Networks - Week 7 - Scaling Properties of Networks & Decomposing Networks

Antonio León Villares

November 2023

Contents

Based on the notes by Renaud Lambiotte and Pete Grindrod, Chapters 8 & 10

1 Growing Networks

1.1 Scaling Laws: Combining Graphs from Set

1.1.1 Notation

This section focuses on how we can construct **new graphs** by **successively** merging smaller graphs together (in a way this is dual to the process of **community detection**). For this:

- let $\mathcal{S}$ be the **set of undirected graphs** over a **finite** number of **vertices**
- if $S \in \mathcal{S}$, then:
  - $V(S)$ denotes its **vertex set**
  - $E(S)$ denotes its **edge** set
- $\diamondsuit$ denotes a **binary** and **commutative** operation on **graphs** in $\mathcal{S}$ (for example, a **disjoint union** of 2 graphs)
- if we apply $\diamondsuit$ to $S \in \mathcal{S}$ n times, we denote this via:

$$nS = \underbrace{S \diamondsuit \ldots \diamondsuit S}_{n}$$

- a **Banach Space** χ is a **complete normed vector space**:
  - **complete**: every Cauchy Sequence in χ converges to some limit in χ
  - **normed**: there exists a **norm** $\|\cdot\|$, such that for any vector in the space, the norm is **non-negative**, **positive definite**, satisfies the **triangle inequality** and if λ is a scalar, $\|\lambda \underline{v}\| = |\lambda| \|\underline{v}\|$

  For example, $\mathbb{R}$ is a **Banach Space**

1.1.2 Definition: Well-Behaved Scoring Function

When dealing with growing networks, we want to have some function which tells us how certain network properties (both mathematical, like degree distribution, but also properties inherent to the data encoded in the network, such as when analysing biological networks. For this we need functions which are well-behaved: that is, they behave smoothly given small changes in the edge set.

---

*Let $\mathcal{B}$ be a **Banach Space**, and consider functions*

$$Q : \mathcal{S} \to \mathcal{B}$$

*Then, Q is **well-behaved** with respect to the **edge set** if:*

$$\exists C > 0 \ : \ \forall S_1 S_1 \in \mathcal{S}, \qquad \|Q(S_2)\|_{\mathcal{B}} - \|Q(S_1)\|_{\mathcal{B}} \leq C \frac{|E(S_2 \setminus S_1)|}{|E(S_1)|}$$

where:

- *$S_2 \in \mathcal{S}$ is the result of **adding edges** of S_1*
- *$|E(S_2 \setminus S_1)| = |E(S_2) - E(S_1)|$ is the change in the number of edges*

In particular, this says that Q is well-defined whenever if $E(S_1) \to \infty$ and adding a fixed number of edges becomes negligible, we get that:

$$\|Q(S_2)\|_{\mathcal{B}} - \|Q(S_1)\|_{\mathcal{B}} \to 0$$

- **If Q is a function counting the number of connected components in $S \in \mathcal{S}$, is Q well-defined on the edge set?**
  - suppose G is a graph composed of many disconnected components
  - if we add a **single edge** connecting 2 such components, then:

    $$\|Q(S_2)\|_{\mathcal{B}} - \|Q(S_1)\|_{\mathcal{B}} = 1$$

  - however:

    $$C \frac{|E(S_2 \setminus S_1)|}{|E(S_1)|} = \frac{C}{|E(S_1)|}$$

    so no such $C > 0$ can exist for **every** $S_1 \in \mathcal{S}$, since we'd require that:

    $$1 < \frac{C}{|E(S_1)|}$$

    always

1.1.3 Definition: Scaling Function

A ***scaling function*** *for:*

- *a* ***Banach Space*** $\mathcal{B}$
- *a* ***well-defined map*** $Q : \mathcal{S} \to \mathcal{B}$
- *a* ***binary*** *and* ***commutative*** *operation on* $\mathcal{S}$ $\diamondsuit$

is a ***mapping***

$$H : \mathcal{B} \times \mathbb{Z}^+ \to \mathcal{B}$$

describing how Q *behaves on* nS*, given how* Q *behaves on* S*:*

$$\forall S \in \mathcal{S}, n \in \mathbb{Z}^+, \qquad Q(nS) = H(Q(S), n)$$

For any $(\mathcal{S}, \diamondsuit, \mathcal{B}, Q)$*, there is no* ***guarantee*** *that a* ***scaling function*** *even exists.*

