On Phase Transitions in the Approximation Ratio for MAX 2-SAT

Comfort Ohajunwa

Under the direction of

Mitchell Harris
Massachusetts Institute of Technology
Department of Mathematics

Research Science Institute
August 2, 2021

Abstract

In the Boolean Maximum 2-Satisfiability (MAX 2-SAT) problem, we consider a Boolean formula $\mathcal{F}$ in conjunctive normal form with clauses that contain only two literals each, and ask for the maximum number of clauses that can be satisfied in $\mathcal{F}$. MAX 2-SAT is an NP-complete problem, and it has been identified that it undergoes a phase transition where the hardness shifts from underconstrained to overconstrained. The hardness of MAX 2-SAT depends on the clause to variable ratio c, such that MAX 2-SAT undergoes a phase transition when $c = 1$. Meanwhile, there exist approximation algorithms to estimate solutions to MAX 2-SAT in polynomial time using relaxations. One such relaxation involves the use of semidefinite programs (SDP), in which the algorithm optimizes over the cone of positive semidefinite matrices, or matrices with nonnegative eigenvalues. Semidefinite programming was first incorporated into an approximation algorithm for MAX 2-SAT by Goemans and Williamson, which produces approximations such that the exact solution is at least 0.87856 times the SDP upper bound. Yet, despite our knowledge of MAX 2-SAT's behavior over different regions of constrainedness, less can be said about how the ratio between the exact solution and the approximated solution, or the approximation ratio, changes over these regions. Thus, we studied how the approximation ratio changes over c to identify if there exists a phase transition for the approximation ratio. Ultimately, we found that the approximation ratio approaches 1 as MAX 2-SAT becomes more constrained. In addition, when the SDP upper bound is restricted such that it cannot produce solutions greater than the number of clauses in an instance, there is an easy-hard-easy pattern for the approximation ratio. We also found that when the negation of variables in MAX 2-SAT instances are "unbalanced," such that variables are negated with a probability of 32%, the rate of convergence between the exact solution and upper bound is faster and the approximation ratio is greater for larger c.

Summary

The Boolean Maximum 2-Satisfiability (MAX 2-SAT) problem is a computational problem that is NP-complete, meaning it is very difficult to solve. However, it has been found that the hardness of MAX 2-SAT undergoes a phase transition at a specific critical point, such that the problem's hardness shifts from easier to hard. Meanwhile, there are approximation algorithms that allow for quicker and easier estimations through techniques that reduce the problem's constraints, or requirements. One approximation algorithm, in particular, provides an upper bound solution. Though we understand how MAX 2-SAT behaves as the problem becomes harder, little is known about the behavior of the ratio between the exact solution and approximated solution. Hence, in this work, we investigated changes in the ratio between the exact solution and the approximated upper bound solution. Altogether, we found that as MAX 2-SAT becomes harder, the approximation algorithm's upper bounded solution gets closer to the exact solution.

1 Introduction

Consider a Boolean formula $\mathcal{F}$ over n variables with m clauses in conjunctive normal form. This means that each clause contains literals, which are variables or their negation, that are in disjunction with each other. In addition, the clauses are in conjunction with each other, such that all the clauses in $\mathcal{F}$ must be satisfied for $\mathcal{F}$ to be true. Variables in $\mathcal{F}$ can be assigned truth values through a vector $\vec{x} \in \{0,1\}^n$.

The Boolean 2-Satisfiability (2-SAT) problem considers a formula $\mathcal{F}$ with clauses that contain only two literals each and asks if there exists a variable assignment that satisfies $\mathcal{F}$. While 2-SAT is solvable in polynomial time, receiving considerable attention is the optimization version, which is NP-complete [1]. For the Boolean Maximum 2-Satisfiability (MAX 2-SAT) problem, rather than considering whether a formula $\mathcal{F}$ can be satisfied, we ask for the maximum number of clauses that can be satisfied.

Cheeseman et al. [2] have determined that several NP-complete problems undergo phase transitions, which are critical points where the problem's hardness changes. Specifically, phase transitions are defined in terms of control parameters and split the space of problem instances into an underconstrained region, where instances are easier to solve and an overconstrained region, where instances are harder to solve.

The hardness of MAX 2-SAT, in particular, can be thought of in terms of the clause to variable ratio c. It was theoretically proven by Coppersmith et al. [3] that MAX 2-SAT undergoes a phase transition at the critical point where $c = 1$. When $c < 1$, the problem is underconstrained, and it can be expected that nearly all the clauses in a given instance can be satisfied. More precisely, only $\Theta(1/n)$ clauses are unsatisfied. This value increases sharply around $c = 1$, such that when $c > 1$, the number of satisfied clauses approaches $((3/4)c + \Theta(\sqrt{c}))n$. In addition, it was identified, first for 2-SAT by Bollobás et al. [4] and later for MAX 2-SAT by Coppersmith et al. [3], that there is a scaling window around the

phase transition. This scaling window is a small margin around $c = 1$ and is within $1 \pm \Theta(n^{1/3})$ inside of which, it is expected that $\Theta(1)$ of the clauses are unsatisfied. These results were later justified by a empirical study by Shen and Zhang [5].

Meanwhile, many algorithms have been developed for NP-complete problems, including MAX 2-SAT, to approximate solutions in polynomial time. These algorithms use relaxation techniques, which reduce restrictions on the problem and allow for a wider range of potential solutions. One relaxation technique involves the use of semidefinite programming, meaning the algorithm optimizes over the cone of positive semidefinite matrices—matrices with nonnegative eigenvalues. Goemans and Williamson [6], in particular, presented in a groundbreaking paper an algorithm that uses semidefinite relaxations for MAX CUT and MAX 2-SAT. This was the first time that a semidefinite program (SDP) was used for approximation algorithms and it provided a more precise estimation than previous algorithms, such that the exact solution is at least 0.87856 times the estimation. Since then, more improved approximation algorithms have been developed with better estimates. Nonetheless, we still apply Goemans and Williamson [6]'s SDP for our study.

Though we understand how MAX 2-SAT behaves in the underconstrained region, overconstrained region, and scaling window around $c = 1$, little is known about how the ratio between the exact solution and approximated solution behaves over these regions. Therefore, we investigate how the ratio between the exact solution and the SDP upper bound solution of Goemans and Williamson [6]'s algorithm, or the approximation ratio, changes over c. Using code, we randomly generated MAX 2-SAT instances and calculated the average approximation ratio between the exact solution and SDP upper bound for a range of c values. Ultimately, we aim to determine whether the approximation ratio undergoes a phase transition, and if so, where that transition occurs.

