THESIS

IN PURSUIT OF INDUSTRIAL LIKE MAXSAT WITH REDUCED MAX-3SAT RANDOM GENERATION

Submitted by

Noah R. Floyd

Department of Computer Science

In partial fulfillment of the requirements

For the Degree of Master of Science

Colorado State University

Fort Collins, Colorado

Fall 2024

Master's Committee:

Advisor: Darrell Whitley

Sarath Sreedharan
David Aristoff

ABSTRACT

IN PURSUIT OF INDUSTRIAL LIKE MAXSAT WITH REDUCED MAX-3SAT RANDOM GENERATION

In the modern landscape of MAXSAT, there are two broad classifications of problems: Random MAX-3SAT and Industrial SAT. Random MAX-3SAT problems by randomly sampling variables with a uniform probability and randomly assigning signs to the variable, one clause at a time. Industrial MAX-SAT consists of MAX-3SAT problems as encountered in the real world, and generally have a lower nonlinearity than random MAX-3SAT instances. One of the goals of recent research has been to figure out which rules and structures these industrial problems follow and how to replicate them randomly. This paper builds off of the paper"Reduction-Based MAX-3SAT with Low Nonlinearity and Lattices Under Recombination" [1], implementing its approach to MAX-3SAT clause generation and determining what it can reveal about industrial MAX-13SAT and random MAX-3SAT. This builds off of the transformation from SAT to MAX-SAT problems and hopes to create random MAXSAT problems that are more representative of industrial MAXSAT problems. All this would be in the pursuit of random MAX-3SAT that more accurately maps onto real-world MAX-3SAT instances so that more efficient MAX-3SATsolvers can be produced.

ACKNOWLEDGEMENTS

I would like to thank my advisor Darrell Whitley, who supported me and believed in my abilities, even when I lacked that belief. I would like to to my family who supported my endeavors and lent aid when they could. I would also like to acknowledge the graduate school for motivating me to pursue lofty ambitions and work on research. I would also like to thank my fellow graduate students who motivated me with their many examples of expertly done projects and boundless motivation; without you all as my role models, I would not have aimed so high, nor strived so fiercely.

DEDICATION

In dedication to the pursuit of knowledge and that which aids in it—Espresso.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

Chapter 1

Introduction

1.1 Background Information

Boolean Satisfiability problems, often simplified to SAT problems, are NP-complete problems that focus on solving a collection of clauses of propositional logical expressions containing some combination of boolean variables. MAX-3SAT Clauses are generally written in conjunctive normal form (CNF). This means that the clauses are formulated where each clause "or's" together the variables in that clause and then "and's" together those clauses. For example (1.1):

$$(\neg x_1 \vee x_2 \vee x_3) \wedge (x_2 \vee x_4 \vee \neg x_5) \wedge (x_1 \vee x_6 \vee \neg x_2) \tag{1.1}$$

A solver's goal is to determine if some formulation of the variables x1-x6 exists that satisfies all of the individual clauses. These problems appear in three general categories, industrial MAX-3SAT, random MAX-3SAT, and handcrafted MAX-SAT. Industrial MAX-3SAT is encountered in the real world, in areas of circuit design, artificial intelligence, and software validation [3]. This is contrasted with random MAX-3SAT which is made by randomly generating clauses from a pool of variables and is used mainly in the development, testing, and benchmarking of MAX-3SAT solvers. Handcrafted MAX-SAT problems are intended to mimic industrial problems, but are time-consuming to make and often more limited in scale than random MAX-3SAT problems. One large goal in the area of MAX-3SAT is to figure out how to bridge the gap between random generation and the industrial class of problems. Random MAX-3SAT can then be adapted to better reflect real-world problems and MAX-3SAT solvers can be made efficient and robust. The current generation of random MAX-3SAT problems seems to poorly reflect the actual structure of real-world MAX-3SAT, which is causing a split in performance between the solvers created based on industrial MAX-3SAT problems and random MAX-3SAT problems. This split in performance tends to be divided between exact and inexact solvers. Exact solvers work exhaus-

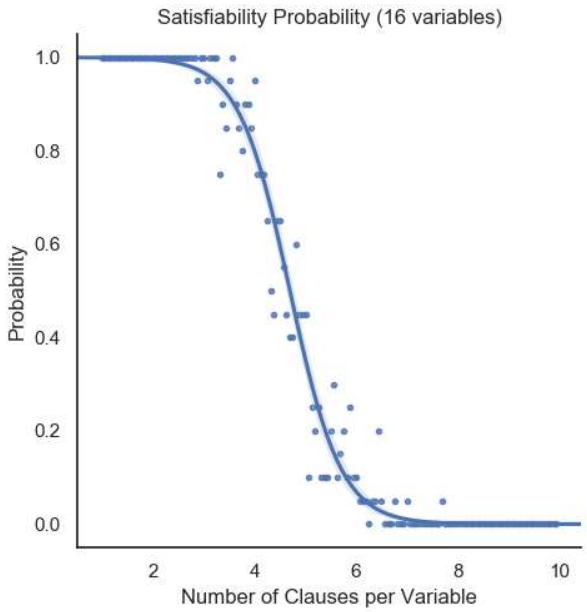

Figure 1.1: 4.27 clause-to-variable phase transition for Random MAX-3SAT with each dot representing a randomly generated SAT problem [2]

