CS4244: Knowledge Representation and Reasoning **Spring 2020**

CS4244 Project: Stage 2

Lecturer: Kuldeep S. Meel *Students: Darren Ong Yun Kai, Nguyen Duy Son*

1 Introduction

In stage 1 of the project, a Boolean Satisfiability Problem (SAT) Solver using Conflict Driven Clause Learning (CDCL) algorithm has been implemented and analysed. This SAT solver has been used to solve various Conjunctive Normal Form (CNF) formulas. These CNF formulas can either be random generated or generated with a purpose such as Einstein's puzzle.

So, in this stage 2, we want to understand the behaviour of random CNF formulas. A set of CNF formulas is randomly generated with variables involved in the generation to analyse its behaviour. These variables are number of literals in a clause, k, and number of clauses which is decided by a non-negative real number, r. Each formulas are generated with n distinct variables, which is 150 in this project.

2 Random CNF Generation

A **RandomCNFGenerator.java** is created to generate a CNF Formulas provided with value k, n and r. Each CNF formulas consist of $\lceil rn \rceil$ k-clauses where r is a non-negative integer, $r \in (0, 10)$ with an incremental of 0.2, and k is an integer, $k \in [3, 5]$. Each clause has k distinct variables and each variable can be either true or false. A set of 50 formulas are generated for each r and k for a large enough sampling. The generations of the sets of 50 formulas is generated with **AutomatedCNFGeneratorRunner.java**. This has helped to ease the process of generating **2450 Formulas** for each k, which is **7350 formulas** in total (Woah! That's a lot.).

3 Project Finding

After the generation of the CNFs are done, the CNFs are being executed by a SAT Solver. This SAT Solver is an open source library which is found online, namely sat4j. This SAT Solver is used mainly due to its performances is better than the SAT Solver built in Stage 1. During the execution of the SAT Solver, the execution time and the probability of a satisfiable formula occurred over 50 CNF formulas. The data needed has been collected using **CNFAnalyzer.java**. Due to the large sets of CNF formulas creates, it took one night, roughly 6 hours, to analyse. Moreover, each set of data collected is printed into a csv file which is located in the stats folder. The data collated is restructured and visualised in the form of bar graphs which is shown below.

3.1 Probability of Satisfiable

The probability of satisfiable formula is collected for each k-CNF formulas. The graphs visualised below shows the probability of SAT, UNSAT and ERROR of a CNF formulas for each r.