As such, the paper is structured as follows. In Section 2, we review preliminaries such as the SDP formulation for Goemans and Williamson [6]'s MAX 2-SAT approximation al-

gorithm. In Section 3, we discuss our methodology and results. In Section 4, we summarize our work, draw conclusions from the results, and present avenues for further study.

2 Preliminaries

MAX 2-SAT

Let $\mathcal{F}$ be a Boolean formula with n variables $\{x_i\}_{i=1}^n$. A Boolean variable x has a value of either true or false. A variable can be negated $\neg x$, so that if x is true, then $\neg x$ is false, and vice versa. Variables in $\mathcal{F}$ are assigned truth values, which can be represented as a vector $\vec{x} \in \{0,1\}^n$, where 0 corresponds to false and 1 corresponds to true. A literal consists of one variable or its negation. Literals can be placed in disjunction with each other to form clauses, such that if at least one of the variables in a clause is true, the entire clause is satisfied, or true. Clauses can be put in conjunction with each other to construct a formula in conjunctive normal form.

Definition 2.1. A Boolean formula $\mathcal{F}$ is in *conjunctive normal form* if it consists of a conjunction of clauses, which are disjunctions of literals.

We assume throughout that our formulas are in conjunctive normal form, which can be seen in Example 2.1.

Example 2.1. *A formula $\mathcal{F}$ in conjunctive normal form could be:*

$$(x_1 \vee x_2) \wedge (\neg x_1 \vee x_2) \wedge (\neg x_1 \vee \neg x_2).$$

In this example, the variables are x_1 and x_2, and the clauses are $(x_1 \vee x_2)$, $(\neg x_1 \vee x_2)$, and $(\neg x_1 \vee \neg x_2)$. For the entire formula to be satisfied, truth values must be assigned to the variables such that all the clauses are satisfied.

Definition 2.2. The k-Satisfiability (k-SAT) problem is a decision problem that considers a

formula $\mathcal{F}$ with clauses that have only k distinct variables, and asks if there exists a variable assignment that satisfies $\mathcal{F}$.

Except when $k = 2$, k-SAT is NP-complete and cannot be solved in polynomial time [1] unless $P = NP$. Furthermore, the optimization version of k-SAT, the Boolean Maximum k-Satisfiability (MAX k-SAT) problem, is also NP-complete and cannot be solved in polynomial time unless $P = NP$.

Definition 2.3. MAX k-SAT is an optimization problem that asks for the maximum number of clauses that can be satisfied in a given formula $\mathcal{F}$. MAX k-SAT is formulated as:

$$\Phi(\mathcal{F}) = \max_{\{x_i\}_{i=1}^n \in \{0,1\}^n} \sum_{C_j \in \mathcal{F}} C_j(x_1, \ldots, x_n) \tag{1}$$

where $C_j(x_1, \ldots, x_n)$ represents the truth value of clause j in $\mathcal{F}$ with variable assignments equal to $\{x_i\}$. Here, we optimize over the sum of the truth values of the clauses in $\mathcal{F}$, constraining x_i for $(i = 1, ..., n)$ to a value of 0 for false or 1 for true.

Semidefinite Programming Formulation

Approximation algorithms aim to estimate solutions to NP-complete problems, such as MAX 2-SAT, through techniques that relax the problem's constraints. Several approximation algorithms use semidefinite programming as defined in [7].

Definition 2.4. A semidefinite program (SDP) is a type of optimization problem that takes the form:

$$\begin{aligned} \max \quad & \mathbf{tr}(CX) \\ \text{s.t.} \quad & \mathbf{tr}(A_i X) = b_i, \quad i = \{0, 1, ..., n\} \\ & X \succcurlyeq 0, \end{aligned} \tag{2}$$

where C, X, and A_i (for $i = 1, ..., n$) are symmetric n by n matrices.

Goemans and Williamson [6] were the first to use SDP for an approximation algorithm for MAX 2-SAT, which we present in this section. Recall that in a given F, we have a set of Boolean variables $\{x_1, ..., x_n\}$. Let every variable x_i (for $i = 1, ..., n$) be associated with a value $y_i \in \{-1, 1\}$. In addition, we introduce a new variable y_0, such that when $y_i = y_0$, x_i is true.

Now we define $v(C)$ for a given Boolean expression C, such that when $v(C) = 1$, the expression is true and when $v(C) = 0$, the expression is false. Here, $v(x_i) = \frac{1+y_0y_i}{2}$ and $v(\neg x_i) = 1 - v(x_i) = \frac{1-y_0y_i}{2}$. So, for an expression $(x_i \vee x_j)$:

$$\begin{aligned} v(x_i \vee x_j) &= 1 - v(\neg x_i \wedge \neg x_j) && (3)\\ &= 1 - v(\neg x_i)v(\neg x_j) && (4)\\ &= 1 - \frac{1 - y_0y_i}{2}\frac{1 - y_0y_i}{2} && (5)\\ &= \frac{1}{4}(3 + y_0y_i + y_0y_j - y_iy_j). && (6) \end{aligned}$$

For other clauses, we still use Equation 6, using $-y_i$ for a negated Boolean variable $\neg x_i$. With this expression, we write an integer program for MAX 2-SAT:

$$\begin{aligned} \max \quad & \sum_{C_j \in \mathcal{F}} v(C_j) \\ \text{s.t.} \quad & y_i \in \{-1, 1\} \quad \forall i \in \{0, 1, ..., n\}. \end{aligned} \tag{7}$$

Here y_i is constrained to only two discrete values: -1 and 1. However, we can relax y_i to be a n-dimensional vector of unit norm u_i. Specifically, u_i lies in a n-dimensional sphere of radius 1, S_n. So, we can replace y_iy_j with the dot product between u_i and u_j, or $u_i \cdot u_j$. Hence, we can rewrite the program Equation (7) as:

$$\begin{aligned} \max \quad & \sum_{i<j} \frac{1}{4}[3 + u_0 \cdot u_i + u_0 \cdot u_j - u_i \cdot u_j] \\ \text{s.t.} \quad & u_i \in S_n \quad \forall i \in \{0, 1, ..., n\}. \end{aligned} \tag{8}$$

Finally, we reformulate Equation (8) as a semidefinite program.