tively and are guaranteed to find the best possible solution if one exists, however, there is no guarantee to find a solution in polynomial time. Inexact solvers work on heuristic-based approaches, and are generally faster, but have no such guarantee of finding a solution. Random MAX-3SAT has many avenues for potential investigation. The potential avenues stem from the ability to manipulate parameters, allowing many different variations and minor modifications to be tested quickly. Random MAXSAT has a special metric, "the phase transition," that is not present in industrial MAX-3SAT. The phase transition is the point where MAX-3SAT problems go from generally being satisfiable to generally being unsatisfiable. This point is measured by the clause-to-variable ratio and for random MAX-3SAT, sits at 4.27 [4]. This is graphically represented in Figure 1.1

For example, a random MAX-3SAT problem generated from 100 variables with 300 clauses will very often be satisfiable, whereas a MAX-3SAT problem generated from 100 variables with 500 clauses will rarely be satisfiable. This phenomenon illustrates an aspect of random MAX-3SAT. As the ratio of clauses to variables increases, the problem's constraints increase significantly, leading to a higher chance of unsatisfiability.

MAXSAT is a similar problem to SAT, yet is distinct. Both problems are considered NP-complete problems looking at the satisfiability of a problem. MAX-SAT seeks to answer the question, "How many clauses can be satisfied" whereas SAT more narrowly asks "Can all clauses be satisfied". The overlap between the two variations is large, both working with clauses in CNF and being NP search problems. MAXSAT, however, is an NP-hard problem, that tends to be focused more on the optimization of the number of clauses able to be satisfied if the problem as a whole can be satisfied.

1.2 Motivation

One of the overarching goals in the study of MAX-3SAT is to bridge this divide and create random MAX-3SAT instances that are representative of industrial MAX-3SAT problems [5]. Reduced 3SAT, a novel approach to generating MAX-3SAT problems, is another construction that hopes to bridge that gap even if only slightly [6]. Another motivation is to further understand some of the general trends of both random MAX-3SAT and industrial MAX-3SAT to better develop the different solvers.

1.3 Previous Work

The textbook “Algorithm” by Cormen et al. [7] has an example of converting a 3 variable expression (1.2) and shows how it can be reduced to four clauses in CNF.

The first step is to start with the formula not in CNF for example:

$$y_1 \Leftrightarrow (y_2 \wedge \neg x_2) \tag{1.2}$$

The Truth table for this can be represented as and is used to transform this into CNF:

The next step, after generating Table 1.1, is to convert the problem into disjunctive normal form (DNF). To find the DNF of $y_1 \Leftrightarrow (y_2 \wedge \neg x_2)$ you simply look at all the different classes where

Table 1.1: Truth Table for $y_1 \Leftrightarrow (y_2 \wedge \neg x_2)$

y_1	y_2	x_2	$y_1 \Leftrightarrow (y_2 \wedge \neg x_2)$
1	1	1	0
1	1	0	1
1	0	1	0
1	0	0	0
0	1	1	1
0	1	0	0
0	0	1	1
0	0	0	1

this problem is unsatisfied. This works out to the collection of clauses:

$$(y_1 \wedge y_2 \wedge x_2) \vee (y_1 \wedge \neg y_2 \wedge x_2) \vee (y_1 \wedge \neg y_2 \wedge \neg x_2) \vee (\neg y_1 \wedge y_2 \wedge \neg x_2) \tag{1.3}$$

To convert from this DNF formula to the CNF formula so that the problem can be in the correct nomenclature for MAX-3SAT, Demorgans law needs to be applied. Demorgan's law is:

$$\begin{aligned} \neg(a \wedge b) &= \neg a \vee \neg b \\ \neg(a \vee b) &= \neg a \wedge \neg b \end{aligned} \tag{1.4}$$

When Demorgan's is applied to the DNF formula it converts the problem into the correct CNF form:

$$(\neg y_1 \vee \neg y_2 \vee \neg x_2) \wedge (\neg y_1 \vee y_2 \vee \neg x_2) \wedge (\neg y_1 \vee y_2 \vee x_2) \wedge (y_1 \vee \neg y_2 \vee x_2) \tag{1.5}$$

This formulation of four clauses in CNF with all clauses sharing the same variables will be referred to as a Block. When this block is transformed into a single string using values change to a single subfunction(1.6), then the result can be used to figure out the linearity by applying a Walsh transform.

$$f_i(y_1, y_2, x_2) = \overbrace{4}^{000}, \overbrace{4}^{001}, \overbrace{3}^{010}, \overbrace{4}^{011}, \overbrace{3}^{100}, \overbrace{3}^{101}, \overbrace{4}^{110}, \overbrace{3}^{111} \tag{1.6}$$

This vector of threes and fours represents the number of clauses that are solvable for the given variation of the bit string. For example, when the function mapped over y_1 = False, y_2 = True, x_1 = False, then all four of the clauses can be satisfied, while y_1 = True, y_2 = False, x_1 = True, would result in 3 of the four clauses being satisfied, and the block being unsatisfied. This can be demonstrated in the Truth Table, Table 1.2

