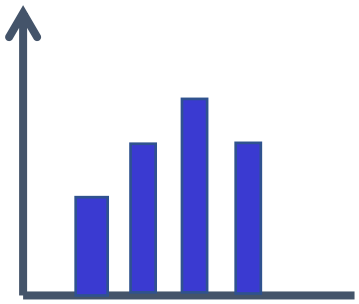
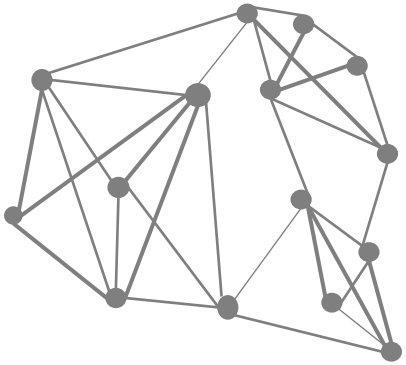
Python package for proteomics data clustering

- Introduction
- Preprocessing
- Network Enhancement
- Clustering
- Usage
- Reference

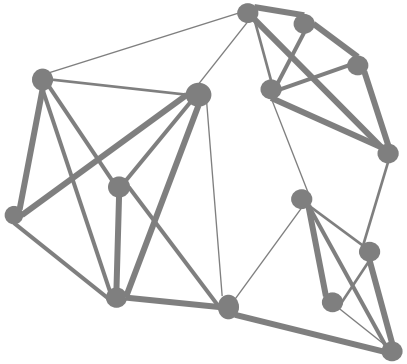
Overview



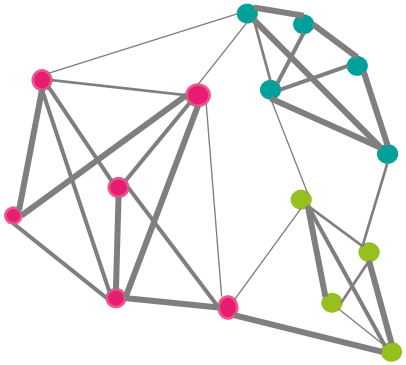
TMT Data



Graph



Denoised Graph



Modules

Annotation of a graph

- Graph G is consisted of 3 components : Nodes V, Edges E, and Weights W

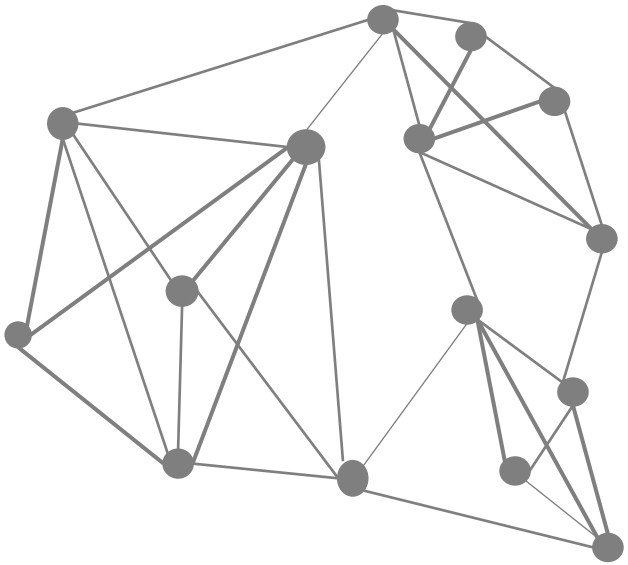
$$G = (V, E, W)$$

- Define the number of nodes as n

$$|V| = n$$

- Edges are defined as pairs of nodes

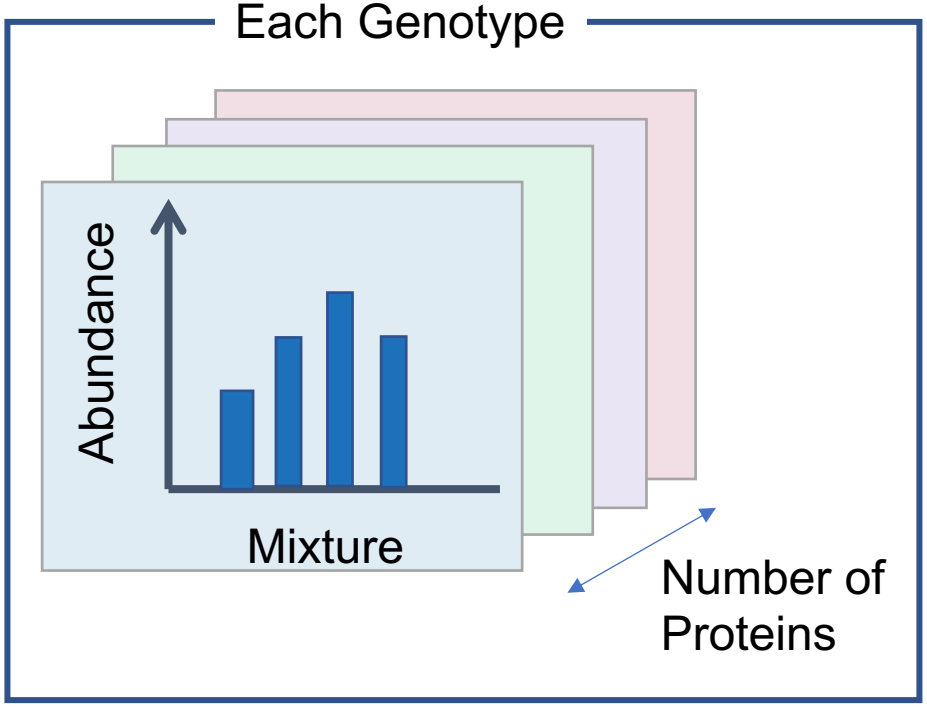
$$e_{i,j} = (v_i, v_j) \quad e_{i,j} \in E, v_i, v_j \in V$$



- If each edges have weight $w_{i,j}$ as a scalar value, weights are defined as a nxn matrix $W \in \mathbb{R}^{n \times n}$

Preprocessing

Data structure of TMT



Annotation

- Proteins
  $p \in \boldsymbol{P}, |\boldsymbol{P}| = n$
- Mixture/ Fraction
  $m \in \boldsymbol{M}$
- Genotype
  $g \in \boldsymbol{G}$
- Abundance vector of
  i-th protein in j-th genotype
