GEO-SAT: A New Approach for Knowledge-Based Agent Decision Making

Thomas C. Henderson, Amelia Lessen, Ishaan Rajan, Tessa Nishida, Kutay Eken, Xiuyu Fan, David Sacharny, Amar Mitiche and Thatcher Geary

[a] *University of Utah, School of Computing, Salt Lake City, 84112, UT, USA*

Abstract

Logical agents base their action selection decisions on inferences made over a logical knowledge base. Given a propositional logic knowledge base expressed in Conjunctive Normal Form (CNF), the knowledge can be converted into a geometrical format, and subsequent analysis takes place as geometrical operations on the feasible region in that representation. Two geometric representations are presented: the n-dimensional hypercube in Euclidean geometry and the n-dimensional Poincaré disk in non-Euclidean geometry. Based on these representations, two novel methods are proposed to: (1) find SAT solutions for the knowledge base (i.e., a truth assignment to each logical variable which makes the CNF sentence true), and (2) find a reasonable approximation to the atom probabilities given the current set of information. This allows agents to determine the semantics (truth) of the world as well as to estimate the probability of truth. The geometric method provides an efficient heuristic approach to solving SAT for CNF knowledge bases, and provides polynomial-time solutions of probabilistic SAT for independent variables, and good PSAT estimates for non-independent logical variables.

Keywords: Knowledge representation, geometric methods, SAT, PSAT

1. Introduction and Background

Given a propositional calculus knowledge base represented in Conjunctive Normal Form (CNF), an agent can find out information about the world by finding new sentences that are entailed by the current knowledge. In this way, an agent can determine proper actions. To do so, it may be useful to determine whether the CNF sentence has a solution (satisfiable) or not (unsatisfiable). This is called the SAT problem (for SATisfiability) problem. The SAT problem is NP complete [21].

Another aspect of interest to the agent is the probability that a specific atom (logical variable) is true. For example, for the CNF $S = A \vee B$, there are three solutions, (0, 1), (1, 0), (1, 1), which satisfy S and, assuming equal likelihood for all solutions, the probability of A is 2/3, and the probability of B is 2/3 Note that this is the mean of the models (truth assignments to variables) which satisfy the sentence. One way to determine the atom probabilities is to solve the *Probabilistic SAT* (PSAT) problem when each conjunct, C_i, is given a probability, p_i [9, 17]. That is, given n logical variables, there are 2^n unique truth assignments (also called models or the complete conjunction set) to the variables. The set of all models is called Ω, and ω_i is the model with binary assignments corresponding to the binary representation of $i-1$; e.g., ω_1 is all zero assignments - all false. The PSAT problem consists of determining a probability distribution, $\pi : \Omega \rightarrow [0,1]$ such that $\sum_{i=1}^{2^n} \pi(\omega_i) = 1$, and $\sum_{\omega_i \models C_j} \pi(\omega_i) = p_j$, for all conjunct probabilities, p_i. The probability of an atom is then found as $Prob(A) = \sum_{\omega_i \models A} \pi(\omega_i)$. We have also previously described how to solve the Probabilistic Sentence Satisfiability Problem (PSSAT) [12] which in certain cases provides a PSAT solution (i.e., given independent variables). The methods we are proposing differ from standard methods in that we solve linear or nonlinear systems of equations rather than having to consider the full joint probability distribution over the variables (e.g., like Bayesian networks [18] or Markov Logic Networks [7]).

The purpose of this study is to investigate *Chop-SAT* as an alternative way to answer these questions about SAT and PSAT [13, 15]. Work on the use of cutting planes started with Gomory [11] who sought integer solutions for linear programs. Given the semantics of the literals in a disjunction, then 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 was discovered independently and is based on a different set of insights into the nature of the CNF form.

Based on the *Chop-SAT* approach, the contributions here provide:

1. A method to determine whether a SAT solution exists, and
2. A method to determine an approximation to the atom probabilities.

2. Chop-SAT

A CNF sentence is the conjunction of a set of disjunctions where each disjunction is a literal (i.e., either an atom or the negation of an atom). A CNF sentence is then represented as $S = C_1 \wedge C_2 \wedge \ldots \wedge C_m$, where $C_i = L_{i,1} \vee L_{i,2} \vee \ldots \vee L_{i,k_i}$ where $L_{i,j} = a_p$ or $L_{i,j} = \neg a_p$, and a_p is an atom.

The satisfiability of a CNF sentence, S, over n variables can be converted to a geometric problem as follows. Consider the hypercube of dimension n centered at $[\frac{1}{2}, \frac{1}{2}, \ldots, \frac{1}{2}]$; call it H_n. Then each vertex of H_n represents a model for n variables. The vertexes also represent assignments of probability 0 or 1 for the truth of each variable. Every other point in H_n can be considered to give a probability on the interval $[0, 1]$ for each variable. E.g., the center of H_n represents a probability assignment of $1/2$ for each atom.

Next, consider a clause, C_i, of S with k_i literals. Since it is a disjunction, there is only one truth assignment over its literals which makes it false: namely, where the atom of each literal is assigned the value which makes the literal false. However, every atom not represented by a literal in C_i can take on either truth value and not change the truth of the clause. Thus, there is a sub-hypercube of dimension $n - k_i$, that is, a shifted, scaled instance of H_{n-k_i}, whose vertexes are not solutions for S. It turns out that there is an $(n-1)$-dimensional hyperplane which can be constructed so as to separate solutions from these non-solutions for C_i. This chopping hyperplane can be positioned anywhere along the edges connecting H_{n-k} to the rest of H_n. Figure 1 shows an example of chopping one vertex, [1,0,1], from H_3. The chop corresponds to finding the intersection of H_n with the set of points in the non-negative half-space defined by the hyperplane. Each conjunct provides a corresponding half-space, and the set of all half-spaces, along with H_n are intersected to produce the feasible region for solutions.

Once all the chops are made (i.e., the intersection of H_n with the non-negative half-spaces defined by the hyperplanes), the resulting convex set is called the *feasible region*. Note that it is necessary to include hyperplanes which define the faces of H_n in order to keep the feasible region bounded.

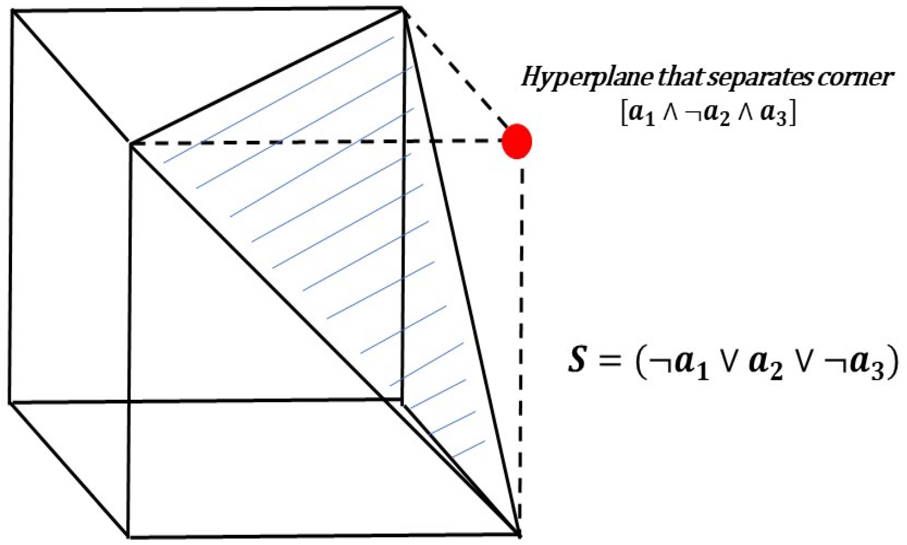
Hyperplane that separates corner
$[a_1 \wedge \neg a_2 \wedge a_3]$
$S = (\neg a_1 \vee a_2 \vee \neg a_3)$

Figure 1: Example of Chopping [1;0;1] from H_3. The chopping hyperplane goes through the neighbors of $[1, 0, 1]$.

Figure 2 shows a simple 2-D example of the feasible region resulting from chopping off each corner. The hyperplanes in this case are lines in 2-D. Each row provides the coefficients a, b, c of the standard form equation of a line: $ax + by + c = 0$:

```
-0.7071   -0.7071    1.0607
-0.7071    0.7071    0.3536
 0.7071   -0.7071    0.3536
 0.7071    0.7071   -0.3536
 1.0000         0         0
-1.0000         0    1.0000
      0    1.0000         0
      0   -1.0000    1.0000
```

where the first four are the corner cuts, and the last four define the faces (sides) of the square. The chops are made midway along the edges connecting the vertex to be removed and its neighbors.

