Lecturer: Kuldeep Meel (Sem 2 AY20/21)

CS4244 Project Stage 2
Behavior of Random CNF Formulas

Evelyn Yi-Wen Chen (A0184698E), Yang Zi Yun (A0184682U)
{e0313687, e0313671}@u.nus.edu

1. Project Statement

To conduct an empirical study followed by an analysis of the behavior of the satisfiability of random variable $F_k(n, rn)$.

$F_k(n, rn)$is a random k-CNF formula on n variables and $L = \lceil rn \rceil$ clauses. Each clause is chosen uniformly and independently from all $(nC_k)2^k$ possible k-clauses over n variables.

2. Experiment Setup

We generated the random formulas with provided k, n and r values. We have the integer n fixed at the value of 150, corresponding to the number of variables. The integer k is chosen within the range of [3,5], which stands for 3-CNF, 4-CNF and 5-CNF. r is a non-negative real number in the range of (0, 10), incremented at intervals of 0.2.

For every fixed k and r, we've sampled 50 formulas and used CryptoMiniSat SAT Solver[1] to solve the formulas generated. We generated 2450 (49*50) CNF formulas for each k, and in total 7350 formulas.

The output of the SAT Solver was written into the csv file, including the time taken to solve the formula, and whether the result was satisfiable, unsatisfiable or timeout (if more than 10 minutes). After which we calculated the probability of satisfiability of $F_k(n, rn)$for the fixed k, n, r values. Lastly, for every k, we've plotted a graph of satisfiability of $F_k(n, rn)$vs. r. Out of curiosity, we've also plotted a curve for the average time taken to solve the CNF formulas.

3. Results

3.1. Probability of SAT

The probability of formulas being satisfiable has been collected for each 3-CNF, 4-CNF and 5-CNF; where the horizontal axis is the value of r, and the vertical axis is the probability of SAT.