Table 1.2: Truth Table for Equation 1.4

y_1	y_2	x_2	$(\neg y_1 \vee \neg y_2 \vee \neg x_2)$	$(\neg y_1 \vee y_2 \vee \neg x_2)$	$(\neg y_1 \vee y_2 \vee x_2)$	$(y_1 \vee \neg y_2 \vee x_2)$	**Sum**
F	F	F	T	T	T	T	4
F	F	T	T	T	T	T	4
F	T	F	T	T	T	F	3
F	T	T	T	T	T	T	4
T	F	F	T	T	F	T	3
T	F	T	T	F	T	F	3
T	T	F	T	T	T	T	4
T	T	T	F	T	T	T	3

This equation can be changed by a constant to yield:

$$f_i(y_1, y_2, x_2) = (1,1,0,1,0,0,1,0) \tag{1.7}$$

As shown in Table 1.2, there are only eight combinations of true or false values for three variables. This leads to $\binom{8}{4}$ different combinations or seventy possible sets of unique clauses. This also means that when the sub-function is applied there exists seventy unique 8-bit long strings [7].

The Walsh transform is a function that processes a set of signals, in this case, the truth table output of a set of four clauses, and transforms them into a polynomial. This is a common transform applied in signal processing and can be used to determine the linearity and nonlinearity of a function. The first step is to take the previously generated bit strings and apply the Walsh transform to them.

Given a Boolean function $f : \{0,1\}^n \to \{0,1\}$, the ith Walsh coefficient for f is defined as:

$$w_i^{(f)} = \frac{1}{2^n} \sum_{x \in \{0,1\}^n} f(x)(-1)^{x \cdot i} \tag{1.8}$$

where $x \cdot i$ denotes the dot product of the binary string x and the binary string represented by integer i.

Lemma 1. *adapted from [1]: Let $f : \{0,1\}^n \to \{0,1\}$ be a Boolean function with c unsatisfiable input combinations (zeroes). Let $\bar{f}$ represent its complement. Then the Walsh coefficients of f and $\bar{f}$ are related as follows:*

$$w_i^{(\bar{f})} = -w_i^{(f)} \quad \forall i \neq 0, \tag{1.9}$$

$$w_0^{(\bar{f})} = 1 - w_i^{(f)} = \frac{c}{2^n}. \tag{1.10}$$

Proof. Using (1.7) the equation is as follows:

$$\begin{aligned} w_i^{(\bar{f})} &= \frac{1}{2^n} \sum_{x \in \{0,1\}^n} (1 - f(x))(-1)^{x \cdot i} \\ &= \frac{1}{2^n} \sum_{x \in \{0,1\}^n} (-1)^{x \cdot i} - \frac{1}{2^n} \sum_{x \in \{0,1\}^n} f(x)(-1)^{x \cdot i} \\ &= \delta_i^0 - w_i^{(\bar{f})} \end{aligned}$$

this assumes the fact that $\sum_{x \in \{0,1\}^n} (-1)^{x \cdot i} = 2^n \delta_i^0$ (see [8]).

According to Lemma 1, if a function with $n = 3$ and $c = 4$ unsatisfying assignments, which describes all seventy of the different possible 8-bit strings, is computed then:

$$w_0^{(\bar{f})} = 1 - w_0^{(f)} = \frac{4}{8} = 0.5.$$

When this is applied to all the seventy unique 8-bit patterns, then there becomes 35 unique Walsh polynomials, because half of the seventy patterns are the complement of the other values.

This means that to evaluate the nonlinearity of the different polynomials only 35 of them need to be assessed. Table 1.3