Definition 2.5. A matrix A is *positive semidefinite* if the matrix is symmetric and has nonnegative eigenvalues. For such a matrix, there exists another matrix B, such that A can be decomposed as $B^T B$.

Let Y represent a positive semidefinite matrix, such that $Y_{ij} = u_i \cdot u_j$ and $Y_{ii} = 1$. Matrix Y can be decomposed into $U^T U$, where the columns of U correspond to the vectors $\{u_0, ..., u_n\}$. With this, we can complete our SDP:

$$\Psi(\mathcal{F}) = \begin{aligned} \max \quad & \sum_{i<j} \frac{1}{4}[3 + Y_{0i} + Y_{0j} - Y_{ij}] \\ \text{s.t.} \quad & Y_{ii} = 1 \quad \forall i \in \{0, 1, ..., n\} \\ & Y \succcurlyeq 0. \end{aligned} \tag{9}$$

Solving for the objective function of this program provides us with an upper bound solution for MAX 2-SAT. In particular, Goemans and Williamson [6] proved that the exact solution for MAX 2-SAT is at least 0.87856 times this upper bound.

3 Methodology and Results

For our tests, we created code in Python to generate random 2-SAT formulas and run a complete solver for the exact solutions and the approximation algorithm to derive the upper bound approximation. Specifically, we used the RC2 complete solver from PySAT [8], a toolkit for SAT solvers. For the approximation algorithm, we implemented it in CVXPY [9, 10], a package for modeling convex optimization problems that uses SCS [11, 12], a backend solver for numerical optimization.

We worked with MAX 2-SAT instances with $n = 20$ for c ranging from 0.1 to 38. For each c, we generated 100 random instances; each instance was inputted into both the complete solver and the approximation algorithm to produce the exact solution, $\Phi(\mathcal{F})$, and the SDP upper bound solution, $\Psi(\mathcal{F})$, respectively. Then the approximation ratio, or $\frac{\Phi(\mathcal{F})}{\Psi(\mathcal{F})}$, was calculated. Afterwards, the average approximation ratio for a given c was determined out of

the approximation ratios calculated for the 100 instances.

In Figure 1, we show a plot of the average approximation ratio over c. Notably, the average approximation ratio increases quickly for small c, especially when $c < 2$. Afterwards, it gradually increases and as c gets larger, the average approximation ratio approaches 1. This result indicates that as c increases, the SDP upper bound solution gets closer to the exact solution.

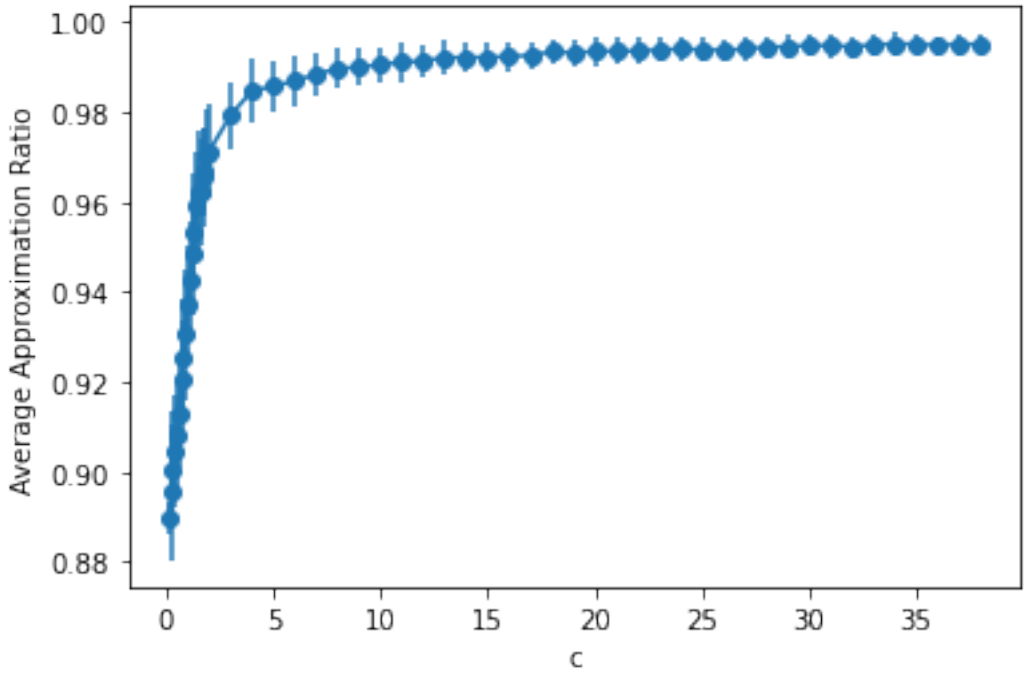

Figure 1: Average approximation ratios for $n = 20$. The error bars indicate the standard deviation of the approximation ratio from the average approximation ratio.

This result is supported by Figure 2, where we plot the average fraction of satisfied clauses out of the total number of clauses as calculated by the complete solver and approximation algorithm over c in Figure 2. Note that as c gets larger, both curves converge towards each other, corresponding to the convergence towards 1 that is seen in Figure 1.

