CS 170 Efficient Algorithms and Intractable Problems
Fall 2022 J. Demmel and J. Nelson Project

Problem Statement

The penguins of Algorithmia are intertwined in a complex web of disputes. The only way to settle them is with a snowball fight. Your job is to organize the penguins into teams such that the tournament resolves the most significant disputes among them. Your team assignments are judged under 3 criteria:

1. We would like the tournament to resolve disputes, so it is best to put disputing penguins on different teams.
2. There should be as few teams as possible, so the tournament can be over quickly.
3. Teams should be roughly equal in size, so no team has a competitive advantage.

You are given an undirected graph $G = (V, E)$ where the vertices are penguins and the edges are the disputes. The edges have weight w_{ij} representing the significance of the dispute between i and j. If there is no edge between i and j, w_{ij} is 0. You will produce a partition $\pi : V \to \mathbb{Z}^+$ that assigns to each vertex v a team number $\pi(v) > 0$.

We define $k \in \mathbb{N}$ as the number of teams in π and $\vec{p} \in \mathbb{N}^k$ as a vector of team sizes:

$$k = \max_{v \in V} \pi(v)$$

$$p = |\{v \mid \pi(v) = i\}| \text{ for } 0 < i \leq k$$

The cost of the partition π is the sum of 3 parts, one for each of the 3 criteria:

$$C = \sum_{\substack{(i,j) \in E \\ \pi(i) = \pi(j)}} w_{ij} + 100 \exp(0.5k) + \exp\left(70 \left\| \frac{\vec{p}}{|V|} - \frac{\mathbb{1}}{k} \right\|_2\right)$$

Where $\mathbb{1} \in \mathbb{N}^k$ is a vector of all ones. You can view the third term of this cost as exponentially punishing the difference (in the L2 norm) between the uniform distribution and your distribution of penguins among the teams.

Your goal is to find a partition π with minimal cost C. You do not need to implement this cost function yourself as it is implemented as the `score` function in the starter code.

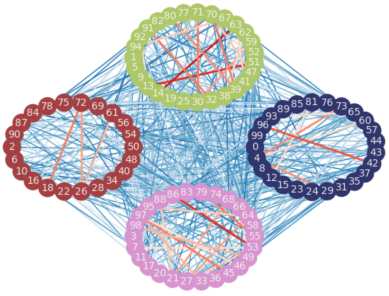

Figure 1: A visualization of a team assignment among 100 penguins. Blue edges are disputes between teams and red edges are disputes within teams. This was generated using `visualize` function in the starter code.

Phase 1: Input Phase (Due 11/16, 11:59pm)

This is a team project. You may have up to 3 members in your team. Your team members and name must be finalized before the Phase 1 deadline.

Overview

Your team will submit three inputs of different sizes. The inputs you are submitting are instances of the problem (i.e. weighted undirected graphs) that are to be solved by you and your peers in Phase 2. In Phase 2, your team will be graded based on how well your solver performs compared your classmates' solvers on these inputs, *so it is in your favor to construct difficult/tricky inputs*.

Input Format

The file extension for inputs is `.in` and the file format is JSON. The input graphs will be encoded in node-link format, which is handled for you by the `read_input` and `write_input` functions in the starter code. The vertices are increasing natural numbers, starting from 0.

The maximum edge weight allowed is 1,000 and the maximum number of edges allowed is 10,000. We require that the sum of edge weights in the graph is at least 500,000. No self-loops are allowed in the graph. You can check that your inputs are valid using the `validate_input` function from the starter code. The number of nodes in your three inputs must be 100, 300, and 1,000 for `small.in`, `medium.in` and `large.in`, respectively.

Submission Details

You will receive full points for your inputs so long as they follow the input format and submission guidelines. **Only one member of your team should submit, and they should add the other members on Gradescope.** In addition to submitting the three `.in` files to their respective Gradescope assignments, you will submit a `.txt` file containing your team name. **Please keep team names civil, PG-13 and no more than 30 characters long**. In order to get out a complete list of inputs to students in a timely fashion, there will be **no late submissions**.

Phase 2: Solver Phase (Due 12/4, 11:59pm)

We strongly suggest you start working on Phase 2 early. You should begin working on solvers *before* the Phase 1 deadline.

Overview

You will be provided the pool of inputs generated by the class. Your team will submit an output (i.e. a partition) for each input provided in a `.tar` file. The Gradescope autograder will check whether your solutions are valid and evaluate their costs. Your grade for this phase will be determined in part by how your solver performs compared to those of other students. In addition to your outputs, you will **write a reflection** on the approaches you tried and how they performed.

