GEO-SAT: A Geometric Approach to Satisfiability

Thomas C. Henderson, David Sacharny, Xiuyi Fan,
Amar Mitiche and Thatcher Geary
University of Utah

UUCS-23-003

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

10 October 2023

Abstract

Boolean Satisfiability (SAT) is posed as a geometric problem and geometric solutions are sought. The approach is based on the fact that Conjunctive Normal Form (CNF) sentences over n logical variables (a standard representation for SAT) can be converted to a bounded convex feasible region in n-dimensional space. Linear programming methods (with polynomial time complexity) are applied in order to find solutions. Methods are given to convert a CNF sentence into both Euclidean and non-Euclidean geometries, and the advantages and disadvantages of this approach discussed.

1 Introduction

SAT can be defined as follows: Given a logical sentence over n variables, determine if there is an assignment of truth values to the variables which makes the sentence true. Note that for an n-variable sentence, there are 2^n possible complete truth assignments (also called models). The complete set of models (or complete conjunction set) can be mapped onto the vertexes of a unit n-dimensional hypercube, called H_n in a straightforward way: the truth assignment values serve as the coordinates in n-dimensional space. Assume the logical sentences are represented as CNF sentences, i.e., a conjunction of disjunctions of literals. Then any assignment of truth values which makes a disjunction false renders the CNF sentence false. If the disjunction has k literals, then there is one truth assignment to the atoms of these literals which makes the disjunction false; however, the variables not in the disjunction can take on either truth value, and so there are 2^{n-k} complete conjunctions which make the sentence false. This set is in fact a sub-hypercube of H_n.

The Euclidean method presented here shows how each disjunction in the CNF sentence gives rise to a hyperplane which separates the non-solution vertexes (on the negative side of the hyperplane) of H_n from the solution vertexes (on the non-negative side of the hyperplane); i.e., the intersection of the non-negative half-spaces of these hyperplanes results in a convex feasible region which must contain any solution which exists. The non-Euclidean method shows how H_n can be projected onto the n-dimensional unit hypersphere considered as an n-dimensional Poincaré Disk. The advantage of this approach is that the vertexes of H_n are mapped onto the surface of the disk and are thus at infinite distance (in terms of hyperbolic geometry) from the center of the disk. The idea is that this property makes the solutions more readily identifiable.

2 Background

For a detailed discussion of the SAT problem and its complexity, see [11]. Related work on a geometric approach started with Gomory [7] who sought integer solutions for linear programs. Given the semantics of the literals in a disjunction, a linear inequality can be formed summing x_i for atoms in the clause and $(1 - x_i)$ for negated atoms in the clause and setting this to be greater than or equal to 1. Next, a $\{0, 1\}$ solution is sought resulting in an integer linear programming problem. If a non-$\{0, 1\}$ solution is found, Gomory proposed a way to separate (via a *cutting plane*) that solution from all integer solutions. This method has been used in finding lower complexity ways to provide theorems for proving the boundedness of polytopes, cutting plane proofs for unsatisfiable sentences, pseudo-Boolean optimization, etc. (see [2, 3, 4, 5, 6]). The *Chop-SAT* method has been proposed as a way to solve SAT and PSAT [8, 9]. The *Chop-SAT* method was developed independently of Gomory and Chvatal's work, and is based on fundamentally different geometric insights.

3 The Euclidean Approach

The CNF SAT problem is cast as a linear programming problem:

Minimize $\mathbf{f}^T x$

Subject to: $Ax \leq c$

where each constraint is given by:

$-\alpha_i \cdot x \leq c_i$

A solution for the SAT sentence exists iff a solution exists for the LP problem such that every component of x has a value equal to 0 or 1.

Given a set of m conjuncts, $C_i, i = 1 : m$, each conjunct is used to produce a hyperplane of dimension $n - 1$ which separates the solutions (i.e., some subset of vertexes of H_n) from non-solutions. The hyperplane for the i^{th} conjunct is:

$$\alpha(i) \cdot x + c = 0$$

Each of these hyperplanes produces an inequality:

$$-\alpha(i) \cdot x \leq c_i$$

for which the signed distance of a point is used to separate solution from non-solution vertexes. A matrix, A, is produced where each row is the $1 \times n$-tuple $\alpha(i)$, *the unit normal to the hyperplane*. An $n \times 1$ vector, c, is constructed where the i^{th} element of c is c_i.

The way these hyperplanes are constructed, it is now possible to run the interior-point method for linear programming to find feasible points which minimize $f^T x$ for $x \in \mathcal{F}$, where $\mathcal{F}$ is the feasible region and f is a unit vector in the desired projection direction. Note that if neither of the projection onto the positive and negative directions of some basis vector results in a 0 or 1 value, respectively, then the CNF has no solution. However, there are feasible regions for unsatisfiable sentences which do have such 0,1 projections, so this is a sufficient but not necessary condition.

