**CS 463G SAT Assignment Write Up**

Raymond Goodman

**Important Notes**

* I created a sample cnf file folder with two very basic cnf files. One is very easy to satisfy, the other is unsatisfiable.
* The code is not fully functional as all three algorithms do not run as intended all the time. They will work in spots, however.

**Algorithm Descriptions**

DPLL

The DPLL algorithm is a complete, backtracking-based approach. It starts with the set of clauses and an empty truth assignment, then recursively applies two simplification techniques: unit propagation and pure literal elimination. Unit propagation assigns values to variables that appear as unit clauses (i.e., clauses with only one literal), while pure literal elimination assigns values to variables that appear in only one polarity across all clauses. After applying these techniques, DPLL either simplifies the formula to find a solution or continues by selecting an unassigned variable and deciding. It recursively explores two possibilities: assigning True to the variable and, if that fails, assigning False. The algorithm proceeds with this search, backtracking, if necessary, until it either finds a satisfying assignment or determines that no solution exists for the given formula.

Simulated Annealing

The Simulated Annealing algorithm starts with a random assignment but takes a probabilistic approach to explore solutions. In each iteration, it randomly selects a variable to flip, and if this flip improves the number of satisfied clauses, it’s accepted. If the flip worsens the solution, it might still be accepted depending on a probability that decreases as the algorithm progresses, based on the cooling temperature. This allows the algorithm to escape local optima in search of a global solution. The algorithm continues iterating, gradually reducing the acceptance of worse solutions until it either finds a satisfying assignment or reaches the iteration limit.

GSAT

The G-SAT algorithm starts by randomly assigning truth values to all variables and then tries to improve this assignment through repeated variable flips. It checks if the current assignment satisfies all clauses, and if it does, it returns the solution. If not, it examines each variable by temporarily flipping its value, determining how many clauses are satisfied after each flip. The variable flip that results in the highest number of satisfied clauses is selected. This process is repeated up to a specified limit of flips, and if no satisfying assignment is found by then, the algorithm reports failure.

**Graphs**

Summary

The graphs show the performance results of the algorithms on the sample files I created and 2 files in the cnf folder. In the test1 file, it was solved extremely fast, so that no noticeable data is displayed. In the uf20-0162, a solution was found quickly for simulated annealing, hence the lack of a bar in the bar graph.

A screen shot of a computer

Description automatically generated

A screen shot of a graph

Description automatically generated

A screen shot of a computer

Description automatically generated

A screen shot of a computer

Description automatically generated

**Learning Outcomes**

The biggest learning outcome I received is how to use heuristic based approaches to make decisions on actions to solve exceptionally large problems.