic line, thus is the boundary of an n-dimensional hypersphere. The vertices of the hypercube will be projected directly

from the corners of the hypercube onto the surface of the hypersphere in the direction of the center of the Poincaré disk. The $2n$ bounding faces goes as follows.

Given a KB with n unique atoms, we seek to construct an n-dimensional hypersphere that bounds the feasible region. The center of these hypersphere faces will lie along the positive and negative side of each coordinate axis. Thus, let $v_1 \in \mathbb{R}^n$ be a unit vector from the center of the disk to one of the vertices of the bounding faces, and $v_2 \in \mathbb{R}^n$ be a unit vector in the direction of the coordinate axis of the desired bounding face. The angle between two vectors in $\mathbb{R}^n$ is given by:

$$\cos(\theta) = \frac{v_1 \cdot v_2}{\|v_1\|\|v_2\|}$$

since v_1 and v_2 are both unit vectors we have:

$$\cos(\theta) = v_1 \cdot v_2$$

The solution for $v_1 \cdot v_2$ can be found by observing that the entries of v_1 will be $\pm\frac{\sqrt{n}}{n}$ since $\|v_1\| = 1$, and v_1 is a vector pointing at the vertex. Since v_2 is a vector with 1 for one entry and rest is 0, we have that

$$|v_1 \cdot v_2| = \frac{\sqrt{n}}{n}$$

The distance from the origin to the center of the bounding hypersphere can be obtained by:

$$d = \frac{1}{\cos(\theta)} = \frac{1}{|v_1 \cdot v_2|} = \frac{n}{\sqrt{n}} = \sqrt{n}$$

The radius, r is the distance between v_1 and $\sqrt{n}v_2$ which is given by:

$$\|v_1 - \sqrt{n}v_2\| = \left\| \left(\pm\frac{\sqrt{n}}{n}, \pm\frac{\sqrt{n}}{n}, \ldots, \pm\frac{\sqrt{n}}{n} \right) - (0, \ldots, 0, \sqrt{n}, 0, \ldots, 0) \right\|$$

$$= \|(\pm\frac{\sqrt{n}}{n}, \pm\frac{\sqrt{n}}{n}, \ldots, \pm\frac{(n-1)\sqrt{n}}{n}, \pm\frac{\sqrt{n}}{n}, \ldots, \pm\frac{\sqrt{n}}{n})\|$$

$$= \sqrt{\left(\pm\frac{\sqrt{n}}{n}\right)^2 + \left(\pm\frac{\sqrt{n}}{n}\right)^2 + \cdots + \left(\pm\frac{(n-1)\sqrt{n}}{n}\right)^2 + \left(\pm\frac{\sqrt{n}}{n}\right)^2 + \cdots + \left(\pm\frac{\sqrt{n}}{n}\right)^2}$$

$$= \sqrt{\left(\pm\frac{(n-1)\sqrt{n}}{n}\right)^2 + (n-1)\left(\pm\frac{\sqrt{n}}{n}\right)^2}$$

$$= \sqrt{(n-1)^2 \left(\pm\frac{\sqrt{n}}{n}\right)^2 + (n-1)\left(\pm\frac{\sqrt{n}}{n}\right)^2}$$

$$= \sqrt{(n-1)\left(\pm\frac{\sqrt{n}}{n}\right)^2 (n-1+1)} = \sqrt{(n)(n-1)\left(\pm\frac{\sqrt{n}}{n}\right)^2} = \sqrt{(n)(n-1)\left(\frac{n}{n^2}\right)}$$

$$= \sqrt{(n-1)\left(\frac{n^2}{n^2}\right)} = \sqrt{(n-1)}$$

The centers of the hyperspheres for the bounding faces are $(0, 0, \ldots, \pm\sqrt{n}_i, 0, \ldots)$ along the i^{th} coordinate axis, with a radius of $\sqrt{n-1}$. This process is repeated for $2n$ bounding faces. The vertices are located at all combinations of $\pm$ for $(\pm\frac{\sqrt{n}}{n}, \pm\frac{\sqrt{n}}{n}, \ldots, \pm\frac{\sqrt{n}}{n}) \in \mathbb{R}^n$ (notice that this is 2^n combinations). The example for this in $n = 2$ dimensions is depicted in Figure 3.3 below.

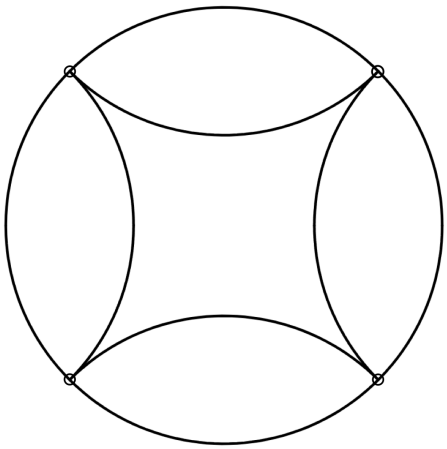


Figure 3.3: Bounding faces for n = 2.