4 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).
- $\ldots$
- 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 $\mathcal{I} \subseteq A$. Given $C_i = L_1 \vee L_2 \vee \ldots \vee L_k$, then define α_i, the hyperplane normal vector, as follows.

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

$$\forall m \notin \mathcal{I}, \alpha_i(m) = 0$$
$$\alpha_i = \frac{\alpha_i}{\parallel \alpha_i \parallel}$$

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:

$$d = \parallel \xi \frac{\overline{\mathbf{b_k}}}{k} \parallel$$

where $\overline{b_k}$ is a k-tuple of 1's. Next:

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

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

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:

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

4.1 The Chop SAT Algorithm

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 $x \in \mathcal{F}$, where $\mathcal{F}$ is the feasible region, using $f = \mathbf{e}_k$, i.e., the unit vector in the k^{th} 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 $x \in \mathcal{F}$, where $\mathcal{F}$ is the feasible region, using $f = -\mathbf{e}_k$, i.e., the unit vector in the k^{th} 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.

Algorithm Chop SAT
On input:
S: CNF sentence
On output:
res: for each dimension, the min and max values found
sol: 1 if complete SAT solution found, else 0 *begin*
A = matrix of negated hyperplane normals (1 per row)
b = vector of hyperplane constants *for each* atom $a \in$ S
d is dimension associated with a
e_d is unit vector in dimension d
x_{10} = linear programming solution projected on -e_d
x_{11} = linear programming solution projected on e_d
res(d,1) = $x_{10}(d)$
res(d,2) = $x_{11}(d)$
if x_{10} or x_{11} is complete 0/1 solution
sol = 1;
end
end

Now consider the time complexity of the approach. Converting the conjuncts to hyperplanes is clearly polynomial given that there are m conjuncts, and each has at most n literals. Given the sizes of A and b, the interior-point method for linear programming requires only polynomial time (see Potra[10]).

4.2 Some Examples

4.2.1 2D One Solution

Consider the two clauses in modus ponens:

$1.a_1$

$2.\neg a_1 \vee a_2$

Then the feasible region is shown in Figure 1. The hyperplane found for conjunct 1 (with $\xi = 0.9$)

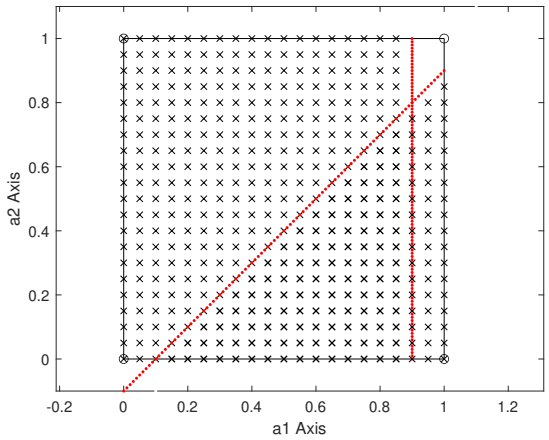

Figure 1: The Feasible Region for Modus Ponens.

is:

$1.0a_1 + 0a_2 - 0.9 = 0$

while the hyperplane for conjunct 2 is:

$-0.7071a_1 + 0.7071a_2 + 0.7071 = 0$

The solutions are:

$x_{10}(1) = 0.9$

and

$x_{11}(1) = 1$

4.2.2 2D No Solution

For a second example, consider a CNF sentence with no satisfying solution:

$1.\neg a_1 \vee \neg a_2$

$2.\neg a_1 \vee a_2$

$3.a_1 \vee \neg a_2$

$4.a_1 \vee a_2$

Then the feasible region is shown in Figure 2. The hyperplane found for conjunct 1 is:

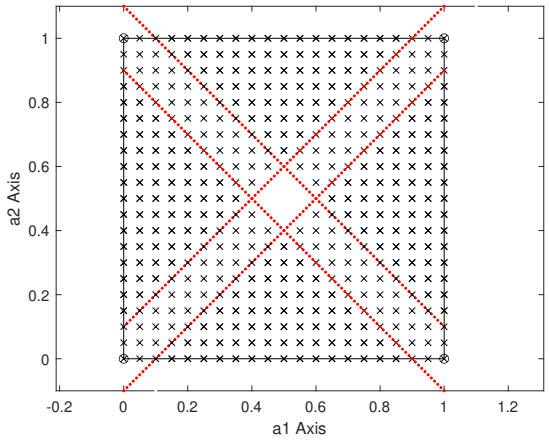

Figure 2: The Feasible Region for an Unsatisfiable CNF Sentence.