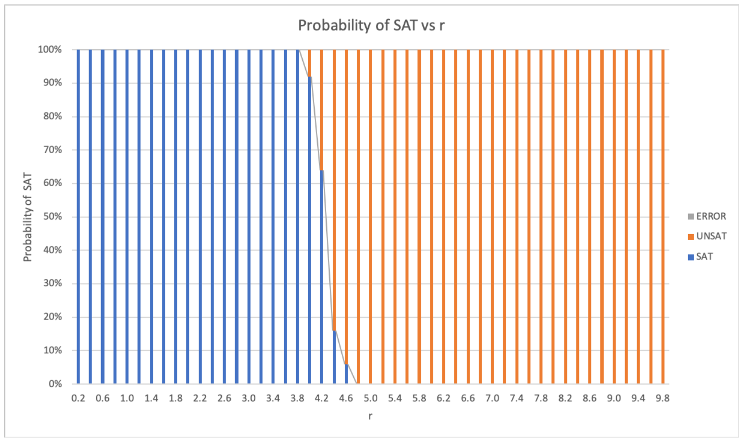

Figure 1: Probability of Satisfiable of 3-CNF Formulas vs r

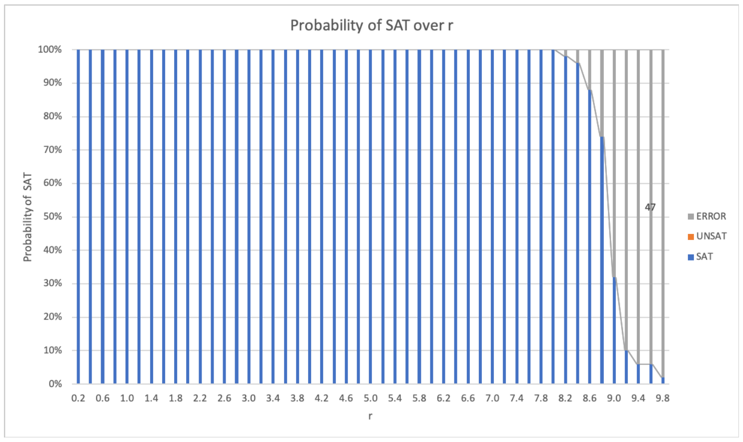

Figure 2: Probability of Satisfiable of 4-CNF Formulas vs r

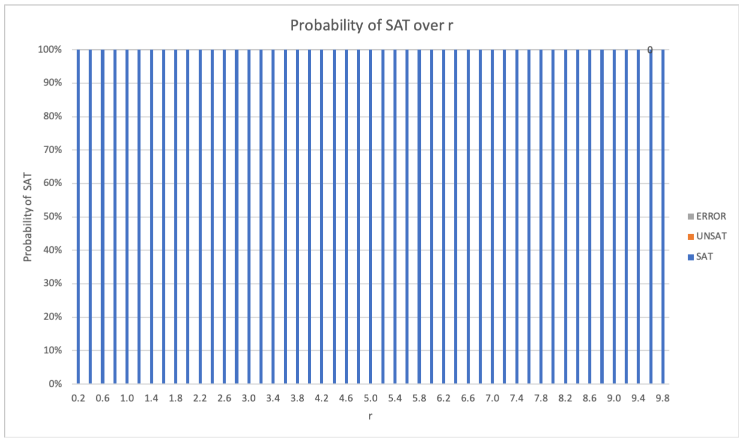

Figure 3: Probability of Satisfiable of 5-CNF Formulas vs r

3.2 Average Execution Time

The average execution time needed to find whether a formula is satisfiable is recorded for each k-CNF formulas. The graphs visualised below shows the average execution time needed of the 50 formulas for each r. However, the execution time for the error formulas is not collected and computed. This may lead to some slight inaccuracy in the data

