CS-E4600 — Programming project

Autumn 2019

Submission deadline: Thursday, Dec 5, 2019, midnight

Instructions

In this programming project you have to design and implement your own method to partition a graph into communities. The tasks you have to complete for the programming project are the following:
(1) design and implement your own graph-partitioning method;
(2) submit your solution, which should include the source code you developed and a report; and
(3) use your implementation to participate in a graph-partitioning competition.

To receive full points you **only** have to complete Tasks (1) and (2). Task (3) is **optional**, however, by participating in the competition you can receive **up to 50 extra points**.

You may work on the project individually or in two-person teams. You can find your own teammate or ask us to find one for you. To do so please send an email to TAs before Friday, November 8.

Each **team** needs to return **only one** solution, i.e., we do not require a separate solution from each of the team members. Detailed instructions on the content and format of the report will soon follow.

The amount of code required for the project is not a lot, however, you may have to think carefully about your algorithm. Do not leave the project for the last moment. Start early!

It is also possible to work on a different programming project than the one proposed here. In this case, you should propose your own project. Your project needs to be related to the topic of the course, *algorithmic methods for data analysis*, it needs to be of equivalent difficulty to the one described here, it should not be already implemented, and it should not be used in another course.

If you wish to propose an alternative project, you should write a project description, send it to the course instructor, Aris Gionis (`aristides.gionis@aalto.fi`), and have it approved. The deadline to propose your own project is Friday, November 8.

Remember that there is a budget of 5 late days, which you can use in any way you want for the three homeworks and the project. Weekend days count as late days. Late days are counted individually (so a team has to return the project by the time required so that none of the team members exceed their budget).

Project description

We will use graphs from the Stanford Network Analysis Project (SNAP)

`http://snap.stanford.edu/data/index.html`

in particular, we will work with the following 5 graphs:

`ca-GrQc`, `Oregon-1`, `roadNet-CA`, `soc-Epinions1`, and `web-NotreDame`.

These graphs are not necessarily connected and they may contain a large number of connected components, which might result in trivial solutions (in particular, when the value of k is less than the number of connected components). Henceforth, if a graph is not connected, by convention, we will work with its largest connected component. For your convenience, we have computed the largest connected component of the 5 SNAP graphs.

`http://bit.ly/amdm-2019-dataset`

Graph-partitioning task: Given an undirected graph $G = (V, E)$ and an integer $k > 1$ we want to partition the set of vertices V into k communities $V_1, \dots, V_k$, so that $\bigcup_{i=1}^{k} V_i = V$ and $V_i \cap V_j = \emptyset$ for all $i \neq j$. We want the communities $V_1, \dots, V_k$ to be as much separated from each other as possible. We also want that the communities have roughly equal size. Thus, we will evaluate the goodness of a partition $V_1, \dots, V_k$ using the objective function

$$\phi(V_1, \dots, V_k) = \sum_{i=1}^{k} \frac{|E(V_i, \overline{V_i})|}{|V_i|},$$

where for $S, T \subseteq V$ with $S \cap T = \emptyset$ we define $E(S, T)$ to be the set of edges of G with one endpoint in S and the other endpoint in T, i.e., $E(S, T) = \{(u, v) \in E \mid u \in S \text{ and } v \in T\}$, and we also define $\overline{V_i} = V \setminus V_i$.

You may want to think why the small value of the objective ϕ indicates that the partition has well-separated communities which are well balanced.

Your first task is to implement a program that reads a problem instance (graph G and integer k) and produces a partition $V_1, \dots, V_k$ for which $\phi(V_1, \dots, V_k)$ is as small as possible.

Implementation: Your program should output a partition $V_1, \dots, V_k$ as a text file with one line per vertex of the form

```
vertexID clusterID
```

There is no restriction on the programming language — you may use the programming language of your preference.

A basic algorithm that you may want to use in the project is the spectral algorithm discussed in class (slide set 11). This algorithm involves forming the Laplacian of the adjacency matrix of the graph, computing and Eigen-decomposition of the Laplacian, and then applying k-means on the vector representation of the vertices provided by the Eigenvectors. Notice however, that this algorithm does not make any attempt to ensure a balanced partition. So, you may want to think about how to improve it in order to obtain a balanced partition (and thus obtain smaller values of the objective ϕ).

You can also implement a different algorithm, if you choose to do so. In this case, to get full points, you will need to justify your proposed algorithm and explain why your algorithm gives small edge cuts and why it favors a balanced partition.

Graph partitioning is a standard graph-mining task and you can find lots of freely available software that implement different algorithms. Obviously, you cannot use those implementations. The objective of the project is to implement your own algorithm. However, you can use existing implementations of “standard” data-mining tasks. For example, for implementing the basic spectral algorithm, you can use an existing implementation for computing the Eigen-decomposition of a matrix, or an existing implementation of the k-means algorithm.

If you are in doubt whether it is OK to use some specific library or routine, please check with the TAs.

Instructions for submitting the project report

The submitted report *must* contain the *name(s)*, *student number(s)*, and the *email id(s)* of the team member(s). Additionally, the report should contain a section describing the individual contributions of each team member. It is acceptable to discuss and collaborate with others, however, code and report must be written entirely by the members of a team. In the case of collaboration with other teams, the nature of collaboration should be explicitly stated and credits should be given to the individuals. Using other’s code and returning almost-identical or paraphrased reports is not acceptable.