$$-0.7071a_1 - 0.7071a_2 + 0.7778 = 0$$

The hyperplane found for conjunct 2 is:

$$-0.7071a_1 + 0.7071a_2 + 0.0707 = 0$$

The hyperplane found for conjunct 3 is:

$$0.7071a_1 - 0.7071a_2 + 0.0707 = 0$$

The hyperplane found for conjunct 4 is:

$$0.7071a_1 + 0.7071a_2 - 0.6364 = 0$$

The linear programming solutions are:

$$x_{20}(1) = 0.4999$$

and

$$x_{21}(1) = 0.5001$$

indicating there is no satisfying solution for S.

4.2.3 3D Two Solutions

As a final example, consider the case with 3 variables, and such that $(\neg a_1 \wedge a_2 \wedge a_3) \vee (a_1 \wedge \neg a_2 \wedge \neg a_3)$ is true. Re-writing this in CNF yields:

$1. : a_1 \vee a_2$

$2. : a_1 \vee a_3$

$3. : \neg a_1 \vee \neg a_2$

$4. : \neg a_2 \vee a_3$

$5. : \neg a_1 \vee \neg a_3$

$6. : a_2 \vee \neg a_3$

Then the hyperplane equations are:

$C_1 : 0.7071a_1 + 0.7071a_2 - 0.7071 = 0$

$C_2 : 0.7071a_1 + 0.7071a_3 - 0.7071 = 0$

$C_3 : -0.7071a_1 - 0.7071a_2 + 0.7071 = 0$

$C_4 : -0.7071a_2 + 0.7071a_3 + 0 = 0$

$C_5 : -0.7071a_1 - 0.7071a_3 + 0.7071 = 0$

$C_6 : 0.7071a_2 - 0.7071a_3 + 0 = 0$

The linear programming solutions are:

$x_1 = [0; 1; 1]$

and

$x_2 = [1; 0; 0]$

This approach has been tested on thousands of randomly generated CNF sentences (both consistent and inconsistent) and always returned a solution where there was one, and gave the empty set where there was none. In addition, a test was made on a consistent 1000-atom, 30,000-conjunct CNF in which the solution was found in about five minutes (this was in Matlab with no special optimizations). On the other hand, for a CNF from satcompetition.org/2002 with 450 variables and 2025 clauses (each with 3 literals), *Chop SAT* produced a 0/1 in each dimension, but no complete SAT solution.

Several ways to guarantee finding a solution have been considered, but found lacking:

- Find vertexes of the feasible region and test them as solutions. $\rightarrow$ usually m enough greater than n so this is NP.
- Find projection axis so that projected value indicates a solution. $\rightarrow$ too many directions, so this is NP
- Find minimal volume circumscribing ellipsoid for feasible region. $\rightarrow$ this problem is NP
- Find maximal length stick (line segment) in feasible region. $\rightarrow$ this problem is NP
- Find maximal volume inscribed ellipsoid in feasible region. $\rightarrow$ although this problem is polynomial time, the maximal volume requires maximizing the minimal semi-axis length, and not maximizing the maximal semi-axis length, so the ellipsoid does not point to a solution vertex. This method maximizes the determinant of an appropriate matrix; if the trace could be maximized, then perhaps this would lead to the desired result.

Thus, to this point, no algorithm guaranteeing a solution in polynomial time has been found using the Euclidean approach.

4.3 Probabilistic SAT

Points in H_n provide more than just solutions at the vertexes. Every point in the hypercube corresponds to a probability assignment to the atoms. The feasible region defines the set of such assignments that are consistent with the logical sentence. More formally, the Probabilistic SAT (PSAT) problem is defined as follows. Assume that each CNF clause, C_i has a probability, p_i, assigned to it. Then we seek a probability function, π, that maps the complete conjunctions, Ω, to $[0,1]^n$ such that:

1. $0 \leq \pi(\omega_j) \leq 1$
2. $\sum_{j=0}^{2^n-1} \pi(\omega_j) = 1$
3. $p_i = \sum_{\omega_j \vdash C_i} p(\omega_j)$

where $\omega_j \in \Omega$.

For example, if $S = A \vee B$, then since the models $\{(0,1),(1,0),(1,1)\}$ satisfy S, and if the models are equally likely, then the probability of A is 2/3, as is the probability of B. It turns out that the mean of the points found by linear programming projection in the positive and negative directions of all the basis axes directions provides a useful estimate of the actual atom probabilities; in this case, the mean of the unique extreme points is in fact the actual probabilities [2/3,2/3]. For a set of independent variables, this can be used to produce a PSAT solution since every model probability is the product of n literal probabilities. This method has been exploited with knowledge-based decision making agents [9].