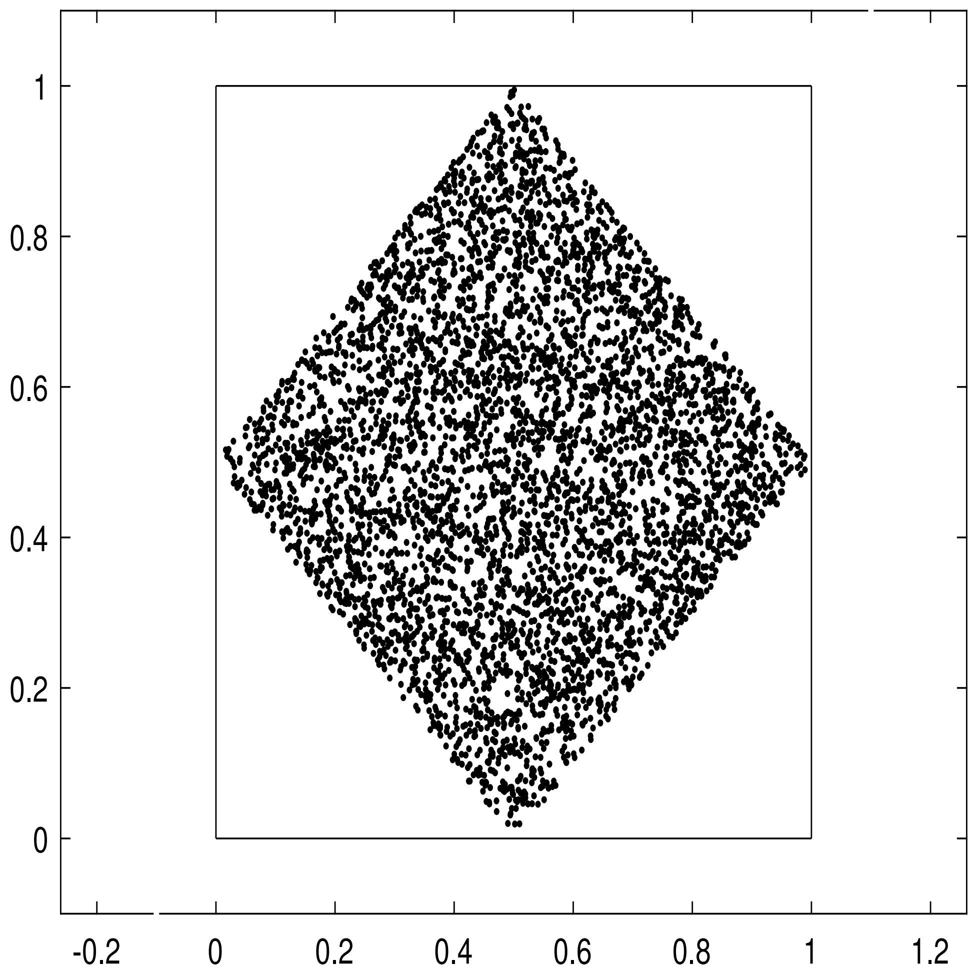

Figure 2: Example 2-D Feasible Region Resulting from Chopping all Corners o H_2.

3. Solving SAT

Given a feasible region, $\mathcal{F}$, corresponding to a CNF sentence (or knowledge base), it is possible to probe that region to determine if there is a SAT solution (i.e., a corner of H_n in $\mathcal{F}$). Note that the feasible region for any unsatisfiable sentence has no point farther than $\frac{\sqrt{n-2}}{2}$ from the center of H_n.

Our approach takes advantage of the fact that the Maximal Volume Inscribed Ellipsoid (MVE) of a full-dimensional convex polytope defined by a finite set of affine inequalities can be found in polynomial time [23], and the MVE will, in general, have its major semi-axes aligned with the most elongated parts of $\mathcal{F}$. Moreover, Prof. Zhang provided us with a Matlab function which produces an ellipsoid representation consisting of x and E, where x is the center of the ellipsoid and E is the matrix such that the points, $\mathcal{P}$, of the ellipsoid are defined as:

$$\mathcal{P} = \{x + Es | s \in \Re^n \ni \|s\| \leq 1\}$$

The Singular Value Decomposition of E yields the semi-major axes of the MVE. In order to increase the volume, it is possible to move the hypercube face constraints outward to allow a larger volume for the ellipsoid. Figure 3 demonstrates this idea in 2-D where there is just one solution ([0;0]), and the sides of the square have been moved out 10 units.

To test the practicality of this approach, the following experiment was performed:

- A set of 1000 random knowledge bases over 20 variables was generated.
- The corresponding feasible region was determined for each KB.
- The MVE was found for each feasible region.
- The major semi-axes were found.
- The feasible region boundary points along both directions of the major semi-axes were found.
- If any point was greater than $\frac{\sqrt{n-2}}{2}$ away from the center, then a solution is known to exist.

A solution was found in this way for every knowledge base. Note that this is only a heuristic and is not guaranteed to find a solution. If the Minimal Volume Circumscribing ellipsoid were found instead, then this would guarantee a solution, but this problem is NP hard [8].

To see how *Chop-SAT* relates to standard SAT solvers, consider DPLL. DPLL is a complete backtracking search algorithm which as its major step assigns a value to a variable, then determines if there is a solution (i.e., it reduces the conjuncts, assigns necessary values to resulting unit clause variables, and assigns values to pure variables with only one polarity across all conjuncts). *Chop-SAT* can be viewed as an alternative geometric approach

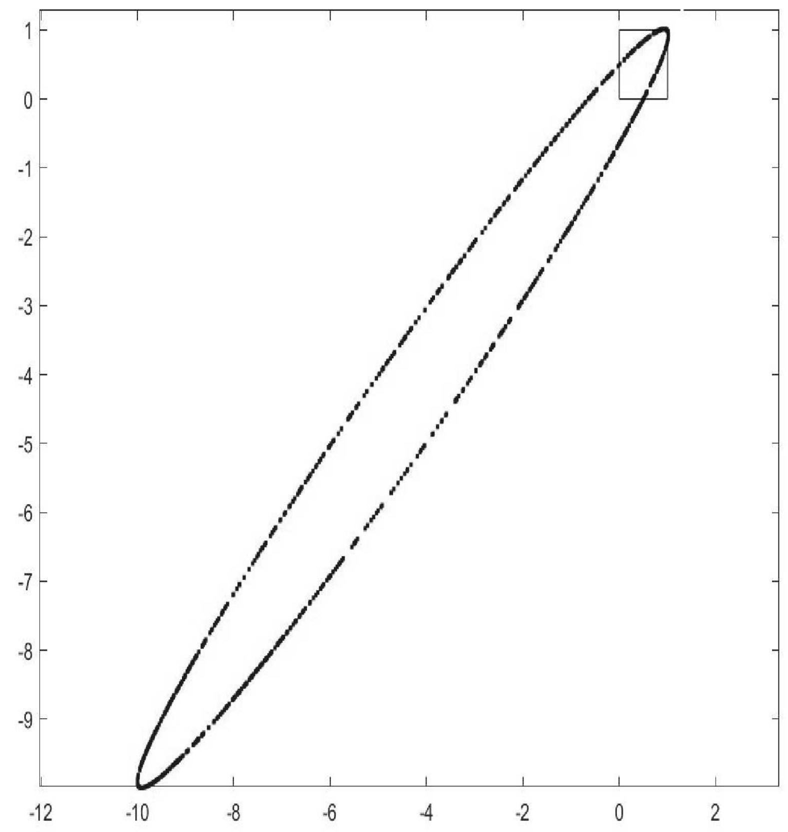

Figure 3: Example 2-D MVE Showing Major Semi-Axis in Direction of Solution.

to this solution determination step only with no need, in general, to be embedded in a complete search.

Chop-SAT represents the SAT problem as a convex feasible region where if this region has a corner from the original n-D hypercube, then a solution exists; if not, then the sentence is unsatisfiable. *Chop-SAT* works by using linear programming (linprog in Matlab) to find extremal vertexes by projecting the feasible region onto a selected vector. Let $\mathcal{F}$ be the feasible region of a CNF sentence; then linear programming finds$x \in \mathcal{F}$ which minimizes $f^T x$ such that $Ax \leq b$ and $lb \leq x \leq ub$, where $lb = 0$ and $ub = 1$.

For example, if $f = [1, 0, 0]^T$ then the feasible region is projected onto the

x-axis (in 3-D). Of course, *Chop-SAT* when embedded in a search algorithm like DDLL is also complete, but we use it as a one-step algorithm since this usually finds a solution. Note that we can also make *Chop-SAT* complete by projecting onto all the diagonal axes of the n-D hypercube, but our original hope was that geometry would provide a polynomial time solution. *Chop-SAT* can determine that a feasible region is from an unsatisfiable sentence because of the following property: For every feasible region arising from an unsatisfiable CNF sentence, there is no point at distance greater than $\frac{\sqrt{n-1}}{2}$ from the center of the hypercube; this means the feasible region can undergo any rotation about the center of the hypercube and all points of the feasible region remain within the bounds of the n-D hypercube -– this is not the case for feasible regions containing solutions.