3.5 Clause Chops

CHOP-SAT uses each clause to construct a cut in the feasible region by cutting the non-solution vertices by a hyperplane that separates them from the feasible region. In the Poincare disk, these would be parts of hyperspheres. Given a CNF each clause would represent each cut or hypersphere that separates the feasible region from the non-solution vertices. The algorithm for getting all the Non-Euclidean chops is given below.

CHOP-SAT leverages each clause to create a partition in the feasible region by placing a hyperplane between the non-solution vertices and the feasible region. In the Poincaré disk, these partitions would manifest as segments of hyperspheres. For a given CNF, each clause corresponds to a separate partition or hypersphere that delineates the feasible region from the non-solution vertices.

The corners that are within the radius of the hypersphere cuts are considered to be part of the non-solution vertices, while the feasible region comprises the points outside these hyperspheres. Thus, the hyperspheres effectively serve as boundaries between the feasible region and the non-solution vertices. The algorithm for generating all the Non-Euclidean partitions is outlined below. Given a KB with clause:

$Cl = L_1 \vee L_2 \vee L_3 \vee \ldots \vee L_k$

let $V = [v_1, v_2, \ldots, v_n]$ where n is the number of unique atoms, and a_i is the i^{th} atom:

$$v_i = \begin{cases} -1, & \text{if } \exists \text{j s.t. } \neg a_i = L_j \\ 0, & \text{if } a_i \notin Cl \text{ and } \neg a_i \notin Cl \\ 1, & \text{if } \exists j \text{ s.t. } a_i = L_j \end{cases}$$

Let $\alpha = -V$ and $u_\alpha = \frac{\alpha}{||\alpha||}$. This is a unit vector pointing towards the center of the hypersphere that chops off the desired vertex or vertices. To find the distance d, start by selecting a vertex to be chopped. Set $\alpha_{-1\leftarrow 0}$ to α but, replacing any 0's with -1 and $chop\_pt = \frac{\alpha_{-1\leftarrow 0}}{||\alpha_{-1\leftarrow 0}||}$, where $chop\_pt$ represents a vertex to be chopped. To find the center a similar process to the bounding faces could be done. Let:

$$\cos(\theta) = u_\alpha \cdot chop\_pt$$
$$d = \frac{1}{\cos(\theta)}$$

Since the angle between the Euclidean line from the origin of the Poincaré disk to $chop\_pt$ must be orthogonal to the Euclidean line with length r from the center of the hypersphere to the surface, by the Pythagorean theorem we have:

$$r = \sqrt{d^2 - 1}$$

Finally the center of the hypersphere chop, C is given by:

$$\mathrm{C} = d u_\alpha$$

Let $\text{Cl}_1 = a \vee b$. Then $\text{V}_1 = [1,1]$, $\alpha_1 = [-1,-1]$, $u_{\alpha,1} = [-0.7071, -0.7071]$, $chop\_pt_1 = [-1,-1]$ We have that $d_1 = \frac{1}{0.7071} = 1.414$ and $r_1 = \sqrt{1.4141^2 - 1} = 1$. Thus, we get:

$\text{C}_1 = d_1 u_{\alpha,1} = 1.414[-0.7071, -0.7071] = [-1,-1],\ r_1 = 1$

Equation for circle 1 is :$(x-(-1))^2 + (y-(-1))^2 = 1$

Let $\text{Cl}_2 = \neg a$. Then $\text{V}_2 = [-1,0]$, $\alpha_2 = [1,0]$, $u_{\alpha,2} = [1,0]$, $chop\_pt_2 = [0.7071, -0.7071]$ We have that $d_2 = \frac{1}{0.7071} = 1.414$ and $r_2 = \sqrt{1.4141^2 - 1} = 1$. Thus, we get:

$\text{C}_2 = d_2 u_{\alpha,2} = 1.414[1,0] = [1.414, 0],\ r_2 = 1$

Equation for circle 2 is :$(x-1.414)^2 + (y)^2 = 1$

Since the points $(0.7071, 0.7071)$ and $(0.7071, -0.7071)$ are included in circle 1 and the point $(-0.7071, -0.7071)$ is included in circle 2, these points are all non-solution vertices. The example of the bounding faces is depicted in black, and the chops are highlighted in red, as shown in Figure 3.4.

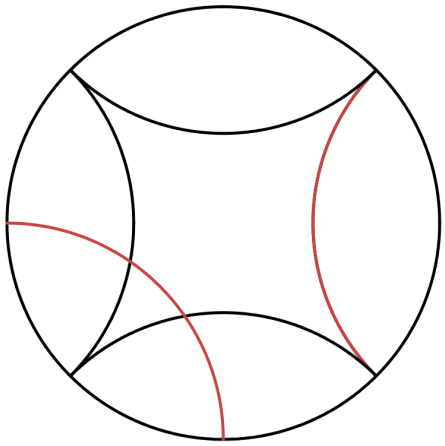


Figure 3.4: Chops for $\text{Cl}_1 = a \vee b$ and $\text{Cl}_2 = \neg a$

The clause chops serve two purposes: one is to eliminate the non-solution vertices from the feasible region, and the other is to facilitate the detection of the solution vertices. Therefore, our goal is to remove as much of the feasible region as possible to simplify the identification of the solution vertices. Two methods have been devised to accomplish this task: the Directional Center Push method and the Non-Chopped Neighbor method.

