Singleton Elimination: A Geometric Approach to Boolean Satisfiability

Ishaan Rajan
University of Utah

UUCS-23-001

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

12 April 2023

Abstract

This study investigates methods of reducing the complexity of solving the Boolean satisfiability problem (SAT) from the geometrical representation of SAT as an n-D hypercube H_n. Recently, a geometric representation of solving SAT called CHOP-SAT was proposed, in which a convex polytope feasible region F is produced by separating (chopping) unsatisfiable models from H_n with a hyperplane. Such a feasible region represents the solution space for a knowledge base (KB) in conjunctive normal form (CNF). The goal of this study is to identify and eliminate singleton points in the feasible region and explore the effectiveness of reducing the possible solution space. Volume of F, inscribed and circum- scribed ellipsoids in and about F, containment of F in a no-solution polytope, and detection of singleton points in a sub-hypercube $H_k \subset H_n$ used to determine the efficiency of singleton elimination in distinguishing satisfiable and unsatisfiable logic sentences. The results suggest that singleton elimination and maximum volume inscribed ellipsoids are efficient methods for determining the satisfiability of KBs.

SINGLETON ELIMINATION: A GEOMETRIC APPROACH TO BOOLEAN SATISFIABILITY

by

Ishaan Rajan

A Senior Thesis submitted to the faculty of
The University of Utah
in partial fulfillment of the requirements for the degree of

Bachelor of Science

Department of Computer Science

The University of Utah

Thomas Henderson Thesis Faculty Supervisor	Mary Hall Director, School of Computing
H. James de St. Germain Director of Undergraduate Studies	

ABSTRACT

This study investigates methods of reducing the complexity of solving the Boolean satisfiability problem (SAT) from the geometrical representation of SAT as an n-D hypercube H_n. Recently, a geometric representation of solving SAT called *CHOP-SAT* was proposed, in which a convex polytope feasible region $\mathcal{F}$ is produced by separating (chopping) unsatisfiable models from H_n with a hyperplane [10]. Such a feasible region represents the solution space for a knowledge base (KB) in conjunctive normal form (CNF). The goal of this study is to identify and eliminate singleton points in the feasible region and explore the effectiveness of reducing the possible solution space. Volume of $\mathcal{F}$, inscribed and circumscribed ellipsoids in and about $\mathcal{F}$, containment of $\mathcal{F}$ in a no-solution polytope, and detection of singleton points in a sub-hypercube $H_k \subset H_n$ are used to determine the efficiency of singleton elimination in distinguishing satisfiable and unsatisfiable logic sentences. The results suggest that singleton elimination and maximum volume inscribed ellipsoids are efficient methods for determining the satisfiability of KBs.

For family, friends, and faculty that have supported me along my way.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

Given a Boolean logic formula (or logical sentence from propositional calculus), the *Boolean Satisfiability Problem (SAT)* is the method of determining whether there exists an assignment of truth values to the logical variables such that the formula evaluates to true. Recently, a geometric approach called *CHOP-SAT*[10] was proposed for solving SAT. This method produces a convex polytope feasible region whose properties may expose solutions to SAT or PSAT. This thesis explores the efficacy of such a geometric interpretation of SAT, with a focus on dimensionality reduction.

The standard representation for a Boolean formula to facilitate analysis is conjunctive normal form (CNF). A CNF sentence is a conjunction of disjunctions of literals, where each conjunct is called a clause. In the geometric view, a clause allows the removal of a subset of models that do not satisfy the sentence. This is realized as the intersection of the corresponding half space with H_n. A feasible region $\mathcal{F}$ is the intersection of H_n with all half spaces associated with the clauses. The feasible region is a convex polytope contained within H_n.

A singleton point is defined as a point in the feasible region that is not a corner of H_n and is the only point in some sub-hypercube $H_k \subset H_n$. Given such a geometric representation of SAT, the goal is to identify singleton points in some H_k to qualify the removal of that H_k from the feasible region. This results in a lower volume feasible region.

Singleton detection is achieved using linear programming to solve:

1. The identification and validation of a singleton point candidate

$$\min f'x \text{ subject to } Ax < b$$

where x is a point in the feasible region, f is an n-tuple, and the system $Ax < b$ defines the half-space cuts. If the resulting point x contains all {0,1} elements, then x is a solution to the CNF and not a singleton point. If x contains all fractional elements outside an error ϵ, more formally $\{x \in \mathbb{R}^n \mid 0 + \epsilon < x_i < 1 - \epsilon\}$, then the sentence is unsatisfiable. When the resulting point x has a mixture of 0, 1, and fractional elements, it is a singleton point candidate.

2. The existence of a sub-hypercube $H_k \subset H_n \mid k > 0$ containing the singleton point that can be removed.

The SAT problem is of significant importance in the field of theoretical computer science and artificial intelligence as it is the first problem proven to be NP-complete[4]. This means that no efficient algorithm is known for solving SAT in all cases, and it is likely that none exists. However, SAT can be solved in specific cases using specialized algorithms such as the DPLL algorithm and its variants. Understanding the complexity of SAT and the limits of efficient algorithms is important for understanding the limits of computation in general. Additionally, SAT has practical applications: SAT solvers are widely used in many fields, such as software verification, circuit design, and automated theorem proving. SAT solvers are used to solve real-world problems in various fields, including scheduling, planning, and resource allocation. For example, a SAT solver can be used to find a schedule that satisfies a set of constraints, such as the availability of resources and the precedence relations between tasks. Many other problems in computer science can be reduced to SAT, meaning that solving SAT efficiently can be a useful step in solving other problems efficiently. One such problem is *Probabilistic Boolean Satisfiability (PSAT)*, an extension of the SAT problem that takes into account the probability of its variables. PSAT has a wide variaty of applications in probabilistic reasoning, probabilistic planning, probabilistic inference, and probabilistic learning.

This thesis explores how the identification and elimination of singleton points contributes to the efficiency and effectiveness of SAT solving. At a high level, singleton elimination supports the search for a solution as follows:

- First, a knowledge base (KB) is defined as a CNF sentence, where each conjunct is assigned a probability. In the case of solving SAT (as opposed to PSAT), each conjuct will have a probability of 1.
- Next, a hyperplane representation of the KB is used to produce a feasible region determined by the intersection of the non-negative half-spaces of the hyperplanes. Such a feasible region is a convex polytope contained in H_n.
- Next, sub-hypercubes containing singleton points are removed and the number of iterations to find a solution is observed. The volume of the feasible region in each iteration is also monitored.
- The utility of maximum volume inscribed ellipsoids within the feasible region and minimum volume circumscribed ellipsoids about the feasible region are explored in determining where a solution may be found.

This approach is based on the fact that the feasible region represents the solution space for the KB. If any of the original hypercube corners remain present in the feasible region, the knowledge base is satisfiable, otherwise it is unsatisfiable. The following chapters of this thesis will provide a comprehensive explanation and critical evaluation of this method and its effectiveness.

CHAPTER 2

BACKGROUND AND RELATED WORK

2.1 Boolean Satisfiabilty

The SAT problem poses the following question: given a Boolean logic formula, does there exist an assignment of truth values to its variables such that the formula evaluates to true? A Boolean logic formula ϕ is composed $\boldsymbol{n}$ variables $\{x_i\}_{i=1}^{n}$ that take on values $\{\text{false, true}\}$ and is defined over the operators $\{\wedge, \vee, \neg\}$. An assignment of $\{\text{false, true}\}$ to all the x_i is called a model or complete conjunction. A Boolean formula is said to be satisfiable if there exists at least one model that makes the formula true, and unsatisfiable if no such model exists. The Boolean satisfiability problem was proved to be NP-complete by Stephen Cook in 1971[4]. This essentially proved that any claimed satisfiable assignment of Boolean values can be verified in polynomial time by a deterministic Turing machine *and* every NP problem can be reduced to an instance of a SAT problem in polynomial time. NP-completeness can be defined as a complexity class of decision problems for which certificates (solutions) can be checked for correctness, by an algorithm whose run time is polynomial in the size of the input. No other NP problem is more than a polynomial factor harder than another NP problem. A significant outcome of this theorem is that if there exists a deterministic polynomial-time algorithm for solving the Boolean satisfiability problem, then every NP problem can be solved by a deterministic polynomial-time algorithm.

