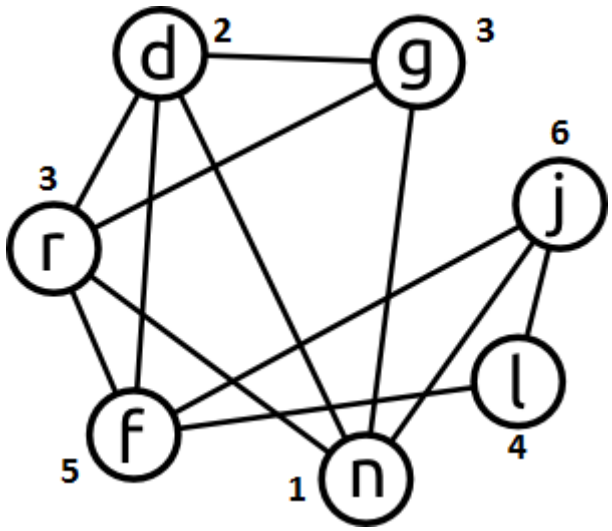
## CSE 7004 – Term Project

(Due date will be announced later, electronic submission only, to ahozer AT gmail DOT com)

In this project, you are required to solve the Maximum Weighted Independent Set Problem (MWISP). In graph theory, an independent set is a set of vertices in a graph, no two of which are adjacent. That is, it is a set S of vertices such that for every two vertices in S, there is no edge connecting the two. In the MWISP, you are required to find the independent set of a given graph such that the sum of the weights of the nodes is the maximum [1].

For instance, for the below graph




the nodes g and f form an independent set with a weight sum of 8, whereas nodes r and j form another independent set with a weight sum of 9.

In this project, you are required to solve MWISP for a given graph (i) optimally using GUROBI Optimizer (you can obtain a free academic license) and (ii) using a greedy heuristic and (iii) using a genetic algorithm-based heuristic. You should design an experiment comparing the solutions found by the proposed heuristic solutions to the optimal solutions.

For this project, you are required to implement:

1. A random undirected graph generator for MWISP: The generator should get two parameters:

    a. the number of nodes in the graph,
    b. the density of the graph;

and generate a random undirected graph with the given number of nodes and the given density. The weights of the nodes should be determined using uniform random distribution on the interval [0-1].

Your generator should save the graph in a text file, the format of which is as follows:

Number of nodes // The first line

Number of edges // The second line

List of node weights (format: X W where W is the weight of node X, X takes values from 0 to |V|-1)

List of edges (format: X Y which indicates an edge from node X to node Y)

e.g.

```
3
4
0 0.32
1 0.45
2 0.89
0 1
0 2
1 2
1 3
```

Since your graph is undirected, do not include duplicate edges such as 1 3 and 3 1, since edge 1 3 is same as edge 3 1.

The algorithm for generating a random graph is as follows:

```
For each vertex u:0 to Number_of_Vertices-1
    For each vertex v: u+1 to  Number_of_Vertices-1
        Generate a uniform continuous random number between 0 and 1
        If this number is smaller than (the density value), add an edge between u and v.
```

You are required to generate at least 20 graph files using the following parameters:

| Number of Nodes | Density |
|---|---|
| 500 | 0.1, 0.3, 0.5, 0.7, 0.9 |
| 1000 | 0.1, 0.3, 0.5, 0.7, 0.9 |
| 1500 | 0.1, 0.3, 0.5, 0.7, 0.9 |
| 2000 | 0.1, 0.3, 0.5, 0.7, 0.9 |

2. A front-end for Gurobi Optimizer: The front-end will read the given graph file and solve the MWISP using Gurobi MIP Solver as a library.

The integer programming formulation of the MWISP is as follows:

$$\max \sum_{v \in V} w_v x_v$$

$$\text{s.t.} \quad x_u + x_v \leq 1 \quad \forall \{u, v\} \in \text{E}$$

$$x_v \in \{0,1\} \qquad \forall v \in \text{E}$$

where $x_v$ is the decision variable indicating whether the vertex v is included in the independent set or not, and $w_v$ is the weight of the vertex v.

Please check Gurobi examples for building your model in your preferred programming language https://www.gurobi.com/resource/functional-code-examples/.

Reference manual can be found in https://www.gurobi.com/documentation/9.0/refman/index.html

For each instance, provide a maximum time limit of 1 hour ( set TimeLimit parameter to 3600 seconds before calling optimize method - https://www.gurobi.com/documentation/9.0/refman/java_parameter_examples.html). If the optimal solution cannot be found, then use the suboptimal solution and indicate this in the report.

2. A greedy heuristic solver: The heuristic solver will read the given graph file and provide a greedy heuristic solution. You may implement any greedy heuristic.

3. A genetic algorithm based heuristic solver: The heuristic solver will read the given graph file and provide a heuristic solution.

Your genetic algorithm based heuristic method should get the following five inputs:

a. Name of the graph file
b. Number of generations
c. Population size
d. Crossover probability
e. Mutation probability

For each of the graphs, your program should generate 50, 100, 150 generations with a population of size 50 and 100. For selection of parents, you are required to implement the binary tournament selection method.

For crossover, you should use uniform crossover operator with a probability of 0.5. Finally, for the mutation, bitwise mutation operator should be used with a probability of 1/n (n:number of nodes in the graph). Any desired repair function can be used (you should also report details of your repair function).

Therefore, for each of the twenty graphs, you should run your program 6 times (once for each of the above parameter configurations). You should submit fully commented source code along with a report containing tables, graphs and discussion of the results and the effect of parameters. Further details will be discussed after the lectures.

References:

[1] https://en.wikipedia.org/wiki/Independent_set_(graph_theory)