Table 1.3: Walsh Transform of the Unique 8-bit Patterns

Unique bit string								w_1	w_2	w_3	w_4	w_5	w_6	w_7	w_8
0	0	0	0	1	1	1	1	0.5	0	0	0	-0.5	0	0	0
0	0	0	1	0	1	1	1	0.5	-0.25	-0.25	0	-0.25	0	0	0.25
0	0	1	0	0	1	1	1	0.5	0	-0.25	-0.25	-0.25	0.25	0	0
0	0	0	1	1	0	1	1	0.5	0	-0.25	0.25	-0.25	-0.25	0	0
0	0	0	1	1	1	0	1	0.5	-0.25	0	0.25	-0.25	0	-0.25	0
0	0	0	1	1	1	1	0	0.5	0	0	0	-0.25	-0.25	-0.25	0.25
0	0	1	0	1	0	1	1	0.5	0.25	-0.25	0	-0.25	0	0	-0.25
0	0	1	0	1	1	0	1	0.5	0	0	0	-0.25	0.25	-0.25	-0.25
0	0	1	0	1	1	1	0	0.5	0.25	0	-0.25	-0.25	0	-0.25	0
0	0	1	1	0	0	1	1	0.5	0	-0.5	0	0	0	0	0
0	0	1	1	0	1	0	1	0.5	-0.25	-0.25	0	0	0.25	-0.25	0
0	0	1	1	0	1	1	0	0.5	0	-0.25	-0.25	0	0	-0.25	0.25
0	0	1	1	1	0	0	1	0.5	0	-0.25	0.25	0	0	-0.25	-0.25
0	0	1	1	1	0	1	0	0.5	0.25	-0.25	0	0	-0.25	-0.25	0
0	0	1	1	1	1	0	0	0.5	0	0	0	0	0	-0.5	0
0	1	0	0	0	1	1	1	0.5	-0.25	0	-0.25	-0.25	0	0.25	0
0	1	0	0	1	0	1	1	0.5	0	0	0	-0.25	-0.25	0.25	-0.25
0	1	0	0	1	1	0	1	0.5	-0.25	0.25	0	-0.25	0	0	-0.25
0	1	0	0	1	1	1	0	0.5	0	0.25	-0.25	-0.25	-0.25	0	0
0	1	0	1	0	0	1	1	0.5	-0.25	-0.25	0	0	-0.25	0.25	0
0	1	0	1	0	1	0	1	0.5	-0.5	0	0	0	0	0	0
0	1	0	1	0	1	1	0	0.5	-0.25	0	-0.25	0	-0.25	0	0.25
0	1	0	1	1	0	0	1	0.5	-0.25	0	0.25	0	-0.25	0	-0.25
0	1	0	1	1	0	1	0	0.5	0	0	0	0	-0.5	0	0
0	1	0	1	1	1	0	0	0.5	-0.25	0.25	0	0	-0.25	-0.25	0
0	1	1	0	0	0	1	1	0.5	0	-0.25	-0.25	0	0	0.25	-0.25
0	1	1	0	0	1	0	1	0.5	-0.25	0	-0.25	0	0.25	0	-0.25
0	1	1	0	0	1	1	0	0.5	0	0	-0.5	0	0	0	0
0	1	1	0	1	0	0	1	0.5	0	0	0	0	0	0	-0.5
0	1	1	0	1	0	1	0	0.5	0	0	0	0	0	0	0

Table 1.3 shows all of the unique Walsh polynomials generated from the original 8-bit patterns. These Walsh coefficients represent either linear, quadratic, or cubic interactions. The following table Table 1.4 shows which 3-bit pattern of each Walsh coefficient is associated with and the variables.

Table 1.4: Walsh Coefficients and Checked Variables

x_1	x_2	x_3	Associated Walsh Coefficient	Interactions
0	0	0	w_0	
0	0	1	w_1	x_3
0	1	0	w_2	x_2
0	1	1	w_3	x_3, x_2
1	0	0	w_4	x_1
1	0	1	w_5	x_1, x_3
1	1	0	w_6	x_1, x_2
1	1	1	w_7	x_1, x_2, x_3

For the Reduced MAX3SAT problems, these values are less than one, and more than half of the time are zero. This means the amount of nonlinearity is low. When this is compared to the Standard MAXSAT the difference is stark. For reference, this is a singular standard MAXSAT, when converted into a Walsh polynomial.

Table 1.5: Example of a Random MAXSAT clause's Walsh transform

Example clause	w_0	w_1	w_2	w_3	w_4	w_5	w_6	w_7
1 1 1 0 1 1 1 1	0.875	0.125	0.125	-0.125	-0.125	0.125	0.125	-0.125

To express the nonlinearity of a function the expected number of Walsh Coefficients Per Clause (WPC) can be observed. To get the WPC value, first let v denote the clause-to-variable ratio of a MAX-SAT instance so that m = vn [1].

Theorem 1. Adapted from [1]: Assume that all variable interactions are unique for each clause or for each Reduced MAX-3SAT subfunction and that all linear coefficients are non-zero. Assume $m = vn$. Then the Walsh Coefficients Per Clause (WPC) for Reduced MAX-3SAT in-

stances is WPC $< \frac{1}{v} + 0.5$. The Walsh Coefficients Per Clause (WPC) for standard random MAX-3SAT instances is WPC $= 4 + \frac{1}{v}$.

Proof. [1] There are n linear coefficients. If $m = vn$, the WPC for linear coefficients is $\frac{n}{m} = \frac{1}{v}$. For Reduced MAX-3SAT, there are only 3 quadratic coefficients for every 4 CNF clauses- w_3, w_5, w_6. Most of these are going to be zero, as seen in Table 1.3 Thus, the number of quadratic terms is bounded by $\frac{3}{4}m \cdot 0.5$ and WPC $< \frac{3}{8}$ for the quadratic coefficients. the number of cubic terms is bounded by $\frac{m}{4} \cdot 0.5$ and WPC $< \frac{1}{8}$. Since we assume that all variable interactions per clause are unique, a normal random MAX-3SAT clause has 3 quadratic and 1 cubic term per clause, all with weight ± 0.125 and WPC $= \frac{1}{v} + 4$.

If m = 4n, the standard random MAX-3SAT instance will have an expected approximate WPC = 4.25. The Reduced MAX3SAT will have a WPC = 0.75, which is much more linear and in line with industrial MAX3SAT problems. [1]

Chapter 2

Block Generation

2.1 Motivations for Blocks

The previous work displays the utility of these reductions [1], allowing for random generation of clauses while displaying characteristics similar to industrial MAX-SAT. This unique generation of clauses also allows for metrics like the phase transition to be measured. This method of generating MAX-3SAT clauses should bridge some of the gaps between random MAX-3SAT and allow for a more intensive study of the properties of industrial MAX-3SAT outside of industrial uses.