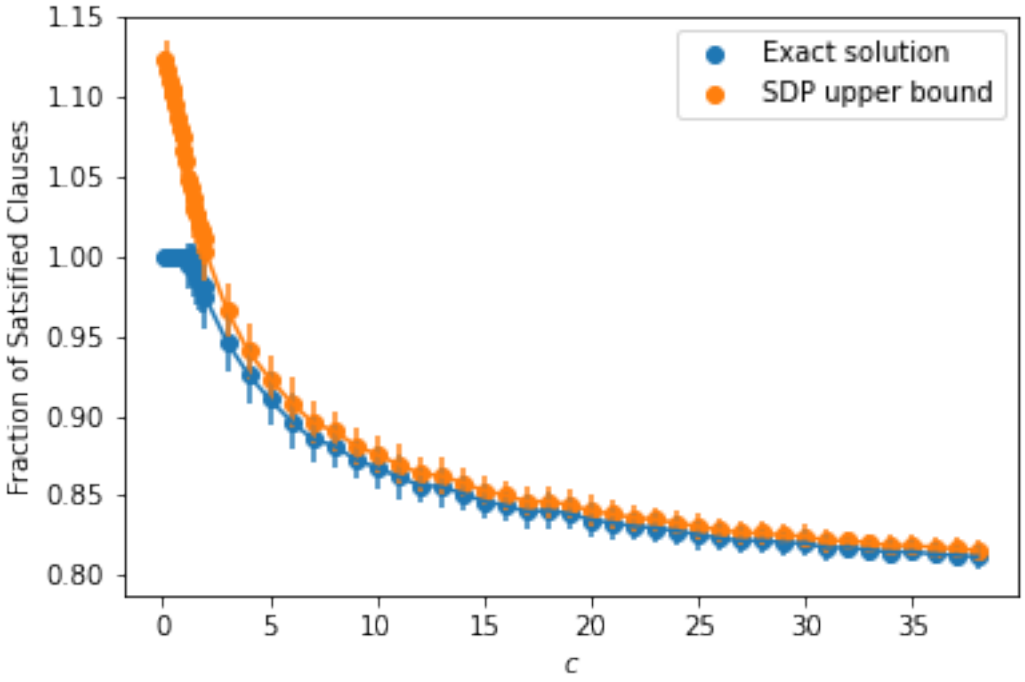

Figure 2: Fraction of satisfied clauses out of the total number of clauses. The blue points correspond to the average fraction of satisfied clauses as determined by the complete solver, while the orange points indicate the approximated average fraction of satisfied clauses estimated by the SDP. The error bars indicate the standard deviation from the average.

Both curves are expected to approach a fraction 0.75 as c increases, based on previous results by Coppersmith et al. [3] and Shen and Zhang [5], which indicated that the expected fraction of satisfied clauses as c approaches infinity is $\frac{3}{4}$. We also prove this in the following theorem.

Theorem 3.1. *The expected number of satisfied clauses approaches $\frac{3}{4}m$ as c gets larger.*

Proof. Recall that $C_j(x_1, \ldots, x_n)$ is the truth value of clause j in $\mathcal{F}$, where $C_j = 1$ when clause j is satisfied and $C_j = 0$, when clause j is not satisfied. We can express the expected

value of a MAX 2-SAT instance as follows:

$$\mathbb{E}[\sum_{j=1}^{m} C_j]. \tag{10}$$

By the linearity of expectation, we can rewrite the above as:

$$\sum_{j=1}^{m} \mathbb{E}[C_j].$$

Suppose we fix the variable assignment for a given $\mathcal{F}$ and randomly add a new clause to $\mathcal{F}$. The variables in the clause has a 50% chance of being negated, so the probability that a literal is true or false for a given variable assignment is $\frac{1}{2}$. As such, the probability that a clause is unsatisfied is equal to the probability that both literals are false, which is $(\frac{1}{2})^2 = \frac{1}{4}$. Thus, for any variable assignment, the probability that a clause is satisfied is $1 - \frac{1}{4} = \frac{3}{4}$.

So, we can express the expectation as:

$$\begin{aligned}
\sum_{j=1}^{m} \mathbb{E}[C_j] &= \sum_{j=1}^{m} (1)\mathcal{P}(C_j = 1) + (0)\mathcal{P}(C_j = 0) \\
&= \sum_{j=1}^{m} \left[(1)\frac{3}{4} + (0)\frac{1}{4} \right] \\
&= \frac{3}{4}m.
\end{aligned}$$

□

Furthermore, Coppersmith et al. [3] reported that the number of satisfied clauses for large values of c is expected to be $((3/4)c + \Theta(\sqrt{c}))n$. As such, the average fraction of satisfied clauses from the complete solver align with the line $0.75 + \frac{k}{\sqrt{c}}$ as in Figure 3.

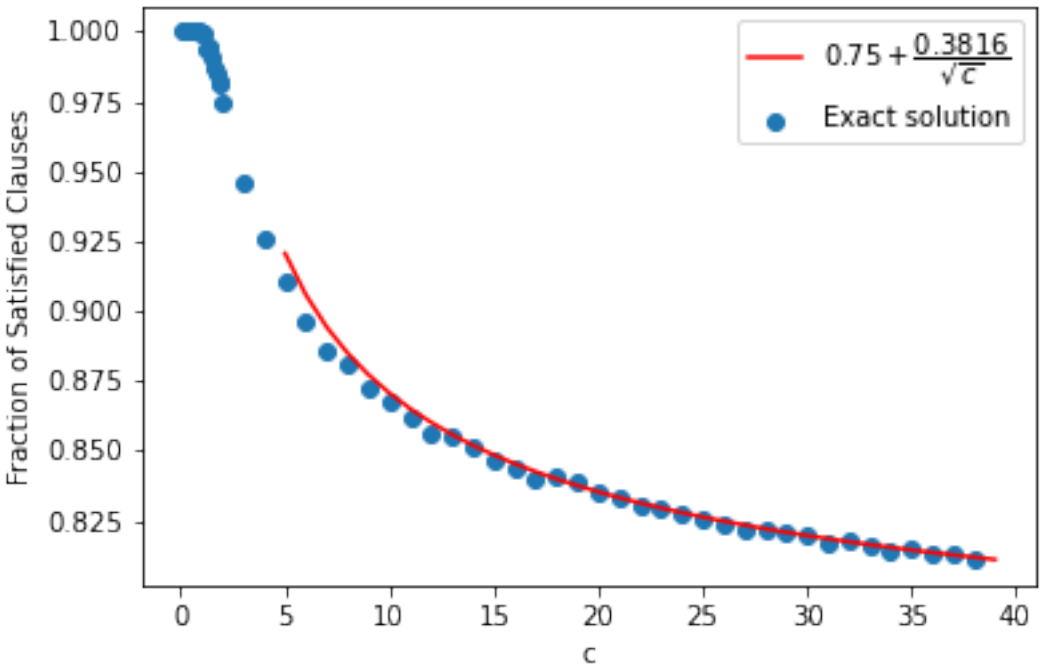

Figure 3: Average fraction of satisfied clauses out of the total number of clauses as computed by the complete solver. The blue points, which indicate the computed average, fit to the red line corresponding to the graph $y = 0.75 + \frac{k}{\sqrt{c}}$, where $k = 0.3816$.

Though the approximation algorithm provides an upper bound for the exact solution, it appears to overestimate more for small c, especially $c < 2$. Up to around $c = 2$, the approximation algorithm frequently estimates a higher number of satisfied clauses than there are clauses themselves in the formula. This can be seen in the following example.

