Appendix: Quantifying and Reducing Imbalance in Networks

Yoosof Mashayekhi
Bo Kang
yoosof.mashayekhi@ugent.be
bo.kang@ugent.be
Ghent University
Ghent, Belgium

Jefrey Lijffijt
Tijl De Bie
jefrey.lijffijt@ugent.be
tijl.debie@ugent.be
Ghent University
Ghent, Belgium

A NETWORK IMBALANCE IS EQUIVALENT TO EMD

Earth Mover's Distance (EMD) is a measure of the distance between two distributions. In this case, EMD computes the minimum amount of work (the distance of movement of nodes in the embedding space) to match the embedding of the source nodes and the target nodes. We show that EMD computes the same value as the network imbalance ψ. To show that, we first introduce EMD.

EMD was invented to solve certain kinds of transportation problems [?]. The transportation problem is a particular type of linear programming where the objective is to minimize the cost of transporting any commodity from one group of sources to another group of destinations. Formally:

Definition 1 (Earth Mover's Distance [?]). *Assume two distributions represented by signatures $\boldsymbol{P} = \{(\boldsymbol{p}_1, w_{p1}), ..., (\boldsymbol{p}_m, w_{pm})\}$ and $\boldsymbol{Q} = \{(\boldsymbol{q}_1, w_{q1}), ..., (\boldsymbol{q}_r, w_{qr})\}$, where $\boldsymbol{p}_i$ and $\boldsymbol{q}_j$ are cluster representatives and w_{pi} and w_{qj} are weights of clusters. Let $\boldsymbol{D} = [d_{ij}]$ be the distance matrix between $\boldsymbol{P}$ and $\boldsymbol{Q}$. EMD is a linear program whose goal is to find a flow $\boldsymbol{F} = [f_{ij}]$ between two distributions $\boldsymbol{P}$ and $\boldsymbol{Q}$ that minimizes the overall cost:*

$$C = \sum_{i=1}^{m} \sum_{j=1}^{r} f_{ij} d_{ij}$$

$$s.t. \quad f_{ij} \geq 0, \ \ 1 \leq i \leq m, \ \ 1 \leq j \leq r,$$

$$\sum_{j=1}^{r} f_{ij} \leq w_{pi}, 1 \leq i \leq m, \tag{1}$$

$$\sum_{i=1}^{m} f_{ij} \leq w_{qj}, 1 \leq j \leq r, \tag{2}$$

$$\sum_{i=1}^{m} \sum_{j=1}^{r} f_{ij} = min\{\sum_{i=1}^{m} w_{pi}, \sum_{j=1}^{r} w_{qj}\}. \tag{3}$$

The optimal flow $\boldsymbol{F}$ is found by solving this linear optimization problem. D_{EMD} is defined as the work normalized by the total flow:

$$D_{EMD}(\boldsymbol{D}, \boldsymbol{P}, \boldsymbol{Q}) = \frac{\sum_{i=1}^{m} \sum_{j=1}^{r} f_{ij} d_{ij}}{\sum_{i=1}^{m} \sum_{j=1}^{r} f_{ij}}.$$

EMD, which is a cross-bin distance measure, computes the distance between two distributions (source nodes and target nodes in our case). Note that bin-bin distance measures such as Kullback–Leibler divergence do not take the distance between values into account.

We use D_{EMD} to compute imbalance in a network and we refer to it as $\boldsymbol{\psi_{EMD}}$. We assign equal weights to each node in S and also equal weights to each node in T in a way that the sum of the weight of nodes in S and T are equal. Formally:

Definition 2 (Imbalance Measure ψ_{EMD}). *Given a network $G = (V, E)$, two disjoint sets of nodes, namely source nodes $\emptyset \subset S \subset V$ and target nodes $\emptyset \subset T \subset V$, and the cost of matching each pair of nodes $\boldsymbol{D} = [d_{ij}]$, we define two signatures $\boldsymbol{C_S} = \{(i, w_i = \frac{1}{|S|}) | i \in S\}$ and $\boldsymbol{C_T} = \{(j, w_j = \frac{1}{|T|}) | j \in T\}$. We define imbalance measure ψ_{EMD}:*

$$\psi_{EMD}(\boldsymbol{D}, S, T) = D_{EMD}(\boldsymbol{D}, \boldsymbol{C_S}, \boldsymbol{C_T}).$$

Proposition 1 (Equivalence of ψ_{EMD} and ψ). *ψ_{EMD} equals to ψ.*

Proof. We show that ψ_{EMD} solves the same problem as ψ. Since $w_i = \frac{1}{|S|}, i \in S$ and $w_j = \frac{1}{|T|}, j \in T$ (Definition ??), $\sum_{i \in S} w_i = \sum_{j \in T} w_j = 1$. Since constraint ?? becomes $\sum_{i \in S} \sum_{j \in T} f_{ij} = \sum_{i \in S} w_i = \sum_{j \in T} w_j = 1$, constraints ?? and ?? also change to $\sum_{j \in T} f_{ij} = w_i, i \in S$ and $\sum_{i \in S} f_{ij} = w_j, j \in T$. Hence, we can rewrite ψ_{EMD} as the linear program:

$$C = \sum_{i \in S} \sum_{j \in T} f_{ij} d_{ij}$$

$$s.t. \quad f_{ij} \geq 0 \quad \forall (i,j) \in S \times T,$$

$$\sum_{j \in T} f_{ij} = w_i \quad \forall i \in S, \quad \sum_{i \in S} f_{ij} = w_j \quad \forall j \in T,$$

which solves the same problem as ψ. □

B GRAB ALGORITHM

Algorithm ?? shows the greedy link selection in `GraB`. Algorithm ?? the complete generic method `GraB` to select k links to add to the network.

Algorithm 2: Graph Balancing (GraB)

```
Input: $NE$(network embedding method), $\boldsymbol{A}$ (adjacency matrix), $S$ (source nodes), $T$ (target nodes), $U$ (auxiliary nodes), $k$ (number of links to add), $b$ (batch size), $l$ (batch size coefficient)
Output: $links$ (k links)
Function GraB($NE, \boldsymbol{A}, S, T, U, k, b, l$):
    $links$ = [], $links\_idx$ = 1
    $\boldsymbol{X}$ = $NE$.Embeddings($\boldsymbol{A}$)
    while $links\_idx \leq k$ do
        $candidate\_list$ = SelectCandidateLinks($NE$, $\boldsymbol{A}$, $S$, $T$, $U$, $b \cdot l$)
        $\boldsymbol{X\_new}$, $\boldsymbol{A\_new}$ = $NE$.ReEmbed($\boldsymbol{X}$, $\boldsymbol{A}$, $candidate\_list$) // Re-embed the network after adding the $candidate\_list$ links
        $current\_batch\_list$ = [], $idx$ = 1
        foreach $(i, j)$ in $candidate\_list$ do
            if $\delta(\boldsymbol{x}_i, \boldsymbol{X}, S, T) < \delta(\boldsymbol{x\_new}_i, \boldsymbol{X\_new}, S, T)$ then
                $links[links\_idx]$ = $(i, j)$
                $links\_idx$ += 1
                $current\_batch\_list[idx]$ = $(i, j)$
                $idx$ += 1
                if $idx > b$ then
                    break
                end
            end
        end
        $\boldsymbol{X}$, $\boldsymbol{A}$ = $NE$.ReEmbed($\boldsymbol{X}$, $\boldsymbol{A}$, $current\_batch\_list$)
         // Re-embed the network after adding the $current\_batch\_list$ links
    end
    return $links$
```