A *SAT solver* is an algorithm for establishing satisfiability. A solver takes a Boolean formula f in CNF as input and returns *SAT* if it finds an assignment of variables that can satisfy f or *UNSAT* if it can demonstrate that no such assignment exists. Most modern SAT solvers use some form of the Davis-Putnam-Logemann-Loveland (DPLL) algorithm as a basis of execution. At its core, the DPLL algorithm is a method that combines both search and deduction to determine satisfiability of CNF formulae.

The DPLL algorithm is outlined below:

1. If possible, “choose” a variable from the input formula that has not already been chosen
2. Find all unit clauses created from the previous assignment and assign the needed value
3. Iteratively do step 2 until there is no change – in other words, found a transitive closure
4. If the current assignment cannot yield true for all clauses, backtrack and retry a different assignment
5. Go to 1

The DPLL algorithm terminates when either:

1. The algorithm is unable to backtrack and change a variable assignment; return no solution
2. All clauses are satisfied and a solution is found; return

Operation	Assignment	Formula
		$(x_1 \vee x_2 \vee \neg x_3) \wedge (x_2 \vee x_4) \wedge (\neg x_1 \vee \neg x_2) \wedge (\neg x_1 \vee \neg x_3 \vee \neg x_4), 1$
choose x_1	x_1	$(x_1 \vee x_2 \vee \neg x_3) \wedge (x_2 \vee x_4) \wedge (\neg x_1 \vee \neg x_2) \wedge (\neg x_1 \vee \neg x_3 \vee \neg x_4), 1$
choose $\neg x_2$	$x_1, \neg x_2$	$(x_1 \vee x_2 \vee \neg x_3) \wedge (x_2 \vee x_4) \wedge (\neg x_1 \vee \neg x_2) \wedge (\neg x_1 \vee \neg x_3 \vee \neg x_4), 1$
choose x_3	$x_1, \neg x_2, x_3$	$(x_1 \vee x_2 \vee \neg x_3) \wedge (x_2 \vee x_4) \wedge (\neg x_1 \vee \neg x_2) \wedge (\neg x_1 \vee \neg x_3 \vee \neg x_4), 1$
choose x_4	$x_1, \neg x_2, x_3, x_4$	$(x_1 \vee x_2 \vee \neg x_3) \wedge (x_2 \vee x_4) \wedge (\neg x_1 \vee \neg x_2) \wedge (\neg x_1 \vee \neg x_3 \vee \neg x_4), 0$
		Inconsistency Found
backtrack x_3	$x_1, \neg x_2$	$(x_1 \vee x_2 \vee \neg x_3) \wedge (x_2 \vee x_4) \wedge (\neg x_1 \vee \neg x_2) \wedge (\neg x_1 \vee \neg x_3 \vee \neg x_4), 1$
choose $\neg x_3$	$x_1, \neg x_2, \neg x_3$	$(x_1 \vee x_2 \vee \neg x_3) \wedge (x_2 \vee x_4) \wedge (\neg x_1 \vee \neg x_2) \wedge (\neg x_1 \vee \neg x_3 \vee \neg x_4), 1$
		Solution found

Figure 2.1: DPLL algorithm execution on an example Boolean formula input.

2.2 Probabilistic SAT

Probabilistic Boolean Satisfiability (PSAT) is a variation of SAT in which the input is a Boolean logic formula in CNF with a probability associated with each conjunct. The goal here is to find a p_i, a probability distribution over the models such that the probability of each conjunct C_i is equal to the sum of the probabilities of the models that satisfy the conjunct. Much like the SAT problem, PSAT is NP-hard, meaning that it is computationally difficult to solve exactly in all cases, but can be approached using approximate methods such as Monte Carlo sampling, Markov Chain Monte Carlo, and Belief Propagation.

Given any point $p \in H_n$ (the n-dimensional hypercube), p can be considered as a probability assignment for the atoms. PSAT is defined as follows: given a logical sentence in CNF and a p_i associated with each conjunct C_i, find a function $\pi : \Omega \longrightarrow [0,1]$, where Ω is the set of all complete conjunctive sentences such that $0 \leq \pi(\omega_i) \leq 1$ and $p_i = \sum_{\omega_i \models C_i} \pi(w_i)$. A matrix A represents solutions to clauses in the input KB, meaning that the i^{th} row of A shows which complete conjunctions make the i^{th} clause in the KB true. The probability of the i^{th} clause is the sum of the probabilities of the models that make it true, which corresponds to multiplying A by the column of model probabilities. Consider the following example:

$$\text{Let } \pi : \Omega \longrightarrow [0,1] \text{ be a valid probability distribution, that is}$$

$$0 \leq \pi_i \leq 1 \text{ and } \sum_{i=1}^{2^n} \pi_i = 1$$

$$\text{KB } = P \vee Q \text{ [0.8]}$$

$$\text{Given } A\pi = p \longrightarrow [0\ 1\ 1\ 1] \begin{bmatrix} P(\omega 1) \\ P(\omega 2) \\ P(\omega 3) \\ P(\omega 4) \end{bmatrix} = 0.8$$

In this example, the KB contains only one clause, so matrix A contains only one row. Models 2, 3 and 4 make $P \vee Q$ true, so row elements 2, 3 and 4 of A are set to 1. Some solutions include:

$$\pi = [0\ .8\ 0\ 0]^T$$
$$\pi = [0\ .4\ .4\ 0]^T$$
$$\pi = [0\ .2\ .2\ .4]^T$$

Consider Modus Ponens in the context of PSAT where logical variable independence is assumed:

$$\begin{aligned}
&C_1 : P\ [p_1 = 0.7] \\
&C_2 : \neg P \vee Q\ [p_2 = 0.7] \\
0.7 &= \Pr(P) \\
0.7 &= \Pr(\neg P \vee Q) = \Pr(\neg P) + \Pr(Q) - \Pr(\neg P)\Pr(Q) \\
&= (1 - \Pr(P)) + \Pr(Q) - (1 - \Pr(P))\Pr(Q) \\
&= 0.3 + \Pr(Q) - 0.3\Pr(Q) \\
&\longrightarrow \Pr(Q) = (0.7 - 0.3)/(0.7) = 0.571
\end{aligned}$$

Note that this Modus Ponens example does not require search and is *not exponential complexity*.

The geometrical view of Probabilistic Boolean Satisfiability (PSAT) provides a mathematical framework for understanding the problem by mapping it to a geometric space. Consider a PSAT formula with n variables and m clauses. Let X = $\{x_1, x_2, ..., x_n\}$ be the set of variables and let F = $\{C_1, C_2, ..., C_m\}$ be the set of clauses in the PSAT formula. Each clause C_i is assigned a probability value $p_i \in [0, 1]$; from these, we would like to determine a set of reasonable probabilities for the atoms. The set of all possible probability assignments for the variables can be represented as an n-dimensional unit hypercube, denoted by $H_n = \{p \in \mathbb{R}^n : 0 \leq p_i \leq 1, i = 1, 2, ..., n\}$. Each clause C_j in the CNF formula defines a $n - 1$ dimensional hyperplane which separates solutions of the clause from non-solutions.