5 The Non-Euclidean Approach

The motivation for using non-Euclidean geometry is that it allows us to put the solutions at a unique location: at infinite distance from the origin in terms of non-Euclidean distance). That is, SAT is mapped onto the Poincare Disk as follows (see Figure 3). The corners of the n-dimensional

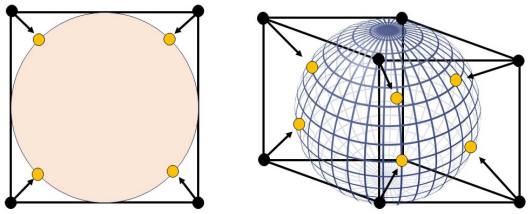

Figure 3: The Vertexes of the Hypercube are Projected onto the Hypersphere.

hypercube are projected onto the n-dimensional hypersphere, D_n, as ideal points (i.e., points on the surface of the hypersphere - note that these are not points in the Poincaré Disk). These ideal points are an infinite distance from the center of the unit hypersphere. Unlike in Euclidean geometry where the hyperplane chops usually produce a bounded convex polytope for the feasible region whether or not a solution exists, in the case of the Poincaré Disk representation, the feasible region is only bounded when no solution is in the feasible region. Given this fact, the goal is to find low-complexity algorithms to determine whether or not there exists a sequence of points in the feasible region such that in the limit their distance from the origin is infinite (in hyperbolic geometry).

The goal is to provide a representation in terms of the Poincaré Disk which allows the solution vertexes to be found through efficient geometric algorithms. To set up this representation, a few basic facts concerning the Poincaré Disk geometry must be defined.

5.1 Poincaré Disk Distance

Consider the 2-dimensional Poincaré Disk (Figure 4, upper left circle). The distance between two points, p and q, in the Poincaré Disk is defined in terms of Euclidean distance on points given as complex numbers:

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

where a and b are the intersection points with the unit circle of the unique circle (Figure 4, lower right circle) through p and q which is orthogonal to the unit circle. Moreover, the points are arranged in the order a, p, q and b along the circle. An alternative formulation which does not require the

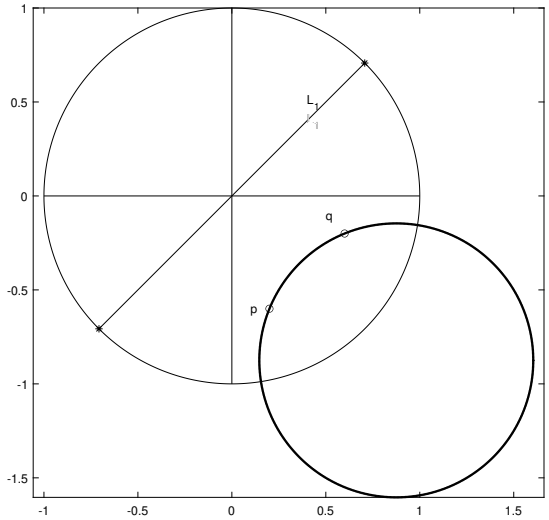

Figure 4: The 2D Poincaré Disk; L_1 is a line through the center, $\mathcal{O} = (0, 0)$, i.e., a diameter; L_2 is a circular arc which is orthogonal to the unit circle.

orthogonal circle is given by:

$$d(p, q) = acosh(1 + \frac{2|pq|^2|r|^2}{(|r|^2 - |op|^2)(|r|^2 - |oq|^2)})$$

where $r = 1$ for the unit disk, and $|op|$ and $|oq|$ are the Euclidean distances of p and q from the origin, respectively.

5.2 The Orthogonal Circle through Two Points

The angle between two circles (defined in Euclidean coordinates) is given by:

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

where r_1 is the radius of the first circle, r_2 the radius of the second circle, and C_1 and C_2 are the centers of the two circles. Two circles are said to be orthogonal if $\theta = \frac{\pi}{2}$.

