**Fair Spectral Clustering: Methodology and Implementation Guide**

**1. Problem Statement & Fairness Motivation**

**Goal:**  
Partition a graph (e.g., social network) into clusters (communities), but **ensure every demographic group (e.g., gender)** is proportionally represented in each cluster—i.e., clusters should be “fair” (not segregated or imbalanced).

**Standard Spectral Clustering:**  
Find clusters using graph Laplacian eigenvectors, unconcerned with demographics.[[1]](#fn1)

**Fair Spectral Clustering:**  
Impose an extra **linear constraint**—forcing clusters to have demographic proportions similar to the entire dataset.[[1]](#fn1)

**2. How Fairness Constraints Are Incorporated**

**A. Mathematical Framework**

* **Spectral Clustering**: Embed nodes in lower-dimensional space using first eigenvectors of (normalized or unnormalized) graph Laplacian, then apply k-means clustering.
* **Fairness via Linear Constraint**:
  + Let be the number of demographic groups (e.g., gender is 2: M/F).
  + For group , define a group-membership vector .
  + Impose that, **for every group and every cluster**, the proportion in the cluster matches the proportion in the whole graph.
* This translates into a **linear constraint**: the embedding matrix (rows: nodes, columns: k embedding dimensions) must lie in the nullspace of a matrix encoding the fairness constraints.[[1]](#fn1)
* **Algorithm Sketch:**
  + Build Laplacian (unnormalized: , normalized: see Appendix Algorithm 3).
  + Compute matrix capturing group memberships minus overall group proportions.
  + Project Laplacian into the fairness-constrained subspace using the nullspace of .
  + Find smallest eigenvectors of the projected Laplacian (and, for normalized, some extra technical steps).
  + Cluster rows, as usual, using k-means.

**3. Python Implementation Walkthrough**

**Common Steps**

* **Load Network and Metadata:**
  + Friendship graph: edges are friendships between students.
  + Metadata: node IDs with gender labels.
* **Preprocessing:**
  + Remove nodes with unknown gender.
  + Keep only nodes with metadata.
  + Keep the largest connected component for analysis.
* **Construct Adjacency Matrix , Degree Matrix , Laplacian** .

**A. Unnormalized Fair Spectral Clustering (unnormalized-friendshipnet.py)**

**Algorithm (corresponds to Alg. 2 in paper):**

1. **Build group-membership vector ():**
   * Binary vector: 1 if female, 0 if male.
   * Center: minus proportion of females.
2. **Build fairness constraint matrix :**
   * Just one column if two groups.
3. **Find nullspace () of :**
   * Only embeddings orthogonal to this fairness direction are allowed.
4. **Project Laplacian:**
   * Compute .
5. **Get Eigenvectors:**
   * Smallest eigenvectors of .
6. **Get Fair Embedding:**
   * (Y = eigenvector matrix).
   * Row-normalize for k-means.
7. **Run k-means for clustering.**
8. **Evaluate Clusters:**
   * *Balance*: For each cluster, min (, ); higher means more proportional.
   * *RatioCut*: Standard spectral cut objective (lower is better).
9. **Repeat for multiple (number of clusters) and multiple random initializations to average results.**
10. **Compare vs. standard spectral clustering (no fairness constraint).**
11. **Plot and print outputs.**

**B. Normalized Fair Spectral Clustering (normalized-friendshipnet.py)**

**Algorithm (corresponds to Alg. 3 in paper):**

1. **Same as above to find fairness constraint () and nullspace ().**
2. **Compute Projected Degree Matrix (), Square Root and Inverse (, ).**
3. **Form projected Laplacian:**
4. **Find eigenvectors and use for clustering (follow the same steps for embedding, normalization, and k-means).**
5. **Evaluate using Ncut (Normalized Cut) and balance.**
6. **Repeat for multiple , average results, and compare vs. standard normalized spectral clustering.**

**4. Core Functions**

* **load\_and\_preprocess\_data:** Loads graph/metadata and filters data.
* **compute\_balance:** Measures fairness (proportionality of groups) for each cluster.
* **compute\_ratiocut / compute\_ncut:** Standard clustering quality metrics.
* **run\_analysis\_and\_get\_results / run\_normalized\_analysis:** Runs clustering, collects and averages metrics per .
* **print\_and\_plot / print\_and\_plot\_normalized:** Produces tabular output and dual-axis plots showing tradeoff between fairness and clustering quality.

**5. Interpreting Results**

* **Balance Improvement**: Plots show fair spectral clustering (dashed line) achieves higher cluster balance than standard SC across a range of cluster numbers.
* **RatioCut/NCut Cost**: Often, enforcing fairness does **not substantially worsen** the objective (cut value)—sometimes increased fairness is achieved at minimal quality cost.[[1]](#fn1)
* **Baselines**: Dataset’s demographic balance is shown as reference; higher cluster balance is desirable.

**6. Reproducibility, Links to Theory**

* Each step directly implements the algorithms from the paper, including fairness constraint matrix construction, Laplacian projection, and nullspace embedding.[[1]](#fn1)
* The normalized and unnormalized variants correspond to Algorithms 2 and 3 (main text and Appendix).
* Evaluations and plotting match those in the paper Figures 5 and related experiments.

**7. How to Adapt/Extend**

* The framework supports **arbitrary group assignments** (not just gender).
* For more than two groups, add more columns to (see paper).
* Can use for other networks, other metadata, or other fairness metrics, as long as you adjust group membership definition.

**8. Summary Table: Implementation Mapping**

|  |  |  |  |
| --- | --- | --- | --- |
| Step | Code function/section | Paper (Algorithm) | Output/Plot |
| Load/prep data | load\_and\_preprocess\_data | Data input | Network graph |
| Group fairness setup | F, null\_space(F.T) | Eq. (6), (19) | - |
| Laplacian | Laplacian computations | Alg. 2 & 3 | - |
| Projection | L\_fair, or L\_fair\_sym | Algorithms 2 & 3 | - |
| Eigenvectors | eigendecomposition | Algorithms 2 & 3 | - |
| Embedding/clustering | kmeans, normalize/embed | Algorithms 2 & 3 | Cluster labels |
| Evaluation | compute\_balance, cut funcs | Figures 2, 5 | Balance, Cut |
| Plotting | print\_and\_plot | Matches paper figures | Balance vs. Cut |

**This guide will help users/readers understand both the mathematical logic and the exact code implementation, and shows how fairness was built into spectral clustering on your real-world dataset, following the referenced paper’s framework.**[[1]](#fn1)

⁂