A half-space can be mathematically represented as a set of points lying on one side of the hyperplane, i.e., a linear inequality of the form:

$$a_1 p_1 + a_2 p_2 + ... + a_n p_n \geq b$$

where $a_i \in \mathbb{R}, i = 1, 2, ..., n$ and $b \in \mathbb{R}$ are coefficients[15]. Assume the solutions to the clause lie on the non-negative side of the hyperplane that satisfy the inequality. The feasible region for the PSAT formula is defined as the intersection of all these half-spaces, denoted by

$$P = \cap_{j=1}^{m} p \in H_n : a_{j,1} p_1 + a_{j,2} p_2 + ... + a_{j,n} p_n \geq b_j.$$

The geometrical view of PSAT provides a rich mathematical structure for the problem, and enables the use of tools from linear programming, convex optimization, and geometric algorithms to design efficient algorithms and heuristics for solving PSAT instances. It also highlights the connections between PSAT and other areas of optimization and machine learning, such as linear programming, neural networks, and support vector machines.

2.3 ChopSAT

Gomory [7] introduced the cutting plane method as an extension to Dantzig's linear programming approach to solving discrete variable extremum problems [5]. Gomory provided a method so that when the objective function was maximized (e.g., using the simplex method), if the result was a non-integral solution, then a new constraint plane was found algorithmically to separate (cut) the non-integer solution from the integer solutions. Chvatal [2], [3] developed this method further, proved supporting theorems for bounded polytopes, and applied the results to solve combinatorial problems. Note that integer programming is in NP [14].

In terms of solving SAT, Cook et al. [4] examined the complexity of cutting plane proofs, and in particular, those for the unsatisfiability of formulae in propositional calculus. This was done in terms of resolution theorem proving using the cutting planes method. Note that it is straightforward to produce an equation from a disjunction; e.g., consider $(a \vee b)$ which results in the equation $a + b \geq 1$. Each conjunct gives rise to an equation, and the set forms an integer programming problem where a and b take on $\{0, 1\}$ values. Note that linear programming relaxation can be used to allow the a and b to take on values in the interval $[0, 1]$ – i.e., relaxing the $\{0, 1\}$ requirement. Hooker [12] provided a way to handle generalized resolution by observing that the resolvent of two clauses corresponds to a certain cutting plane in integer programming. Buss et al. [1] describe an extension to Cook's cutting plane refutation approach that applies to threshold logic analysis. For some recent work using cutting planes, see Devriendt et al. [6] who use the method to improve state-of-the-art pseudo-Boolean optimization. All these methods still yield exponential time complexity for finding SAT solutions.

The Boolean Satisfiability problem can be interpreted geometrically. Given a set of n variables, and a CNF with a set of m conjuncts $\{C_i\}_{i=1}^{m}$, each conjunct can be represented as a hyperplane of dimension $n - 1$. This hyperplane separates satisfactory models from unsatisfactory models (corners of H_n). Each conjunct C_i has at least one variable assignment that evaluates it to false. Furthermore, if there are k literals in the conjunct, there are 2^{n-k} assignments that make it false. A hyperplane can be produced for each conjunct that separates solutions from non-solutions. The intersection of the solution side of the hyperplane with the hypercube H_n (representing the untouched original feasible region) produces a convex feasible region. Applying this method to each conjunct results in a convex feasible region that may or may not contain a corner of the original hypercube.

Given n logical variables, there are 2^n models. These models are represented as n-tuples in n-dimensional space, where satisfying models are the corners of H_n. This essentially means that the i^{th} axis corresponds to the values that can be assigned to the i^{th} variable, and these values are in $[0, 1]$ for $i = 1, 2, ..., n$. Constructing the hyperplanes this way makes it possible to separate a subset $\mathcal{S}$ of vertices of H_n from vertices that are not a solution. This idea stems from the fact that every conjunct is itself a disjunction, and a disjunction is made false by assigning a falsifying model. The set of models which do not satisfy this conjunct includes both the falsifying models as well as all the possible assignments to the literals not appearing in that conjunct. This turns out to be a sub-hypercube which can be separated, or chopped, from the H_n by a hyperplane. The feasible region represents the solution space for the KB. In the context of SAT solving, the key indicator is the existence of any of the original hypercube corners in the feasible region: if there are any corners of H_n in the feasible region, then the KB is satisfiable, otherwise it is unsatisfiable. I_n, a special convex polytope, is defined as the largest feasible region for *any* unsatisfiable sentence. Moreover, every feasible region produced by an unsatisfiable sentence is contained in I_n.

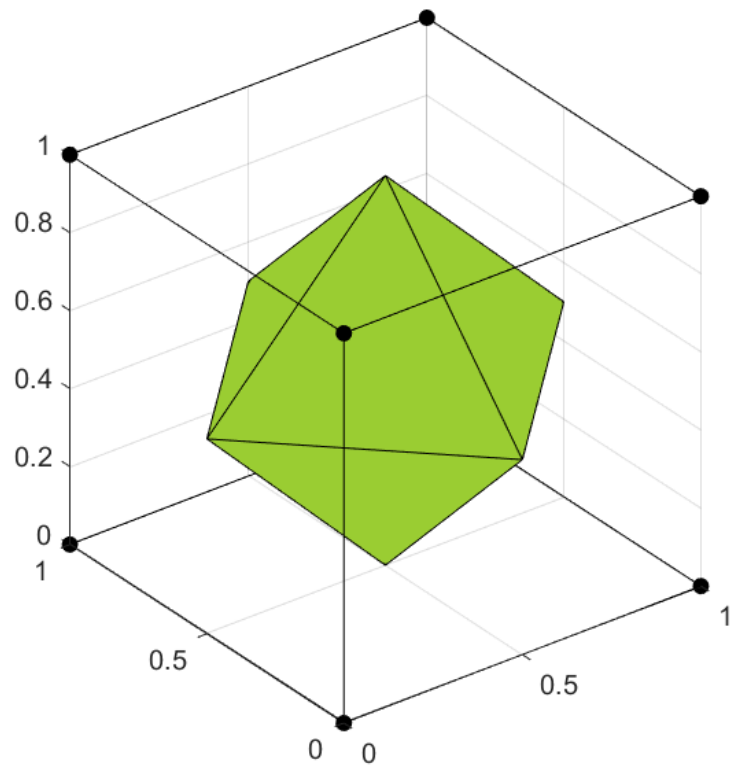


Figure 2.2: I_3: H_3 with all corners (solutions) chopped off individually.

The geometric approach to SAT developed some properties of I_n and showed that no H_0 or H_1 components remained within such a feasible region[10]. Previous approaches of converting clauses to systems of linear and non-linear equations resulted in a fundamental problem: the need to add constraints between variables, resulting in exponential complexity[9]. However, an important observation to make is that every corner of H_n is $\frac{\sqrt{n}}{2}$ distant from the center of H_n. In fact, every satisfiable sentence results in a feasible region which contains an H_n corner. This means that there exists some point in the feasible region that is $\frac{\sqrt{n}}{2}$ distant from the center of H_n, and this fact enables the determination of sentence satisfiability. The question arises as to what the maximal distance is of any feasible point after the cuts are applied to a CNF with no solution. This thesis maintains the assertion that the maximal possible distance occurs if the feasible region of an unsatisfiable CNF sentence contains the center of a 2-D hypercube (a square) which is at distance $\frac{\sqrt{n-2}}{2}$. In summary, if it can be determined that no point in the feasible region is $\frac{\sqrt{n}}{2}$ distant from the center of H_n, then the CNF is not satisfiable.

CHAPTER 3

SINGLETON ELIMINATION

3.1 Singleton Detection

The detection of singleton points in the sub-hypercube H_k contained within H_n may help gain insight into the existence of a SAT solution. Any vertex of a feasible region $\mathcal{F}$ which has a non-integer coordinate may be a singleton point. A singleton point is defined as a point in the feasible region that is not a corner of H_n and is the only point in some sub-hypercube H_k of H_n. Formally, given a CNF sentence, S, over n variables, each geometric chop (see [10]) removes some sub-hypercube of H_n. The resulting feasible region, $\mathcal{F} \subseteq H_n$, is a convex set, and if bounded, is a convex polytope. $\mathcal{F}$ can be defined as a set of vertexes, V such that:

$$\begin{aligned} V &= V_B \cup V_F \text{ where} \\ V_B &= \{\bar{v} \mid \bar{v} \text{ is a vertex of } H_n\} \\ V_F &= \{\bar{v} \mid \bar{v} \in \mathcal{F} \text{ but } \bar{v} \text{ is not a vertex of } H_n\} \end{aligned}$$

If there exists a sub-hypercube $H_k \subset H_n$ such that:

(i) $\bar{v} \in V_F$

(ii) $\bar{v} \in H_k$

(iii) $\bar{x} \in H_k \cap \mathcal{F} \implies \bar{x} = \bar{v}$

then $\bar{v}$ is a singleton point.