[1] https://github.com/msoos/cryptominisat

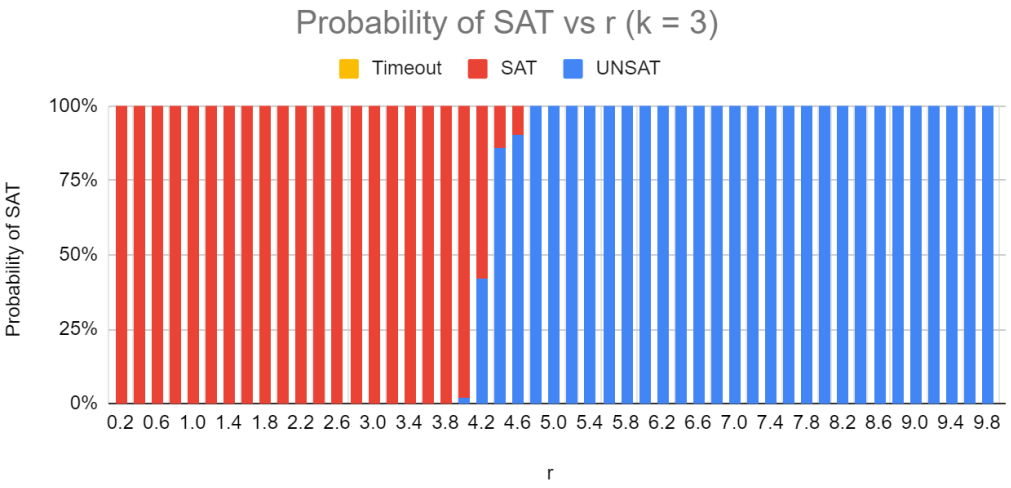

Figure 1. Probability of SAT of 3-CNF formulae vs r

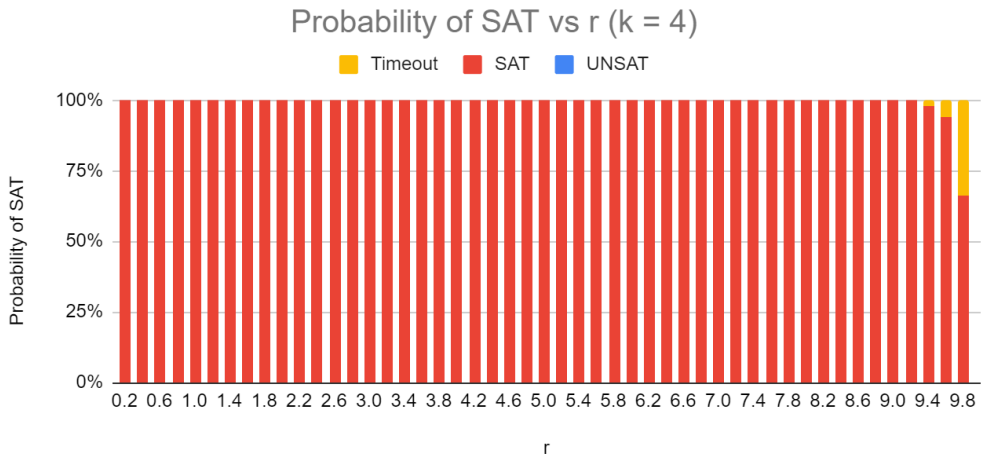

Figure 2. Probability of SAT of 4-CNF formulae vs r

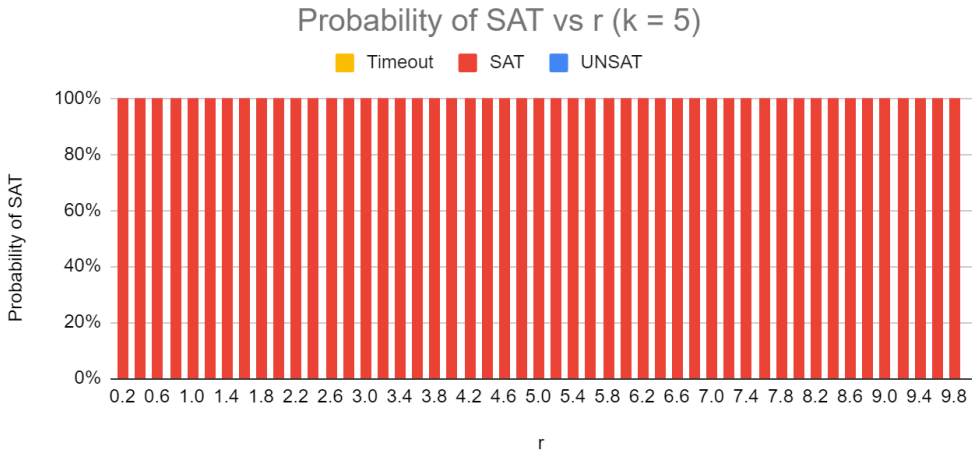

Figure 3. Probability of SAT of 5-CNF formulae vs r

3.2. Average time taken

The average time taken to solve the 50 formulas have been collected for each 3-CNF, 4-CNF and 5-CNF, where the horizontal axis is the value of r, and the vertical axis is the time taken. It is important to note that the timeouts in solving 4-CNF are excluded in the calculation of average time.