4. Solving for Atom Probabilities

Another important problem for an agent is to determine the probability of the state of the world. For example, in the Wumpus World, the agent will die if it enters a cell with a pit. Here we describe a method to determine such probabilities using the feasible region arising from a CNF sentence knowledge base.

For any given satisfiable knowledge base, the atom probabilities are just the average of the 0/1 truth assignments of the models which make satisfy the CNF sentence as described in the introduction. However, since it is too computationally costly to determine all possible solutions and take their average, we propose the following approximations that can be found within a feasible region $\mathcal{F}$:

- the analytic center of $\mathcal{F}$.
- the p-center of $\mathcal{F}$.
- the center of the MVE.
- mean of samples from $\mathcal{F}$.

Analytic Center of $\mathcal{F}$

The *analytic center* of $\mathcal{F}$ is defined as:

$$c = max_y \prod_h (a_h \cdot \begin{bmatrix} y \\ 1 \end{bmatrix})$$

where $y \in \mathcal{F}$ and a_h is the hyperplane coefficient vector for hyperplane h.

P-Center of $\mathcal{F}$

The p-center of the feasible region is defined as follows [16]: (1) choose an initial point $x \in \mathcal{F}$; (2) for each hyperplane constraint find the line normal to the hyperplane which goes through x; (3) find the two most distant points in $\mathcal{F}$ on this line; (4) Average all the points found this way and assign the value to x; (5) stop if x does not change significantly.

Chop-SAT Mean Center of $\mathcal{F}$

In this approach, a number of samples are found in the feasible region and their average determined. For example, solve the following:

$$min_x f^T x$$

such that $x \in \mathcal{F}$ and f is taken as the positive and negative unit vector along each axis of the n-D space.

MVE Center of $\mathcal{F}$

The MVE provides both the center of the MVE as well as the directions of the semi-axes. Here, the center of the MVE is used to approximate the atom probabilities.

4.1. Experimental Study

For this study, 1000 random knowledge bases were generated with 5 atoms, a maximum of 10 clauses, and at most three literals in a clause. Figure 4 shows the Euclidean distance between the atom probability vector and the four proposed approximations: (1) the analytic center (has lowest error), (2) the p-center (next lowest error), (3) the Chop mean center (similar to p-center error), and (4) the MVE center (has the most error). The center errors over all trials was 0.25, 0.52, 0.57, and 1.07, respectively.

Figure 5 shows how the centers cluster. Since the vectors are all 5-dimensional, they are converted to a 2-dimensional representation which maintains spatial coherence. As can be seen, the analytic centers spread similarly to the actual atom probabilities, the MVE centers form a tight central cluster, the p-centers skew somewhat away (up) from the actual atom probabilities, while the Chop mean centers skew a good bit away from the

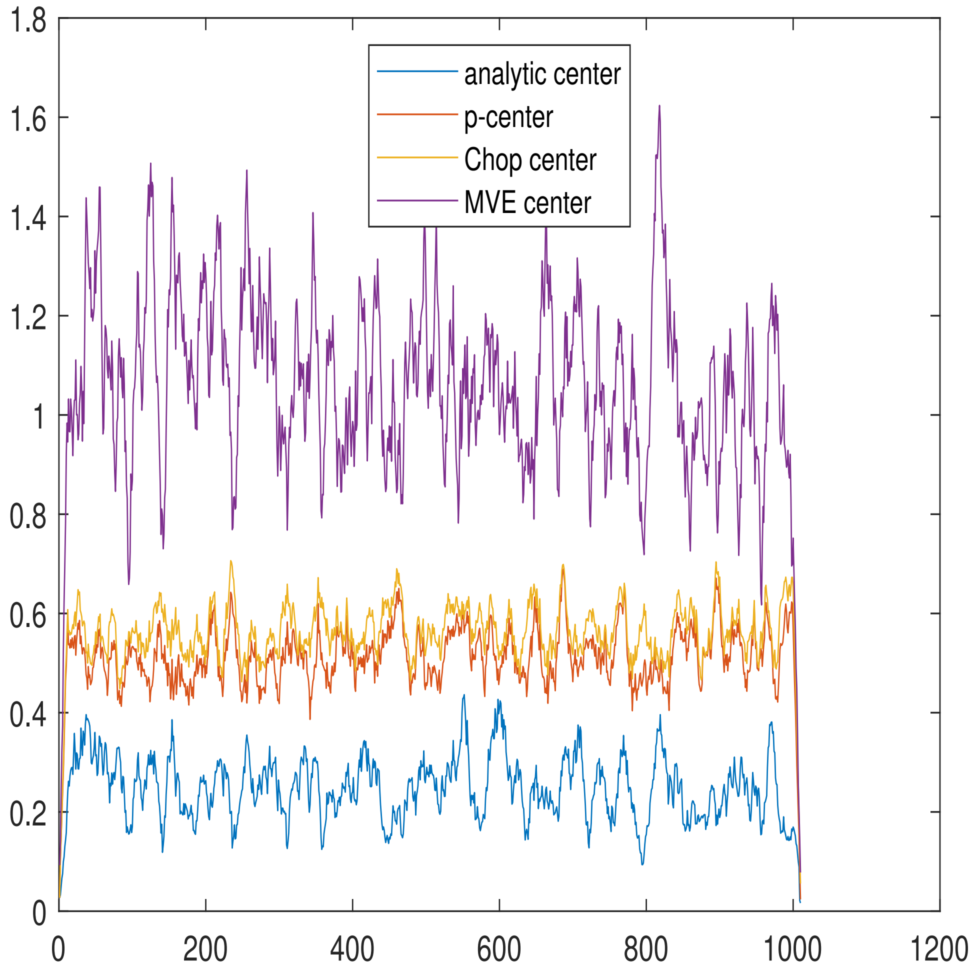

Figure 4: Error of Approximation Methods: Analytic Center (blue), P-Center (red), Chop center (mustard), MVE center (purple).

actual probabilities. The experiments indicate that the Analytic Center is the best approximation to the actual atom probabilities.

5. Wumpus World Experiment

To demonstrate the effectiveness of this approach, agent decision making was tested in the Wumpus World framework. Wumpus World was proposed by Yob [22], and has been used as a standard agent testbed for some time [19]. Wumpus world here is a 4x4 grid of cells; each cell may contain a pit

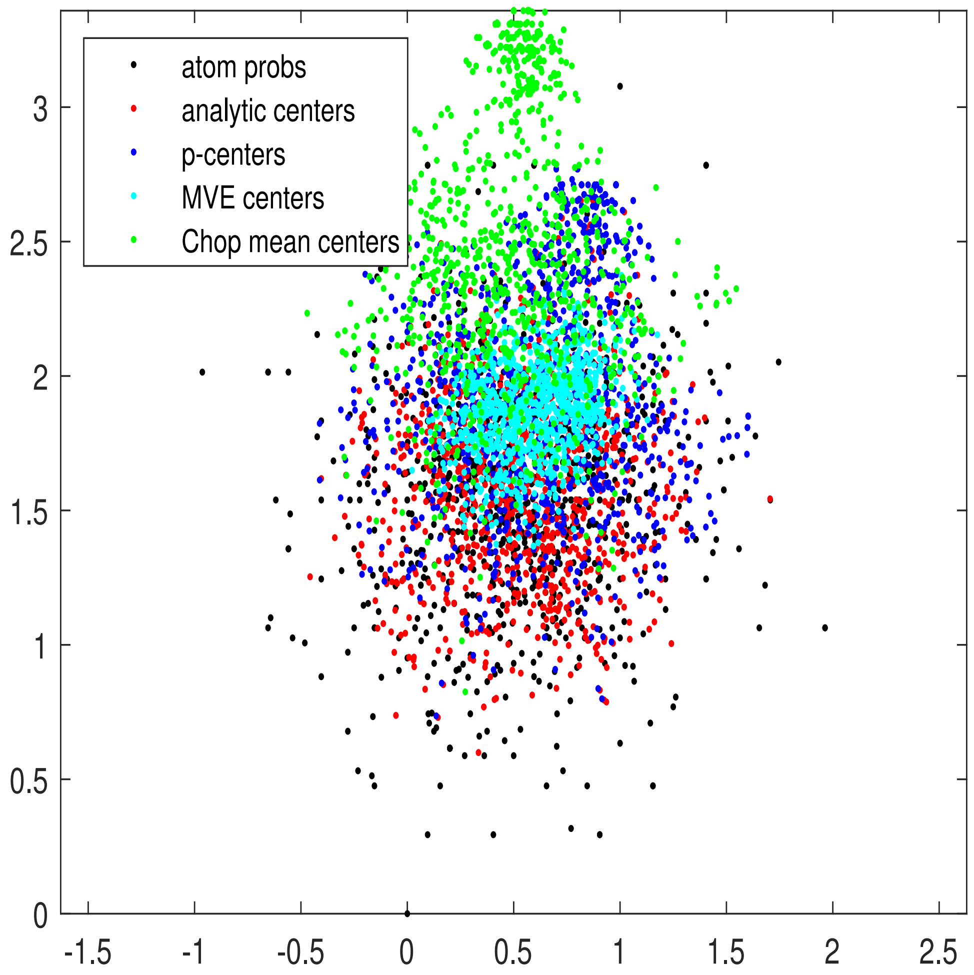

Figure 5: The center distributions.

