Final project
The Maximum Edge Weight Clique Problem

Introduction

Given an undirected simple connected graph $G = (V, E)$, a clique is a complete maximal induced subgraph of G. The Maximum Clique Problem consists in finding a clique of maximum size. This is a classic graph theory problem that has many real-life applications in various fields such as social networks, chemistry, bioinformatics. In addition, this problem is NP-Hard and it has been studied as a combinatorial optimization problem, being very important in operations research and theoretical computer science.

Now let $G = (V, E, w)$ be an undirected simple weighted connected graph such that $|V| = n$, $|E| = m$ and $w : E \to \mathbb{R}_{[1,100]}$ is the weight function. That is, G is a graph with bounded weights on its edges. The Maximum Edge Weight Clique Problem (MEWC) consists in finding a clique that maximises the sum of the weights of its edges. Clearly MEWC is also a NP-Hard problem since the Maximum Clique Problem is the special case in which all weights are equal.

Exercises

1. Describe real-life situations that can be modelled as MEWC.
2. Develop and implement an exact algorithm to solve the MEWC with at least one improvement as explained in class.
3. Develop and implement a constructive heuristic to solve the MEWC.
4. Develop and implement a local search heuristic to solve the MEWC.
5. Develop and implement an algorithm that uses the Tabu Search meta-heuristic [1] for the MEWC. Consider the use of different sizes for the tabu list and analyze possible stop criteria. Optionally, you can add aspiration level, intensification and diversification techniques. **SET** all the parameters involved (size of the tabu list, stop criteria, etc.) through experiments. **JUSTIFY** your elections and the choice of the test instances considered to decide your parameters.
6. For each of the methods 2 to 5:
   - Explain in detail the ideas and how your algorithm works. **DO NOT** give the code, use pseudo-code instead.
   - Calculate its time complexity using the pseudo-code given before.
   - Try to describe instances of the MEWC for which the method does not provide an optimal solution. How bad the obtained solution can be with respect to the optimal solution?
   - Run experiments that allow to observe the performance of the algorithm. Compare the execution time with its theoretical time complexity. Also, compare the quality of the obtained solutions and the execution time with respect to the input size (and other parameters if appropriate). Within the cases of test should also be included, as pathological cases, those described in the previous item. In case the algorithm has some configurable parameters that determine its behaviour (the meta-heuristic for example, although it remains open to others as well), you should run experiments varying the values of the parameters and choose, if possible, the settings that provide best results for the instances used. Present the results obtained using appropriate graphics.

7. Once you have chosen the best configuration values for each heuristic, run experiments on a **new set of instances** to observe the performance of the methods and compare again the quality of the solutions obtained and the execution time. Present all the results obtained by using appropriate graphics and present conclusions about your work.

General guidelines

1. A report tackling all points described above should be written (in english, of course) and uploaded in pdf format. You should follow the guidelines described in the document "Normes de mise en page des dossiers", Version 3.2, by Matéo Sorin. Use of LaTeX [2] is **highly recommended**.

2. You can use any programming language you prefer to implement your algorithms **BUT** use the same for all of them. When you upload you project, you should include within the source code, a file called "`readme.txt`" that explains how to compile/execute the source code.

3. Your algorithms should read an input file (check the examples below) that contains $m + 1$ lines as follows:

   - The first line has the number n of vertices and m of edges separated by one space.
   - The next m lines, each contain three numbers u, v, w_{uv} separated by one space, where u,v represent two vertices ($1 \leq u \neq v \leq n$) and w_{uv} represents the weight of the edge uv ($1 \leq w_{uv} \leq 100$).

4. Your algorithms should write a file containing the solution (check the examples below) as follows:

   - The name of the file should be "`instance_method.out`" where "`instance`" is the name of the input file (without the extension if any) and "`method`" should be one among "`exact`", "`constructive`", "`local_search`" and "`tabu_search`".
   - The file should contain two lines. The first one contains the size and the weight of the clique separated by one space. The second line contains the number of the vertices in the clique separated by one space.

5. You should handout as well the test instances you have used to set the configurable parameters and the final set of instances used to compare performances (as they must be referenced in the report).