3.5.1 Directional Center Push Method

Finding the center and radius of the hypersphere is possible as long as we have the direction vector. We aim to establish a parameter $0 \leq \Delta < 1$, where as Δ approaches 1 the chops gradually remove more of the feasible region. The Directional Center Push method will derive the same parameters as the regular clause chops, with one modification in the method where the distance d is determined by:

$$\cos(\theta) = u_\alpha \cdot chop\_pt$$
$$d = \frac{1}{\cos(\theta)(1-\Delta)}$$

The equation to obtain the radius and center still follows:

$$r = \sqrt{d^2 - 1}$$
$$\mathrm{C} = du_\alpha$$

While this method does produce legal chops that effectively separate the feasible region from the non-solution vertices, as Δ approaches 1, the distance value escalates significantly. This could make it more challenging to manage these hyperspheres due to the large numerical values involved. Therefore, it would be preferable to find another method for adjusting the chops.

3.5.2 Non-Chopped Neighbor Method

Consider the Euclidean n-dimensional hypercube, denoted as H_n. To identify the neighboring vertex to a given chop point *chop_pt*, we invert a non-zero element of the vector α and replace any 0's with -1's. Let $\overline{\alpha}_{-1\leftarrow 0}$ represent this operation, and define nei_pt $= \overline{\alpha}_{-1\leftarrow 0}$. Determine a projection point, *proj_pt*, by moving along the edge of H_n that connects *chop_pt* and *nei_pt* by a percentage $0 < \Delta < 1$. This yields:

$$proj\_pt = (1-\Delta)chop\_pt + \Delta nei\_pt$$

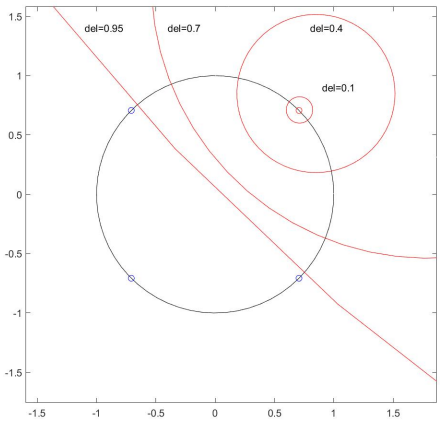
The point, *proj_pt* can be projected onto the surface of the Poincaré disk by normalizing it.

$$proj\_pt\_PD = \frac{proj\_pt}{||proj\_pt||}$$

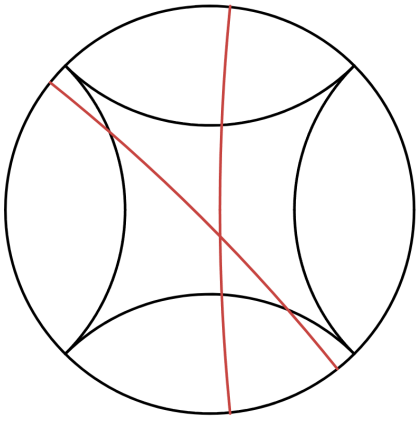
the hypersphere can be obtained similarly by:

$$\cos(\theta) = u_\alpha \cdot proj\_pt\_PD$$
$$d = \frac{1}{\cos(\theta)}$$
$$r = \sqrt{d^2 - 1}$$
$$\mathrm{C} = du_\alpha$$

This method allows a more precise method for accurately picking what percent of the feasible region would we want to chop off for each cut without big numerical problem present in the Directional Center Push method. The impact of Δ on the chops is shown in Figure 3.5 below.



(a) Impact of Δ on single chop



(b) Chops for $\mathrm{Cl}_1 = a \vee b$ and $\mathrm{Cl}_2 = \neg a$ with $\Delta = .9$

Figure 3.5: Impact of Δ on chops

3.5.3 Method

In Euclidean Geometry, the detection of projected corners could be accomplished using linear programming. However, in the Poincaré disk model, where lines are curved, linear programming is not applicable because it deals with optimizing linear objective functions and representing our chops using hyperspheres introduces non-linearity. Therefore, an alternative approach is needed to find the corners. We employ the Boyd Force Field method to address this challenge, as detailed in Appendix B. The implementation of the method is done using Matlab.

Given a CNF sentence, the problem solution could be found as follows:

1. The set of $2n$ bounding faces and the hypersphere chops produced by the CNF clauses are constructed.

2. A Barrier Method type algorithm is devised to navigate through the constraint set, aiming to maximize projection in the selected direction while utilizing the hyper-sphere surfaces as barrier constraints. Implemented as a force field method, it is elaborated upon in the Appendix B and the subsequent section. Additional forces, such as those directing away from the origin, may also be incorporated to enhance convergence.

3. A solution is considered to exist if the convergence point is sufficiently close enough to a solution vertex.

CHAPTER 4

BARRIER METHOD

4.1 Initial Parameters

The Barrier Method requires initial values of starting point x, forcing vector f, magnitude of forcing vector t, the rate of increase for forcing vector μ, and the stopping criterion ϵ. This section delves into the methodology employed to derive these values for the Barrier Method.