2.2 Definition of a Block

In the previous section, it was shown that a set of three variables can be transformed into a collection of 4 unique clauses, resulting in a unique block. See (1.5). That section mentioned that seventy unique patterns of clauses can be generated. Each of those different patterns can be thought of as an independent block, that has some formulation of variable assignments that satisfies them. These blocks form the base structure of a Reduced 3SAT problem.

This section describes how the Reduced 3SAT blocks are generated in a practical application to understand how they impact the performance of MAX-3SAT solvers. An important factor in the Reduced 3SAT clauses is which block patterns to include. For Reduced MAX3SAT, seventy possible blocks can be generated, but not all blocks are created equal. Including certain blocks can cause the structure of the problem to vary massively.

The beginning for all of the different forms of block generation start the same, randomly generating three variables from an equal distribution of values. As expressed in (2.1)

$$(x_1, x_2, x_3) \tag{2.1}$$

The next step of the process is to generate a block of four unique clauses that match one of the seventy unique patterns mentioned previously. The choice of these clauses determines the amount of hard constraints the resulting problem has. When referring back to the original space of all possible Reduced MAX3SAT patterns, six of those seventy patterns will introduce unit constraints. A unit constraint is a forced state on a variable or else the problem would be unsatisfied. For example, if a MAX-3SAT problem has to have x_1 be True, then x_1 would be considered a unit constraint. These unit constraints force certain variables, which can quickly lead to a contradiction if those variables have an interaction with another block. The six patterns that cause a unit constraint for example variables x_1,x_2, and x_3 are shown in Table 2.1

Table 2.1: Blocks that generate a unit constraint

Block 1	**Block 2**	**Block 3**	**Block 4**	**Block 5**	**Block 6**
$x_1 \vee x_2 \vee x_3$	$x_1 \vee x_2 \vee x_3$	$x_1 \vee x_2 \vee x_3$	$-x_1 \vee x_2 \vee x_3$	$x_1 \vee -x_2 \vee x_3$	$x_1 \vee x_2 \vee -x_3$
$x_1 \vee x_2 \vee -x_3$	$x_1 \vee x_2 \vee -x_3$	$x_1 \vee -x_2 \vee x_3$	$-x_1 \vee x_2 \vee -x_3$	$x_1 \vee -x_2 \vee -x_3$	$x_1 \vee x_2 \vee -x_3$
$x_1 \vee -x_2 \vee x_3$	$-x_1 \vee x_2 \vee x_3$	$-x_1 \vee x_2 \vee x_3$	$-x_1 \vee -x_2 \vee x_3$	$-x_1 \vee -x_2 \vee x_3$	$-x_1 \vee -x_2 \vee -x_3$
$x_1 \vee -x_2 \vee -x_3$	$-x_1 \vee x_2 \vee -x_3$	$-x_1 \vee -x_2 \vee x_3$	$-x_1 \vee -x_2 \vee -x_3$	$-x_1 \vee -x_2 \vee -x_3$	$-x_1 \vee -x_2 \vee -x_3$

These blocks in Figure 2.1, when included in an MAX-3SAT problem, introduce unit constraints. For a fleshed-out example, Table 2.2 is one of those unique patterns that cause unit constraints, as mentioned above in Table 2.1. This block in particular is block four:

Table 2.2: Block generating a unit constraint

Block
$-x_1 \vee -x_2 \vee -x_3$
$-x_1 \vee x_2 \vee -x_3$
$-x_1 \vee -x_2 \vee x_3$
$-x_1 \vee x_2 \vee x_3$

when simplified these four clauses can be represented as seen in equation (2.2):

$$-x_1 = \text{True} \tag{2.2}$$

This means that the appearance of this block forces a strong constraint on x_1 restricting the possible solutions immensely. These constraints occur in six out of the seventy blocks. This means that if 100 blocks were chosen at random 8.57% or eight of the blocks would be expected to impose unit constraints on the problem. One of these blocks being included in a problem does not immediately imply that the problem is unsatisfied, but it does limit the possible solutions. This problem is exasperated if two blocks have an interaction. This can cause more unit constraints to be generated across multiple different variables.

Chapter 3

Testing the variations of clause generation

This section explores how modern MAX-3SAT solvers perform on the Reduced 3SAT problems, including with the blocks generating unit constraints and removing the blocks that generate unit constraints. This analysis is crucial for understanding the impact of different constraints on the solvability of Reduced 3SAT problems. Through examining the performance of MAX-3SAT solvers with all seventy block patterns, it can be determined how the diversity of block configurations influences the solvers' ability to find solutions. These patterns introduce a variety of relationships among the variables which forces the MAX-3SAT problems to be Unsatisfiable.