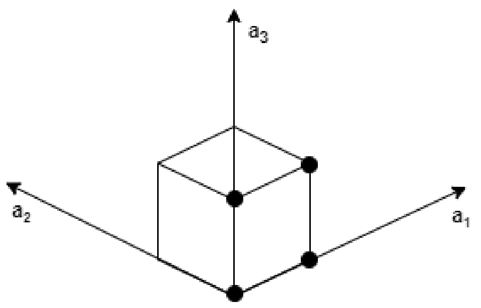
To verify candidate singleton points, it is necessary to examine the sub-hypercube H_k that contains the singleton point. Determining H_k is straightforward: say we are presented with a candidate singleton point $\bar{p}_s = [b, f_2, b, b, f_5, b, f_7]$ where $b \in \{0, 1\}$ and $f_i \in \{x \mid 0 < x < 1\}$. In this case, H_k would be an H_3 defined by the fractional coordinates $f_2, f_5,$ and f_7. After determining the sub-hypercube H_k, a cut is added to H_n such that only H_k is kept. If the projection onto each axis of H_n results in $\bar{p}_s$, then $\bar{p}_s$ is the only point in H_k, and thus must be a singleton point. Algorithm SDA for detecting singleton points is given below, as well as an example.

Algorithm 1 Singleton Detection Algorithm (SDA)

On input: hp: a hyperplane representation of the KB; $\bar{x}$: a candidate singleton point
On output: s: a Boolean indicating if $\bar{x}$ is a singleton point; h: the subspace that contains $\bar{x}$
Find fraction axes: iterate over $\bar{x}$ and note which indices are not $\in \{0, 1\}$
Chop non sub-hypercube: fix the $0/1$ dimensions so that when linear programming is used it only considers the dimensions with fractional values.
Project: Project all fractional axes onto that normal hyperplane; if any solution point of the linear program problem is not $\bar{x}$, then $\bar{x}$ is not a singleton point

Suppose we are given the following KB and corresponding hyperplane representation where unsatisfiable solutions are shown as black dots.

$$\begin{aligned} \text{KB} = &[1, 2, 3] \\ &[-1, 2, 3] \\ &[1, 2, -3] \\ &[-1, 2, -3] \end{aligned}$$



Chopping off the unsatisfiable solutions individually results in a singleton point contained in an H_2 sub-hypercube (specifically, the square where $a_2 = 0$). The output of Algorithm SDA run on the above KB with the projection vector $[0, 1, 0]$ is shown below:

$s = 1$ $\qquad\qquad$ $h = 0 \quad 1.000 \quad 0$

confirming that a singleton point exists in the H_2 sub-hypercube.

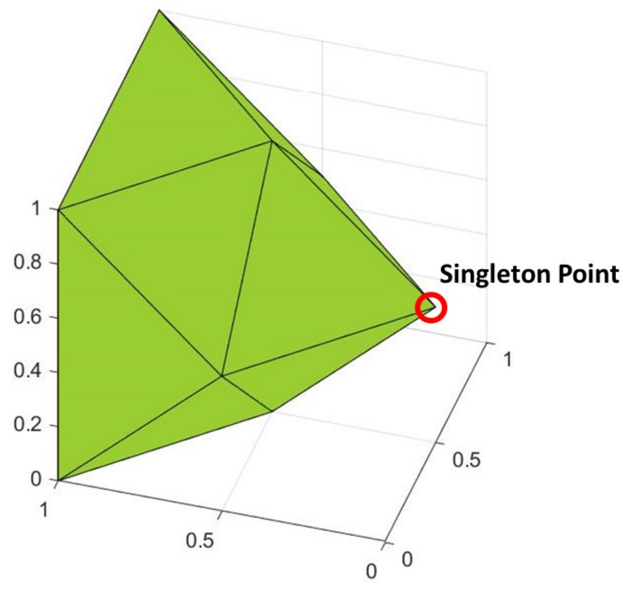


Figure 3.1: The chopped polyhedron. It has corners:
[0.5,0,0.5; 0,0.5,0.5; 0.5,0.5,0; 1,0.5,0.5; 0.5,0.5,1; 1,1,1; 0,1,1; 0,1,0; 1,1,0]. The rightmost point in the figure is a singleton point.

3.2 Singleton Elimination

If both a singleton point and the corresponding containing H_k can be identified, then the H_k can be eliminated from H_n, reducing the feasible region. This is done by adding H_k as a constraint to the hyperplane representation of the KB. The question examined here is whether the removal of singleton points iteratively leads to an effective determination of the (un)satisfiability of the input CNF sentence S. In this chapter, key metrics used to monitor the performance of singleton elimination include number of iterations until no singleton point is found, and volume of the feasible region $\mathcal{F}$ per iteration. Algorithm SEA for eliminating singleton points is given below.

Algorithm 2 Singleton Elimination Algorithm (SEA)

```
Input:  hp: hyperplane representation of KB
Output: $hp'$: hyperplane representation of KB with singleton points removed
        $sol$: 1 if solution found; else 0
        $\bar{x}$: solution (if one is found)
        $count$: number of iterations of algorithm
  $hp' \leftarrow hp$
  $sol \leftarrow 0$
  $count \leftarrow 0$
  $done \leftarrow False$
  $P \leftarrow$ set of projection vectors
  while $\neg$ done do
      $count \leftarrow count + 1$
      $done \leftarrow 1$
      for $k = 1 : |P|$ do
          $\bar{p} \leftarrow k^{th}$ vector in $P$
          $\bar{x} \leftarrow$ linprog solution for $hp'$ and $\bar{p}$
          if $\bar{x}$ is $\varnothing$ then                    ▷ the feasible region is empty
              $hp' \leftarrow \varnothing$
              return
          else if $\bar{x}$ is solution then
              $sol \leftarrow 1$
              return
          else if $\bar{x}$ is singleton point in $H_k$ then
              update $hp'$ by removing $H_k$
              $done \leftarrow 0$
          end if
      end for
  end while
```

3.3 Unsatisfiable CNFs and Geometry

An unsatisfiable Boolean formula in CNF produces a convex region $\mathcal{F} \subset H_n$ such that no point in $\mathcal{F}$ is at a distance greater than $\frac{\sqrt{n-2}}{2}$ from the center of H_n. The center of any $H_k \subseteq H_n$ is at distance $\frac{\sqrt{n-k}}{2}$ from the center of H_n. For example, consider H_3, a cube; in order of largest to smallest distance from the center of H_3, we find: corners of H_3, edge centers, 2-D face centers, and finally the center of the 3-D hypercube itself. If every point in $\mathcal{F}$ is less than or equal to $\frac{\sqrt{n-2}}{2}$ distant from the center of H_n, then the Boolean formula is not satisfiable. It is conjectured that if $\mathcal{F}$ is the feasible region of an unsatisfiable sentence, then for every rotation of $\mathcal{F}$ about the center of H_n, all points of the rotated set are contained in H_n. If a point in the rotated feasible region is not contained in H_n, then a solution must exist.

If a Boolean formula S is inconsistent, then $I_s \subseteq I_n$. Because there are no corners of H_n in an inconsistent feasible region, projected points via linear programming that are not completely composed of {0,1} elements are searched for and marked as candidate singleton points.