1.1.4 Proposition: Lipschitz Condition on Scaling Functions

A function $f : \mathbb{R} \to \mathbb{R}$ *is Lipschitz Continuous if there exists a constant* L *such that*

$$\forall x, y \in \mathbb{R}, \qquad |f(x) - f(y)| \leq L|x - y|$$

Lipschitz Continuous functions are smooth and very well-behaved. We translate Lipschitz Continuity to our functions $Q : \mathcal{S} \to \mathcal{B}$*, and use this to provide a sufficient condition for the existence of sacling functions.*

Let $\mathcal{B}$ *be a* ***Banach Space****, and consider a well-defined function on edges:*

$$Q : \mathcal{S} \to \mathcal{B}$$

Suppose that $\forall S \in \mathcal{S}$*, there exists* $C(S) > 0$ *such that:*

$$\forall S_1, S_2 \in \mathcal{S}, \qquad \|Q(S_1 \diamondsuit S) - Q(S_2 \diamondsuit S)\|_{\mathcal{B}} \leq C(S_0)\|Q(S_1) - Q(S_2)\|_{\mathcal{B}}$$

Then, there exists a ***scaling function***

$$H : \mathcal{B} \times \mathbb{Z}^+ \to \mathcal{B}$$

satisfying:

$$Q(nS) = H(Q(S), n)$$

1.1.5 Proposition: Power-Law Scaling Functions

A solution to the ***functional equation***

$$Q(nS) = H(Q(S), n)$$

given the ***initial condition****:*

$$H(Q, 1) = Q$$

is given by:

$$H(Q, n) = Qn^{-\alpha} + \beta(1 - n^{-\alpha})$$

where $\alpha \in \mathbb{R}, \beta \in \mathcal{B}$ *are* ***constants****.*

Proof. We begin by noting that:

$$Q(nmS) = Q(n(mS)) = H(Q(mS), n) = H(H(Q, m), n)$$

so a (defining) property of scaling functions is:

$$Q(nm) = H(H(Q, m), n)$$

(where we recover the original definition by setting $m = 1$ and using the initial condition $H(Q, 1) = Q$).

Now, for any $\alpha \in \mathbb{R}$, we claim that:

$$H(Q, n) = Qn^{-\alpha}$$

satisfies the properties of a scaling function. Indeed, for any $S \in \mathcal{S}$:

$$Q(nm) = Q(nm)^{-\alpha} = (Qm^{-\alpha})n^{-\alpha} = (H(Q, m))n^{-\alpha} = H(H(Q, m), n)$$

and it also satisfies the initial condition:

$$H(Q, 1) = Q$$

Now, assuming H is non-trivial (so that $\alpha \neq 0$), we may assume that the (general) form of H is given by:

$$H(Q, n) = Qn^{-\alpha} + g(n)$$

where so that H satisfies the initial condition

$$g(1) = 0$$

Assuming that for $n, m \in \mathbb{R}$ we have:

$$Q(nm) = H(H(Q, m), n)$$

we must have that:

$$Q(nm)^{-\alpha} + g(nm) = H(Q, m)n^{-\alpha} + g(n) = (Qm^{-\alpha} + g(m))n^{-\alpha} + g(n)$$

which implies that:

$$g(nm) = g(m)n^{-\alpha} + g(n)$$

If we differentiate with respect to m:

$$g'(mn)n = g'(m)n^{-\alpha} \implies g'(mn) = g'(m)n^{-\alpha-1}$$

Now, setting $m = 1$:

$$g'(n) = g'(1)n^{-\alpha-1}$$

If we define:

$$g'(1) = \alpha\beta$$

for some $\beta \in \mathcal{B}$, we thus obtain:

$$g'(n) = \alpha\beta n^{-\alpha-1}$$

This is a differential equation, which can be solved by separation of variables:

$$g(n) = \int \alpha\beta n^{-\alpha-1} dn = -\beta n^{-\alpha} + C$$

and using the initial condition $g(1) = 0$ we have that

$$0 = -\beta + C \implies C = \beta$$

so:

$$g(n) = \beta(1 - n^{-\alpha})$$

Thus:

$$H(Q, n) = Qn^{-\alpha} + \beta(1 - n^{-\alpha})$$

as required.