  $\boldsymbol{a_{p_i,g_i}} = \left(a_{\boldsymbol{p_i,g_i,m}} \mid m \in \boldsymbol{M}\right)$

Correlation matrix

Think about single genotype $\mathcal{g}$.

The correlation coefficient between i-th and j-th protein can be defined as :

$$\mathrm{corr}\left(\boldsymbol{a}_{p_i}, \boldsymbol{a}_{p_j}\right) = \frac{\sum_{m \in M}(a_{m,p_i} - \widehat{\boldsymbol{a}_{p_j}})(a_{m,p_j} - \widehat{\boldsymbol{a}_{p_j}})}{\sqrt{\sum_{m \in M}(a_{m,p_i} - \widehat{\boldsymbol{a}_{p_i}})^2 \sum_{m \in M}(a_{m,p_j} - \widehat{\boldsymbol{a}_{p_j}})^2}}$$

Correlation matrix $W \in \mathbb{R}^{n \times n}$ is defined as:

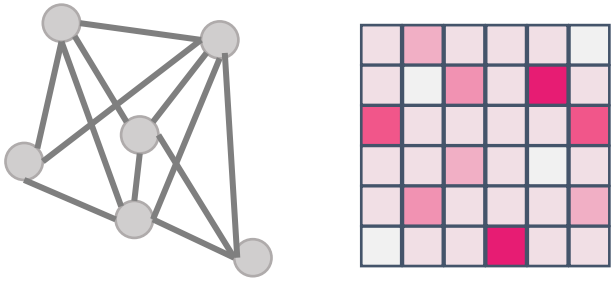
$$W_{i,j} = \mathrm{corr}\left(\boldsymbol{a}_{p_i}, \boldsymbol{a}_{p_j}\right)$$

By regarding correlation coefficients as weights and proteins as nodes, the sets (P, E, W) is regarded as a graph.

Network Enhancement

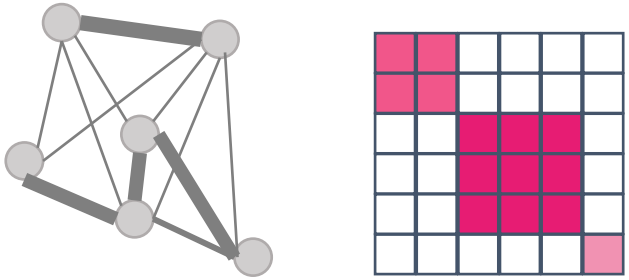
Goal : Getting new wights matrix W^*

$$G = (P, E, W)$$



Initial graph

$$G^* = (P, E, W^*)$$



Weighted graph

- Weights should be positive : $W_{i,j} \geq 0$
- ”Heavy” edge should be heavier, “right” edge should be lighter