3.1 General Formulation

To gain a comprehensive understanding of how the Reduced 3SAT clauses impact the MAX-3SAT space, a series of MAX-3SAT problems were generated following the formulations outlined previously. The MAX-3SAT solver employed for all experiments was the Lingeling solver [9], which is integrated into the Pysat Python library [10]. Lingeling was chosen for its exceptional speed and robust performance, having consistently demonstrated its efficacy in numerous MAX-3SAT competitions [9], making it a reliable choice for evaluating the performance of MAX-3SAT instances. The testing methodology involved generating 100 MAX-3SAT problems using the Reduced 3SAT method with varying numbers of variables for each set while maintaining a consistent clause-to-variable ratio. This allowed for a controlled comparison across different instances, providing a clearer picture of how the potential phase transition is influenced. Additionally, multiple clause-to-variable ratios were explored to examine their effects on the satisfiability of the problems.

3.2 Reduced 3SAT problems with all seventy block patterns

This collection of MAX-3SAT problems is generated using the complete Reduced 3SAT generation method, using all seventy block patterns, including the ones that impose unit constraints on the MAX-3SAT problem. The problems were generated with a clause-to-variable ratio of 1.2, 1.4, 1.6, 1.7, and 2.0. Figure 3.1 illustrates a unique aspect of all Reduced 3SAT prob-

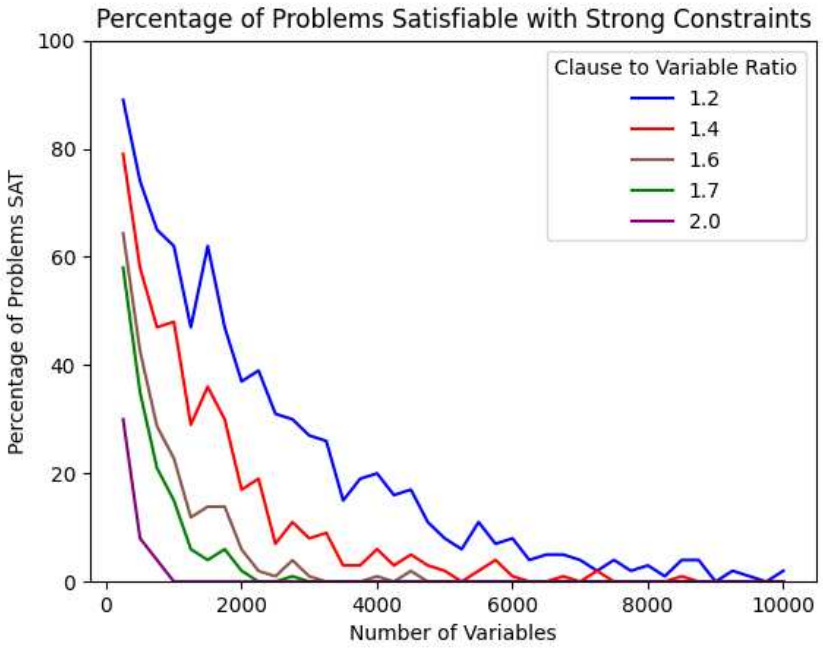

Figure 3.1: Percentage of Problems Satisfiable with Unit Constraints

lems and some features unique to Reduced 3SAT problems that use all seventy block patterns in their generation. The key aspect that can be seen is the phase transition. For standard random 3SAT, the phase transition sits close to a clause-to-variable ratio of 4.27. This phase transition holds steady at 4.27 across different numbers of variables. For example, a random MAX-3SAT problem with 100 variables and 427 clauses will more often than not be satisfiable, and a problem with 1000 variables should be able to be solvable up to approximately 4300 clauses. These Reduced 3SAT clauses do not display this property of being invariant to the problem size, with the percentage of solvable problems decreasing as the number of variables increases. If they

were like standard MAX-3SAT the individual lines should be roughly horizontal. Another aspect of these MAX-3SAT problems is the rapid decrease in solvability that accompanies the growth of the variables. Most of these MAX-3SAT problems have a significant drop in satisfiability very quickly. For example, the problems with a clause-to-variable ratio of 1.7 go from just below 60% solvable to less than 10% solvable when there are 2000 variables.

Another aspect that can be seen in this graph is the difference in the sharpness of the satisfiability transition for the different clauses in variable relations. For example, a clause-to-variable ratio of 1.2 approaches being completely unsolvable, at a rate much slower than 1.4, even though they both start comparably satisfiable. What this could imply is that these reduced 3SAT's transition between satisfiable and unsatisfiable is smoother based on the clause-to-variable relationship.

3.3 Reduced 3SAT problems without Unit Constraint Blocks

This next collection of MAX-3SAT problems is generated using the Reduced 3SAT generation method while removing the six-block patterns that generate unit constraints. This means that the problem only has sixty-four of the original seventy block patterns, which causes it to generate fewer unit constraints. This is because those six problematic blocks mentioned in Table 2.1 implicitly force a unit constraint on a problem. Removing these blocks was done to attempt to create MAX-3SAT problems that better mimicked industrial MAX-3SAT problems while being more invariant to the problem size. The graph in Figure 3.2 was generated with a clause-to-variable ratio of 1.7, 1.8, 1.9, 2.0, and 2.1. These values start higher than the values in Figure 3.1 because at those clause-to-variable ratios, they are almost all satisfiable.