4.1.1 Starting Point

The hyperbolic lines in the Poincaré disk, as discussed in section 2.5.2 have two types of lines based on Euclidean geometry, a straight Euclidean line that goes through the origin, or a segment of a Euclidean circle that intersects the Poincaré unit disk at a right angle. It is arguable that the origin can serve as an initial starting point x. All the chops and $2n$ bounding faces are represented using hyperspheres, thus there are no straight Euclidean hyperplanes that pass through the origin. The hypersphere chops, derived using the Non-Chopped Neighbor method, can never include a chop that affects the origin. The projected point is determined by moving along the edge of H_n connecting the chopped point with the neighboring point that remains unchanged. Since $\Delta < 1$, the projected point can never coincide with a point on a straight Euclidean line. By observing Figure 3.5(a), it's clear that as Δ approaches 1, the cuts converge toward an infinitely large radius hypersphere (straight Euclidean line), but they never actually reach that point, as Δ can never equal 1. Thus, the origin remains unaffected and is neither intersected nor contained within the hypersphere chops and x can be set to the origin.

4.1.2 Forcing Vector

The Barrier Method involves iterating $2n$ times, alternating the forcing vector to point in both positive and negative directions along each dimension. However, directing the forcing vector along the dimensional axis may not be optimal, as projected corners are not there; instead, it would directly target the bounding faces. A more effective approach would be to rotate the vectors by 45° degrees, aligning them relatively towards the corners.

The forcing vectors can be derived by rotating the basis vector into another dimension. Let $v \in \mathbb{R}^n$ being the basis vector for some dimension i, then:

$$v_i = \begin{bmatrix} 0 \\ \vdots \\ 1_i \\ \vdots \\ 0_n \end{bmatrix}$$

To rotate the basis vector v_i into the j dimension, let $\mathbf{A}_{i,j} \in \mathbb{R}^{n \times n}$, the rotation matrix is given by [9]:

$$\mathbf{A}_{i,j} = \begin{bmatrix} 1 & \cdots & 0 & 0 & \cdots & 0 & 0 & \cdots & 0 \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & \cos(\phi)_{i,i} & 0 & \cdots & 0 & -\sin(\phi)_{i,j} & \cdots & 0 \\ 0 & \cdots & 0 & 1 & \cdots & 0 & 0 & \cdots & 0 \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & 0 & \cdots & 1 & 0 & \cdots & 0 \\ 0 & \cdots & \sin(\phi)_{j,i} & 0 & \cdots & 0 & \cos(\phi)_{j,j} & \cdots & 0 \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & 0 & 0 & \cdots & 0 & 0 & \cdots & 1 \end{bmatrix}$$

then we get:

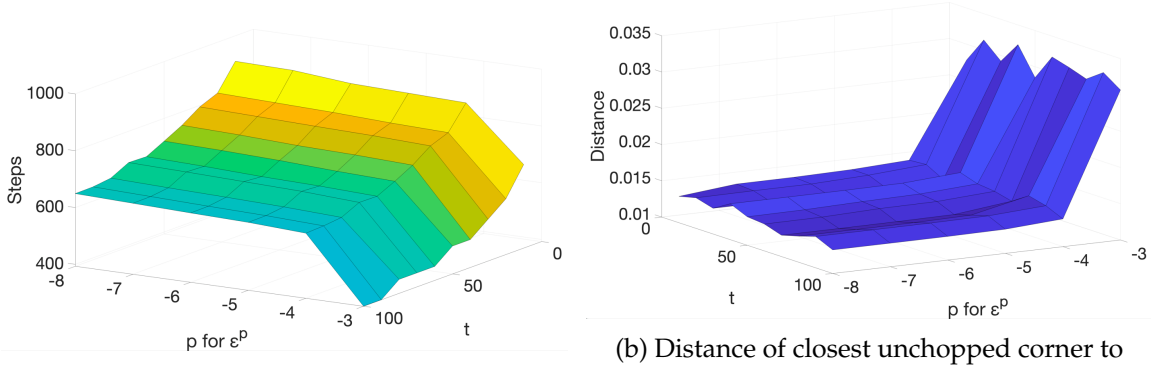
$$f_{i,j} = \mathbf{A}_{i,j} v_i = \begin{bmatrix} 0 \\ \vdots \\ \cos(\phi)_i \\ \vdots \\ \sin(\phi)_j \\ \vdots \\ 0_n \end{bmatrix}$$

The $2n$ forcing vectors can be obtained by rotating each basis vector for dimension i to the next dimension $i+1$ (for $i = n$ rotate back to $j = 1$). To obtain the opposite direction forcing vector, we plug in $-\phi$ for the $f_{i,j}$.

4.1.3 Convergence Constraints

For parameters μ, t, and ϵ, we observed higher variability. We set μ to a fixed value of 1.5, considering it a reasonable scaling factor for t across iterations. However, for t and ϵ, we conducted a test to determine the combination of parameters yielding the best results.

In this test, we varied t from 10 to 100 with increments of 10, and ϵ from 10^{-8} to 10^{-3} with a tenfold increase. We collected the distance from the closest corner and the number of steps of the final convergence for all parameter combinations and plotted the results in a 3D graph. This analysis was carried out using several symmetrically unique chops for $n = 3$.

