COSC 76, Fall 2020, PA-5, Edmund Aduse Poku

Q1b.:

a) Description

GSAT

I implemented GSAT as a class with 4 instance attributes: h (threshold value for randomly selecting variables), var_map (maps clause literals in the given file to integer variables), model (for assigning values to integer variables), and clauses (list of all clauses in the given file). The class has 4 instance methods. The __convert_to_generic_SAT__(self, cnf_file) method takes in a cnf file converts it into a generic SAT problem of integeger variables and sets of clauses. The write_solution(self, filename) method takes in a solution file and maps the solution back in terms of the given problem parameters using the var_map created in the problem conversion method. The __is_clause_satisfied__(self, clause) method takes in a clause and checks if the clause is statisfied against the current model. Similarly, the __are_all_clauses_satisfied__(self, clauses) method takes in all the problem clauses and checks if each of them is satisfied.

SAT

I implemented the SAT class as an extension of the GSAT class. In that sense, I was able to use virtually all the instance variables and instance methodes in the GSAT class in the same way. The only methods that needed more specifications and adjustments are the walksat and __are_all_clauses_satisfied__(self, clauses) methods.

b) I think my algorithm works well and satisfies all boundary cases. However, due to not having enough time to run all the test cases to completion, I am unable to provide concrete data to support how my algorithm performs.