6. **VERY IMPORTANT:** While you're running measured experiments,

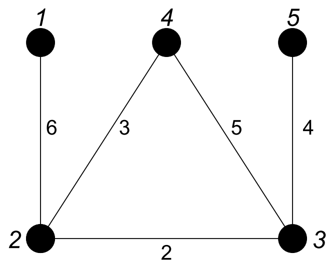
   - **DO NOT** do anything else with your computer **AND** close all unnecessary programs (like chrome, firefox, etc) that might be running. Not doing so might affect the performance of your algorithms and give inaccurate results. **DO NOT** plug/unplug your computer either.
   - **USE** the same computer (and under the same conditions specified above) to compare time execution of algorithms. It makes no sense (therefore it will give you wrong results), if you compare time execution, for example, of algorithm A against algorithm B, in two different computers with different processors, memory size, etc.
   - **DO NOT** waste your time running experiments on single instances for many hours (or days!).

Examples

- Consider the file "`test1.in`":

```
5 5
1 2 6
2 3 2
2 4 3
3 4 5
3 5 4
```

This file corresponds to the following graph on 5 vertices and 5 edges:



Therefore if we run the exact algorithm in that instance, the file "`test1_exact.out`" will be created containing the following two lines only.

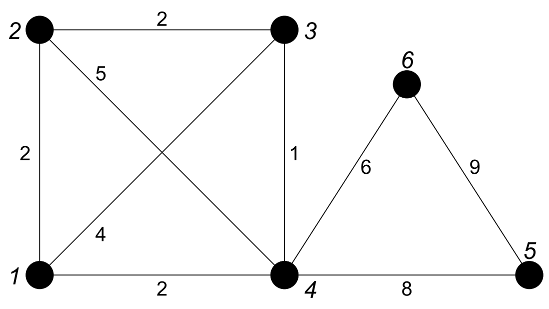
```
3 10
2 3 4
```

This corresponds to the clique $\{2, 3, 4\}$ with total weight $2 + 3 + 5 = 10$.

- Consider now the file "`test2.in`":

```
6 9
1 2 2
1 3 4
1 4 2
2 3 2
2 4 5
3 4 1
4 5 8
4 6 6
5 6 9
```

This file corresponds to the following graph on 6 vertices and 9 edges:



Therefore if we run the exact algorithm in that instance, the file "`test2_exact.out`" will be created containing the following two lines only.

```
3 23
4 5 6
```

This corresponds to the clique $\{4, 5, 6\}$ with total weight $8+6+9 = 23$. Note that the clique $\{1, 2, 3, 4\}$ is bigger (4 vertices) **BUT** its weight is 16 that is smaller than 23.

Deadline & Upload instructions

The deadline of the project is on Monday, May 10, 12h00, on the ENT.
NO FILES WILL BE ACCEPTED AFTER THE DEADLINE.

You should upload only **ONE** compressed file called "`team_XX.zip`", where "`XX`" is the number of your team. Only **ONE** member of the team should upload the file. This file should contain the following folders:

- "`Report`" : containing the pdf file of your report.
- One different folder for each algorithm called "`Method`", where "`Method`" should be "`Exact`", "`Constructive`", "`Local_search`" and "`Tabu_search`". Each of these folders must contain two sub-folders.
  i. Sub-folder "`Source`": containing the source code of the algorithm plus the "`readme.txt`" file with the compiling/executing instructions (**DELETE** executable files).
  ii. Sub-folder "`Instances`": containing the instances you used to test your algorithms individually and/or to set configurable parameters (that should be referenced in the report). This folder should contain as well "`.out`" files with the obtained solutions for each input instance. Compress instances files if their size is very big.
- "`Instances`": containing the **new set of instances** you used to compare performance between algorithms as described before. You should again include "`.out`" files with the obtained solutions for each algorithm and input instance. Again, you should compress these files if their size is big.

If you use `github` or any other repository manager, **DELETE** all hidden files and folders that the manager creates. **Not doing so will lead to a penalty of points**.

References

[1] F. Glover, E. Taillard, and D. de Werra. A user's guide to tabu search. *Annals of Operations Research*, 41:3–28, 1993.

[2] *An introduction to LaTeX*. LaTeX project (`https://www.latex-project.org/about/`).