Example 3.1. *Consider an instance in which $n = 2$, $m = 1$, and the formula $\mathcal{F}$ is $(x_1 \vee x_2)$. For this formula, the optimization problem becomes:*

$$\Psi(x_1 \vee x_2) = \begin{array}{ll} \max & \frac{1}{4}[3 + Y_{0,1} + Y_{0,2} - Y_{1,1}] \\ s.t. & Y_{ii} = 1 \quad \forall i \in \{0, 1, ..., n\} \\ & Y \succcurlyeq 0 \end{array} \tag{11}$$

Solving for this optimization problem produces the following 3 by 3 matrix:

$$Y = \begin{bmatrix} 1 & 0.5 & 0.5 \\ 0.5 & 1 & -0.5 \\ 0.5 & -0.5 & 1 \end{bmatrix}.$$

From this matrix, one can determine that the objective value is $\frac{1}{4}[3 + 0.5 + 0.5 - (-0.5)] = 1.125$*, which is greater than* $m = 1$*.*

We also tested MAX 2-SAT instances with $n = 25$ variables for a smaller range of c values between 0.1 and 10. We used a smaller range due to computational speed. Notably, despite having a larger number of variables, the average approximation ratio is roughly the same over c as seen in Figure 4.

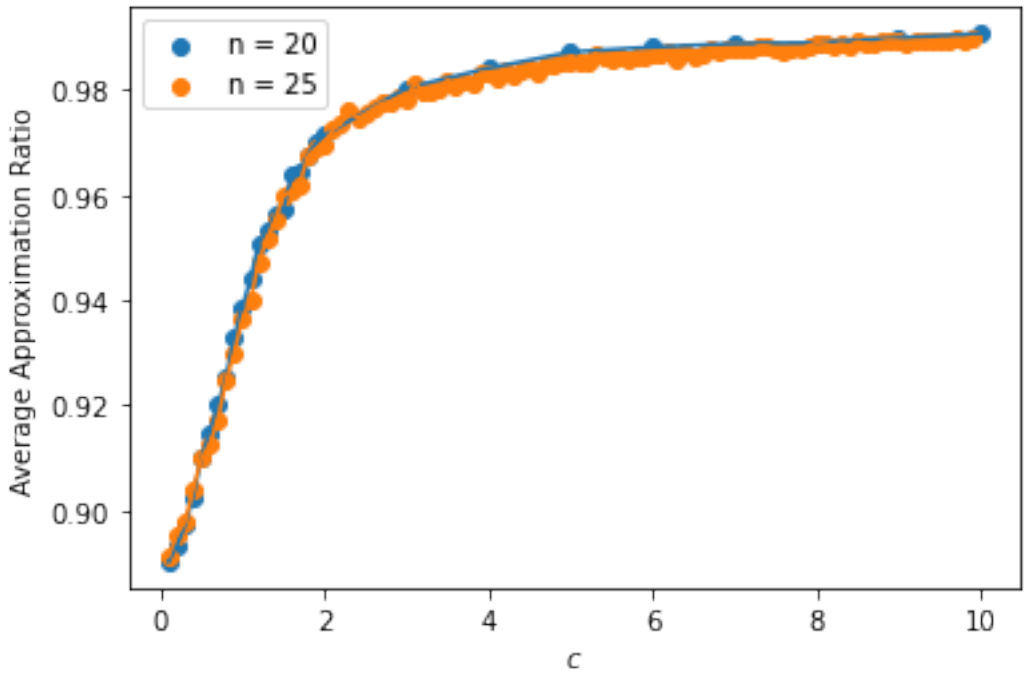

Figure 4: Average approximation ratio for $n = 20$ variables and $n = 25$ variables. MAX 2-SAT instances generated from $n = 20$ variables are indicated in blue, while instances from $n = 25$ are indicated in orange.

Furthermore, we ran tests on MAX 2-SAT such that the maximum SDP upper bound

was limited to the number of clauses in the generated formulas. In other words, if the SDP upper bound solution for a generated MAX 2-SAT formula is greater than the number of clauses in the formula, we set the solution to be equal to the number of clauses in the formula instead. We ran these tests for c values between 0.1 and 38.

Notice that the approximation ratio in Figure 5 initially decreases around $c = 1$, in contrast to the initial increase seen in Figure 1. This decrease continues until around $c = 3$, from which the average approximation ratio increases and steadily approaches 1. This could indicate an easy-hard-easy pattern for the approximation ratio. Where the average approximation ratio is closer to 1, the SDP performs better as it produces solutions that are closer to exact solution. On the other hand, where the approximation ratio is lower, such as around $c = 3$, the SDP performs comparably worse. The approximation ratio also increases more gradually for $c > 3$, compared to the rate of increase found in Figure 1.