with 20% probability, and if there is a pit it is the only thing in the cell. One cell contains some gold, and one cell contains a Wumpus (the gold may be co-located with the Wumpus). An agent starts in cell $(1, 1)$ and explores the grid in order to find the gold. Each cell neighboring a pit has a breeze, and each cell neighboring the Wumpus has a stench. Figure 6 shows an example board with the agent in cell $(1, 1)$ with direction $\theta = 0$, with pits in cells $\{(1, 4), (4, 2), (4, 3)\}$, the Wumpus in cell $(2, 3)$, and the gold in cell $(4, 4)$.

The agent has a set of percepts in a cell (represented as bits): (1) stench, (2) breeze, (3) glitter, (4) bump, and (5) scream. The glitter percept lets the

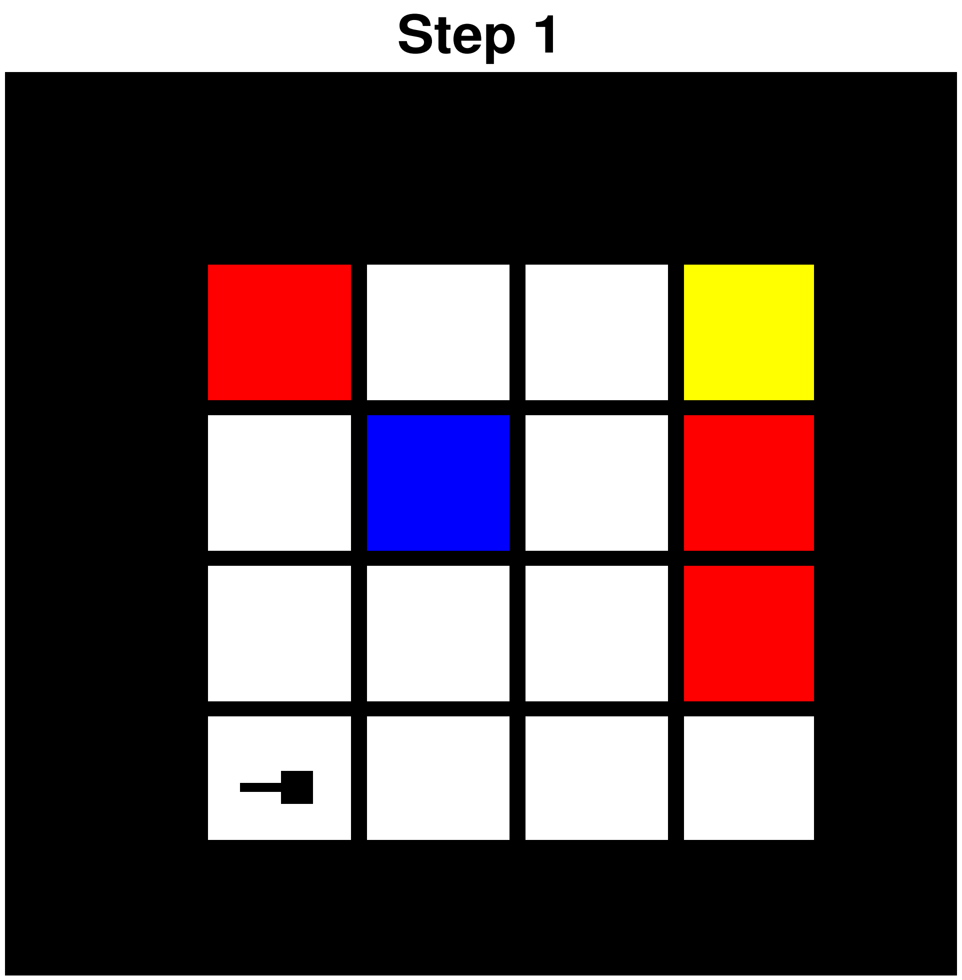

Figure 6: An Example Wumpus World Board.

agent know it's in the cell with the gold. The agent has a state (x, y, θ), where (x, y) is its location in the grid, and θ is its orientation $\theta \in \{0, 90, 180, 270\}$ degrees. The agent has a set of possible actions: (1) rotate left, (2) rotate right, (3) forward, (4) grab, (5) shoot, and (6) climb. If the agent moves into a cell with a pit or the Wumpus, it dies. The agent has one arrow and can shoot the Wumpus.

The agents here all share a Belief, Desire, Intention cognitive architecture [10]. The Belief-Desire-Intention architecture is a hierarchical organization of states and actions (grouped into plans) that was designed specifically for agent models. The architecture not only defines the conceptual structure of a program, but also a process structure that enables dynamic planning. Organizing the program in this way supports both reasoning by the autonomous agent as well as reasoning by human operators. The structure of desires, intentions, beliefs, and plans coincides well with the reasoning of the human operator. For example, a human observer could ask a BDI agent directly, "What is your current plan?" and the agent could respond "Located the Wumpus and pro ceding to kill it."

In the specific scenario here, the desires are: (1) escape, (2) kill the Wumpus, and (3) explore. If it has the gold, its intention will be to escape with the gold. The agent's intention will be to kill the Wumpus if its location is known. Otherwise, its intention is to explore the grid. In order to explore, the agent selects the lowest probability risk unvisited cell. All agents have the same BDI architecture and only differ in how they compute the probabilities for risk: (1) a human produced algorithm to assign pit and Wumpus probabilities, (2) the analytic center, (3) *Chop-SAT* mean, and (4) Monte Carlo simulation based on statistics over sample boards satisfying the known percept information.

The experimental method is to generate 1,000 random Wumpus boards, run each agent type on the boards, and measure successful escape with the gold. Note that the Monte Carlo results will be very close to optimal and serves as the upper bound on success. The results are given in Table 1. As can be seen, the *Chop-SAT* based agent performed the best, and this is a good indication of its efficacy.

Table 1. Results of Wumpus World Experiment.

Number of Boards	**Human**	**Analytic Center**	**Chop-SAT**	**Monte Carlo**
1000	585	598	608	613.3

These experiments from Wumpus world have 80 logical variables (i.e., 2^{80} models for the knowledge base!). The point is to see if the probabilistic method provides adequate variable probabilities to help make good (life preserving) decisions. The baseline of the experiment is given as the Monte Carlo method for determining these probabilities (i.e., random boards are generated that satisfy the knowledge acquired during a game and the exact variable probabilities are estimated from these board configurations). The overall statistic is that 1000 random Wumpus boards are generated, and each algorithm (human, analytic center, *Chop-SAT*) is compared to the ground truth approximated by the Monte Carlo method. Thus, there are three solvers (humans wrote programs to assign probabilities, the analytic center method provides probabilities, and *Chop-SAT* provides probabilities) which are then used to make decisions. Other probabilistic SAT solution methods were not used because they are all exponential complexity and either won't run on this size problem or require max entropy type assumptions which are not valid here.

A current application under study is the use *Chop-SAT* as part of a Belief-Desire-Intention architecture for autonomous Unmanned Aircraft Systems agents [20]. We are conducting experiments this summer at the US Air Force Academy where such an architecture for decision making is used.

6. Non-Euclidean Geometry Approach

The Euclidean method described above 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 projects H_n onto the n-dimensional unit hypersphere considered as an n-dimensional Poincaré Disk (see [14]). 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.