Given two points, p and q, in the Poincaré Disk, the straight line (in hyperbolic terms) through them can be found as follows. If p and q lie on a diameter of the Poincaré Disk, then the line is just the straight Euclidean line through the two points. This can be viewed as a circle of infinite radius through the two points. If p and q do not lie on a diameter line, then there are two circles to consider: the unit disk with $r_1 = 1$ and $C_1 = (0, 0)$, and the circle through p and q with radius r_2 and center $C_2 = (C_x, C_y)$. Since the circles are orthogonal:

$$cos(\theta) = cos(\frac{\pi}{2}) = 0$$

$$\Rightarrow 0 = 1 + r_2^2 - \|C_2\|^2$$
$$\Rightarrow r_2^2 = C_x^2 + C_y^2 - 1$$

Each point, p and q, gives rise to another equation; e.g., for p:

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

and substituting the first into the second:

$$C_x^2 + C_y^2 - 1 = C_x^2 + C_y^2 - 2p_xC_x - 2P_yC_y + p_x^2 + p_y^2$$

$$\Rightarrow p_xC_x + p_yC_y = \frac{p_x^2 + p_y^2 + 1}{2}$$

Likewise:

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

This can be solved as a linear system:
$A = \begin{bmatrix} p_x & p_y \\ q_x & q_y \end{bmatrix}$ $b = \begin{bmatrix} \frac{p_x^2+p_y^2+1}{2} \\ \frac{q_x^2+q_y^2+1}{2} \end{bmatrix}$ and solving for the center, C:

$$AC = b$$

Finally, the radius is found as follows:

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

5.3 Representation of Bounding Faces in the Poincaré Disk

In Euclidean geometry, the SAT problem is posed in terms of removing vertexes from the hypercube. Each conjunct in the CNF sentence provides a hyperplane which divides the feasible region from the non-satisfying solutions for that conjunct. However, in the Poincaré Disk, a way must be found to represent the original complete set of vertexes, and then some way to chop non-solutions from the feasible region. One way to go at this is to use a square like the one shown in Figure 5. The problem arises that this is likely to suffer from the same problem as the Euclidean representation, namely that a projection does not readily reveal solution corners.

Another representation that may allow lower complexity discovery of whether a solution exists or not is given by the feasible region shown in Figure 6. The $2n$ $(n-1)$-dimensional bounding faces of the hypercube are represented in the Poincaré Disk in terms of hyperspheres through the

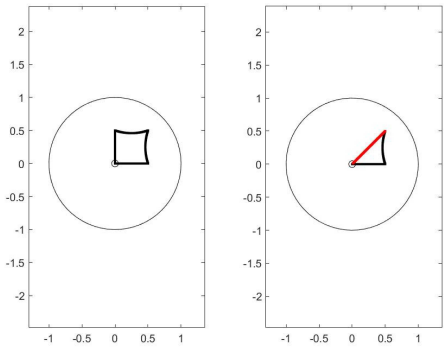

Figure 5: Representing the Complete Feasible Region as a Square in the Poincaré Disk. The left side is the complete feasible region, while the right side shows the feasible region when $(0, 1)$ is not a truth assignment that satisfies the CLF sentence.

corresponding face vertexes. The centers of these hyperspheres will lie along the coordinate axes (one each in the positive and negative directions). Let v_1 be the unit vector from the center of D_n to a vertex of the face, and v_2 be the unit vector in the desired axis direction. We seek the center, C and radius r of the hypersphere through the face vertexes. Given that the desired hypersphere is orthogonal to the unit disk, then the angle, θ, between v_1 and v_2 is given by:

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

which means that the distance, xd, along the axis from the origin to C is:

$$xd = \frac{1}{cos(\theta)}$$

which gives C. This process is done for the $2n$ bounding face constraints on the feasible region.

5.3.1 Clause Chops

As has been described, *lines* in the Poincaré Disk are circular arcs when viewed in Euclidean geometry, and this allows non-solution vertexes to be separated from the feasible region by choosing the appropriate side of the circle which defines the chop. What is required is an algorithm to convert from a CNF conjunct to the equation of a circle which has the non-solutions on one side and the remaining possible solutions on the other. This algorithm is provided below.

Given clause:

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

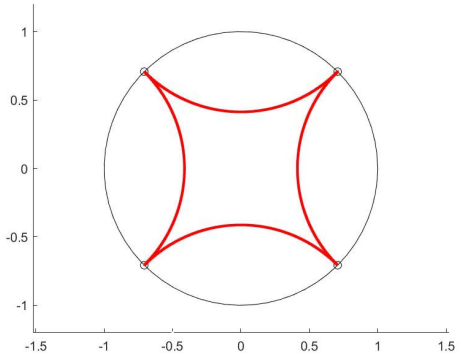

Figure 6: Representing the Complete Feasible Region as a Square with corners at infinity in the Poincaré Disk.

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

$v_i = -1$ *if* $\exists j \ni \neg a_i = L_j$

$v_i = 0$ *if no literal of* $a_i \in C$

$v_i = 1$ *if* $\exists j \ni a_i = L_j$

Let $\alpha = -V$ and $u_\alpha = \frac{a}{\|\alpha\|}$. Obtain an H_n vertex to be chopped by substituting -1 for 0's, if any, in α: $chop\_pt = \alpha_{-1 \leftarrow 0}$.