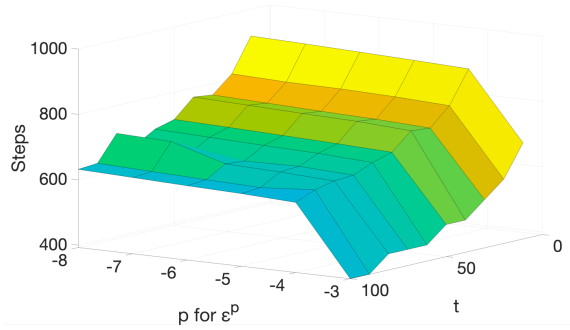


(a) Number of steps for Barrier Method to converge for input t and p for ϵ^p for CNF_1

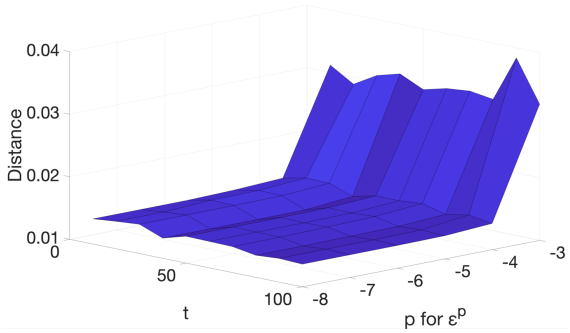
(b) Distance of closest unchopped corner to convergence point of Barrier Method for input t and p for ϵ^p for CNF_1

Figure 4.1: Analysis of Barrier Method Convergence: Exploration of the number of steps and the distance to the closest unchopped corner as influenced by input parameters t and p for ϵ^p, for CNF_1

The experiments indicate that as t increases, the number of steps decreases. Regarding the parameter p for ϵ^p, changes do not notably affect the number of steps, except for when p is very large, where a significant decrease in steps is observed. In terms of distance, a noticeable trend is observed: as p decreases, the minimal distance from the corner also decreases, with a notable spike observed at -3. However, the distance does not seem as affected by the t value, and the number of steps does not seem to be significantly influenced by p, except for extreme values. It is notable that there is a small dip in distance for when $t = 60$. From the experiment it seems like a to maximize efficiency and precision for the Barrier Method picking $\epsilon = 10^{-8}$ and $\mu = 60$.

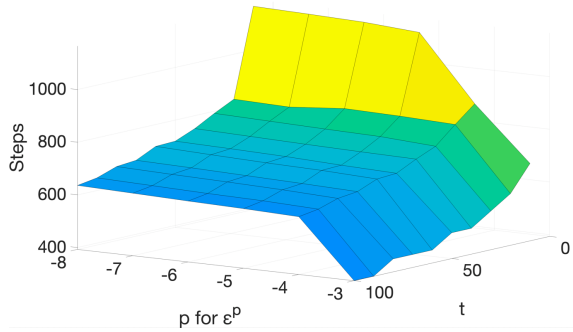


(a) Number of steps for Barrier Method to converge for input t and p for ϵ^p for CNF_2

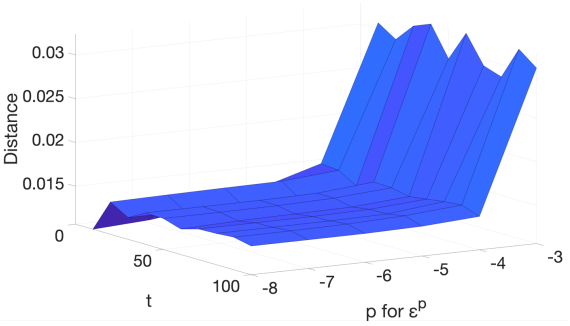


(b) Distance of closest unchopped corner to convergence point of Barrier Method for input t and p for ϵ^p for CNF_2

Figure 4.2: Analysis of Barrier Method Convergence: Exploration of the number of steps and the distance to the closest unchopped corner as influenced by input parameters t and p for ϵ^p, for CNF_2



(a) Number of steps for Barrier Method to converge for input t and p for ϵ^p for CNF_3



(b) Distance of closest unchopped corner to convergence point of Barrier Method for input t and p for ϵ^p for CNF_3

Figure 4.3: Analysis of Barrier Method Convergence: Exploration of the number of steps and the distance to the closest unchopped corner as influenced by input parameters t and p for ϵ^p, for CNF_3

4.2 Analysis

Given a CNF sentence, to determine a solution's existence relies on the convergence point, particularly whether it is sufficiently close to the corners while remaining within the satisfiable region. To investigate this, a test was conducted to evaluate the average convergence of the Barrier Method for both unsatisfiable and satisfiable CNF sentences. The test comprised 100 data points of final convergence for each dimension ranging from $n = 3$ to $n = 10$, to determine if the convergence value varies with the number of dimensions. The convergence value is determined by computing the norm of all final convergence points for the $2n$ directions and selecting the maximum value among them. A random set of satisfiable and unsatisfiable CNF sentences was constructed, and the average convergence value is summarized in Table 4.1 and 4.2 below.