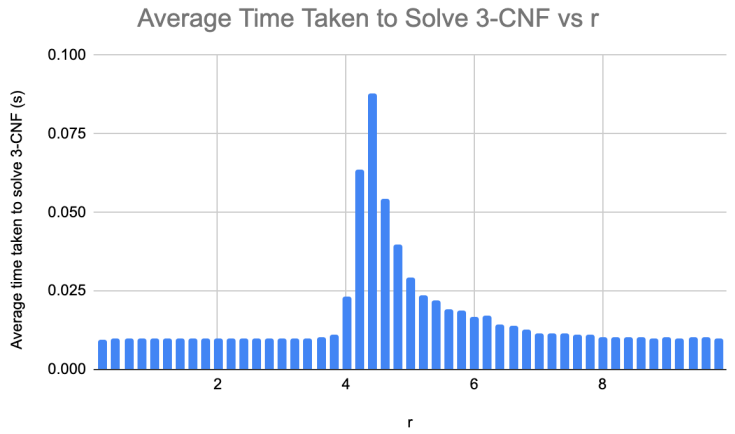

Figure 4. Average Time Taken to Solve 3-CNF formulae vs r

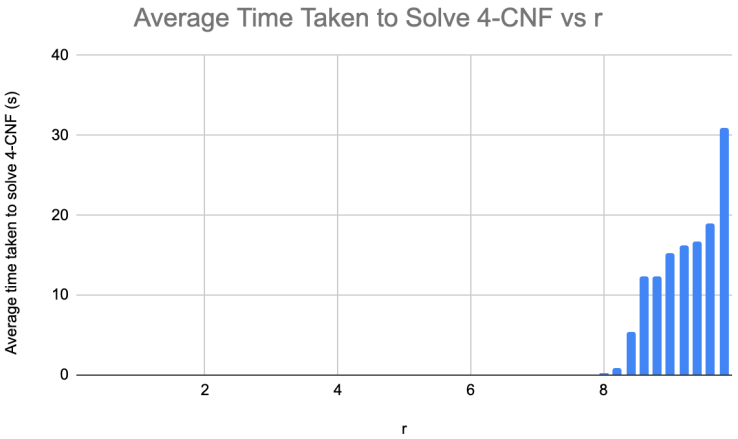

Figure 5. Average Time Taken to Solve 4-CNF formulae vs r

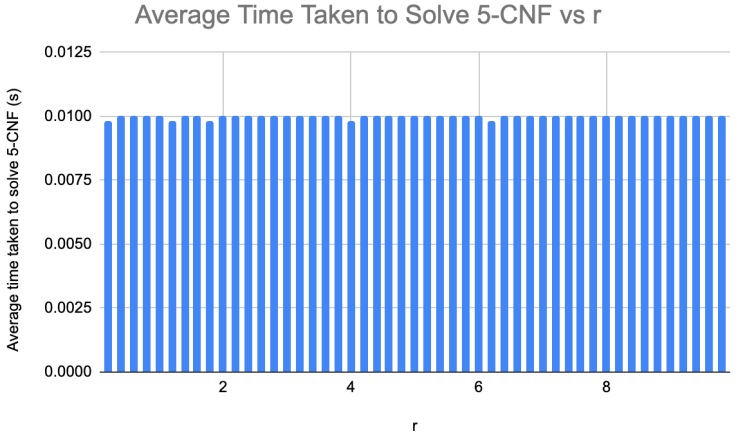

Figure 6. Average Time Taken to Solve 5-CNF formulae vs r

We've also plotted the combined graph of k in the range of [3,5] for both probability of SAT and the average time taken to solve. Unfortunately, the lines overlap quite a bit for the following graphs.

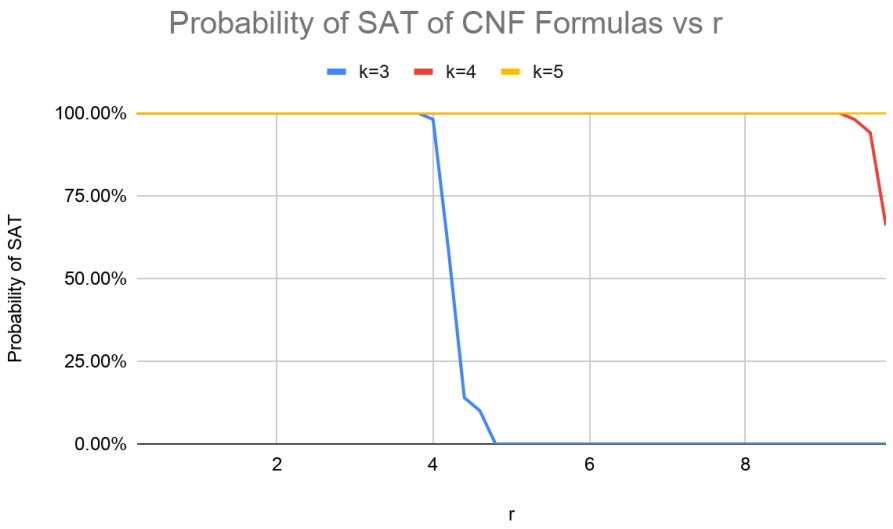

Figure 7. Probability of SAT of CNF formulae vs r

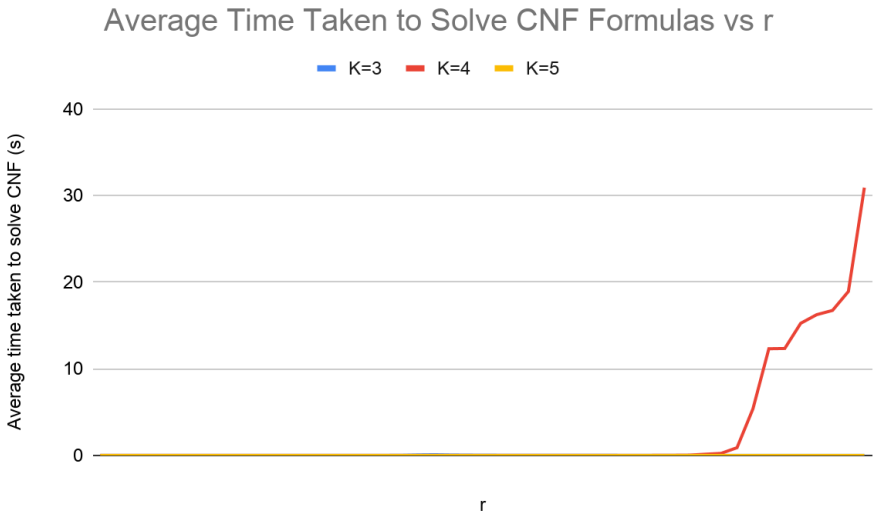

Figure 8. Average Time Taken to Solve CNF formulae vs r

4. Analysis

We observed the above figures, and made some observations, hoping to explain the behaviors from a theoretic point of view.