Teams are free to choose any programming language they wish, so long as the input and output files follow the specified format. Given the variety of possible programming languages and packages, we cannot guarantee that course staff will be able to provide support in office hours. The scores of each team on each input will be updating live on a leaderboard website.

Services, Libraries, Tools

If you use non-standard libraries or computing resources beyond your personal computer, **you may only use free services and tools**. Services and tools which are normally paid are okay if used under free, easily available academic licenses. This includes things like free AWS credits for students. If you use AWS or any other cloud computing service, you must cite how you used it, and how much, in your project report. If you use any non-standard libraries, you need to explicitly cite which you used, and describe in your reflection document why you chose to use them. If you choose to use the instructional machines[1], **you may only use one at a time per team**. **Anybody caught using multiple instructional machines will receive a zero for this part of the project**. The reason we have this rule is that in the past CS170 has overloaded instructional machines and we do not want to make these resources unavailable to other students.

Output Format

The output file for a given input file should have the same file name, but with the `.out` file extension. The file format is JSON, containing only a list of team numbers for each vertex, in order. Once again, the starter code has `read_output` and `write_output` functions to help with parsing and formatting. The validity of a solution can be checked with the `validate_output` function.

Submission Details

You should submit a `.tar` file of your team's output folder to the Gradescope assignment. For your convience, the starter code has a `tar` function to create a `.tar` file from a directory of outputs. You may submit many times; your team's best output for each input will be updated live on a leaderboard website. Adding team members to each submission isn't necessary — once you've submitted your team name in Phase 1, your teammates will automatically be added to each submission in Phase 2.

Reflection

You will also submit a project reflection in addition to your code. **Your reflection should address the following questions:**

- Describe the algorithm you used to generate your outputs. Why do you think it is a good approach?
- What other approaches did you try? How did they perform?
- What computational resources did you use? (e.g. AWS, instructional machines, etc.)

Your reflection should be **no more than 1000 words** long, although it may be shorter as long as you respond to the questions above. You will also submit the code for your solver, along with a README containing precise instructions on how to run it. If we cannot parse your instructions then you run the risk of receiving a penalty to your overall grade. We should be able to replicate all your results using the code you submit as well as your project report. More details on how to submit your code and project report will be released closer to the Phase 2 deadline.

Grading

Overall, this project is worth 5% of your final grade. You will earn these points as follows:

[1] *Note on accessing instructional machines:* to use the instructional machines, first access EECS Acropolis and get your account credentials (e.g. `cs170-abc`). Once you have your account, SSH into one of the instructional machines listed on Hivemind (e.g. `ssh cs170-abc@asbhy.cs.berkeley.edu`). You should now have terminal access to your instructional account.

- 1% will come from your inputs.
- 1% will come from your reflection.
- 3% will come from the the quality of your outputs.

We encourage students to create the best solver they possibly can, and as staff we love to see a healthy competitive spirit in project groups. However, we'd like to emphasize that the point of this project is not to be purely an evaluation of your abilities against those of your peers. We really want to encourage students to view this as an opportunity to apply what they've learned over the semester to an open-ended project in an exploratory way. Do your best, try approaches that sound interesting to you, and have fun!

We will not accept late submissions. Submissions made after the deadline will not be considered.

Academic Honesty

Here are some rules and guidelines to keep in mind while doing the project:

1. No sharing of any files (input files or output files), in any form.
2. No sharing of code, in any form.
3. Informing others about available libraries is encouraged in the spirit of the project. You are encouraged to do this openly, on EdStem.
4. Informing others about available research papers that are potentially relevant is encouraged in the spirit of the project. You are encouraged to do this openly, on EdStem.
5. Informing others about possible reductions is fine, but treat it like a homework question – you are not to give any substantial guidance on how to actually think about formulating the reduction to anyone not in your team.
6. As a general rule of thumb: don't discuss things in such detail that after the discussion, the other teams write code that produces effectively the same outputs as you. This should be a general guideline about how deep your discussions should be.
7. If in doubt, ask us. Ignorance is not an excuse for over-collaborating.

If you observe a team cheating, or have any reason to suspect someone is not playing by the rules, please report it here.

As a final note from the staff, we generally trust that students will make the right decisions when it comes to academic honesty, and can distinguish collaboration from cheating. However, we'd like to point out that what has made this project so interesting in the past is the diversity of student approaches to a single problem. We could have elected to give you yet another problem set, but we believe that the open-ended nature of this project is a valuable experience to have. Do not deprive yourselves (or us) of this by over-collaborating or simply taking the same approaches you find your peers using. Again, try your best and have fun!