Obtain a non-chopped neighbor, nei_pt, by inverting a non-zero element of $chop\_pt$.

Obtain the projection point by sliding along an H_n edge connecting the $chop\_pt$ to the nei_pt by the desired percentage amount $\Delta \in (0, 1]$; that is:

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

To get the projection point on the Poincaré Disk boundary:

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

Then:

$$\theta = acos(u_a \cdot proj\_pt\_PD)$$

$$d = \frac{1}{cos(\theta)}$$

$$r_2 = \sqrt{d^2 - 1}$$
$$C = du_a$$

Consider the following example. Let $C = \neg A \vee \neg B$. Then $V = [-1, -1]$, $\alpha = [1, 1]$, and $\mu_\alpha = [0.7071, 0.7071]$. Then $chop\_pt = [1, 1]$ and $nei\_pt = [-1, 1]$. Let $\Delta = 0.9$, then $proj\_pt = [-0.8, 1]$ and $proj\_pt\_PD = [-0.6247, 0.7809]$ resulting in $\theta = 1.4601$ radians (83.66 degrees), and $d = 9.0554$.

Notice that if single-literal clauses are allowed, then the center of the chop hypersphere may flip sides to satisfy circle orthogonality. If so, the feasible side of the hypersphere is the interior. Such clauses can be avoided by deleting the atom of the literal and substituting its truth value into the other clause where it appears and then reducing these clauses accordingly. Figure 7 shows example 2D and 3D chops. Note that this formulation works in any dimension.

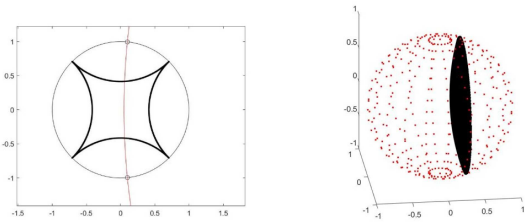

Figure 7: Examples of Chops in 2D and 3D.

The Δ parameter determines how far the chop occurs from the non-solution vertexes which are being cut. Figure 8 shows some example positions for chops. Figure 9 shows the chops for Modus Ponens, Figure 10 shows three edges chopped, and Figure 11 shows a 3D face chop.

5.4 Method

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

- A set of chops are produced for the CNF clauses; these chops are hyperplanes just like those used in Euclidean geometry. Note that they can also be viewed as infinite radius hyperspheres through those neighbor vertexes, however, these hyperspheres are not orthogonal to the unit disk.
- In addition to the constraint surfaces arising from the chopped vertexes, it is also necessary to bound the feasible region to be interior to the transformed unit cube. To this end, a set of face constraint hyperspheres are added; these hyperspheres are orthogonal to the unit disk.

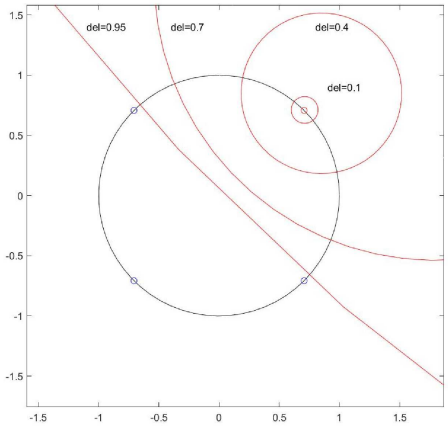

Figure 8: The Impact of $Delta$ on Chop Distance from Non-Solution Vertexes.

- A Barrier Method type algorithm is developed for this constraint set which moves in the selected projection direction (i.e., maximizes the projection) using the hypersphere surfaces as barrier constraints. This is implemented as a force field method and is described in detail below. Other forces may also be introduced to ameliorate the convergence; e.g., a force away from the origin.
- A solution is considered to exist if the convergence point is close enough to a solution vertex; i.e., in the limit, the distance from the origin of the points in the sequence grows without limit.

In Euclidean geometry, existing software tools exist to solve linear programming problems (e.g., the Matlab function *linprog*). However, a variant was implemented here to handle the mix of hyperplane and hypersphere surfaces.