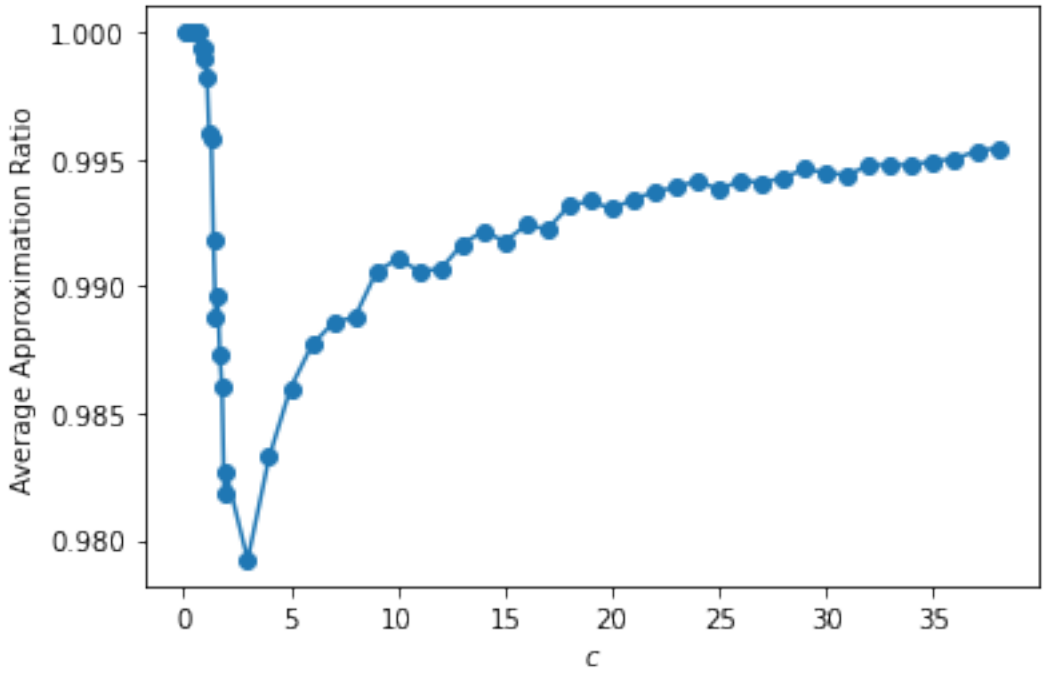

Figure 5: Average approximation ratios for $n = 20$ when the SDP upper bound is restricted such that estimations cannot exceed the number of clauses in an instance.

In Figure 6, we also show the fraction of satisfied clauses for the exact solution and SDP upper bound when the SDP is restricted. Note that for $c < 2$, the SDP upper bound solution is limited to at most 1, since it can no longer exceed the number of clauses in $\mathcal{F}$. Thus, the SDP upper bound matches the exact solution for small c until the exact solution begins to decrease around $c = 1$, where MAX 2-SAT undergoes a phase transition, corresponding to the decrease in the average approximation ratio that was found in Figure 5. However, both curves in Figure 6 do converge as c increases, particularly after $c = 3$.

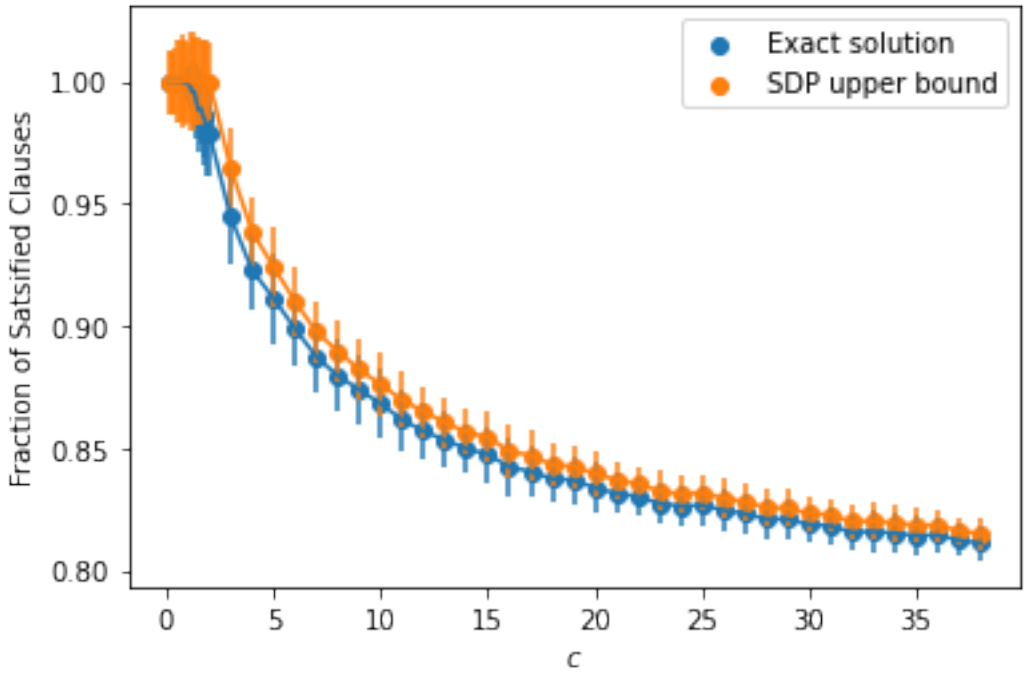

Figure 6: Fraction of satisfied clauses out of the total number of clauses, when the SDP upper bound is restricted such that the bound cannot exceed the number of clauses in an instance. The blue points correspond to the average fraction of satisfied clauses as determined by the complete solver, while the orange points indicate the average fraction of satisfied clauses in the SDP estimate. The error bars indicate the standard deviation from the average.

Now, so far, we have produced formulas such that any given literal is the negation of a

variable with a probability of 50%. We call these MAX 2-SAT instances "balanced," since variables can be negated or not negated with equal probability. However, Austrin [13] found that if this probability is decreased to 32%, it is harder to approximate MAX 2-SAT instances. These instances are "unbalanced." So, we studied the approximation ratio and fraction of satisfied clauses for unbalanced MAX 2-SAT compared to balanced MAX 2-SAT.

In Figure 7, we show the average approximation ratio from balanced and unbalanced MAX 2-SAT instances. Both balanced and unbalanced MAX 2-SAT have similar trends, with their average approximation ratios approaching 1 for larger values of c. However, as c gets larger, the average approximation ratio for unbalanced MAX 2-SAT becomes larger than that for balanced MAX 2-SAT.