3-CNF

The probability of UNSAT began to increase from 0% at r =4.0, and reached 100% at r=4.8. This matches with the general behavior (N=[12,50]) from Critical Behavior in the Satisfiability of Random Boolean Expressions paper, written by Scott Kirkpatrick and Bart Selman.[2] We estimated that when r≈4.3, the probability of SAT and UNSAT is around 50%.

During this range of r values, the time taken to solve the formulas also peaked, indicating that it might be harder to solve for formulas with r in the range of [4.0, 5.0].

4-CNF

Unlike 3-CNF, there weren't any UNSAT data points. This result also matches approximately with the results from the paper mentioned above.

However, TIMEOUT (exceeding 10 minutes) began to occur when r=9.2. Out of interest, we fed the formulae that caused "TIMEOUT" to an online DPLL solver[3], and the DPLL solver was able to solve it immediately, giving a satisfying assignment.

We suspect that the random formula generated for r > 9.2 might have caught the worst case behavior or adversarial example of the CryptoMiniSAT solver, thus resulting in TIMEOUT.

Additionally, the time taken to solve the formulas began to increase dramatically starting at r = 8.0, from less than 1 seconds up to 30 seconds and above, indicating that it's harder to solve for these formulas.

Interestingly, as shown in Figure 8, 4-CNF takes the longest average time to solve.

5-CNF

Surprisingly, all the 5-CNF formulas we've tested are all satisfiable, and have a constantly small solving time of 0.01s on average. We noted from the Critical Behavior in the Satisfiability of Random Boolean Expressions paper that the same behavior we've seen in 3-CNF probability of UNSAT will occur at approximately when r = 20.

---

[2] Scott Kirkpatrick and Bart Selman, Critical Behavior in the Satisfiability of Random Boolean Expressions, (1994) *Science*, New Series, Vol. 264, No. 5163 (May 27, 1994), pp. 1297- 1301

[3] https://www.inf.ufpr.br/dpasqualin/d3-dpll/

5. Discussion

Since n=150 is fixed throughout the experiments, we mainly investigate the impact of different k and r values on the probability of SAT and the execution time.

Case 1: k is fixed

From the above observations and supporting evidence from the mentioned paper, we can conclude that when k is fixed, the chances of getting SAT decreases as the number of clauses increases. Intuitively speaking, when r increases, there are more constraints; thus it's least likely to satisfy all these constraints. Just as it's impossible to please everyone, because everyone has different desires.

Case 2: r is fixed

Another observation we made in Figure 7 was that the probability of SAT is higher at fixed r when k increases. We noted from the Critical Behavior in the Satisfiability of Random Boolean Expressions paper that the same transition behavior we've seen in 3-CNF probability of UNSAT will occur for 4-CNF at approximately when r = 9 and 5-CNF at approximately when r = 18.

The starting of spiking in 4-CNF could be observed at the rightmost part of Figure 5, although our graph is not complete to see the entire curve. This observation makes sense from the probabilistic point of view as satisfying a 3-CNF requires ⅓ of the literal to be satisfied in a clause while a 5-CNF only requires ⅛ of the literals to be satisfied.

Execution time

From Figure 8, 4-CNF takes the longest average time to solve especially towards r = 9. Following this trend, we hypothesised that 5-CNF would take much longer time than 4-CNF when r reaches its transition phase (r = 18).

From Figure 4 and 5, we know that execution time of 3-CNF and 4-CNF spikes during the transition phase from high probability of SAT to UNSAT. The long execution time does not seem to be related to the number of clauses n, the number of literals k, and the number of clauses L, because the execution time will eventually decrease back to the short amount of time after r passes the transition phase.

6. Conclusion

In conclusion, we've tried to analyze and understand the behavior of changing k and r values for solving CNF formulas in terms of the probability of satisfiability and the time it takes to solve. The probability of SAT decreases as r increases, when k is fixed. We've also deduced that we shouldn't try to reduce to 4-CNF as inputs to SAT solver, as the time taken is significantly longer than 3-CNF.

Lastly, we enjoyed exploring the behaviors of random CNF formulas and we would grade ourselves a A-.