□

1.1.6 Remark: Lipschitz Scaling Function for Non-Commutative Operations

*If $\diamondsuit$ is non-commutative, we require that Q is **Lipschitz Continuous** from the left and right. In particular, if $\forall S \in \mathcal{S}$ there exists $C(S) > 0$ such that $\forall S_1, S_2 \in \mathcal{S}$:*

$$\begin{aligned} \max\{&\|Q(S_1 \diamondsuit S) - Q(S_2 \diamondsuit S)\|_{\mathcal{B}}, \\ &\|Q(S \diamondsuit S_1) - Q(S \diamondsuit S_2)\|_{\mathcal{B}}\} \\ &\leq C(S_0)\|Q(S_1) - Q(S_2)\|_{\mathcal{B}} \end{aligned}$$

Then, there exists a scaling function H, such that:

$$Q(nS) = H(Q(S), n)$$

1.2 Scaling Laws: Combining Graphs from Distribution

1.2.1 Notation

- let $\mathcal{S}$ be the **set of undirected graphs** over a **finite** number of **vertices**

- a **random graph** W is a **probability distribution** $P_W(S)$ defined over $\mathcal{S}$ (i.e with an independent probability for each edge, or range-dependent graphs like in the small world configuration)
- let $\mathcal{W}$ denote the set of all **random graphs**
- $\square$ denotes a **binary** and **commutative** operation on **random graphs** in $\mathcal{W}$
- if we apply $\square$ to $W \in \mathcal{W}$ n times, we denote this via:

$$nW = \underbrace{W \square \ldots \square W}_{n}$$

1.2.2 Proposition: Lipschitz Condition on Scaling Functions for Random Networks

*Let $\mathcal{B}$ be a **Banach Space**, and consider a well-defined function on edges:*

$$Q : \mathcal{W} \to \mathcal{B}$$

Suppose that $\forall W \in \mathcal{W}$, there exists $C(W) > 0$ such that:

$$\forall W_1, W_2 \in \mathcal{W}, \qquad \|Q(W_1 \diamondsuit W) - Q(W_2 \diamondsuit W)\|_{\mathcal{B}} \leq C(W_0)\|Q(W_1) - Q(W_2)\|_{\mathcal{B}}$$

*Then, there exists a **scaling function***

$$H : \mathcal{B} \times \mathbb{Z}^+ \to \mathcal{B}$$

satisfying:

$$Q(nW) = H(Q(W), n)$$

1.2.3 Example: Preferential Attachment Stochastic Block Model

- recall the **preferential attachment model**

*The **BA Model** constructs an **evolving network** using the following steps:*

1. *Consider n_0 initial **vertices**, each with degree at least one (i.e a **clique**)*
2. *Add a new **vertex** to the network, with $m < n_0$ **half-edges**. If the network has n' **vertices** (initially $n' = n_0$) with degrees d_i, the probability that a **half-edge** connects to v_i is given by:*

$$\Pi(d_i) = \frac{d_i}{\sum_{j=1}^{n'} d_j}, \quad i \in [1, n]$$

   *This is the **preferential attachment mechanism**: **vertices** with **higher degree** are more likely to get attached to. However, this must be carried out carefully*[a]

3. *Continue repeating step 2 until we reach a desired number of vertices n.*

[a] During this step, we should avoid generating **multiple edges** between 2 vertices. Moreover, it is a design decision whether we need to update the d_i as new edges are generated through this process