SEA was run on feasible regions $I_n, n = 3$ to 13 to observe the performance of SEA in reducing the feasible region. Performance is measured in terms of iterations of SEA, and the results are provided in Table 3.1.

Table 3.1: Iterations to satisfiability

n	3	4	5	6	7	8	9	10	11	12	13
count	3	5	12	20	43	67	100	585	1149	1995	3770

Having established the performance of SEA in reducing the feasible region, we now seek to determine its complexity as a function of n. The hope here is that a non-exponential model with a reasonably low mean squared error (MSE) can be fit. Fitting the data to a general linear model $f(x) = ax_1 + b$ resulted in the coefficients and MSE shown in Table 3.2.

Table 3.2: 1 Degree Polynomial Fit

a	b	MSE
285.39	-1578.67	567.07

Fitting the data to a general 7 degree polynomial model $f(x) = ax + bx^2 + cx^3 + dx^4 + ex^5 + fx^6 + gx^7 + h$ resulted in the coefficients and MSE shown in Table 3.3.
Fitting the data to a general exponential model $f(x) = ae^{(bx)}$ resulted in the following 95%

Table 3.3: 7 Degree Polynomial Fit

a	b	c	d	e	f	g	h	MSE
0.05	-2.47	55.22	-658.55	4518.84	-17802.26	37226.92	-31840.00	18.49

confidence interval for the coefficients and MSE:

$$a = 0.86 \quad [0.25, 1.46]$$

$$b = 0.65 \quad [0.59, 0.70]$$

$$\text{MSE} = 47.22$$

These functions are shown in Figure 3.2.

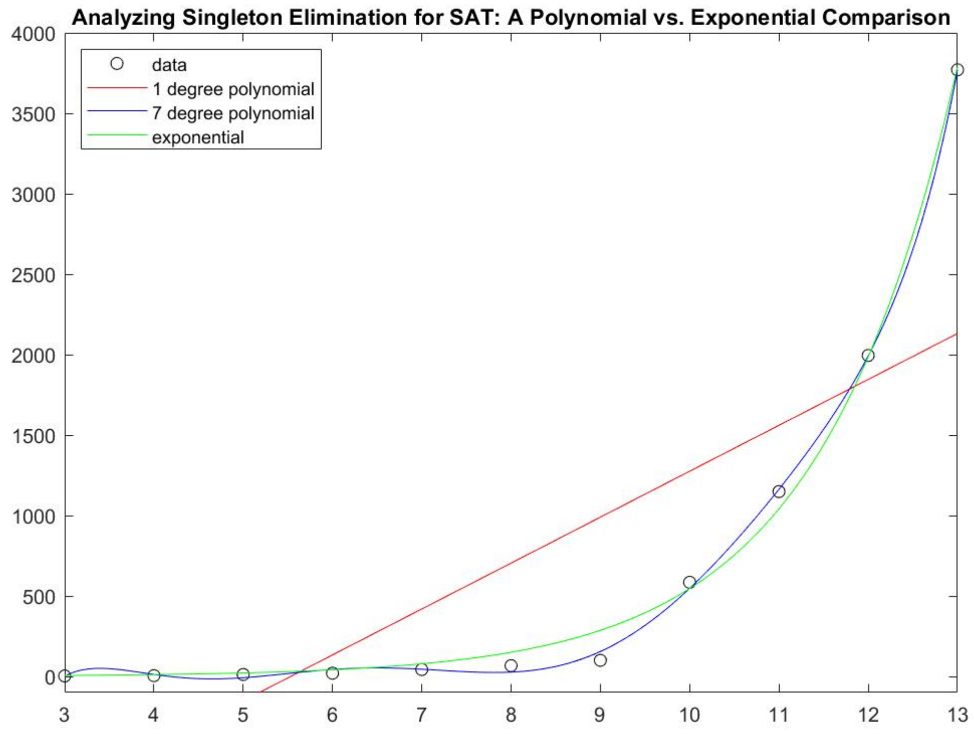


Figure 3.2: Singleton elimination shows evidence of polynomial time complexity.

In addition to measuring the performance of SEA in reducing the feasible region, we also consider the rate of change in the volume of $\mathcal{F}$, denoted by v_f, as an important metric for evaluating its effectiveness. The goal is to determine whether, as SEA iterates, v_f steadily decreases with the removal of more and more singleton points and their corresponding H_k. The volume of the feasible region was approximated every iteration through the hit-or-miss method. The hit-or-miss method is a Monte Carlo algorithm for volume estimation in high-dimensional polytopes. It works by generating uniformly sampled random points within a H_n and counting the number of points that fall inside the feasible region. The algorithm is based on the fact that the ratio of the volume of the hypercube to the volume of the feasible region is equal to the ratio of the number of points that fall inside the feasible region to the total number of points generated. By repeating this process multiple times and averaging the results, an estimate of the volume of the feasible region with a desired level of accuracy can be obtained. The hit-and-miss method is often useful for high-dimensional polytopes of adequate volume as other methods of volume calculation and approximation are computationally infeasible.

In every iteration of SEA, the volume of each I_n reduction is monitored to analyze the efficacy of the algorithm. For each I_n, the number of volume observations is equivalent to the number of iterations in SEA except for I_{13}, which resulted in fewer volume observations compared to SEA iterations for I_{13}. This is illustrated in Figure 3.3 and Figure 3.4, where the volume per iteration for each I_n is shown. The observed behavior of the volume of the feasible region approaching 0 for each I_n is an indication that the algorithm is effective in identifying singleton points and removing the containing sub-hypercubes. This systematic reduction in volume reinforces the efficacy of SEA. For I_n where $n > 5$, the number of generated points needed to be adjusted for the larger dimensions, as using a fixed number of generated points for every I_n resulted in inaccurate volume estimations. If the number of generated points is too low, then the hit-or-miss method may fail to capture important regions of the polytope, resulting in an inaccurate estimate of its volume.