  Increasing the eigengap of W while preserving the eigenvector

3 steps

1. Define probability transition matrix P as:

$$P_{i,j} = \frac{W_{i,j}}{\sum_{k \in \mathcal{N}_i} W_{i,k}} * I\{j \in \mathcal{N}_i\}$$

$\mathcal{N}_i$: k-th neighborhoods of i-th vertex

$$I\{j \in \mathcal{N}_i\} = \begin{cases} 1 & \text{if} \quad j \in \mathcal{N}_i \\ \text{else} & 0 \end{cases}$$

2. Yield doubly stochastic matrix $\mathcal{T}$ as:

$$\mathcal{T}_{i,j} \leftarrow \sum_{k=1}^{n} \frac{P_{i,k} P_{j,k}}{\sum_{v=1}^{n} P_{v,k}}$$

3. Update weight

$$W_{t+1} = \alpha \mathcal{T} \times W_t \times \mathcal{T} + (1 - \alpha)\mathcal{T}$$

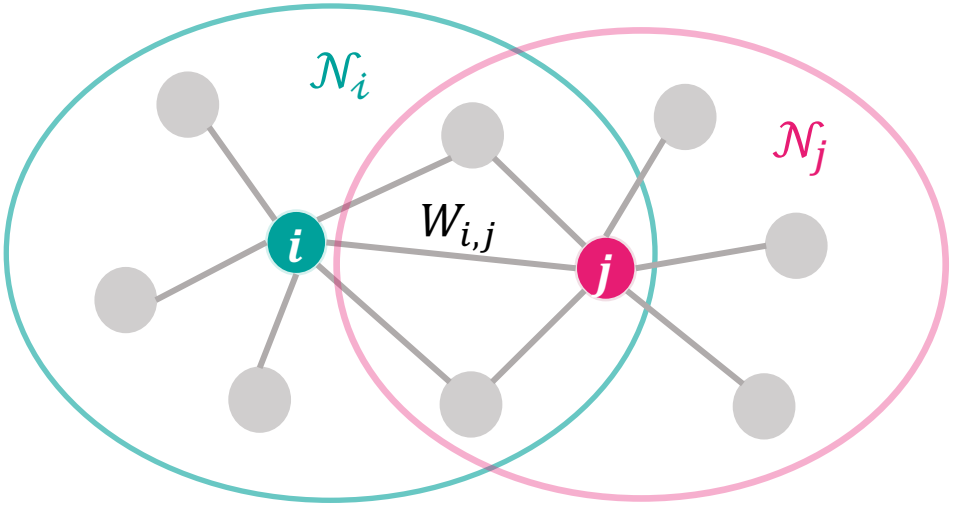
α: Regularization parameter
t : Iteration number

Probability transition matrix

$$P_{i,j} = \frac{W_{i,j}}{\sum_{k \in \mathcal{N}_i} W_{i,k}} * I\{j \in \mathcal{N}_i\}$$