- this can be generalised in the following way:
  - fix parameters K (number of **nodes**) and p (probability of generating an **edge**)
  - these define a **Erdös-Rényi Graph** $G(K, p)$
  - we define an **existing combination graph** (ECG), initiliased as a single graph fdrawn from $G(K, p)$
  - then, we generate a new graph Γ from $G(K, p)$
  - using the **vertex degree distributions** of the ECG and Γ, we select 2 vertices in the ECG and Γ to join via an edge (this is nothing but **preferential attachment**, whereby the vertices of higher degree in the 2 graphs are most likely to be joined)
  - after n steps, this results in a random graph with Kn vertices (called a **PASBM graph**)
- in this case, $\square$ corresponds to degree-biased, random preferential attachment
- moreover, if $K = 1$ (so that we are joining single nodes), this defaults to the standard preferential attachment model

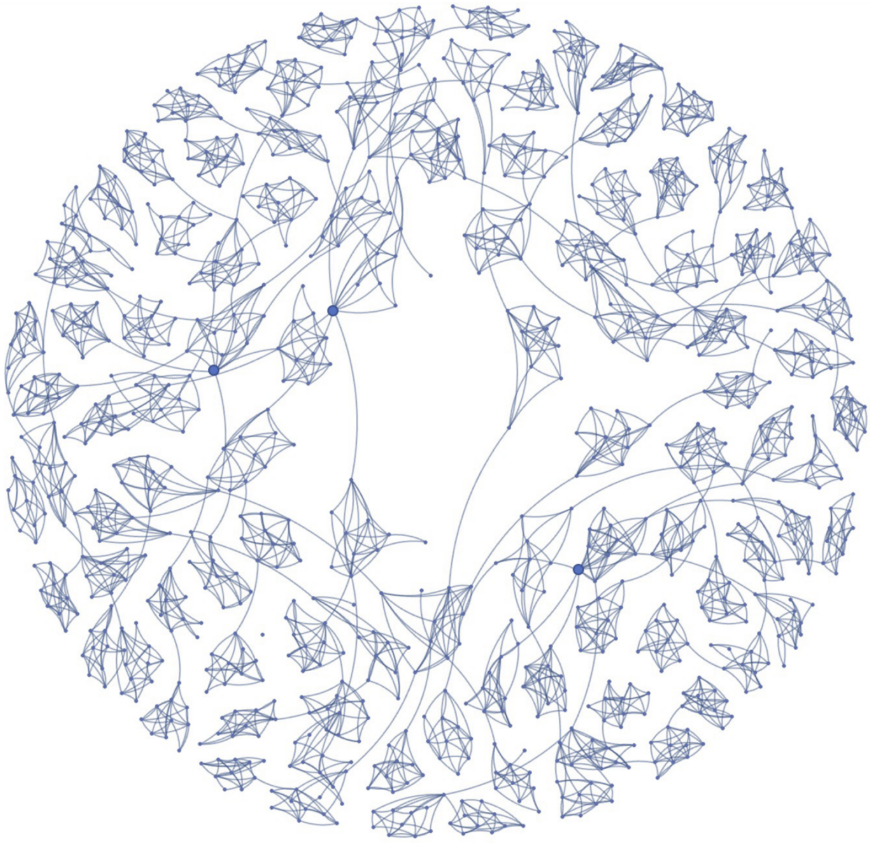

Figure 1: Example of the PASBM model. The 3 vertices with the highest degree have been highlighted. This used $K = 10, p = 0.6, n = 100$.

