Final Project Proposal

Team Members:
Sharath Chandra Vemula - svemula@umail.ucsb.edu
Saikumar Yadugiri - saikumar@umail.ucsb.edu

Summary

With the advent of fast SAT solvers that can take care hundreds of thousands of clauses, NP-complete problems which were previously thought to be quite hard to solve can be practically solved using Karp reductions. In particular, if a reasonably good reduction for an NP-complete problem exists, using that reduction, we can convert the instance of the problem into a (3-)SAT formula. We can then feed this formula to an SAT solver such as Z3 or PySAT and check if there is a satisfying assignment or if the formula is unsatisfiable. The advantage of such reductions is that once we get a satisfying solution, it can be converted back to the solution of the original problem. Thus, we are solving the NP-complete problem using SAT solvers.

For our project, we are looking at the graph 3-coloring problem which is a well-studied NP-complete problem. The idea of the problem is simple. Given a graph, we need color each vertex of the graph using on three colors, red, green, blue such that no two vertices connected by an edge have the same edge. The coloring problem can be thought of as a partition problem in which we are dividing the graph's vertex set into 3 disjoint partitions such that there are no edges within a partition. While the number and the choice of colors are completely arbitrary, the real-world applications of the graph 3-coloring problem are enormous including CPU memory register allocation, frequency allocation for transmitting towers, etc.

We will implement our code in Python using Z3 and PySAT SAT solvers. We will write a script that allows the user to input his graph using the adjacency matrix format and then pass this to another script that implements the Karp reduction to convert it into an SAT formula. This way, we will keep our manual testing, automated testing, and logic files completely isolated. For simplicity, we will only consider simple graphs and in that case, for the graph 3-coloring problem, the weights of the graphs and the direction of the graphs do not matter. Hence, we will transform any non-symmetric adjacency matrices into symmetric adjacency matrices.

Moreover, we will also implement the heuristic algorithm mentioned in [1] which uses the signs of the eigenvectors for the adjacency matrix of a graph. As we are computing all the eigenvectors and values of the adjacency matrix, the time required for this approach might be higher than the SAT solver approach and in any language, the eigenvalues and eigenvectors produced might not be entirely accurate. This might inject a few errors into our solution. On the other hand, the SAT solver, if it halts within the stipulated time limit, will always produce a valid solution that can be readily converted to a valid coloring scheme. Comparing the solutions received from the SAT solver(s) and the heuristic algorithm and the time taken by each method is the main aim of our project.

Experimentation

For finding random graphs to test our implementation, we chose the sparse matrix collection present in [2]. This website contains 1,185 symmetric matrices which can be used as unweighted and undirected graphs. We will modify the matrices to have non-zero diagonal entries as 1 only if there is an edge between the vertices and set all diagonal entries to 0. As the edge weights and directions are irrelevant to graph coloring, this way, we will cover all of the symmetric matrices.

Although the collection only has 1,185 matrices, the individual dimensions of the square matrices varies from 1 to $\sim$ 500,000. Hence, it is vital to have a time-out for the SAT solver evaluation as it might quickly become infeasible to find a solution in reasonable amount of time. We are thinking of setting the

time-out to 300 seconds (5 minutes). This way, we will have a good estimate of when to stop. We will also run the instances which have timed-out for a total of 3 times to account for any random guesses that occur inside the SAT solving process.

We will also use similar time limits for heuristic algorithm presented in [1] to solve the 3-Coloring problem. Our final results will comprise of the number of successfully solved graph instances using SAT solver and the algorithm provided by the [1]. We also plot the time it took for both algorithms to find the solution vs the size of graph. We will present our results in table as well as data chart format.

References

[1] Aspvall, Bengt, and John R. Gilbert. "Graph coloring using eigenvalue decomposition." SIAM Journal on Algebraic Discrete Methods 5.4 (1984): 526-538.

[2] Timothy A. Davis and Yifan Hu. 2011. The University of Florida Sparse Matrix Collection. ACM Transactions on Mathematical Software 38, 1, Article 1 (December 2011), 25 pages. DOI: https://doi.org/10.1145/2049662.2049663