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 7). The corners of the n-dimensional 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

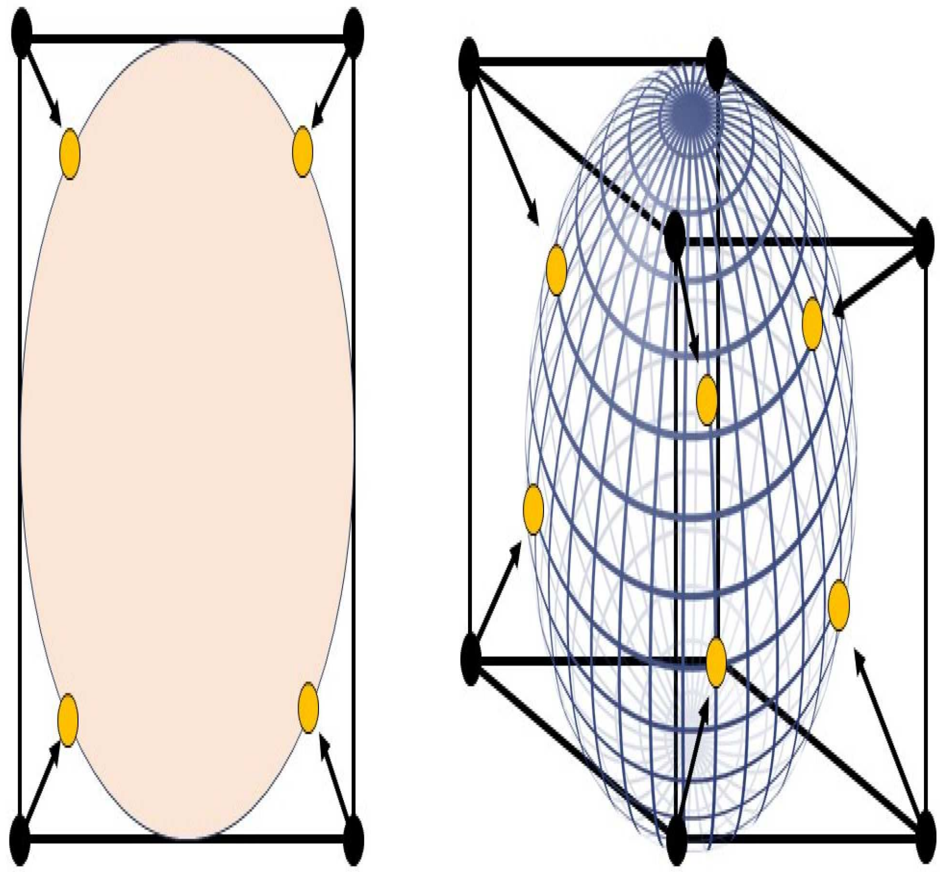

Figure 7: The Vertexes of the Hypercube are Projected onto the Hypersphere.

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.

6.1. Poincaré Disk Distance

Consider the 2-dimensional Poincaré Disk (Figure 8, upper left circle).

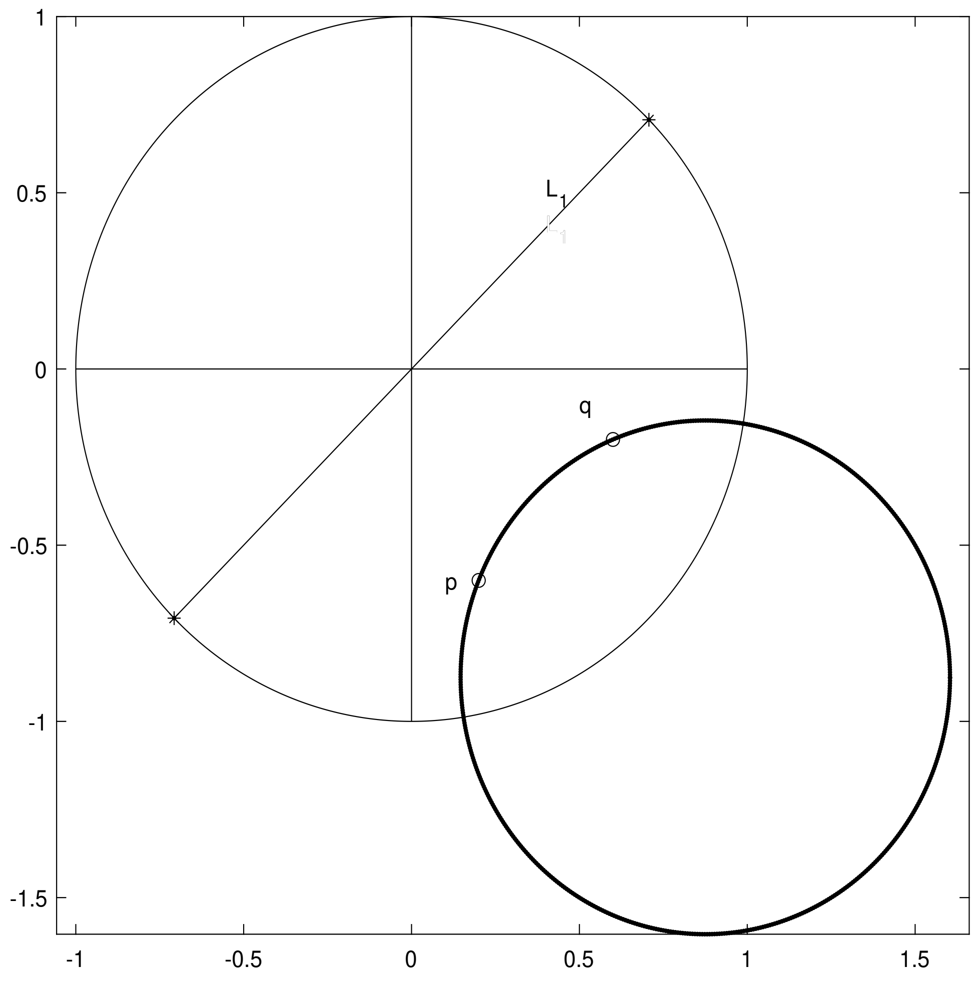

Figure 8: The 2D Poincaré Disk; L_1 is a line through the center, $\mathcal{O} = (0, 0)$, i.e., a diameter; L_2 is a circular arc which is orthogonal to the unit circle.

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 8, 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 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.

6.2. 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:

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

$$\Rightarrow p_x C_x + p_y C_y = \frac{p_x^2 + p_y^2 + 1}{2}$$

Likewise:

$$\Rightarrow q_x C_x + q_y C_y = \frac{q_x^2 + q_y^2 + 1}{2}$$

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:

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

6.3. 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 9. 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 10. The $2n$ $(n-1)$-dimensional bounding faces of the hypercube are represented in the Poincaré Disk in terms of hyperspheres through the 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:

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

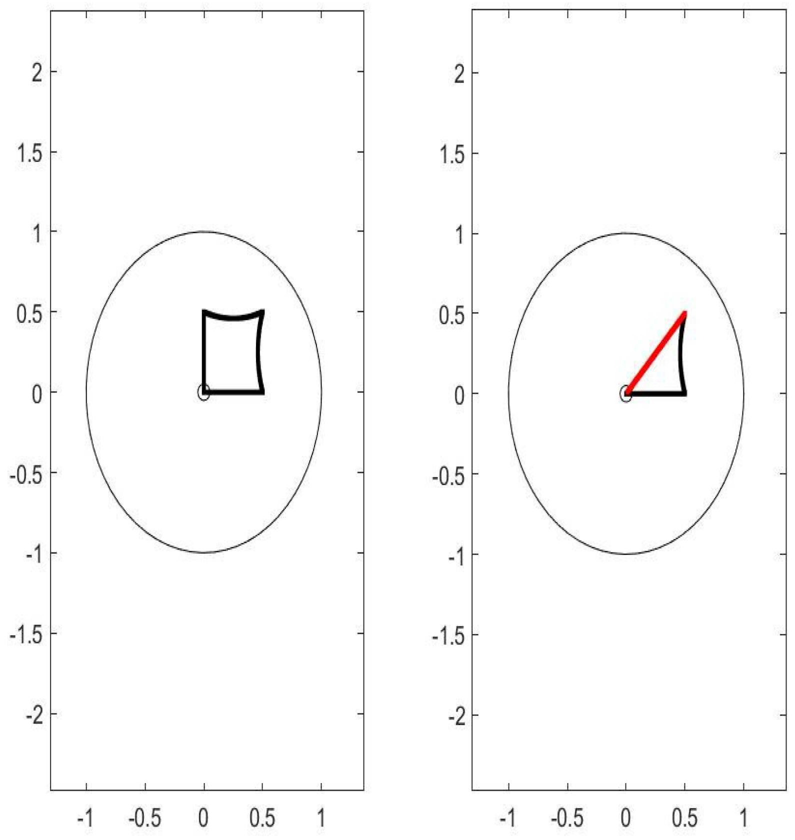

Figure 9: Representing the Complete Feasible Region as a Square in the Poincaré Disk. The left side is the complete feasible region, while the right side shows the feasible region when $(0, 1)$ is not a truth assignment that satisfies the CLF sentence.

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.