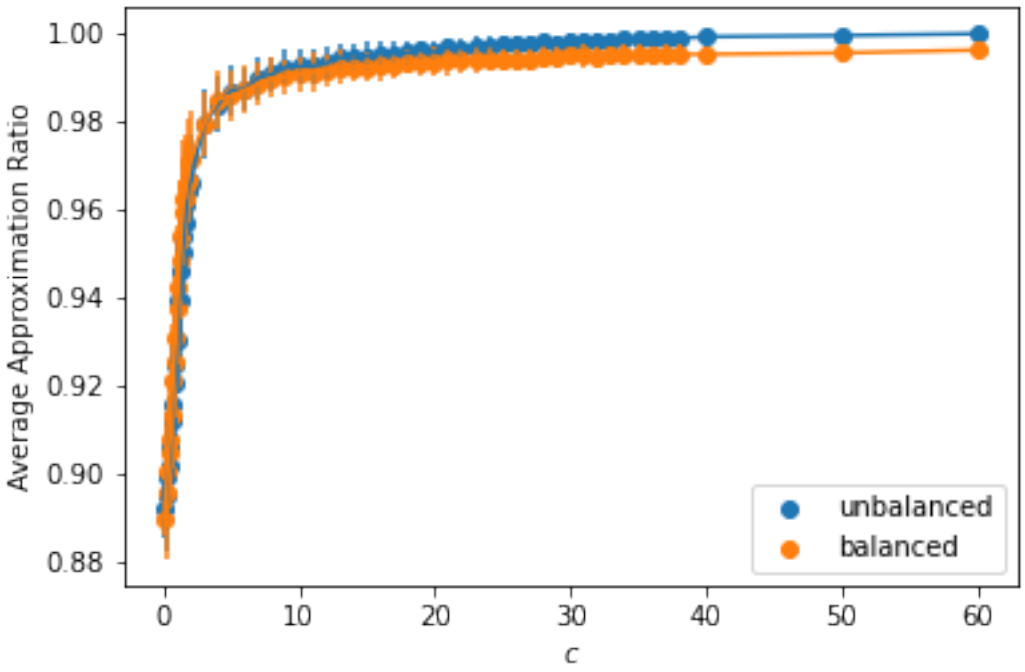

Figure 7: Average approximation ratios of balanced and unbalanced MAX 2-SAT for $n = 20$. The orange points correspond to the approximation ratios for balanced MAX 2-SAT, while the blue points correspond to the approximation ratios for unbalanced MAX 2-SAT. The error bars indicate the standard deviation from the average approximation ratio.

Additionally, there are similar trends between balanced and unbalanced MAX 2-SAT with regards to the fraction of clauses that are satisfied, which are depicted in Figure 8. However, in unbalanced MAX 2-SAT, a larger fraction of clauses are satisfied. In fact, unbalanced MAX 2-SAT appears to have a faster rate of convergence than balanced MAX 2-SAT, which corresponds to the higher average approximation ratios found in Figure 7.

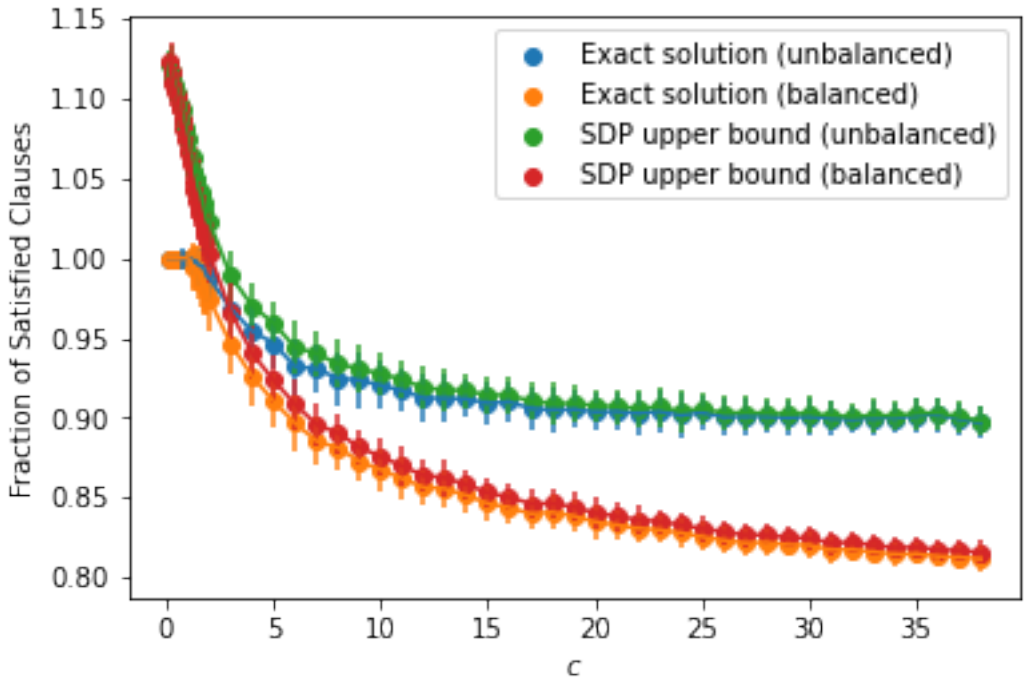

Figure 8: Fraction of satisfied clauses out of the total number of clauses for balanced and unbalanced MAX 2-SAT. The orange and blue points correspond to the average fraction of satisfied clauses as determined by the complete solver for balanced and unbalanced MAX 2-SAT, respectively. The red and green points correspond to the average fraction of satisfied clauses from the approximation algorithm for balanced and unbalanced MAX 2-SAT, respectively. The error bars indicate the standard deviation from the average.

In addition, we ran tests with a larger number of MAX 2-SAT instances, 1000, for c between 0.1 and 10. We used a smaller range of c values due to computational speed. The results seen in Figure 9 are consistent with the results found in Figure 7 for small c, though the variation in the average approximation ratio is smaller as indicated by the shorter error bars.