1.3 Alternative Random Graph Combination Operations

1.3.1 Mechanistic Aggregative Combination

- can be thought of to further **generalise preferential attachment**
- given 2 graphs, we randomly join 2 nodes according to a distribution **proportional** to the rth power of the **vertex degree distributions** of the graphs:
  - as $r \to 0$, this is **non-preferential attachment** (since we uniformly randomly select 2 vertices to join, one from each graph)
  - as $r \to \infty$, this is **most popular vertex attachment** (since we will jsut join the vertices of maximum degree within each graph)
- to further generalise this, at each **step**, we might choose to join $J > 1$ edges, so called **multiple-edge preferential attachment**

1.3.2 Product Combination

- we can add graphs **hierarchically**
- each **random graph** is defined by a **core vertex**; non-core vertices are called **peripheral**
- we initialise a graph G

- at each iteration, we expand G, by generating a new graph Γ, and then attaching **peripheral vertices** of Γ to the **core node** of G
- notice here that the combination operator $\square$ will **not** be commutative

1.3.3 Probability Distribution Combination

- since **random graphs** are **probability distributions**, we can generate new random graphs by combining their probability distributions
- let
$$h : [0,1]^2 \to \mathbb{R}^+$$
is a **commutative map** of 2 variables
- if W_1, W_2 are **random graphs**, then we can define $\square$ via:
$$P_{W_1 \square W_2}(S) = \frac{h(P_{W_1}(S), P_{W_2}(S)))}{\sum_{S' \in \mathcal{S}} h(P_{W_1}(S'), P_{W_2}(S')))}$$
- if $h(x,x) = x$, then $W \square W = W$

2 Decomposing Trees

In this article they discuss how the Hodge decomposition can be used analyse Bitcoin Money Flow.

2.1 Definition: Flow Trees

*A **tree** is a **connected graph** with no **cycles**.*

---

*If a **tree** is a **weighted graph**, we can interpret the **weights** as **flow**: that is, if there is an **edge** $v_i \to v_j$ with weight $k \in \mathbb{R}$, we can interpret k as the amount of **flow** between v_i and v_j.*

2.2 Helmholtz-Hodge Decomposition

2.2.1 Motivation: Hodge Decomposition of Vector Fields

- **What is a vector field?**
  - a mapping, which assigns to each point in some **space** a **vector**
  - for example:
$$\underline{F}(x,y,z) = \left\langle x^2 + y, \frac{z}{e^x}, 2 \right\rangle$$
defines a **vector field**
- **What is the divergence of a vector field?**
  - let $\underline{F}$ be a **vector field** on a n dimensional space, defined by variables $\{x_i\}_{i \in [1,n]}$

- the **divergence** of $\underline{F}$ is defined as:

$$\operatorname{div} \underline{F} = \nabla \cdot \underline{F} = \sum_{i=1}^{n} \frac{\partial F_{x_i}}{\partial x_i}$$

- the **divergence** gives a measure of **flow** at a given point in space, due to $\underline{F}$
- for example, if $\underline{F}$ represents the motion of a fluid, and the **divergence** at a point is **positive**, then the point acts like a **source** (fluid emanates from the point); if the divergence is **negative**, the point acts as a **sink** (fluid goes towards the point)

- **What is the gradient of a vector field?**
  - let $\underline{F}$ be a **vector field** on a n dimensional space, defined by variables $\{x_i\}_{i\in[1,n]}$
  - the **gradient** of $\underline{F}$ is another vector field, defined as:

$$\nabla \underline{F} = \left\langle \frac{\partial F_{x_1}}{\partial x_1}, \ldots, \frac{\partial F_{x_n}}{\partial x_n} \right\rangle$$

  - the **gradient** gives the direction of maximum increase of $\underline{F}$

- **What is the Hodge decomposition of a vector field?**
  - we can **decompose** any vector field as:

$$\underline{F}(\underline{r}) = \underline{G}(\underline{r}) + \underline{R}(\underline{r})$$

    where:
    * $\underline{R}$ has 0 **divergence**

$$\nabla \cdot \underline{R}(\underline{r}) = \underline{0}$$

      This can be interpreted as there being no flow.
    * $\underline{G}$ is the **gradient** of some other vector field:

$$\exists \Phi \ : \ \underline{G}(r) = -\nabla \underline{\Phi}(\underline{r})$$

2.2.2 Proposition: Helmholtz-Hodge Decomposition of Directed Graph

Let T be a ***tree****, with* ***vertex set*** *V. Let:*

- *A be the* ***adjacency matrix*** *of T*
- *B be the* ***weight matrix*** *of T, such that $T_{ij} \in \mathbb{R}^+$ denotes the* ***weight*** *of the edge A_{ij}*