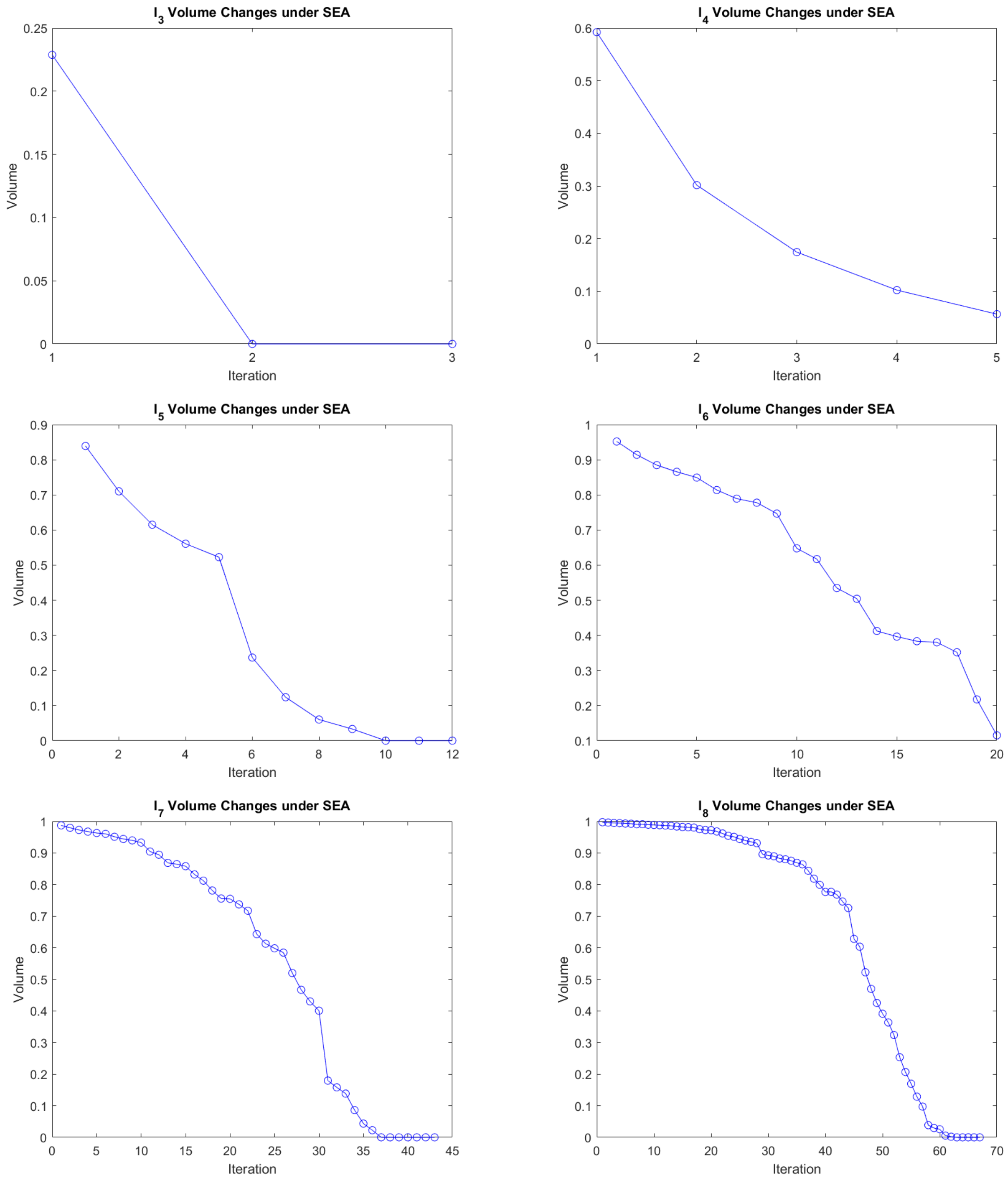


Figure 3.3: Monitoring change in feasible region volume under Singleton Elimination Algorithm, I_3 to I_8.

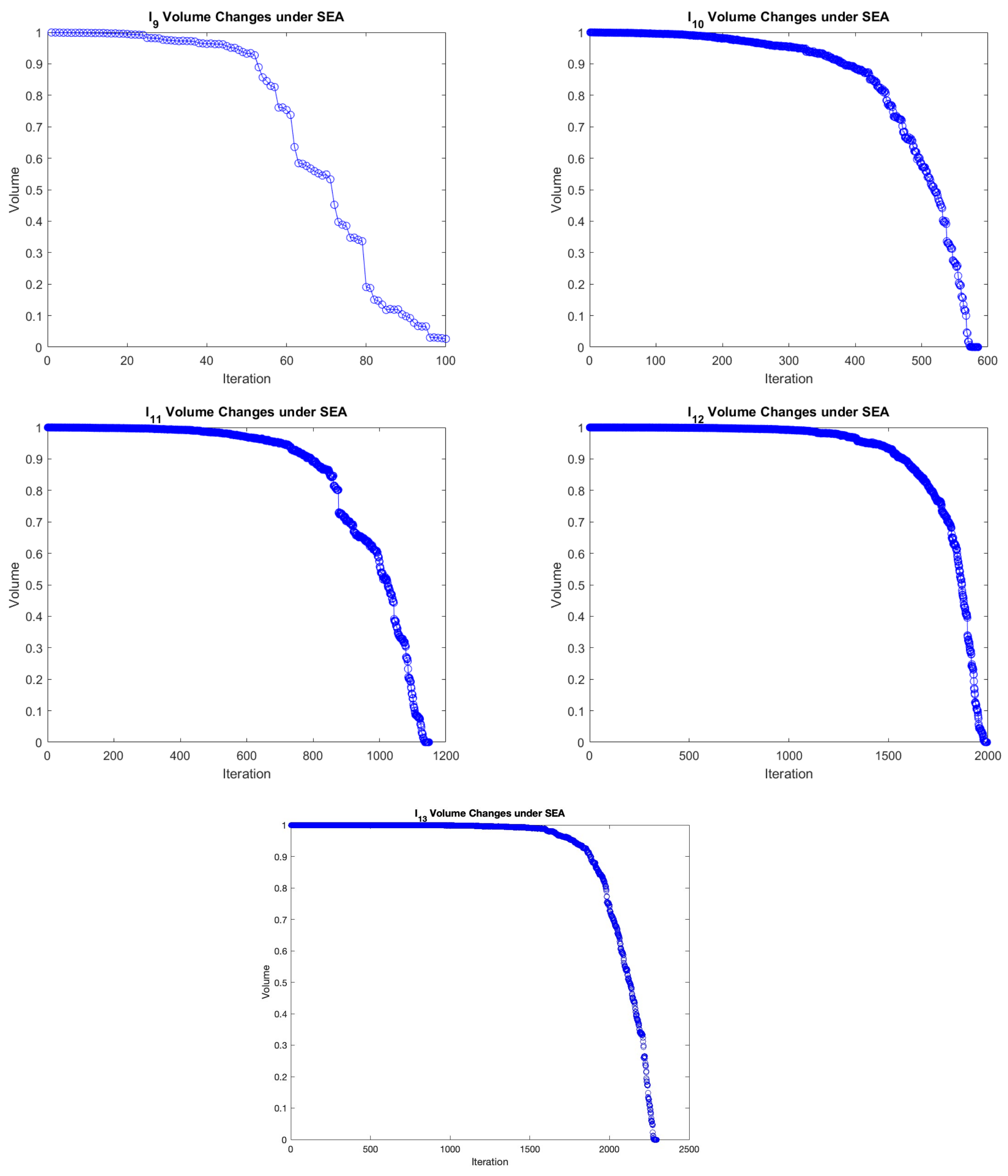


Figure 3.4: Monitoring change in feasible region volume under Singleton Elimination Algorithm, I_9 to I_{13}.

Due to the exponential model having a larger MSE than a 7 degree polynomial model, we can conclude that for dimensions $d \leq 13$, SEA results in polynomial time SAT solving. Regarding the rate of change in the volume of the feasible region during the application of SEA, the data shows that the polytope volume decreases rapidly at the beginning of the algorithm and then levels off as the algorithm removes most of the singleton points. In every experiment, the volume of the feasible region approaches zero. Overall, these results provide insight into the properties of unsatisfiable Boolean formulas in CNF and the effectiveness of SEA in reducing the feasible region. The data and analysis suggest that SEA is a promising method for reducing the feasible region of unsatisfiable Boolean formulas, particularly for low-dimensional problems.

CHAPTER 4

MAXIMUM VOLUME ELLIPSOID

4.1 Maximum Volume Ellipsoid

The maximum volume inscribed ellipsoid (MVE) contained in the convex feasible region $\mathcal{F}$ may be instrumental in determining if a solution to SAT exists. An ellipse is a set of the form

$$E = \{x \in \mathbb{R}^n | (x-c)^T A(x-c) \leq 1\}$$

where A is a $n \times n$ symmetric positive definite matrix and $c \in \mathbb{R}^n$ is a point called the *center* of the ellipsoid. A *semi-axis* of an ellipsoid is defined as the line segment from the center of the ellipsoid to the point where the axes intersect with the surface. For every bounded full dimensional convex body $\Omega \subset \mathcal{R}^n$, there exists both a unique ellipsoid $\underline{E}(\Omega)$ of maximal volume contained within Ω and a unique ellipsoid $\overline{E}(\Omega)$ of minimal volume containing Ω[15]. An inscribed ellipsoid in a convex region can be be mapped to a circumscribed (but not necessarily minimal volume) ellipsoid by enlarging it from its center by a factor n. Similarly, a circumscribed ellipsoid about a convex region can be mapped to a inscribed (but not necessarily maximal volume) ellipsoid by shrinking it from its center by a factor of $1/n$. Computing the minimum volume circumscribed ellipsoid is NP-hard, however computing the maximum volume inscribed ellipsoid is not[8], [13]. If a minimum volume circumscribing ellipsoid could be calculated in polynomial time, it may be possible to determine satisfiability by checking if the maximum semi-axis is less than $\frac{\sqrt{n-2}}{2}$.