Table 4.1: Average convergence value for satisfiable CNF sentences in n dimensions

n	3	4	5	6	7	8	9	10
Average Convergence	0.985	0.984	0.983	0.981	0.975	0.971	0.972	0.974

Table 4.2: Average convergence value for unsatisfiable CNF sentences in n dimensions

n	3	4	5	6	7	8	9	10
Average Convergence	0.149	0.195	0.311	0.394	0.368	0.443	0.381	0.320

The tables above demonstrate that satisfiable CNF sentences consistently achieve a maximum norm of 0.97 using the Barrier Method. Consequently, the Barrier Method attains approximately a 97% convergence rate for them. It's noteworthy that, except for a few CNF sentences with very few clauses, the convergence value for satisfiable ones never dropped below 0.97 across the 100 samples for each $n = 3$ to $n = 10$.

For unsatisfiable CNFs sentences the average maximum convergence ranged from 0.14 to 0.45. While this outcome was somewhat expected, the more crucial statistics would be the greatest maximum convergence value for an unsatisfiable CNF sentence. If an upper bound can be established for the greatest maximum convergence value, it would facilitate the detection of CNF sentence satisfiability. In the 100 samples, the greatest maximum convergence value was recorded for each n, as shown in Table 4.3 below.

Table 4.3: Greatest convergence value for unsatisfiable CNF sentences in n dimensions

n	3	4	5	6	7	8	9	10
Greatest Convergence	0.282	0.425	0.482	0.523	0.552	0.445	0.603	0.581

The table above reveals that the maximum norm never exceeded 0.603 for an unsatisfiable CNF sentence. With these samples in hand, we can now infer the satisfiability of a given CNF sentence based on whether the maximum convergence value surpasses a certain threshold. A threshold range of $0.9 \leq S \leq 0.95$ appears to be reasonable and safe value for detecting satisfiablity.

4.2.1 Performance

The performance of the Barrier method can be assessed by examining how it scales with increasing n. This can be done by counting the total number of iterations of the Barrier method, which is equivalent to summing the total path lengths for all $2n$ directions. In the same experiment mentioned earlier, the average number of total iterations was calculated for 100 samples across dimensions ranging from $n = 3$ to $n = 10$ and subsequently the results were divided by n for time complexity analysis. The results are summarized in Table 4.4 and 4.5 below.

Table 4.4: Average number of iterations for unsatisfiable CNF sentences in n dimensions

n	3	4	5	6	7	8	9	10
Average iterations	6327	7618	8744	9754	11497	12875	14529	16446
Average iterations / n	2109	1905	1749	1626	1642	1609	1614	1645

Table 4.5: Average number of iterations for satisfiable CNF sentences in n dimensions

n	3	4	5	6	7	8	9	10
Average iterations	4508	5901	7398	8776	10337	11858	13489	15176
Average iterations / n	1503	1475	1479	1463	1477	1482	1499	1518

The data indicates that the average number of iterations for both satisfiable and unsatisfiable CNF sentences were similar, with slightly more iterations required for the unsatisfiable ones. Interestingly, the average number of iterations exhibits a linear time complexity, as evidenced by the fact that the average iterations divided by n remains roughly constant.

This observation is not entirely surprising, as the addition of each dimension necessitates the exploration of two additional directions by the Barrier Method, leading to a linear increase in iterations.

It's worth emphasizing that the Barrier Method itself does not exhibit linear time complexity. Each iteration of the method involves computation on all the clauses, resulting in a time complexity of $T(n) = n \times \text{length}(Cl)$, where $\text{length}(Cl)$ represents the length of the clauses. Assuming the clauses are some polynomial function of n, we could expect that for $n \leq 10$ the Barrier Method exhibits a reliable polynomial time complexity for detecting satisfiablity for a CNF sentence.

CHAPTER 5

CONCLUSION AND FUTURE WORK

Achieving a polynomial-time solution to the Boolean satisfiability problem (SAT) would be a groundbreaking advancement with profound implications for computer science, potentially resolving the longstanding question of whether P equals NP in complexity theory. In this thesis, we explored new approaches to SAT solving by leveraging Non-Euclidean geometry as a model for CHOP-SAT. Specifically, we developed a Barrier Method to effectively detect non-chopped corners and navigate the feasible region.

In Chapter 3, we explored the use of the Poincaré disk model to represent CHOP-SAT chops. By leveraging Non-Euclidean geometry, we developed methods for accurately projecting vertices onto hyperspheres and defining feasible regions. Through the Clause Chops methodology, each CNF clause was translated into a hypersphere cut, aiding in vertex elimination and solution identification. We also introduced approaches like the Directional Center Push Method and the Non-Chopped Neighbor Method to refine the chopping process.

Testing and analysis of the Barrier Method was done in Chapter 4, focusing on the determination of initial parameters and evaluating its performance. Convergence constraints, including parameters like μ, t, and ϵ, are discussed, with experimental findings revealing the optimal combination of parameters for maximizing efficiency and precision. Moving on to analysis, the chapter explores the method's performance in converging satisfiable and unsatisfiable CNF sentences, providing insights into its efficacy based on convergence thresholds. The method showed great results for $n \leq 10$, demonstrating an average convergence rate of approximately 97% for satisfiable CNF sentences.