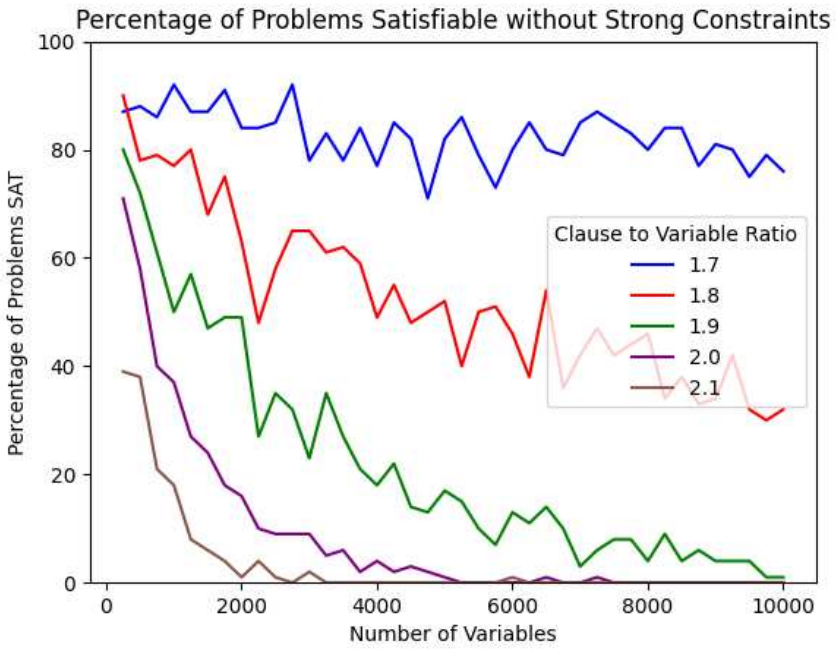

Figure 3.2: Percentage of Problems Satisfiable without unit Constraints

Figure 3.2 shows a marked change from the previous Reduced 3SAT problems. These problems have a higher probability of being satisfiable as the number of variables increases, leading to a slower transition from satisfiable to unsatisfiable. The clearest example of this pheromone

is with the clause-to-variable ratio of 1.7. The line representing a clause-to-variable ratio of 1.7 never hits a phase transition, compared to the previous Figure 3.1 which hits that transition within 1000 variables. The phase transition for this graph is more gradual across the board, with problems with a ratio of 2.0 dropping to almost completely unsatisfiable after 6000 variables when compared to 1000 variables in Figure 3.1.

This shows the impact that those six problematic block patterns have on the solvability of a problem. These problems don't have the added unit constraints that would be implicit in the problematic block patterns, and that allows them to have a lot more variable interactions before they start to generate contradictions. This is why the problems start at a higher clause-to-variable ratio, the unit constraints are not introduced the first time a problematic block is generated, they only come into play after there becomes an interaction between two or more blocks.

Another impact of removing the problematic blocks shows is the effect on the transition from solvable to unsolvable, and how quickly that transition happens. When the problems are generated with a clause-to-variable ratio of 1.8, the solvability hovers around 50% solvable for approximately 4000 variables, being above 60% solvable at 4750 variables and only dropping below 40% solvable after 8000 variables. This means that the phase transition, which tends to be a sharp transition, can be softened and leveled out for these Reduced 3SAT problems with the six implicit unit constraints removed.

Chapter 4

Analysis

4.1 Analysis of the Patterns

The impact of the six-block patterns that force unit constraints is large, making both the phase transition and the solvability of the MAX-3SAT problem low. These patterns also seem to cause the phase transition to be much sharper and more pronounced than when the problematic blocks are not included in the generation of problems. Removing the problematic block patterns does not cause the problems to perform in ways similar to Random MAX-3SAT as one might predict. If, however, these were the only way that unit constraints were generated, then it would be expected that removing them would greatly change the performance, and possibly remove the phenomena of decreased solveability tied to problem size. One of the reasons for this not being the case comes from the generation of reduced MAX-3SAT problems. This generation method is highly susceptible to recreating patterns which create secondary string constraints.

Assume, that two separate blocks are generated, neither of them being one of those six problematic patterns but both of the blocks have an interaction across x_3. Assume also these are not the only blocks being generated with some unknown constraint that constrains the value of x_3. These two hypothetical blocks are below in Table 4.1:

Table 4.1: Hypothetical occurrence inside of a larger problem

Block 1	**Block 2**
$-x_1 \vee -x_2 \vee \ x_3$	$-x_3 \vee \ x_4 \vee \ x_5$
$x_1 \vee \ x_2 \vee -x_3$	$-x_3 \vee \ x_4 \vee -x_5$
$-x_1 \vee \ x_2 \vee -x_3$	$-x_3 \vee -x_4 \vee -x_5$
$-x_1 \vee -x_2 \vee -x_3$	$x_3 \vee -x_4 \vee -x_5$

When looked at jointly, however another strong set of constraints can be observed. Assume that x_3 =0; then the potential solutions only have soft constraints on x_1 and x_2, as well as x_4 and

x_5, this best-case scenario can be seen in equations (4.1).