j is in $\mathcal{N}_i$: $P_{i,j} > 0$

i isn't in $\mathcal{N}_j$: $P_{j,i} = 0$

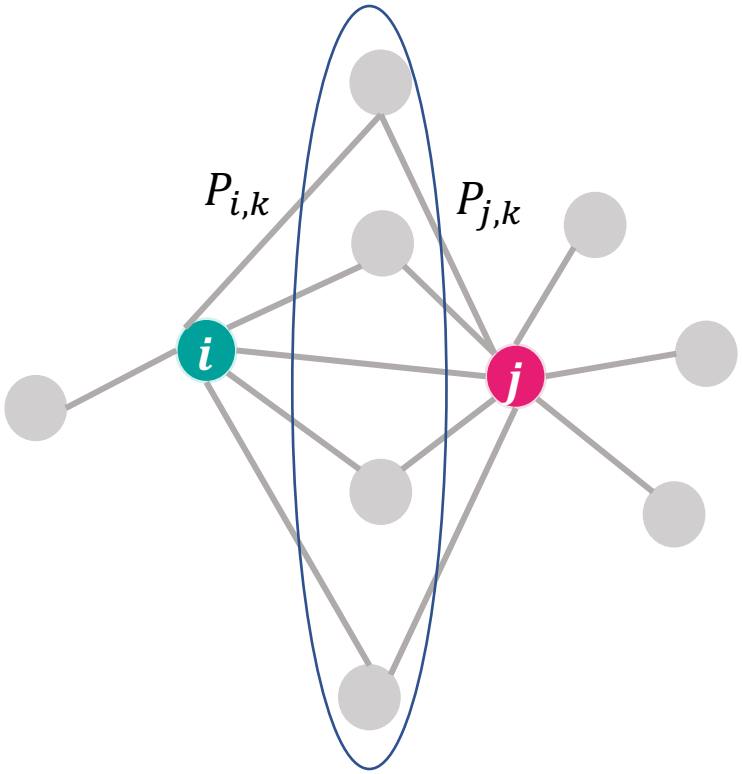


- P isn't necessarily symmetric
- P is sparse

Doubly stochastic matrix

$$\mathcal{T}_{i,j} = \sum_{k=1}^{n} \frac{P_{i,k} P_{j,k}}{\sum_{v=1}^{n} P_{v,k}}$$

$$= \frac{\sum_{k=1}^{n} P_{i,k} P_{j,k}}{\sum_{k=1}^{n} \sum_{v=1}^{n} P_{v,k}}$$



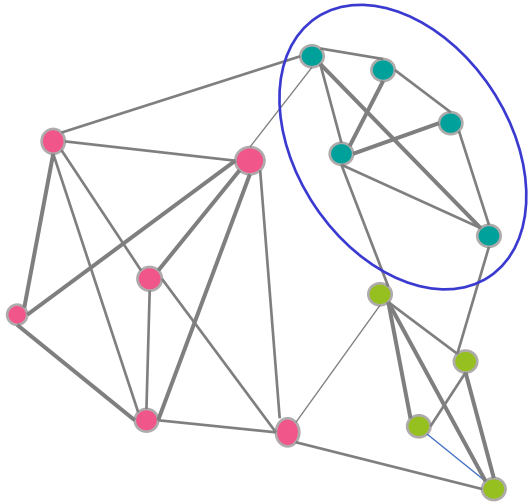
- T is symmetric
- Summation of all lows and columns of T is 1

Update of weights

$$
\begin{aligned}
W_t &= \alpha\mathcal{T}\times W_{t-1}\times\mathcal{T} + (1-\alpha)\mathcal{T} \\
&= \alpha\mathcal{T}\times(\alpha\mathcal{T}\times W_{t-2}\times\mathcal{T} + (1-\alpha)\mathcal{T})\times\mathcal{T} + (1-\alpha)\mathcal{T} \\
&= \alpha^2\mathcal{T}^2 W_{t-2}\mathcal{T}^2 + \alpha\mathcal{T}(1-\alpha)\mathcal{T}\times\mathcal{T} + (1-\alpha)\mathcal{T} \\
&= \alpha^2\mathcal{T}^2 W_{t-2}\mathcal{T}^2 + (1-\alpha)\mathcal{T}\,(\alpha\mathcal{T}^2 + 1) \\
&\cdots \\
&= \alpha^{\mathrm{t}}\mathcal{T}^{\mathrm{t}} W_0\mathcal{T}^t + (1-\alpha)\mathcal{T}\sum_{k=0}^{t-1}(\alpha\mathcal{T}^2)^k
\end{aligned}
$$

$$
W_{t\to\infty} = (1-\alpha)\mathcal{T}(\mathcal{I} - \alpha\mathcal{T}^2)^{-1}
$$

Overview and basic annotation



Graph $G = (V, E, W)$

Community $C_i \subseteq V$

s.t.