We take advantage of the fact that methods exist to calculate the MVE of a bounded full dimension convex body in polynomial time[16]. This allows us to explore properties of the feasible region in the hope that it will aid in the identification of solutions. Computing the MVE of the feasible region provides a tight ellipsoidal approximation of the region by inscribing the ellipsoid within the region in such a way that it has the maximum possible volume. Once the ellipsoid is obtained, it can be used to efficiently check the satisfiability of a new set of constraints. In general, this ellipsoid will have one of its major semi-axis aligned with the most anisotropic part of the feasible region $\mathcal{F}$[11]. This essentially gives a direction in which to look for a solution. Figure 4.1 shows some points sampled from the ellipse, with a major semi-axis aligned with solution $[1, 1]$. It is crucial that the volume of the MVE is detectable. 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 4.2 shows some points sampled from the ellipse moved outward such that the H_2 has a side length of 5, with a major semi-axis aligned with solution $[1, 1]$.

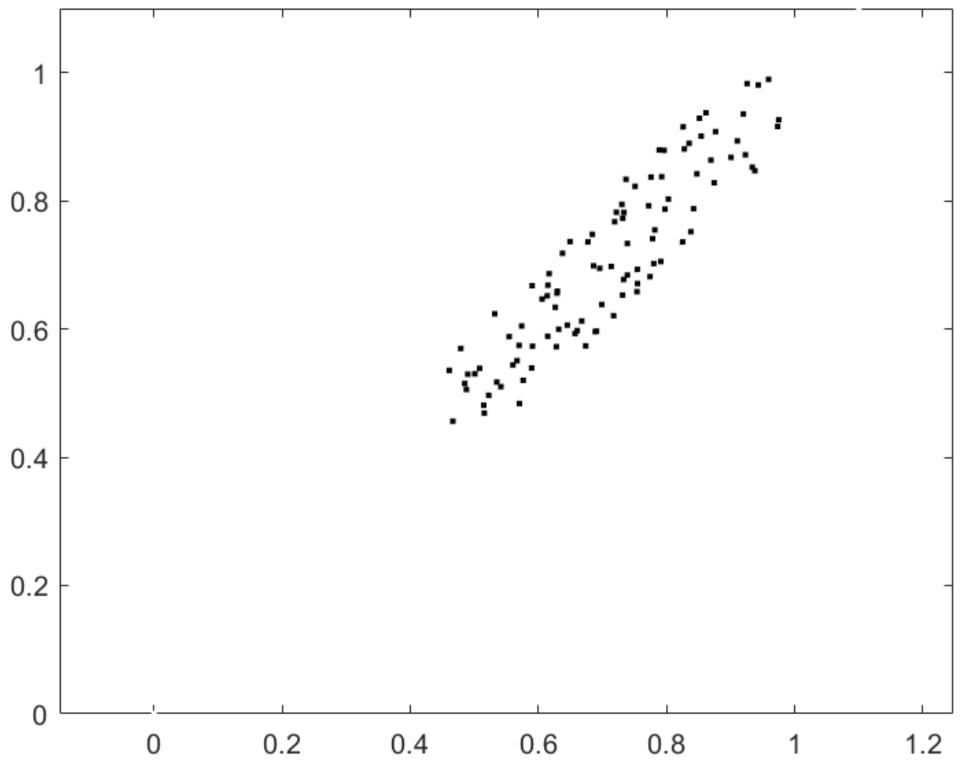


Figure 4.1: The MVE method indicating the direction of a solution in 2-D.

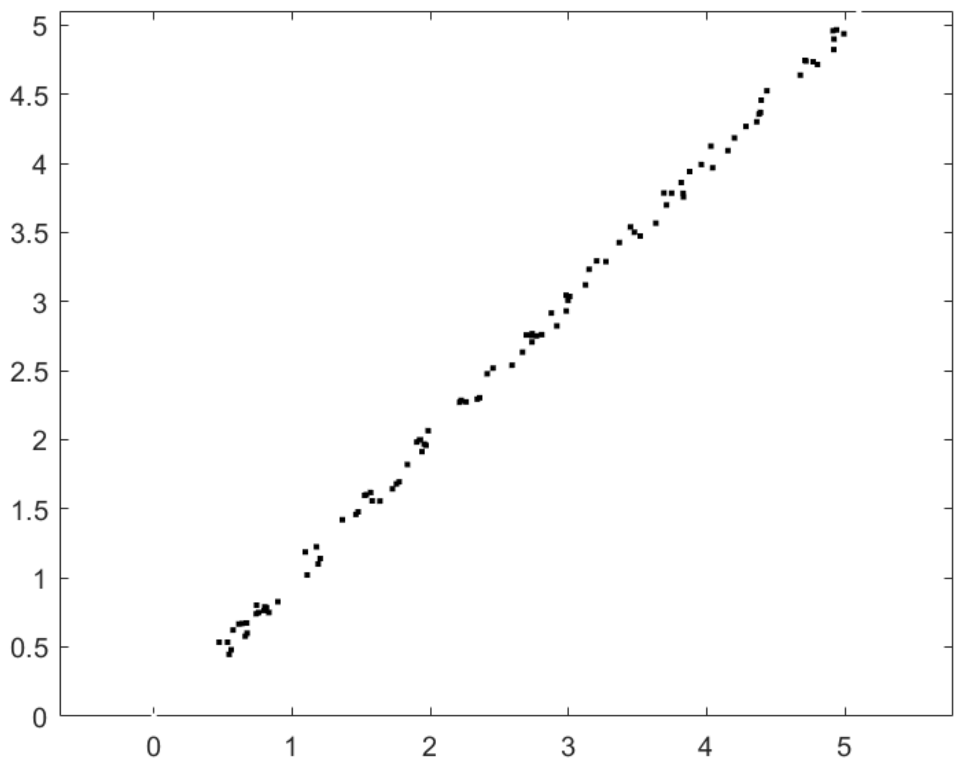


Figure 4.2: Moving the hypercube constraints such that the H_2 has a side length of 5. The volume of the ellipse increases.

Changing the offset at which unsatisfiable solutions are chopped off has an effect on MVE. In Figure 4.3, sampled points from MVE's with different chop offsets are illustrated, with blue representing the smallest offset, green representing the next smallest, and red representing the largest.