Future work may include further assessing the convergence rate of the Barrier Method. Expanding testing to substantially higher dimensions could provide deeper insights into the validity of the algorithm. Alternatively, refining upper bound estimates for the con-

vergence rate of unsatisfiable CNF sentences could help identifying corners still within the feasible region. Another avenue to explore is leveraging hyperbolic distance within the model to take advantage of the projection of corners onto ideal points. While this study primarily relied on Euclidean geometry for describing chops and values, incorporating hyperbolic distance as a metric, rather than just a model, could exploit the advantages offered by corners projected onto ideal points.

APPENDIX A

EUCLIDEAN CHOP-SAT

Given m conjuncts, C_i, $i = 1, \ldots, m$, then let:

$$C_i = L_1 \vee L_2 \vee \ldots \vee L_k$$

Note that any complete truth assignment with $\neg L_1 \wedge \neg L_2 \wedge \ldots \wedge \neg L_k$ makes C_i false. Observe that:

- If $k = n$, then this eliminates 1 solution ($H_0 \equiv$ 0-D vertex).
- If $k = n - 1$, then this eliminates 2 solutions ($H_1 \equiv$ 1-D segment).
- If $k = n - 2$, then this eliminates 4 solutions ($H_2 \equiv$ 2-D square).
- ...
- If $k = 1$, then this eliminates half the solutions in the hypercube (H_{n-1}).

The individual hyperplane is determined as follows. Let $A = \{1, 2, \ldots, n\}$ indicate the atoms, and $I \subseteq A$. Given $C_i = L_1 \vee L_2 \vee \ldots \vee L_k$, then define α_i, the hyperplane normal vector, as follows:

$$\forall i_j \in I, \alpha_i(i_j) = 1 \text{ if } L_j \text{ is an atom } a_{i_j}, \text{ else } -1$$

$$\forall m \notin I, \alpha_i(m) = 0$$

$$\alpha_i = \frac{\alpha_i}{|\alpha_i|_k}$$

In order to get the constant for the hyperplane equation, a point must be found on the hyperplane. This is selected so that the hyperplane cuts the edges of the hypercube at a distance ξ from the non-solution vertex. This distance depends on the number k of literals:

$$d = ||\xi \frac{\bar{\mathbf{b}}_k}{k}||$$

where $\mathbf{b}_k$ is a k-tuple of 1's. Next:

$\forall i_j \in I, p(i_j) = 0$ if L_i is an atom, else 1

$\forall m \notin I, p(m) = 0$

Then p is a non-solution vertex. To find a point, q, on the hyperplane:

$$q = p + d\alpha_i$$

This allows a solution for the constant, c, in the hyperplane:

$$c_i = -(\alpha_i \cdot q)$$

This yields the hyperplane equation:

$$\alpha_i \cdot x + c = 0$$

and the resulting inequality:

$$-\alpha_i \cdot x \leq c$$

Thus, to solve a CNF instance:

1. Find the linear inequality for each conjunct.
2. Set up an $m \times n$ matrix, A, with row i set to $-\alpha_i$ (the negative of the hyperplane normal).
3. Set up an $n \times 1$ vector b with row i set to c_i (the constant from hyperplane i).
4. Apply the interior-point method for linear programming with A and b specifying the inequalities, and with $0 \leq x \leq 1$. Minimize $f^T x$ with $x \in F$, where F is the feasible region, using $f = e_k$, i.e., the unit vector in the kth dimension. Call the resulting n-dimensional solution $x_{k,1}$.
5. Apply the interior-point method for linear programming with A and b specifying the inequalities, no equality constraints, and with $0 \leq x \leq 1$. Minimize $f^T x$ with $x \in F$, where F is the feasible region, using $f = -e_k$, i.e., the unit vector in the kth dimension. Call the resulting n-dimensional solution $x_{k,0}$.
6. If $x_{k,0}(1) = 0$ or $x_{k,1}(1) = 0$, then it is possible there is a solution for the CNF sentence, S. If $x_{k,1}(1) > 0$ and $x_{k,0}(1) < 1$, then there is no satisfying solution.

Steps 4 and 5 are guaranteed to find a solution with $x_{k,1}(1) = 0$ or $x_{k,0}(1) = 1$, if there is such a point in the feasible region; however, this point may be on a face of the hypercube, and not at a corner [12].