Define:

- *a* ***symmetric matrix***

$$W = \frac{A + A^T}{2}$$

- *a* ***net flow matrix***

$$F = B - B^T$$

  where F_{ij} gives the ***net flow*** *from i to j*

The ***Hodge Decomposition*** *of F is:*

$$F_{ij} = W_{ij}(\phi_i - \phi_j) + F_{ij}^{circ}$$

where:

- *$\phi_i \in \mathbb{R}$ is the* ***Hodge potential*** *at* ***vertex*** *v_i*
- *the matrix defined by F_{ij}^{circ} is the* ***divergence free flow*** *of edge i, j, which is defined to satisfy:*

$$\sum_{j=1}^{n} F_{ij}^{circ} = 0$$

- **How can we compute the Hodge decomposition for any matrix?**
  - using the property of **divergence free flow**, we can write:

$$\sum_{j=1}^{n} F_{ij} = \sum_{j=1}^{n} W_{ij}(\phi_i - \phi_j) = \phi_i \sum_{j=1}^{n} W_{ij} - \sum_{j=1}^{n} W_{ij}\phi_j$$

  - if we let L be the **Laplacian** of W:

$$L = \text{diag}(W\,\mathbf{1}) - W$$

    then if we define:

$$w_i = \sum_{j=1}^{n} W_{ij}$$

we have that:

$$\begin{aligned} -\sum_{j=1}^{n} W_{ij}\phi_j &= \sum_{j=1}^{n}(L_{ij} - w_i\delta_{ij})\phi_j \\ &= \left(\sum_{j=1}^{n} L_{ij}\phi_j\right) - w_i\phi_i \\ &= \sum_{j=1}^{n} L_{ij}\phi_j - \phi_i \sum_{j=1}^{n} W_{ij} \end{aligned}$$

- in other words:

$$\sum_{j=1}^{n} F_{ij} = \sum_{j=1}^{n} L_{ij}\phi_j$$

- since L isn't invertible (as it has a non-trivial nullspace - $L\,\mathbf{1} = \underline{0}$), the system defined by:

$$\sum_{j=1}^{n} F_{ij} = \sum_{j=1}^{n} L_{ij}\phi_j$$

  can't be solved
- however, if we restrict the ϕ_i to be orthogonal to $\mathbf{1}$:

$$\sum_{i=1}^{n} \phi_i = 0$$

  then the system has a unique solution for the ϕ_i
- then, from the ϕ_i, F_{ij} and W_{ij}, we can solve for F°_{ij}

2.3 Helmholtz-Hodge Decomposition from Perturbing Tree

- **What is the effect of adding cycles to a tree in the corresponding Hodge Decomposition?**
  - if we connect **vertices** in the **tree**, such that a **cycle** is formed, then the **divergence free flow** will no longer satisfy:

$$\sum_{j=1}^{n} F^{\circ}_{ij}$$

  - this will also visibly alter the **Hodge Potentials**

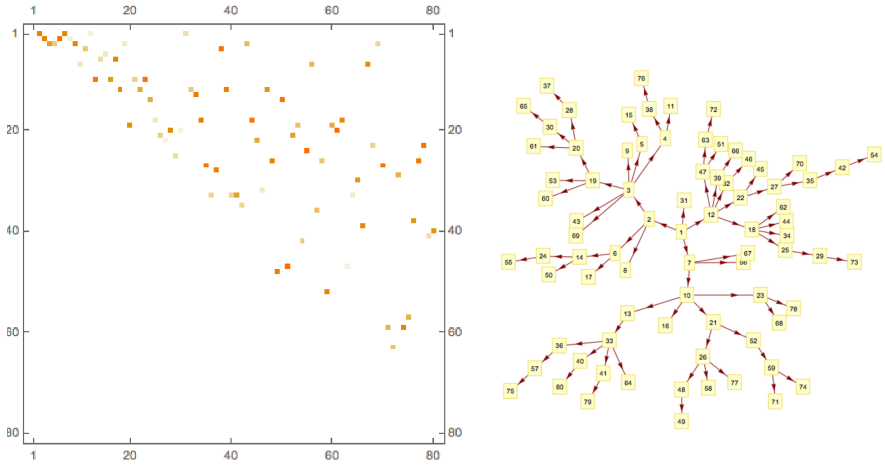

Figure 2: An example of a tree.