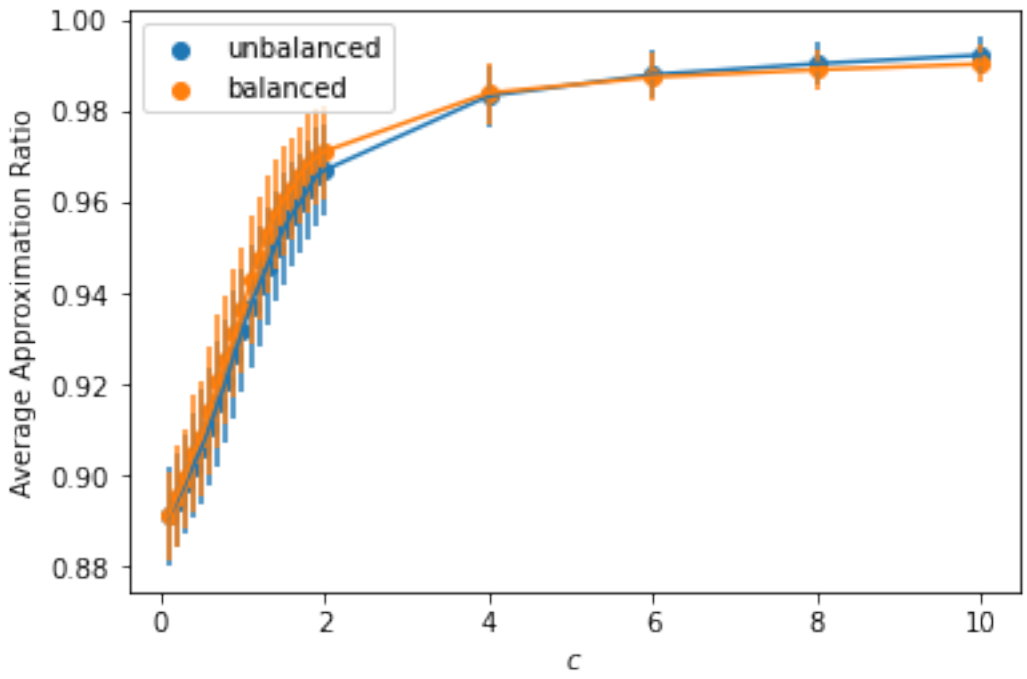

Figure 9: Average approximation ratios of balanced and unbalanced MAX 2-SAT for $n = 20$ for c between 0.1 and 10, using 1000 random instances for each c. The orange points correspond to the approximation ratios for balanced MAX 2-SAT, while the blue points correspond to the approximation ratios for unbalanced MAX 2-SAT. The error bars indicate the standard deviation from the average approximation ratio.

4 Conclusion

In this paper, we explored how Goemans and Williamson [6]'s algorithm perform over MAX 2-SAT's regions of constrainedness through studying the approximation ratio over a range of values for c. We investigated both balanced MAX 2-SAT, in which variables are negated with a probability of 50% and unbalanced MAX 2-SAT, in which variables are negated with a probability 32%. We also considered the approximation ratio when the SDP upper bound is restricted such that it cannot exceed the number of clauses in an instance.

We found that for both balanced and unbalanced MAX 2-SAT, as c increases, the approximation ratio approaches 1, which is depicted in Figure 7. This indicates that the that the SDP upper bound converges towards the exact solution as MAX 2-SAT becomes more constrained as seen in Figures 2 and 8. Additionally, we found that balanced and unbalanced MAX 2-SAT have different rates of convergence with regard to the fraction of clauses that are satisfied as c gets larger. We report that the fraction of clauses that are satisfied in balanced MAX 2-SAT approaches $\frac{3}{4}$ as c gets larger for both the exact solution and SDP upper bound, confirming previous results [3, 5]. Moreover, given that the SDP upper bound and exact solution for unbalanced MAX 2-SAT converges faster than balanced MAX 2-SAT, unbalanced MAX 2-SAT instances have a larger average approximation ratio than balanced MAX 2-SAT as c increases. We also identified an easy-hard-easy pattern in the approximation ratio when we restricted the SDP upper bound, which is seen in Figure 5.

In the future, we would like to explore our results further in order to understand the mechanisms that drive them. In particular, we want to investigate the rate at which the approximation ratio changes, especially the steep increase for small c that is seen in Figure 7. Additionally, we would like to investigate the rate of convergence of the fraction of satisfied clauses between the exact solution and SDP upper bound. In particular, we want to understand the difference between the convergence between the exact solution and SDP upper bound of balanced and unbalanced MAX 2-SAT as c increases. Furthermore, we would like to study the easy-hard-easy pattern identified when the SDP upper bound was restricted and see if this could indicate a phase transition for the approximation ratio.

5 Acknowledgments

Firstly, I would like to express my gratitude towards Mitchell Harris for his guidance and mentorship for this project. In addition, I would like to thank my tutor Dr. Rickert

and the head mentor Dr. Tanya Khovanova for providing helpful advice and suggestions throughout the writing process of this paper. I would also like to acknowledge the readers of my penultimate draft Yunseo Choi, James Skelley, and Natalia Pacheco for giving additional feedback to enhance this paper. Additionally, I would like to thank Prof. David Jerison and Prof. Ankur Moitra for arranging the math mentorships as well as Prof. Pablo Parrilo for providing feedback and advice for the direction of this research. Furthermore, I would like to thank the Research Science Institute, the Center for Excellence in Education, and the Massachusetts Institute for Technology for providing me opportunity to participate in this research. Finally, I am grateful for the sponsorship of the Department of Defense, especially the Director of STEM in the Defense Laboratories and Personnel Mr. Louie Lopez.

References

[1] V. V. Vazirani. *Approximation algorithms*. Springer Science & Business Media, 2013.

[2] P. C. Cheeseman, B. Kanefsky, W. M. Taylor, et al. Where the really hard problems are. In *IJCAI*, volume 91, pages 331–337, 1991.

[3] D. Coppersmith, D. Gamarnik, M. Hajiaghayi, and G. B. Sorkin. Random max sat, random max cut, and their phase transitions. *Random Structures & Algorithms*, 24(4):502–545, 2004.

[4] B. Bollobás, C. Borgs, J. T. Chayes, J. H. Kim, and D. B. Wilson. The scaling window of the 2-sat transition. *Random Structures & Algorithms*, 18(3):201–256, 2001.

[5] H. Shen and H. Zhag. An empirical study of max-2-sat phase transitions. *Electronic Notes in Discrete Mathematics*, 16:80–92, 2003.

[6] M. X. Goemans and D. P. Williamson. . 879-approximation algorithms for max cut and max 2sat. In *Proceedings of the twenty-sixth annual ACM symposium on Theory of computing*, pages 422–431, 1994.

[7] S. Boyd, S. P. Boyd, and L. Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

[8] A. Ignatiev, A. Morgado, and J. Marques-Silva. PySAT: A Python toolkit for prototyping with SAT oracles. In *SAT*, pages 428–437, 2018.

[9] A. Agrawal, R. Verschueren, S. Diamond, and S. Boyd. A rewriting system for convex optimization problems. *Journal of Control and Decision*, 5(1):42–60, 2018.

[10] S. Diamond and S. Boyd. CVXPY: A Python-embedded modeling language for convex optimization. *Journal of Machine Learning Research*, 17(83):1–5, 2016.

[11] B. O'Donoghue, E. Chu, N. Parikh, and S. Boyd. Conic optimization via operator splitting and homogeneous self-dual embedding. *Journal of Optimization Theory and Applications*, 169(3):1042–1068, June 2016.

[12] B. O'Donoghue, E. Chu, N. Parikh, and S. Boyd. SCS: Splitting conic solver, version 2.1.4. `https://github.com/cvxgrp/scs`, Nov. 2019.

[13] P. Austrin. Balanced max 2-sat might not be the hardest. In *Proceedings of the thirty-ninth annual ACM symposium on Theory of computing*, pages 189–197, 2007.