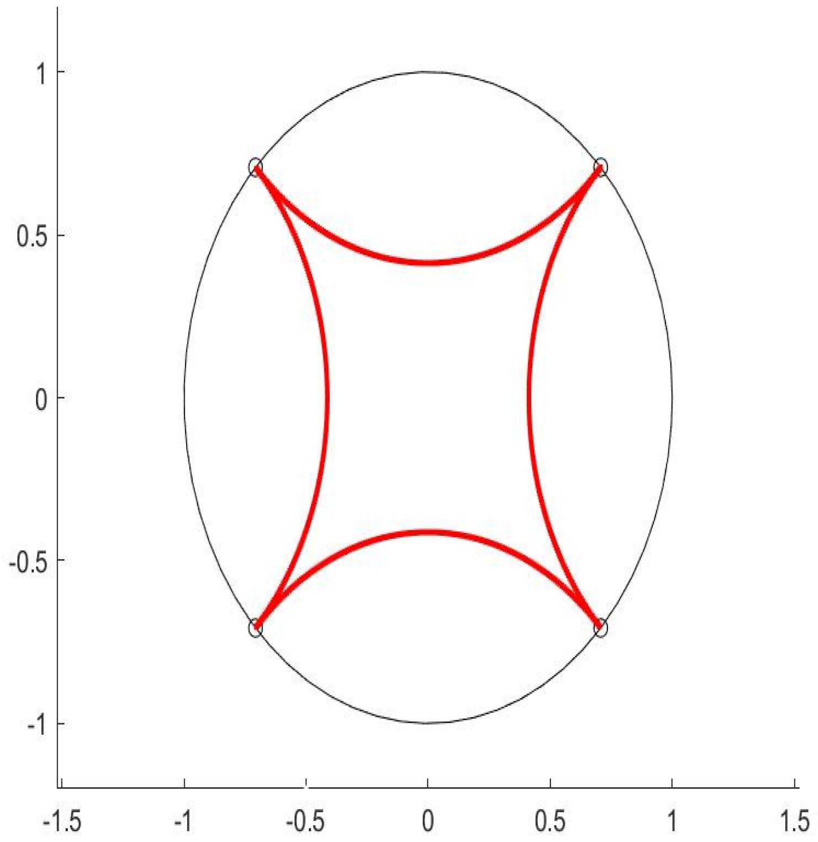

Figure 10: Representing the Complete Feasible Region as a Square with corners at infinity in the Poincaré Disk.

6.3.1. 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:

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

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:

$$v_i = -1 \text{ } \textit{if} \text{ } \exists j \ni \neg a_i = L_j$$

$$v_i = 0 \text{ } \textit{if no literal of } a_i \in C$$

$$v_i = 1 \text{ } \textit{if} \text{ } \exists j \ni a_i = L_j$$

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

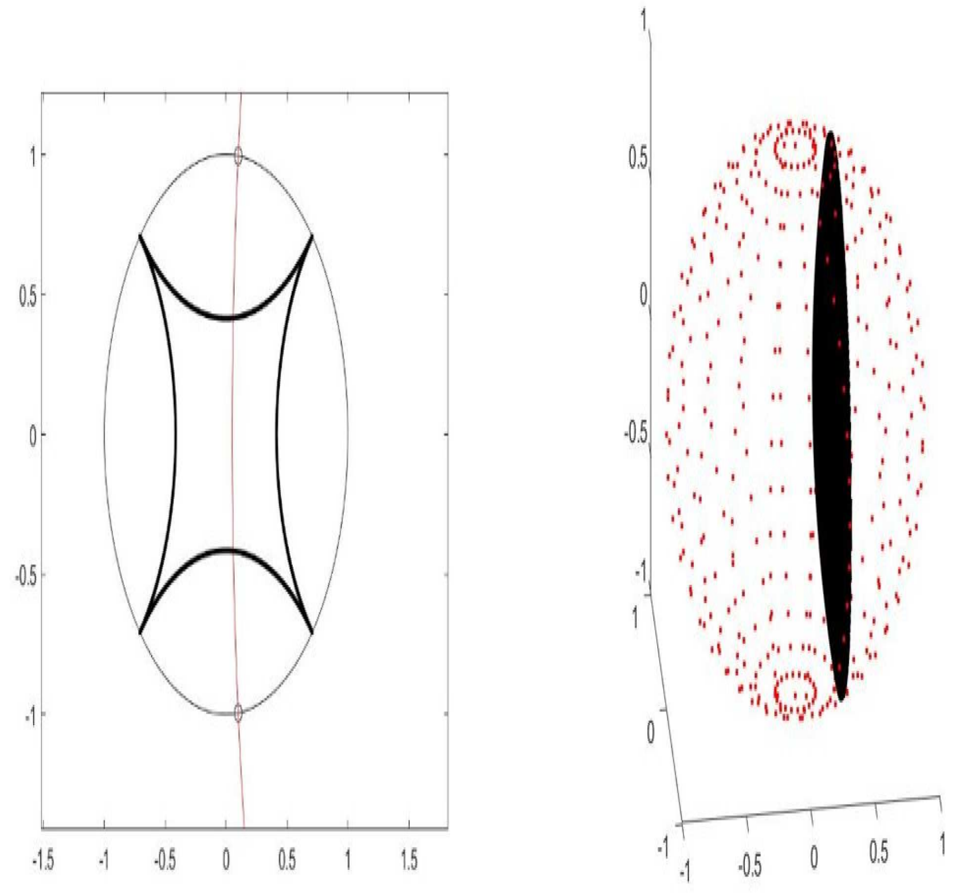

Figure 11: Examples of Chops in 2D and 3D.

other clause where it appears and then reducing these clauses accordingly. Figure 11 shows example 2D and 3D chops. Note that this formulation works in any dimension.

The Δ parameter determines how far the chop occurs from the non-solution vertexes which are being cut. Figure 12 shows some example positions for chops. Figure 13 shows the chops for Modus Ponens, Figure 14 shows three edges chopped, and Figure 15 shows a 3D face chop.

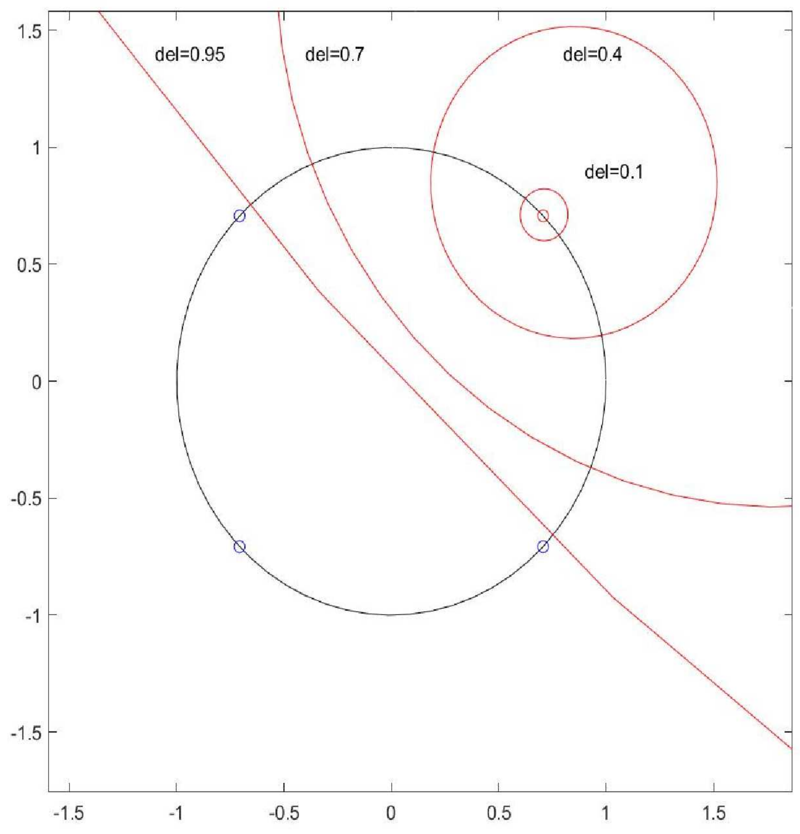

Figure 12: The Impact of *Delta* on Chop Distance from Non-Solution Vertexes.