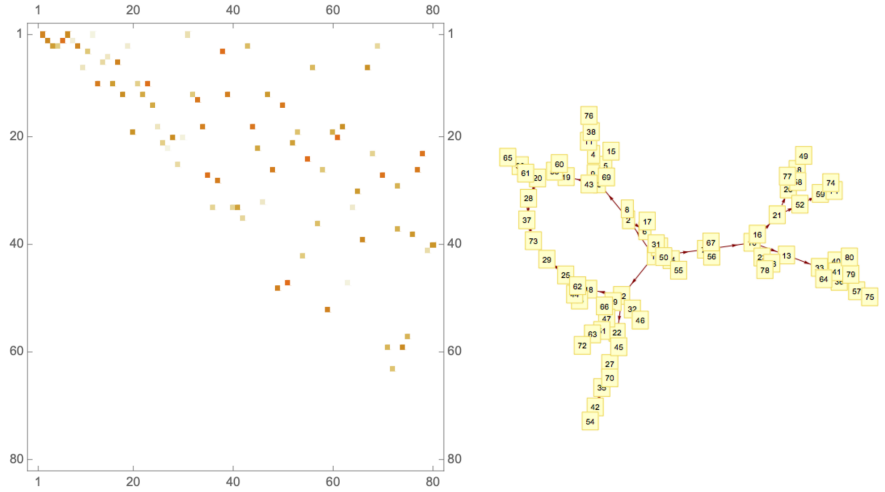

Figure 3: We join 2 vertices together, thus creating a cycle of 12 elements.

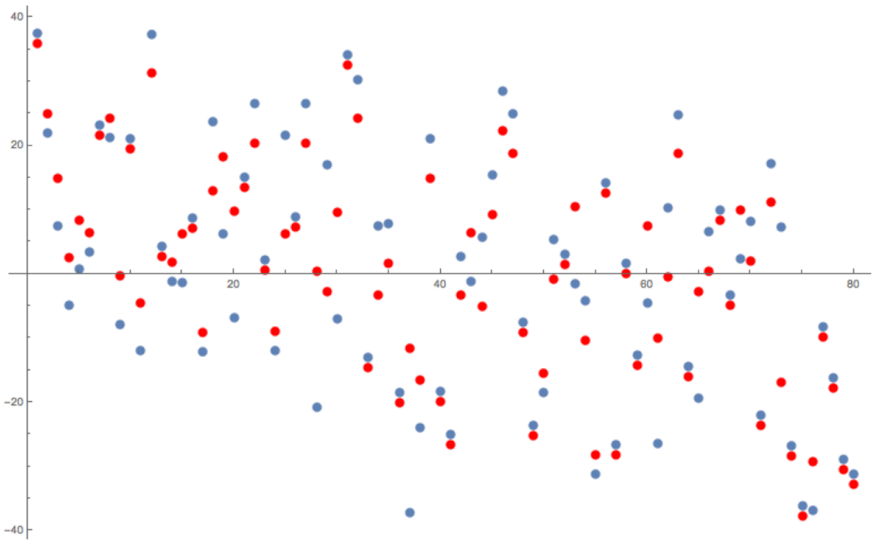

Figure 4: WE then compute the Hodge Potentials. In blue are the potentials for the original tree. In red are the potentials for the graph which now contains a single cycle. Notice the big difference between the before and after.