Your report must be written in a text processor and has to be submitted in a pdf format via mycourses; hand-written reports will not be accepted. You should also need to submit your code and experimental results. In total, your submission to mycourses should consist of one tar/zip bundle containing (i) the report — in pdf format; (ii) a directory `code/` containing your code; and (iii) a directory `results/` containing your results. The code directory should also contain a `README.txt` file describing how one should use the code in order to reproduce the results contained in the results directory. We may also ask for additional results and code in case we find the results to be ambiguous or interesting.

A typical report file should contain the following sections: introduction, literature review, description of your algorithms and methods, experimental results, and conclusion. We expect that you conduct your experimental evaluation on the five networks listed before, i.e., `ca-GrQc`, `Oregon-1`, `roadNet-CA`, `soc-Epinions1`, and `web-NotreDame`.

In addition, in your report, you should indicate whether you participated in the competition, and if yes, what results you achieved.

Grading of the project is based on the quality of the report (30 points), quality of the experiments and results (30 points), and algorithmic approaches used for the project (40 points).

The deadline for submitting your report to mycourses is Thursday, December 5, midnight.

Instructions for participating in the competition

Participating in the competition is optional, but it gives you the chance to receive extra credit points.

For the competition we will use the following graphs from SNAP:

ca-GrQc, Oregon-1, roadNet-CA, soc-Epinions1, and web-NotreDame

There will be 5 competition events, one for each of the above graphs, with a given value for the number of partitions k.

The graph statistics and value of k are described in Table 1.

For the competition, we will use the following input graph file format: a problem instance (G, k) is defined so that G is the graph to be partitioned and k is the number of partitions, and it will be provided as a text file, where the first line specifies the problem parameters (`# graphID numOfVertices numOfEdges k`) and the rest of the lines specify the graph, one line per edge (`vertex1ID vertex2ID`). The input graph file format is also described in Figure 1. If the graph contains n vertices in total then `vertexID` ranges from 0 to $n-1$.

For the purpose of this project, we assume that all graphs are undirected. If a graph dataset is directed we treat it as undirected by ignoring the edge direction. If a graph has multiple edges between the same pair of vertices then we only consider it as one edge (no multi-edges). The graphs may contain self-loops.

Registration

You should register for the competition by updating the necessary information in the spreadsheet:

http://bit.ly/amdm-2019-project-registration

Only one registration per team is needed.

After registration, you will receive an email notification related to a shared google-drive folder where you can upload your results. The registration process and creation of shared folders is not automated and it has to be done manually, so expect some delay. In order to participate in the competition you need to register before Friday, November 22, midnight.

Submission

After receiving the access permissions you can upload your results to the shared drive. The results filename should be in the format `graphID.output` and there can be *at most* 5 output files in your

Table 1: Dataset statistics

Graph	**#vertices**	**#edges**	**Value of k**
ca-GrQc	4 158	13 428	2
Oregon-1	10 670	22 002	5
soc-Epinions1	75 877	405 739	10
web-NotreDame	325 729	1 117 563	20
roadNet-CA	1 957 027	2 760 388	50

Figure 1: Input graph file format

```
# graphID numOfVertices numOfEdges k
vertex1ID vertex2ID
vertex3ID vertex4ID
vertex5ID vertex6ID
...
```

Figure 2: Output file format

```
# graphID numOfVertices numOfEdges k
vertex1ID partition1ID
vertex2ID partition2ID
vertex3ID partition3ID
...
```

shared folder, one for each graph. The maximum allowed storage space per team is 50 Megabytes and this includes all output files. Remember that we are using Google's cloud storage so please do not store unnecessary and large files.

For a given problem instance (G, k), your program is expected to output a graph partitions $V_1, \ldots, V_k$. We will use the following output format: the output file should be provided as a text file, where the first line specifies the problem parameters (`# graphID numOfVertices numOfEdges k`) and the rest of the lines specify the partition, one line per vertex (`vertexID partitionID`). The output format is also described in Figure 2. If an input problem instance contains n vertices and k partitions then `vertexID` ranges from 0 to $n-1$ and `partitionID` ranges from 0 to $k-1$. A partition output is valid if and only if each vertex is assigned to exactly one partition and there exists at least one vertex in each partition.

Evaluation

The evaluation process is automated and a script runs every 4 hours starting from midnight. The scores of your submission will be updated in a shared folder of google drive at the location:

`http://bit.ly/amdm-2019-project-results`

Each file `graphID-results.txt` lists the scores of all submissions for the graph instance `graphID.txt` in ranked order. In total, we have five results files, each one of which lists the scores for each of the datasets in Table 1.

Each results file is in text format and each line of the file has the format (`Rank TeamName Cost`). `Cost` is the value of the objective function ϕ. The lower the value of `Cost` the better is your rank. Our evaluation script only takes into consideration your most recent submission and we do not store your old scores. So, we expect you to resubmit your output files (partitioning results) only if it is strictly improving the objective function value.

A team will receive extra points if it gets placed in the top-10 of at least one competition. In particular, if a team gets placed in the top-10 of i competitions, $i = 1, \ldots, 5$, the team will receive x_i extra points. If a team has two members, both members will receive x_i extra points.

i	1	2	3	4	5
x_i	25	38	45	48	50

Deadline: The competition is closing on Thursday, Dec 5, 2019, midnight.

Questions

If you have any question or need clarifications, please contact the teaching assistants (Ananth Mahadevan, Antonis Matakos, Suhas Thejaswi) through email (firstname.lastname@aalto.fi).