APPENDIX B

BOYD FORCE FIELD METHOD

The Barrier Method is formulated as a force field problem as described by Boyd and Vandenberg [**?**]. For each point $x \in F$, a barrier force is defined for each constraint surface:

$$F_i(x) = \nabla(-\log(-f_i(x))) = \nabla f_i(x)$$

where $F_i(x)$ is the force vector at point x from the ith constraint, and $f_i(x)$ is the (minimal) distance function from x to the ith constraint surface. The projection constraint force (called the forcing direction force) is:

$$F_0(x) = -t\nabla f_0(x)$$

where $F_0(x)$ is the forcing direction force at x and $f_0(x)$ is $f^T x$ where f is the direction of the forcing vector.
These forces are based on the logarithmic (barrier) function:

$$\Phi(x) = -\sum_{i=1}^{m} \log(-f_i(x))$$

and the distance function for hyperplanes is:

$$f_i(x) = a_i^T x + c_i$$

and for hyperspheres:

$$f_i(x) = \|C_i - x\| - r_i$$

where C_i and r_i are the center and radius, respectively, of the hypersphere. Then the force field model is defined in terms of forces generated by the minimization impulse function (to move in a certain direction) and the repulsive force of the constraint surfaces. Boyd gives the hyperplane forces which in our representation are:

$$F_i(x) = \frac{-a_i}{b_i - a_i^T x}$$
$$F_0(x) = tf$$

The hypersphere forces are derived as follows:

$$f_0(x) = f^T x = \sum_{i=1}^{n} f(i)x(i)$$

Therefore:

$$\nabla f_0(x) = \begin{bmatrix} \frac{\partial f_0}{\partial x_1} \\ \frac{\partial f_0}{\partial x_2} \\ \vdots \\ \frac{\partial f_0}{\partial x_n} \end{bmatrix} = \begin{bmatrix} f(1) \\ f(2) \\ \vdots \\ f(n) \end{bmatrix} = f$$

which implies that:

$$F_0(x) = tf$$

In addition:

$$f_i(x) = \left(\sum_{j=1}^{n} (C_i(j) - x(j))^2 \right)^{1/2} - r_i$$

which means that:

$$\frac{\partial f_i(x)}{\partial x_j} = \frac{1}{2} \left((C_i(j) - x(j))^2 \right)^{-1/2} (2(C_i(j) - x(j)))(-1) = \frac{x(j) - C_i(j)}{\|C_i - x\|}$$

and finally:

$$\nabla f_i(x) = \frac{x - C_i}{\|C_i - x\|}$$

and

$$F_i(x) = \frac{x - C_i}{\|C_i - x\|(\|C_i - x\| - r_i)}$$

In order to encourage moving toward the disk boundary, another forcing function may be defined as:

$$F_b(x) = \frac{t_b x}{\|x\|}$$

where t_b is a magnitude value.

Given $x \in F$, $t^{(0)} > 0$, $\mu > 1$, $\epsilon > 0$, then the Barrier Method is [12]:

repeat

1. Centering step: find force equilibrium point $x^*(t)$ of $tf_0 + \Phi$
2. Update: Set x to $x^*(t)$
3. Stopping Criterion: quit if $\mu/t < \epsilon$
4. Increase t: Set t to μt

REFERENCES

[1] James W. Anderson. *Hyperbolic Geometry*. Springer-Verlag, London, 2nd edition, 2005.

[2] Marcel Berger. *Geometry II*. Springer, translated edition edition, 1987. Original work published in 1977.

[3] Armin Biere, Marijn Heule, Hans van Maaren, and Toby Walsch. *Handbook of Satisfiability*. IOS Press, 2008.

[4] Václav Chvátal. Edmonds polytopes and a hierarchy of combinatorial problems. *Discrete Mathematics*, 4:305–337, 1973.

[5] Stephen A. Cook. The complexity of theorem-proving procedures. In *Proceedings of the Third Annual ACM Symposium on Theory of Computing*, STOC '71, pages 151–158, Shaker Heights, Ohio, USA, 1971. Association for Computing Machinery.

[6] G.B. Dantzig. Application of the simplex method to a transportation problem. In T.C. Koopmans, editor, *Activity Analysis of Production and Allocation*, pages 359–373. John Wiley and Sons, New York, 1951.

[7] Euclid. *Elements*. Green Lion Press; Later Printing edition, January 2002. Translated by Thomas L. Heath.

[8] R.E. Gomory. Outline of an algorithm for integer solution to linear programs. *Bulletin of the American Mathematical Society*, 64(5):275–278, 1958.

[9] Andrew J. Hanson. Rotations for n-dimensional graphics. *Computer Science Department, Indiana University*, 1995.

[10] T. C. Henderson, D. Sacharny, A. Mitiche, X. Fan, A. Lessen, I. Rajan, and T. Nishida. CHOP-SAT: A new approach to solving SAT and probabilistic SAT for agent knowledge bases. In *International Conference on Agents and Artificial Intelligence*, Lisbon, Portugal, Feb. 2023.

[11] Thomas C. Henderson, Amelia Lessen, Ishaan Rajan, Tessa Nishida, and Kutay Eken. Chop-sat: A new method for knowledge-based decision making. In *International Conference on Intelligent Autonomous Agents*, Suwon, South Korea, July 2023.

[12] Thomas C. Henderson, David Sacharny, Xiuyi Fan, Amar Mitiche, and Thatcher Geary. GEO-SAT: A geometric approach to satisfiability. Technical Report UUCS-23-003, School of Computing, University of Utah, Salt Lake City, UT, November 2023.

[13] Marijn J.H. Heule and Oliver Kullmann. The science of brute force. *Communications of the ACM*, 60(7):70–79, 2017.

[14] Michael Sipser. *Introduction to the Theory of Computation*. Cengage Learning, Independence, KY, 2012.