2.4 Breaking Cycles in Directed Graphs

- **Given a directed graph, what is the best way of eliminating edges to produce a tree with properties similar to the original graph?**
  - we apply the **Hodge decomposition** to the **graph**
  - we thus obtain a matrix F^{circ}
  - if we define:

$$\beta = \sum_{i,j} (F_{ij})^2$$

    then:

$$\beta = 0 \iff \text{graph is a tree}$$

  - thus, we follow an **iterative process**: at each step, delete the **edge** which leads to the largest decrease in β, until $\beta = 0$
  - this generates a **tree** (which is simpler to work with) but which has **algebraic properties** similar to the original graph

---

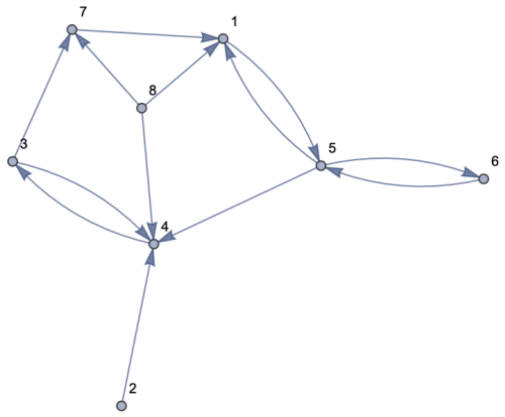

Figure 5: Consider the following (directed) graph.

$$A = \begin{pmatrix} 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \end{pmatrix} \quad B = \begin{pmatrix} 0 & 0 & 0 & 0 & 5.715 & 0 & 0 & 0 \\ 0 & 0 & 0 & 6.651 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.390 & 0 & 0 & 2.356 & 0 \\ 0 & 0 & 2.250 & 0 & 0 & 0 & 0 & 0 \\ 6.897 & 0 & 0 & 0.3282 & 0 & 1.215 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1.511 & 0 & 0 & 0 \\ 5.144 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 3.574 & 0 & 0 & 8.096 & 0 & 0 & 5.891 & 0 \end{pmatrix}$$

Figure 6: The **adjacency** and **weight** matrices are given by A and B respectively.

$$F^{Circ} = \begin{pmatrix} 0 & 0 & 0 & 0 & 0.407 & 0 & 0 & 3.113 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 2.691 & 0 \\ 0 & 0 & 2.691 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.407 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 3.520 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 2.284 & 0 & 0 & 0.829 & 0 \end{pmatrix}$$

Figure 7: If we apply the **Hodge Decomposition**, then we obtain the following matrix for F^{circ}.

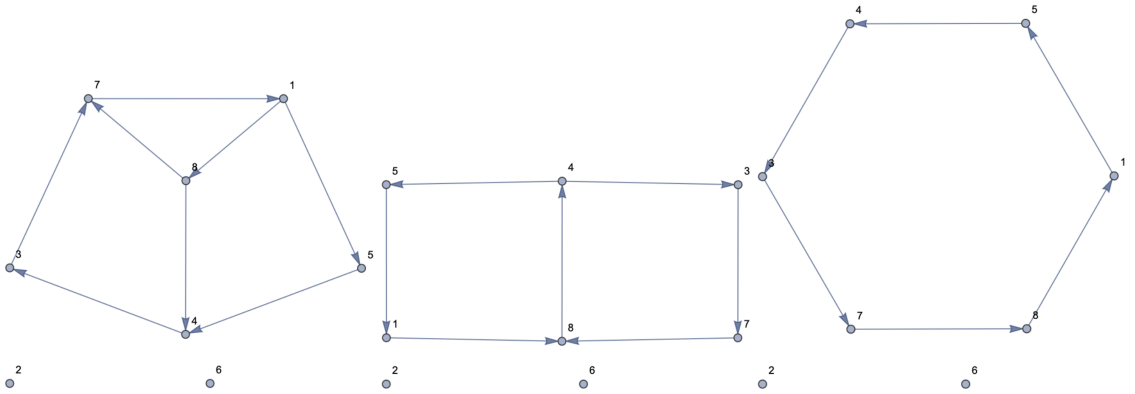

Figure 8: Successively removing the edges which most decrease β, we obtain the following sequence for F^{circ}. At the end, removing any edge will result in a tree. Then, all the **flow** in the tree will come as part of the **potential flow**.