Advanced Algorithm Design CSCE 686 Spring 2020
---------- Homework 3 ----------
Student: **John Beighle** Due Date: 19 May 2020

Problem 1:

As discussed in class size of an instance is an indicator to be considered but not the exclusive consideration. For instance, considering the limit of dimensionality attempting to solve a 2^n search space gets into extreme computation times with n=500. Processing time could move into the hundreds of years. However, instance structure is another important consideration. When a small instance has many connections, it may be more difficult to solve optimally versus a much larger instance with very few or no connections. An instance with 500 nodes could possibly be solved quickly depending on the node structure. There is no way to know these things Apriori. Therefore, much of the theory about these two problems is devoted to identifying special types of graphs that admit more efficient algorithms, or to establishing the computational difficulty of the general problem in various models of computation. To find a maximum clique, for instance, one can systematically inspect all subsets, but this sort of brute-force search is too time-consuming to be practical for networks comprising more than a few dozen vertices. Although no polynomial time algorithm is known for this problem, more efficient algorithms than the brute-force search are known. For instance, the BK algorithm can be used to list all maximal cliques in worst-case optimal time, and it is also possible to list them in polynomial time per clique.

Problem 2:

Results

Observation of clear complimentary relationship in PD between MIS and Clique: Contrasting MIS/Clique problems with the graphprogram's MIS.data output demonstrates the two problems relationship clearly. Using the Turan graph creator as input with node size (n), for instance, in a simple case is n=4 and number of independent sets r=2, the results are proportional e.g. in this case there are 2 Maximal Indep. sets found and 4 cliques of size 2. When n=3 and r=2 Maximal Indepence sets = 3 and only 1 clique of size 3.

Observation of UI: User interface was basic, clean and intuitive. It seems to deliver what it claims. Some simple improvements may to remove the upper bound limits of dimensionality and let the program attempt to solve exceptionally large graphs with full utilization of the underlying OS and hardware. Adding this and a constant memory and core set utilization to the console would make this software much more interesting since this is a class studying computational complexities and optimization. These things are the essentials and would be fairly easy to add. At the very least add a multi-threaded capability and one could open system tools to watch the core/memory showdown.

Observation of the software: As mentioned in class another nice addition on the software side would be a standardized data format for input. There are many common modern formats e.g. these. It would also be interesting to do a study in this space to better understand where graph data format standards are at presently. It was an odd discovery to have performed a good internet search in hopes to find a general purpose, whiz-bang,

graph generator. One that could kick out any style of graph you might want to do computationally intense algorithm design with but the search results proved fairly fruitless. This was surprising.

Visualizations:

Some free lunch here: Graphgrogram barely got warmed up with the 10 graphs it processed during this analysis due to the program's node size limitation. Thus, a nice graphic of meaningful space and time performance analysis would be nothing more than a linear graph of the microscopic changes as node size of the small, medium and "large" graphs used in the exercise. For the case of the medium sized non-planar presentation n=30, r=20, it literally took a 10^{th} of second to find the solution.

Data Analysis:

Observation of graphprogram upper bound limitation: Program refused to process node sizes of significant size. In one of the attempted node sizes of 200 the program's output was basically null. This is surely by design to avert limits of dimensionality 2^n, as in this case n=200 encroaches a potentially very large time and space requirement.

Reproducibility and Conclusion:

Exercise focus is to evaluate graphprogram: Evaluation of AFIT's graphgrogram, because of the bound limit, makes the results above trivial in terms of reproducibility. The primary educational import of the program is the existence of a solid and working codebase to build upon and add learning opportunity optimizations to which is HW4. Will be interesting to discuss further next week the best approach to doing computational analysis with this codebase, especially with the different types of graphs mentioned in HW4 to analyze. Looking into DIMACS benchmarks and recent research of this PD, things like: "Planar graphs form an important class both from a theoretical and practical point of view. The theoretical importance of this class is partly due to the fact that many algorithmic graph problems that are NP-hard in general remain intractable when restricted to the class of planar graphs. In particular, this is the case for the maximum independent set (MIS) problem, i.e., the problem of finding in a graph a subset of pairwise non-adjacent vertices (an independent set) of maximum cardinality. Moreover, the problem is known to be NP-hard even for planar graphs of maximum vertex degree at most 3 or planar graphs of large girth. On the other hand, the problem can be solved in polynomial-time in some subclasses of planar graphs, such as outerplanar graphs [5] or planar graphs of bounded chordality [1]." These areas of optimization where we may begin to appreciate existing benchmarks and what those look like in comparison to our in-class optimization exercises.

References:

1. The Maximum Independent Set Problem in Planar Graphs Vladimir E. Alekseev1, Vadim Lozin, Dmitriy Malyshev3, and Martin Milani