5.4.1 Barrier Method

The Barrier Method is formulated as a force field problem as described by Boyd and Vandenberg [1]. For each point $x \in \mathcal{F}$, a barrier force is defined for each constraint surface:

$$F_i(x) = \nabla(-log(-f_i(x))) = \frac{\nabla f_i(x)}{f_i(x)}$$

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

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

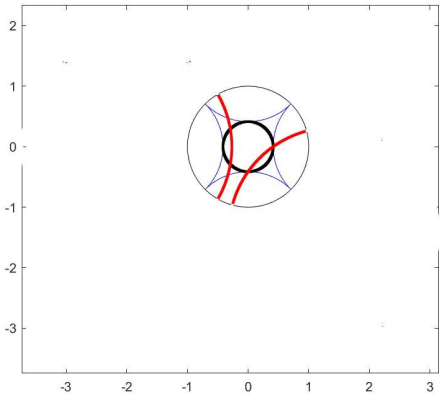

Figure 9: Chops for Modus Ponens.

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:

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

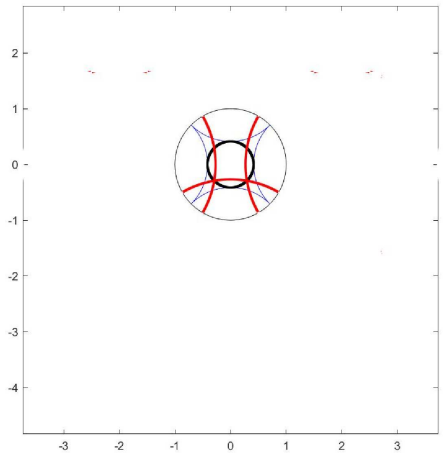

Figure 10: Three Edges Chop.

Therefore:

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

which implies that:

$$F_0(x) = tf$$

In addition:

$$f_i(x) = ((C(1) - x(1))^2 + \cdots + (C(n) - x(n))^2)^{1/2} - r_i$$

which means that:

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

and finally:

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

and

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

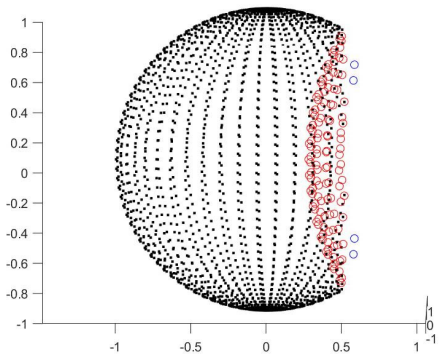

Figure 11: 3D Face Chop.

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

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

Figure 12 shows the forces resulting from only constraint surfaces repulsion forces (left), an upward external force (middle), and a downward external force (right). The basic approach is to use the origin as an initial starting point since the origin is guaranteed to be in the feasible region, and find the equilibrium point when no force is applied (this corresponds to the analytic center). Next, starting from the analytic center and using a forcing direction, follow the resultant forces and move to the equilibrium point for this set of forces. Choosing different forcing directions results in an exploration of the feasible region. The problem is to determine an effective and efficient search strategy.

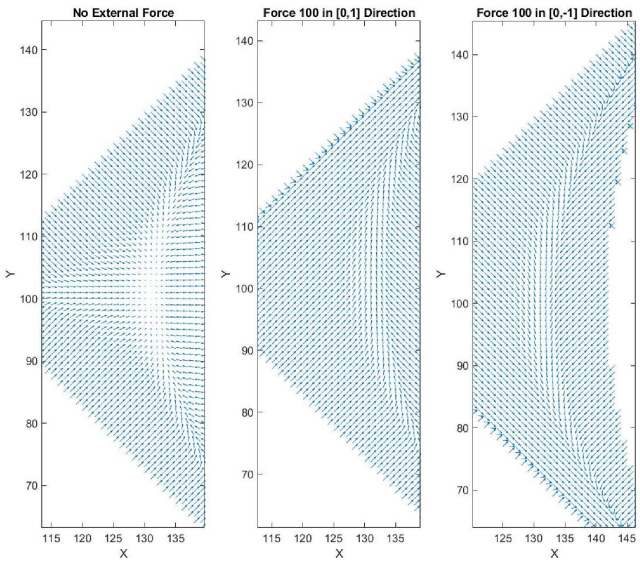

Figure 12: Example Force Fields: Only constraint surface repulsion forces are present (left); An added upward force (middle); A downward external force (right).

Figure 13 shows the Barrier Method applied to finding solutions for a 2D problem with three solutions. A 3-D example path is shown in Figure 14.

6 Conclusions and Future Work

The Geo-SAT approach has been extended to non-Euclidean geometry, and first results indicate that solutions can be found. However, a number of things remain to be done:

1. Determine the convergence properties of the Barrier method in non-Euclidean space.
2. Perform a systematic study of solutions finding behavior in dimensions higher than 3.
3. Explore methods to speed up the discovery of a solution; e.g., as a force path is followed, project the path points onto the appropriate constraint surfaces and determine whether the direction of motion of the projected points can be used to move to a solution vertex more rapidly.

