# CS 830 Intro to AI, Spring 2025

## Written Assignments, Sushma Akoju

### Assignment 5

1. Describe any implementation choices you made that you felt were important. Clearly explain any aspects of your program that aren't working. Mention anything else that we should know when evaluating your work.

   I chose a model dictionary. In the case of dpll, I used a dictionary with each literal as key and corresponding assignment. In the case of walksat, I used a model dictionary for results which contains a variable as the key and its corresponding assignment.

   For dpll, I attempted two algorithms: one of them is vanilla dpll and other is from a modified version of dpll with scoring for choosing between x and NOT x (Michail G. Lagoudakis 2007). The (Michail G. Lagoudakis 2007) work suggests to directly find a single literal clause i.e. until the model finds unit propagation and when choosing single literal - it uses scoring as a way to select branching between x and not x. But it did not run faster than vanilla dpll so I did not include any results here.

   In case of walksat, it seemed like so far the satisfiable assignment can be found, walksat would do reasonably well, but if it was unsatisfiable, walksat could run forever randomly. This actually happens for both my walksat implementation as well as sat-ref walksat.

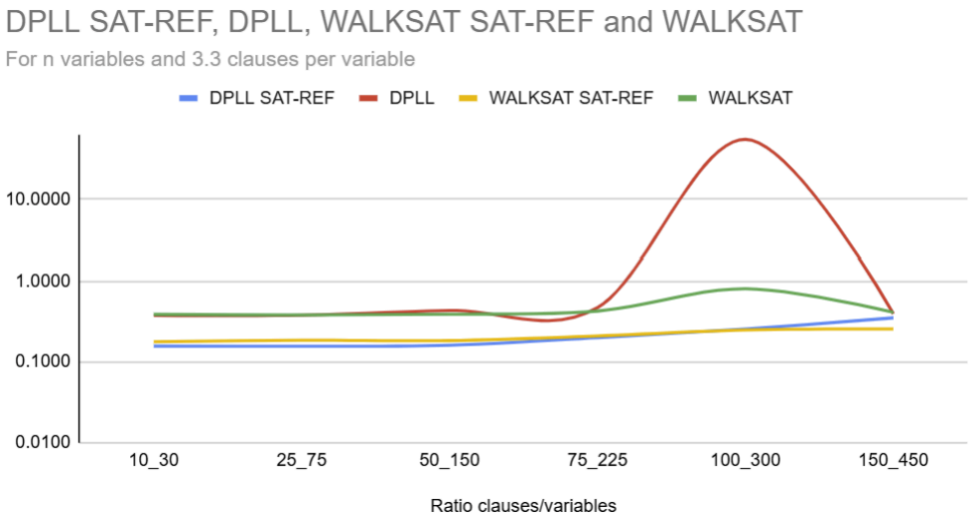
2. What is the size of the state space for this problem?

   Let n = number of variables and m = number of clauses. The state space grows exponentially in the number of variables n. 2^n i.e. tautology.. In the worst case, DPLL explores all possible truth assignments for each one of the variables until a satisfiable assignment is found. It seems that the ordering of branching variables also matters. If I randomly selected a free

literal, the chance that algorithm finds a satisfiable assignment will also be random depending on the number of free literals whether the algorithm removes pure literals or not. Suppose I reduce the clause by repeatedly setting variables only in unit propagation (Michail G. Lagoudakis 2007), then it seems likely to reduce the exponential time but only to some extent. But this Lagoudakis 2007 suggests scoring to choose whether to branch on x or NOT x, which is interesting. (free literal is the one that is not yet assigned, pure literal is the one that does not have complement of the literal in any of the clauses.)

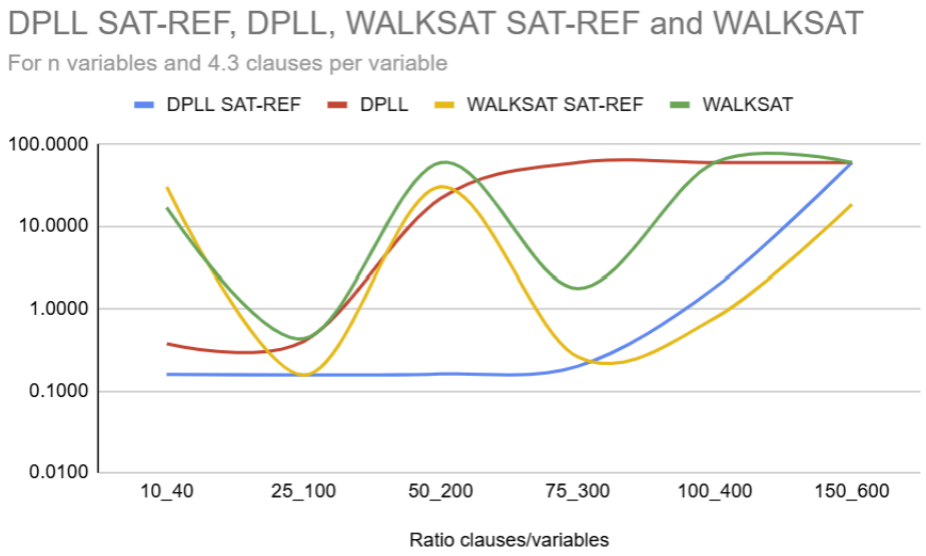
3. When there are around 3.3 clauses per variable, how does the largest 3-CNF formula your DLL can solve compare to the largest for your WalkSAT? (Undergrads can run the reference WalkSAT.)

   DPLL seems to take longer but always provides correct results. DPLL also ends up in timeout for >= 100 variables both with sat-ref and my dpll implementations.
   WalkSAT does not take as long as that of DPLL and especially if it is satisfiable, WalkSAT seems to finish faster than DPLL atleast until 100 variables with 3.3 clauses per variable ratio. WalkSat works fine. I think maxtries need to be adjusted to some constant times exponential in the number of variables. Because walksat
   WalkSAT seems to explore unendingly and is subject to randomization and can loop forever which was happening randomly with 100 variables both with Sat-reference walksat and my walksat.



4. Repeat the previous question at 4.3 clauses per variable, and explain the results you obtain.

   DPLL seems to take longer but always provides correct results.
   WalkSAT does not take as long as that of DPLL and especially if it is satisfiable, WalkSAT seems to finish faster than DPLL atleast until 100 variables with 4.3 clauses per variable ratio.
   WalkSAT seems to explore unendingly and is subject to randomization and can loop forever which was happening randomly with 100 variables both with Sat-reference walksat and my walksat.




5. What suggestions do you have for improving this assignment in the future?

   If there were any additional optimizations that could have benefitted dpll it would be interesting.