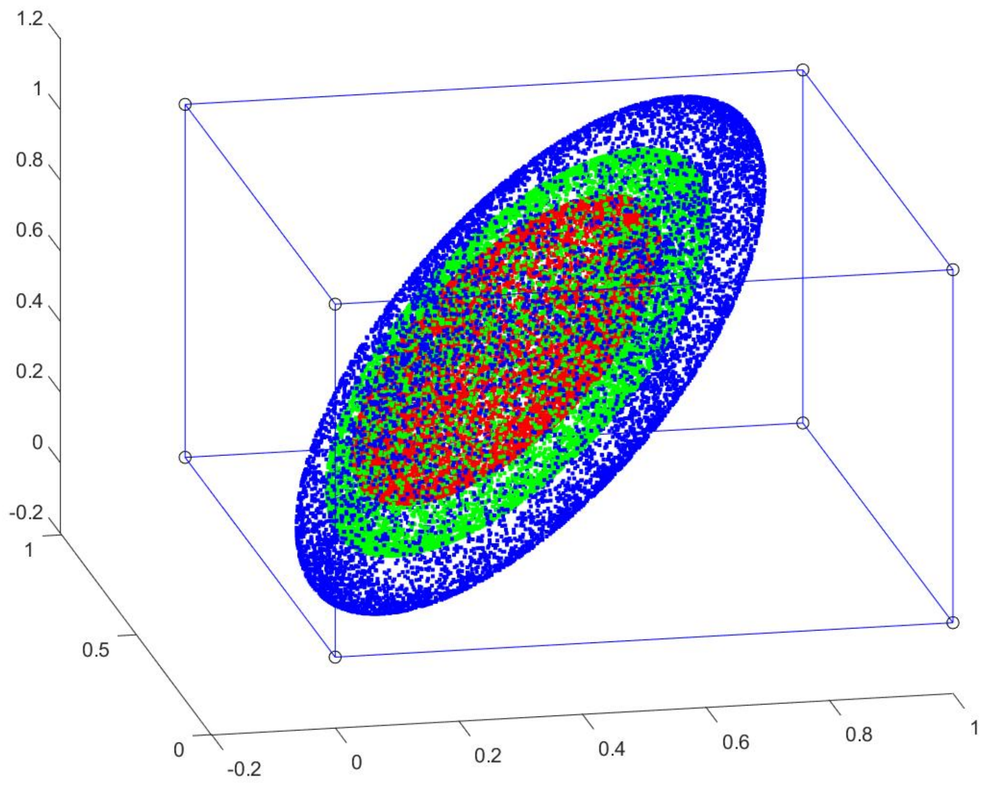


Figure 4.3: Calculated MVE with different offsets in H_3. There are only two solutions: [0,0,0] and [1,1,1].

An experiment was run to investigate the utility of using the maximum volume inscribed ellipsoid to detect satisfiable solutions in randomly generated knowledge bases. For $n = 10$ to 20, two sets of 100 random KB's were generated, one ensuring independence of variables and the other not. Each KB was composed of n atoms and had a maximum of $2n$ clauses where the maximum number of literals in any clause was n. First, each KB was converted into a hyperplane representation, hp. For each hp, the MVE was calculated and a projection vector from the semi-axes of the ellipsoid was derived. This projection vector was then used as a constraint for linear programming to find a solution x_s to the KB. For every direction along the projection vector ($2n$ directions in total), if the solution point is more than $\sqrt{n-2}/2+\epsilon$ distant from the center of the ellipse or if x_s satisfies the KB, then x_s indicates there is a solution. If x_s indicates a solution, then a 1 is returned, else, a 0 is returned. Table 4.1 shows the average number of attempted directions out of $2n$ directions for each set of KBs that led to a solution indication for n ranging from 10 to 20.

Table 4.1: Average number of directions looked along projection vector until a solution is indicated for 100 random KB's with independent variables (Set1) and 100 random KB's with no variable independence constraints (Set2).

n	10	11	12	13	14	15	16	17	18	19	20
Set1	10.53	11.43	12.52	13.85	14.73	15.98	16.99	17.92	18.88	20.15	21.33
Set2	10.33	11.65	12.88	13.79	15.00	16.08	17.44	18.32	19.51	20.97	21.46

The experiment found that for all 100 randomly generated knowledge bases, both independent and non-independent, at least one projection vector obtained from the MVE resulted in an indication of a satisfiable solution when used as an input to a linear program. This suggests that the MVE can be a useful tool for aiding in the detection of satisfiable solutions for generated knowledge bases. The experiment also found that on average, out of the $2n$ axes directions to look along the projection vector, only n were needed to indicate the presence of a solution. This suggests that the MVE is not only useful determining satisfiability, but efficient as well.

CHAPTER 5

CONCLUSIONS AND FUTURE WORK

The significance of solving the Boolean satisfiability problem efficiently lies in its position as one of the most important open problems in theoretical computer science, as a successful resolution of the problem would have significant implications for the field of computational complexity theory. In this thesis, we explored the use of singleton elimination and the maximum volume inscribed ellipsoid as methods for improving the efficiency and effectiveness of SAT solving.

We defined methods to detect and eliminate singleton points in Chapter 3 and tested the efficacy of Algorithm SEA. Promising results were found in the complexity of determining the unsatisfiability of KB's with all corners removed individually under SEA. Further work on more general cases of unsatisfiability needs to be performed. Additionally, through volume estimation, we found that SEA reduces the feasible region in all iterations in every observed dimension, proving its ability to aid in SAT solving. Overall, singleton elimination provided valuable insight into properties of unsatisfiable Boolean formulas and resulted in efficient SAT solving when applied to KB's in CNF.

In Chapter 4, we articulated a novel method of applying a maximum volume inscribed ellipsoid to our geometric interpretation of SAT. We found that the MVE indicated a solution in every observed experiment, requiring only a linear number of checks along its major semi-axis to do so. The results suggest that in the context of geometric SAT, MVE may be a useful and efficient method of determining satisfiability for certain KB's.

Future work includes exploring the applications of SEA in industries and organizations with high dimensional datasets. Another potential direction of solving a geometric representation of SAT is utilizing matrix shearing. We speculate that matrix shearing can be used to transform the coordinates of solutions in such a way that preserves their relative positions and orientations, but changes the shape of the feasible region. This may be a method of radically exposing solutions, especially when a MVE is computed before shearing.

REFERENCES

[1] S. Buss and P. Clote, “Cutting Planes, Connectivity, and Threshold Logic,” *Archive for Mathematic Logic*, vol. 35, pp. 33–62, 1996.

[2] V. Chvatal, “Edmonds Polytopes and a Hierarchy of Combinatorial Problems,” *Discrete Mathematics*, vol. 4, pp. 305–337, 1973.

[3] ——, “Cutting Planes in Combinatorics,” *European Journal of Combinatorics*, vol. 6, pp. 217–226, 1985.

[4] S. A. Cook, “The complexity of theorem-proving procedures,” in *Proceedings of the Third Annual ACM Symposium on Theory of Computing*, ser. STOC ’71, Shaker Heights, Ohio, USA: Association for Computing Machinery, 1971, pp. 151–158, ISBN: 9781450374644. DOI: 10.1145/800157.805047. [Online]. Available: https://doi.org/10.1145/800157.805047.

[5] G. Dantzig, “Discrete-Variable Extremum Problems,” *Journal of Operations Research Society of America*, vol. 5, no. 2, 1957.

[6] J. Devriendt, S. Gocht, E. Demirovic, J. Nordstrom, and P. 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. Gomory, “Outline of an Algorithm for Integer Solution to Linear Programs,” *Bulletin of the Americal Mathematical Society*, vol. 64, no. 5, pp. 275–278, 1958.

[8] J.-Y. Gotoh and H. Konno, “Minimal ellipsoid circumscribing a polytope defined by a system of linear inequalities,” *Journal of Global Optimization*, vol. 34, no. 1, pp. 1–14, Jan. 2006, ISSN: 1573-2916. DOI: 10.1007/s10898-005-3883-8. [Online]. Available: https://doi.org/10.1007/s10898-005-3883-8.

[9] 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*, vol. 278, 2020.

[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] T. C. Henderson, A. Lessen, I. Rajan, and T. Nishida, “Chop-SAT: A New Method for Knowledge-Based Agent Decision Making,” in *International Conference on Autonomous Intelligent Systems*, Suwon, South Korea, Jul. 2023.

[12] J. Hooker, “Generalized Resolution and Cutting Planes,” *Annals of Operations Research*, vol. 12, pp. 217–239, 1988.

[13] L. Khachiyan and M. Todd, "On the complexity of approximating the maximal volume ellipsoid for a polytope," Cornell University, Ithaca, NY, Technical Report TR No. 893, 1990.

[14] C. Papadimitriou, "On the Complexity of Integer Programming," *Journal of the Association of Computing machinery*, vol. 28, pp. 765–768, Oct. 1981.

[15] Y. Ye, *Interior Point Algorithms: Theory and Analysis*. New York, NY: John Wiley & Sons, 2011.

[16] Y. Zhang and L. Gao, "On numerical solution of the maximum volume ellipsoid problem," *SIAM Journal on Optimization*, vol. 14, no. 1, pp. 53–76, 2002.