$$\neg x_3 \implies (-x_1 \vee -x_2) \wedge (-x_4 \vee -x_5) \tag{4.1}$$

Table 4.2: Combined blocks that have unit constraints

Combined Patterns				
x_1	x_2	x_3	x_4	x_5
$-x_1$	$-x_2$	x_3		
x_1	x_2	$-x_3$		
$-x_1$	x_2	$-x_3$	x_4	x_5
$-x_1$	$-x_2$	$-x_3$	x_4	$-x_5$
		$-x_3$	$-x_4$	$-x_5$
		x_3	$-x_4$	$-x_5$

If, however, $x_3 = 1$, then the problem becomes very constrained. This can be seen in the red clauses in Table 4.2, with unit constraints being forced on every single value. The only possible solution for this would be:

$$x_3 \implies (-x_1 \wedge x_2 \wedge x_4 \wedge -x_5) \tag{4.2}$$

This hamstrings the possible solutions and makes the probability of the problem being unsatisfiable exceedingly high.

This demonstrates that, while not initially generating a pattern with a direct unit constraint, these two patterns have managed to impose a strong constraint on the problem. This forces x_3 to be false or forces unit constraints on x_1,x_2,x_4, and x_5. This explains why the difference between problems with the six patterns and problems without the six patterns act so differently with MAX-3SAT solvers when compared to random MAX-3SAT problems.

The ability to generate very strong unit constraints on multiple variables gives insight into why these problems seem to be very sensitive to the total number of variables regardless of the

clause-to-variable ratio. For Reduced MAX3SAT problems, a strong constraint could appear if one variable appears in two separate blocks as shown in Table 4.2 and Table 4.1. Any clause-to-variable ratio where a variable is generated twice has a chance of generating a strong constraint like the one seen in Table 4.2. This by itself won't cause the problem to be unsatisfiable but will make it so that any other block that has an interaction with the offending block will most likely generate a contradiction and cause the problem to be unsatisfiable.

A connection can be drawn here between the Reduced MAX3SAT problems and standard MAX3SAT problems. While the previous phenomenon is theoretically possible for standard Random MAX-3SAT generation to mimic, the statistical probability of this is microscopic; with the probability of simply generating a singular clause with the same 3 variables being $\frac{1}{\binom{n}{3}}$. This doesn't account for this rare statistical occurrence needing to happen at least 4 times to potentially generate a unit constraint. This means that to find this sort of behavior in practice, the size of the MAX-3SAT problem would be massive, beyond the size of anything that could be efficiently searched or solved. In theory, however, this phenomenon would appear as the problem size approached infinity. One question that arises is how tightly tied Standard Random MAX-3SAT is to its phase transition being at a 4.3 clause-to-variable ratio.

Chapter 5

Conclusion

This paper studies the construction and impact of Reduced MAX3SAT instances. It initially explains the construction of Reduced MAX3SAT, as laid out in "Reduction-Based MAX-3SAT with Low Nonlinearity and Lattices Under Recombination" [1]. The paper then explores the impact this generation method for MAX-3SAT problems has on their solvability. This was then further expanded upon with an analysis of the different possible blocks that this generation method creates and the different unit constraints that can occur. The paper also included a comparison of problems where blocks that had a unit constraint implicitly were included and problems where these blocks with unit constraints were removed. It was shown that the inclusion of blocks with unit constraints causes the probability of the problem being unsatisfiable to increase dramatically. It was also shown that removing the blocks with unit constraints caused the problem to perform differently, with the problem remaining solvable with higher clause-to-variable ratios. Finally, the paper looked at the ability of this generation method to create blocks that create unit constraints using blocks that did not initially contain them.

Bibliography

[1] Darrell Whitley et al. Reduction-based max-3sat with low nonlinearity and lattices under recombination. In *Lecture Notes in Computer Science*, pages 113–128. Springer, 2024.

[2] Erik Davis. A look at the 3-sat phase transition. Cadlag Dot Org - A Look at the 3-SAT Phase Transition, April 2019. Accessed: [Access date].

[3] Thomas J. Schaefer. The complexity of satisfiability problems. In *Proceedings of the Tenth Annual ACM Symposium on Theory of Computing*, STOC '78, page 216–226, New York, NY, USA, 1978. Association for Computing Machinery.

[4] S. Kirkpatrick and B. Selman. Critical behavior in the satisfiability of random boolean expressions. *Science*, 264:1297–1301, 1994.

[5] C. Ansótegui, M. L. Bonet, and J. Levy. On the structure of industrial sat instances. In *International Conference on Principles and Practice of Constraint Programming*, pages 127–141. Springer, 2009.

[6] C. Ansótegui, M. L. Bonet, and J. Levy. Towards industrial-like random sat instances. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, volume 9, pages 387–392, 2009.

[7] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms*. The MIT Press, 2022.

[8] A. M. Sutton, L. D. Whitley, and A. E. Howe. A polynomial time computation of the exact correlation structure of k-satisfiability landscapes. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)*, pages 365–372. ACM, 2009.

[9] SATComp Organizing. The international sat competition web page. Accessed: October 5, 2024.

[10] Alexey Ignatiev, Antonio Morgado, and Joao Marques-Silva. Pysat: A python toolkit for prototyping with sat oracles. In *Proceedings of the 21st International Conference on Theory and Applications of Satisfiability Testing (SAT)*, pages 428–437, 2018.