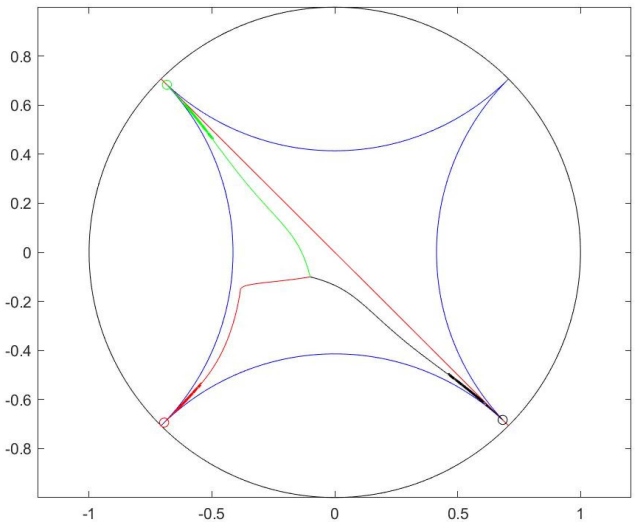

Figure 13: The Paths for the Barrier Method finding Solutions in a 2D Problem.

References

[1] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, Cambridge, UK, 2021.

[2] S.R. Buss and P. Clote. Cutting Planes, Connectivity, and Threshold Logic. *Archive for Mathematic Logic*, 35:33–62, 1996.

[3] V. Chvatal. Edmonds Polytopes and a Hierarchy of Combinatorial Problems. *Discrete Mathematics*, 4:305–337, 1973.

[4] V. Chvatal. Cutting Planes in Combinatorics. *European Journal of Combinatorics*, 6:217–226, 1985.

[5] W. Cook, C.R. Coullard, and G. Turan. On the Complexity of Cutting-Plane Proofs. *Discrete Applied Mathematics*, 18:25–38, 1987.

[6] J. Devriendt, S. Gocht, E. Demirovic, J. Nordstrom, and P.J. Stuckey. Cutting to the Core of Psuedo-Boolean Optimization: Combining Core-Guided Search with Cutting Planes Reasoning. In *Thirty-Fift AAAI Conference on Artificial Intelligence*. Elsevier, 2021.

[7] R.E. Gomory. Outline of an Algorithm for Integer Solution to Linear Programs. *Bulletin of the Americal Mathematical Society*, 64(5):275–278, 1958.

[8] Thomas C. Henderson, Amar Mitiche, Xiuyi Fan, and David Sacharny. Some Explorations in SAT. Technical Report UUCS-21-016, University of Utah, July 2021.

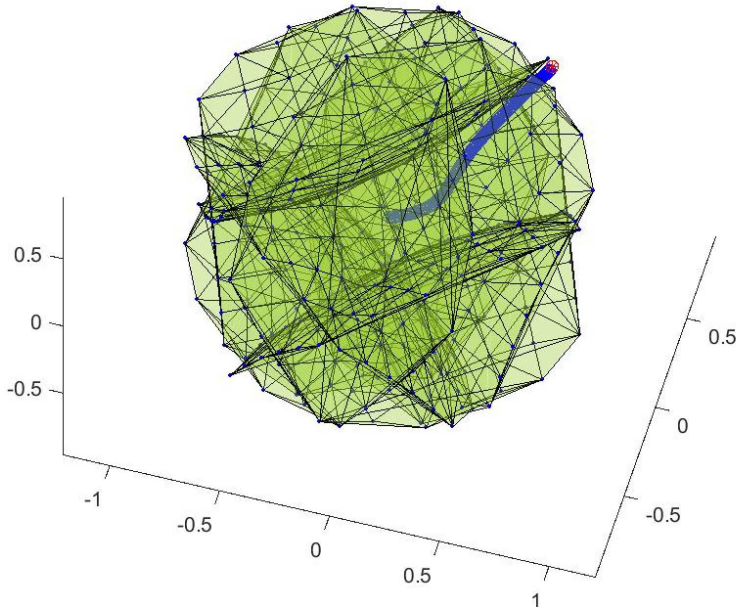

Figure 14: The Paths for the Barrier Method finding a Solution in a 3D Problem.

[9] Thomas C. Henderson, David Sacharny, Amar Mitiche, Xiuyi Fan, Amelia Lessen, Ishaan Rajan, and Tessa 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, Spain, February 2023.

[10] F.A. Potra and S.J. Wright. Interior Point Methods. *Journal of Computational and Applied Mathematics*, 124, 2004.

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