6.4. 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 ver-

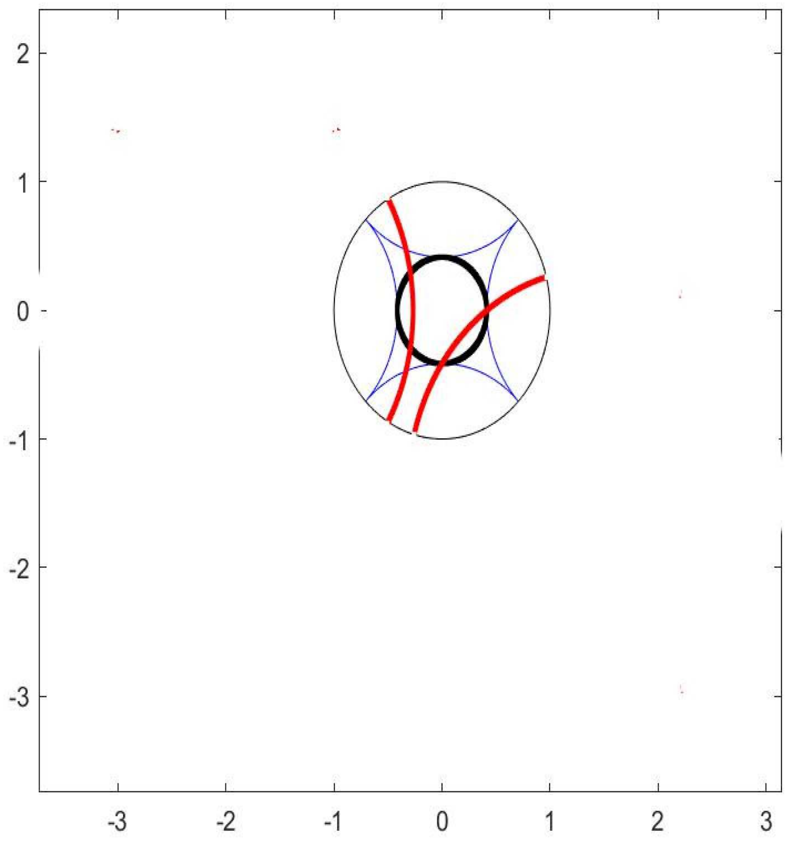

Figure 13: Chops for Modus Ponens.

texes, 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.

- 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.

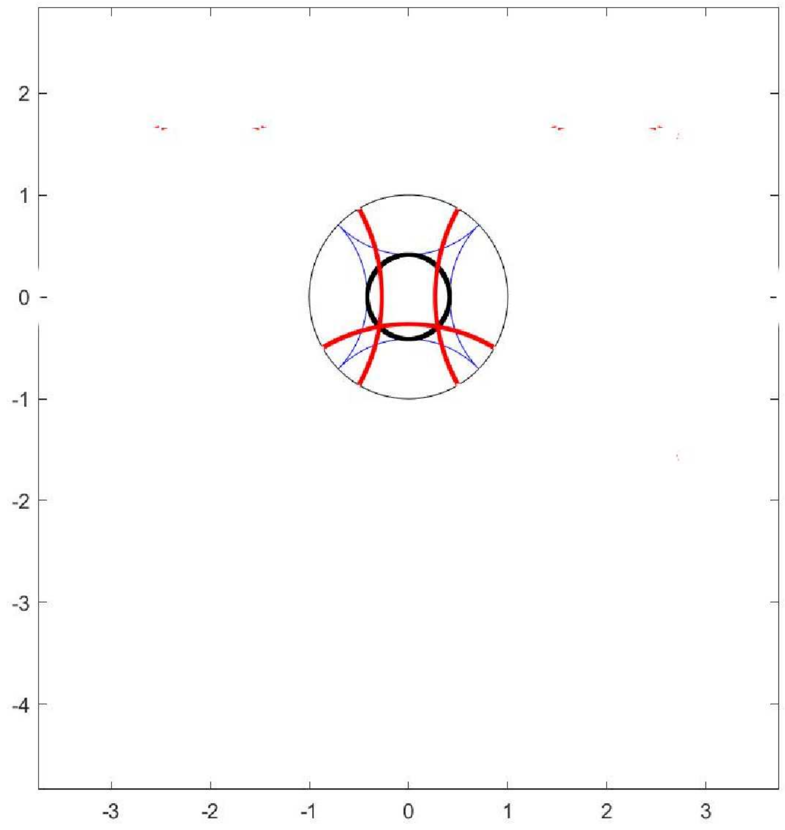

Figure 14: Three Edges Chop.

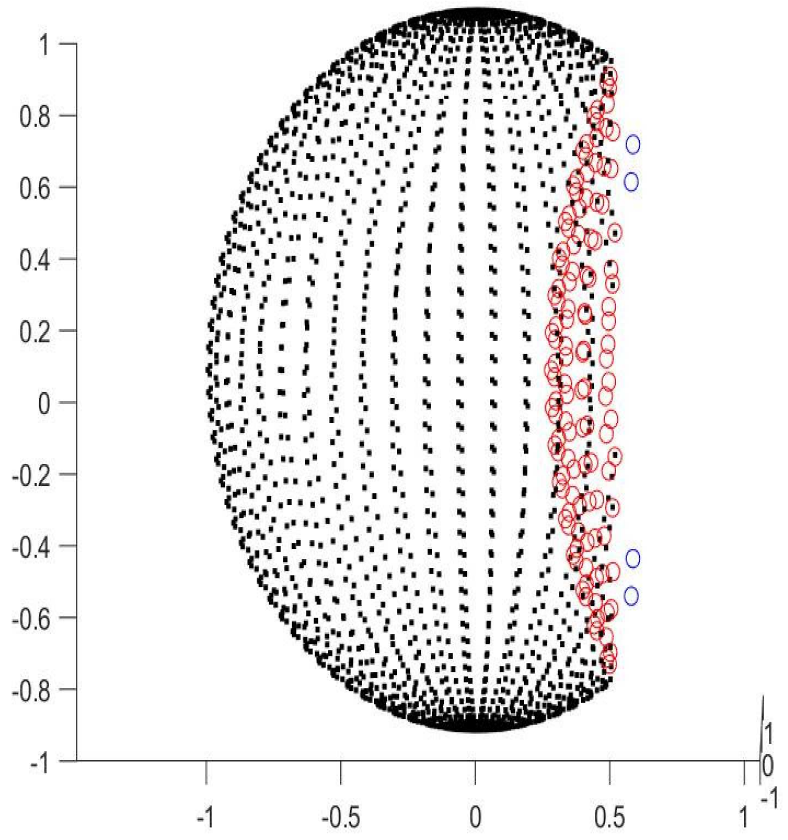

Figure 15: 3D Face Chop.

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.

6.4.1. 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:

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

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:

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

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

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

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 16 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 17 shows the Barrier Method applied to finding solutions for a 2D problem with three solutions. A 3-D example path is shown in Figure 18.

7. Conclusions and Future Work

The results of the experiments indicate that this geometric approach works well at least up to dimension 20, and that the analytic center is the best approximation to the actual atom probabilities on a random set of knowledge bases. However, the probabilities provided by the *Chop-SAT* method performed slightly better in the decision making experiment in Wumpus World, and both the analytic center and *Chop-SAT* methods performed better than a human developed risk probability algorithm. Moreover, the non-Euclidean geometry also allows the determination of solutions, although since the barrier method is applied to a non-convex set, it is not guaranteed to find a vertex solution.

In the future, work will be performed in order to:

- test the SAT solver experiments in higher dimensions,
- seek effective ways to exploit the non-Euclidean geometry representation; e.g., view the feasible region as a domain to be explored by a geo-bot which has a set of range sensors to help it detect feasible points far from the Poincaré disk origin.
- use the atom probability approximation in more complex agent decision making situations (e.g., UAS traffic management) and compare its effectiveness with other methods.
- use the method as the basis for decision-making in a set of autonomous Unmanned Aircraf Systems (UAS) agents,

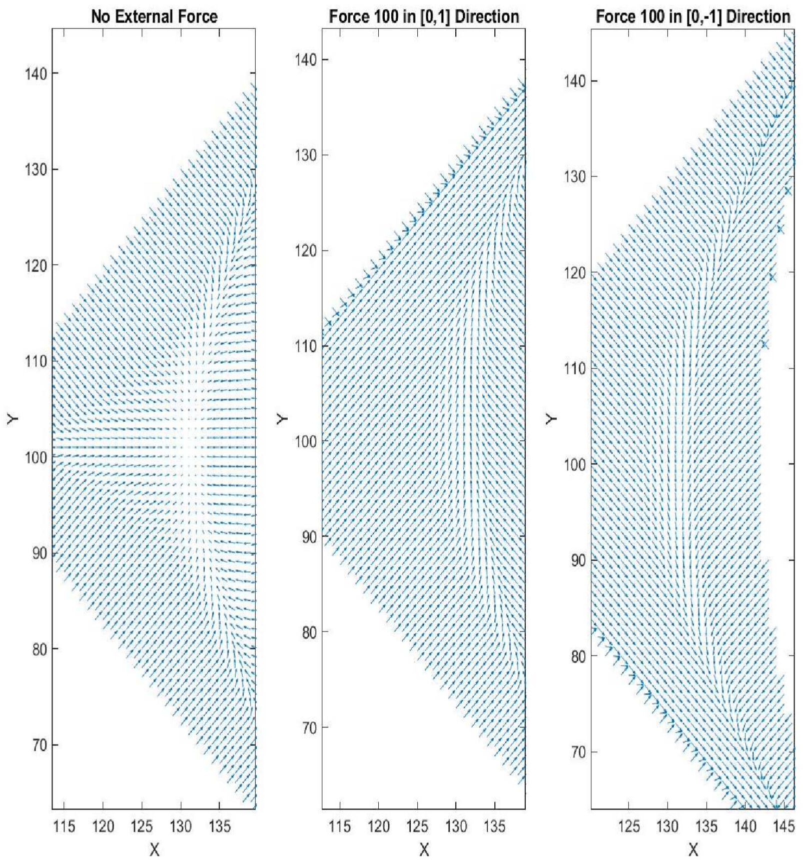

Figure 16: Example Force Fields: Only constraint surface repulsion forces are present (left); An added upward force (middle); A downward external force (right).