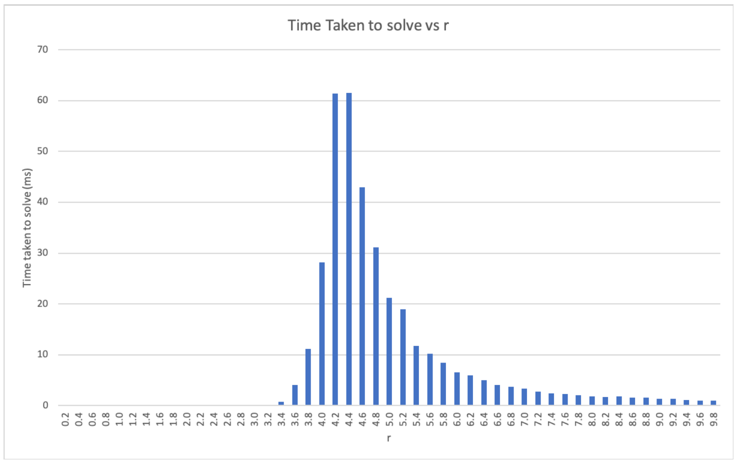

Figure 4: Average execution time of 3-CNF Formulas vs r

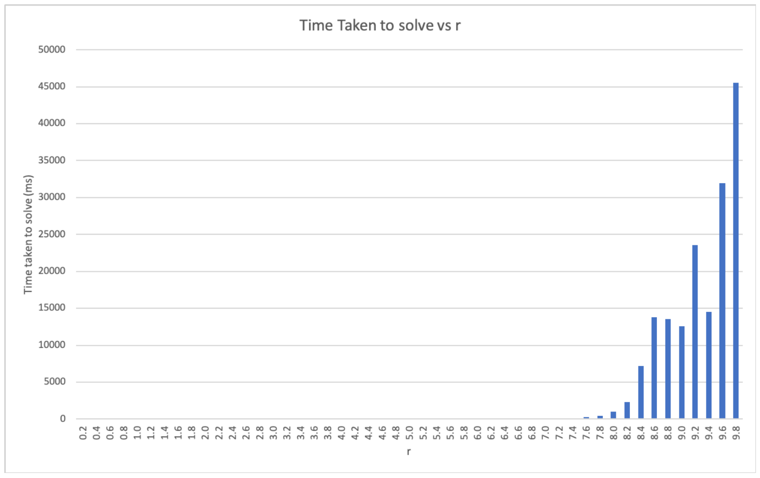

Figure 5: Average execution time of 4-CNF Formulas vs r

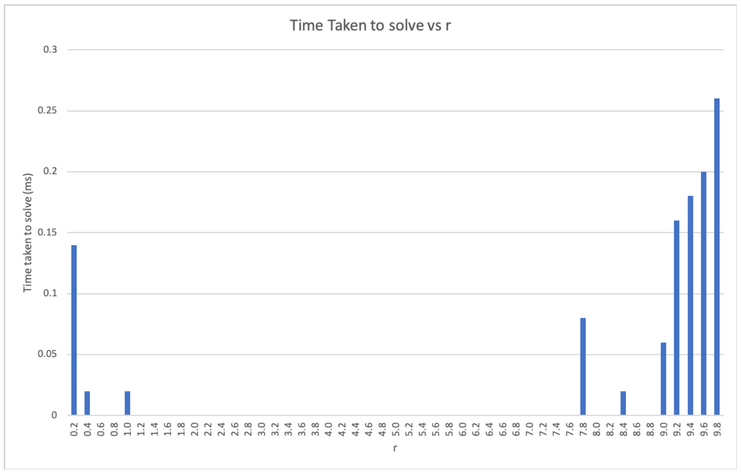

Figure 6: Average execution time of 5-CNF Formulas vs r

4 Analysis

Based on the visualisation above, we could make a few observation and some hypothesis on why such behaviour happens.

1. **3-CNF Formulas**

   Based on Figure 1 above, 3-CNF Formulas starts to have some probability of getting UNSAT Formulas when r = 4.0. The probability of SAT of 3-CNF Formulas drops drastically from 100%, $r = 4.0$ to 0%, $r = 4.8$. The average execution time spikes up starting from approximately 0 ms, $r = 3.8$ to approximately 60 ms, $r = 4.4$. Then, the average execution time taken decreases exponentially from approximately 60 ms, $r = 4.4$ to approximately 0 ms. This is a possible indications that the complexity of the formulas drops significantly.

2. **4-CNF Formulas**

   On the other hand, 4-CNF Formulas has some error produced while being solved when r = 8.2. The error is most likely occured due to timeout. The timeout could be an indication of having issue with finding a satisfiable assignment which is likely to be an UNSAT formula. However, the pattern of the changes in probability of SAT formulas is very similar to 3-CNF Formulas. Furthermore, we could see that the average execution time needed for 4-CNF Formulas has spiked up to approximately 45000 ms (45 s) when $r = 9.8$. This could be an indicator for the clause when r increases and approaching to approximately 9.0. The execution

3. **5-CNF Formulas**

   Finally, all of the 5-CNF Formulas generated are satisfiable and the execution time needed is very few where on average, each formulas is solved within 1ms. However, a clear observation is made that the time taken solve a formula is when $r = 9.0$. This could be a possibility of the formula is getting complicated to find a satisfiable assignment.

Based on the observations made above, we can conclude that there might be an indication on the increasing number of clauses also rises the chance of getting an UNSAT clause when the k is fixed. This may caused by various reasons like increasing in the complexity of the formulas, more clauses needed to be satisfied and other reasons. However, we could not accept the reason on increasing of the complexity of a formula also increases its chance to be unsatisfied. The spiking up of exectuion time eventually goes down when there are more clauses based on Figure 4 indicates that it will not be complex to find a UNSAT clause when the number of clauses gets large enough.

Moreover, we also observed that the probability of a formula is satisifiable is higher when k is larger and r is fixed. This could probably be due to the fact that having more literal in a clause is easier to be satisfied. For example, in a 3-CNF clause you need 1/3 of the literal to be satisfied and the odds of getting any 3 of them to be satisfied is k/n, in this case is $3/150 = 2\%$. On the other hand, in a 5-CNF clause you need 1/5 of the literal to be satisfied and the odds of getting any 5 of them to be satisfied is k/n, in this case is $5/150 = 3.33\%$.

Furthermore, we also realised that the maximum average execution time for 4-CNF formulas is a lot higher that 3-CNFs formulas. Although, the maximum average execution time for 5-CNF formulas is barely significant, which is approximately 0.25 ms, but we could not ignore the fact that the execution time has a signal of spiking up. The maximum average execution time may increase further and higher than the maximum of 4-CNF formulas.

5 Conclusion

With this project, we understand how each different k and r value affects the CNF Fomulas in terms of execution time needed and probability of Satisfiable formulas. The probability of satisfiable formulas is increased when k increases and r is fixed. Moreover, the probability of satisfiable formulas is also increased when r increases and k is fixed.

The behaviour of different CNF formulas with unique k and r value has showed us why every form of CNF formulas has to be reduced to 3-CNF formulas if possible. 3-CNF formulas could be the most simplified and has the shortest execution time needed for solving a SAT formula or UNSAT formula.

Finally, we hope that by learning this knowledge throughout the project can give us some fundamental basics on working some other hard problems.

We grade our project as grade A-.