- explore formal verification of the BDI method.

Acknowledgements

This work was supported in part by National Science Foundation award 2152454.

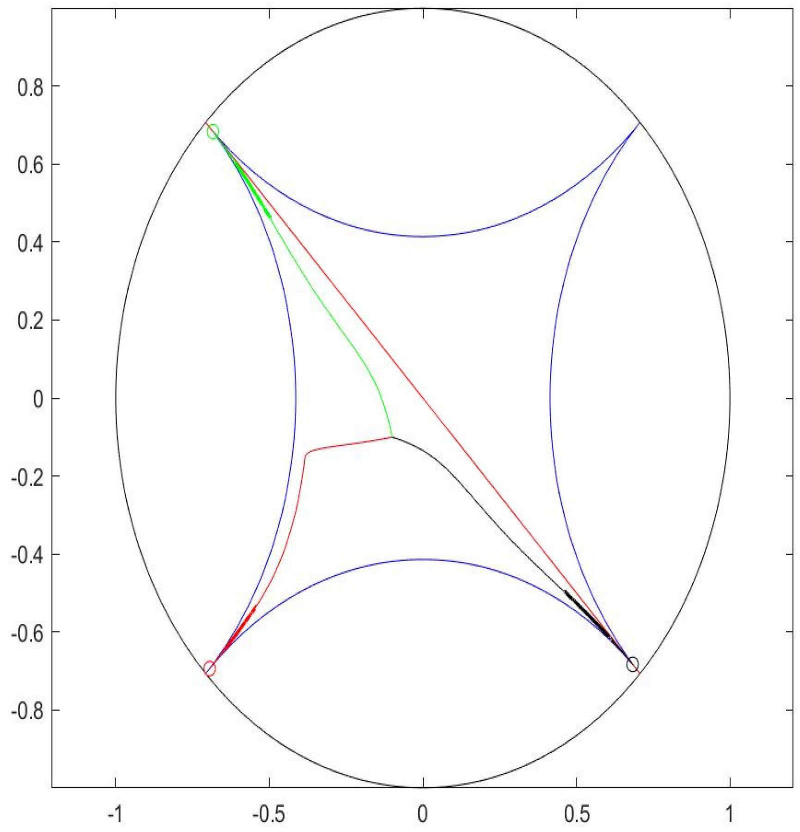

Figure 17: The Paths for the Barrier Method finding Solutions in a 2D Problem.

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.

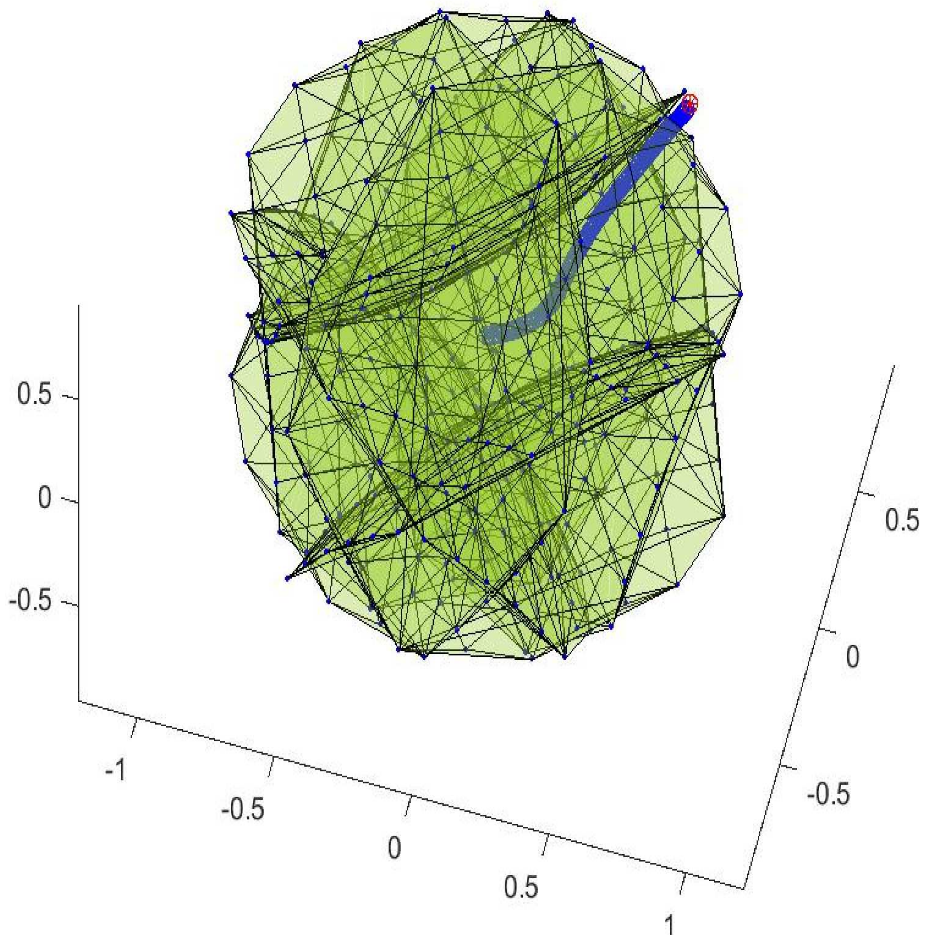

Figure 18: The Paths for the Barrier Method finding a Solution in a 3D Problem.

[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] P. Domingos and D. Lowd. *Markov Logic Networks.* Morgan and Claypool Publishers, Williston, VT, 2009.

[8] R.M. Freund and J.B. Orlin. On the Complexity of Four Polynomial Containment Problems. *Mathematical Programming*, 33:139–145, 1985.

[9] G. Georgakopoulos, D. Kavvadias, and C.H. Papadimitriou. Probabilistic Satisfiability. *Journal of Complexity*, 4:1–11, 1988.

[10] M. Georgeff, B. Pell, M. Pollack, M. Tambe, and M. Wooldridge. The Belief-Desire-Intention Model of Agency. In Jörg P. Müller, Anand S. Rao, and Munindar P. Singh, editors, *Intelligent Agents V: Agents Theories, Architectures, and Languages*, pages 1–10, Berlin, Heidelberg, 1999. Springer Berlin Heidelberg.

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

[12] T.C. Henderson, R. Simmons, B. Serbinowski, M. Cline, D. Sacharny, X. Fan, and A. Mitiche. Probabilistic Sentence Satisfiability: An Approach to PSAT. *Artificial Intelligence*, 278:71–87, 2020.

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

[14] Thomas C. Henderson, David Sacharny, Xiuyi Fan, Amar Mitiche, and Thatcher Geary. GEO-SAT: A Geometric Approach to Satisfiability. Technical Report UUCS-23-003, University of Utah, November 2023.

[15] 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.

[16] A.C. Moretti. A Weighted Projection Centering Method. *Computational and Applied Mathematics*, 22(1):19–36, 2003.

[17] N. Nilsson. Probabilistic Logic. *Artificial Intelligence Journal*, 28:71–87, 1986.

[18] J. Pearl. *Probabilistic Reasoning in Intelligent Systems.* Morgan Kaufmann, San Mateo, CA, 1988.

[19] Stuart Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach.* Prentice Hall Press, Upper Saddle River, NJ, 3rd edition, 2009.

[20] D. Sacharny, Thomas C. Henderson, Michael Cline, and Benjamin Russon. Reinforcement Learning at the Cognitive Level in a Belief, Desire, Intention UAS Agent. In *Intelligent Autonomous Systems Confernce*, Singapore, June 2021.

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

[22] G. Yob. Hunt the Wumpus? *Creative Computing*, September–October 1975.

[23] Y. Zhang. On Numerical Solution of the Maximum Volume Ellipsoid Problem. *SIAM Journal on Optimization*, 14(1), 2003.