$V = \cup_i C_i$, $C_i \cap C_j = \phi$ when $i \neq j$

Partition $P = \{C_1, C_2 \cdots C_r\}$

Node movement $P(v \rightarrow C)$

Goal : To obtain good partition → Optimization of quality function $H(G, P)$

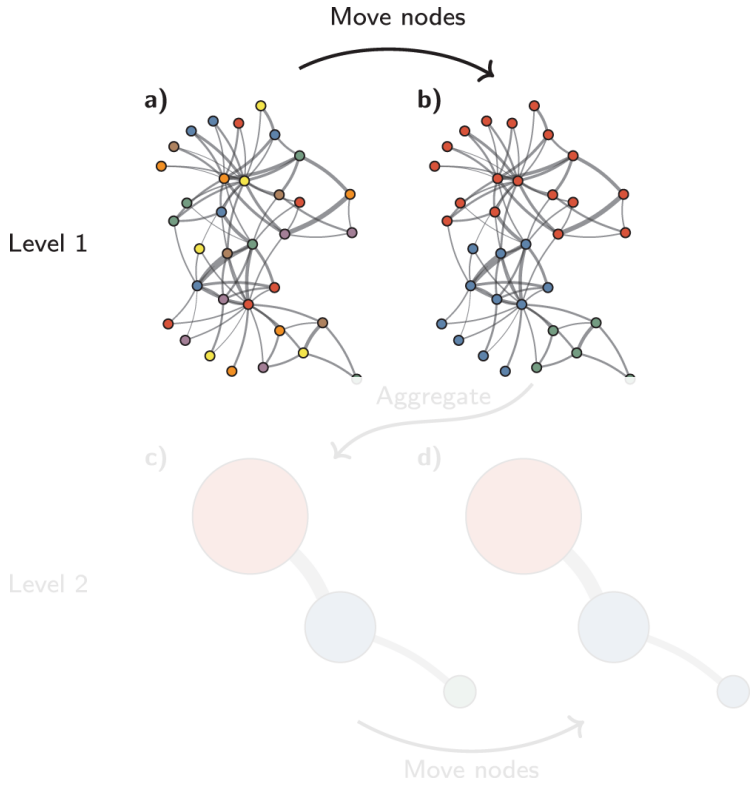
$$H(G,P) = \sum_{C \in P} [E(C,C) - \gamma \binom{||C||}{2}]$$

$$H(G,P) = \sum_{C \in P} [E(C,C) - \frac{\gamma}{2m} \binom{||C||}{2}]$$

Louvain algorithm

```
function Louvain(Graph G, Partition P)
    do
        P ← MoveNodes(G, P)
        done ← |P| = |V (G)|
        if not done then
            G ← AggregateGraph(G, P)
            P ← SingletonPartition(G)
        end if
    while not done
    return flat∗ (P)
end function
```

```
function MoveNodes(Graph G, Partition P)
    do
        H_old = H(P)
        for v ∈ V (G)  do
            C'← argmax_{C∈P∪∅} ΔH_P(v → C) .
            if ΔH_P(v → C') > 0 then
                v→ C'
            end if
        end for
    while H(P) > H_old
     return P
end function
```



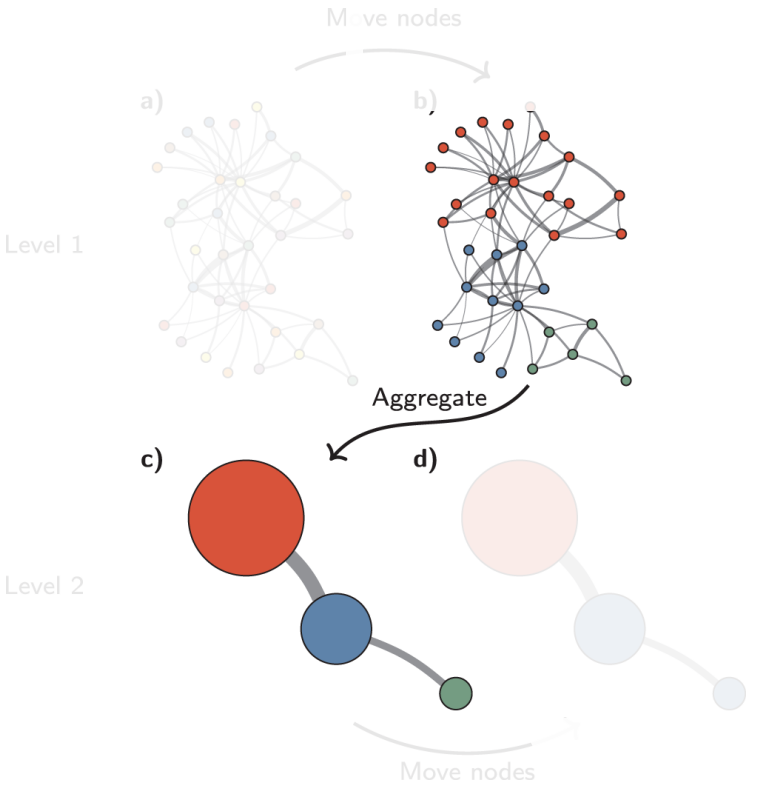
- Assign nodes to communities which maximize quality function

Louvain algorithm

```
function Louvain(Graph G, Partition P)
   do
        P ← MoveNodes(G, P)
        done ← |P| = |V (G)|
        if not done then
              G ← AggregateGraph(G, P)
              P ← SingletonPartition(G)
        end if
   while not done
   return flat∗ (P)
end function
```

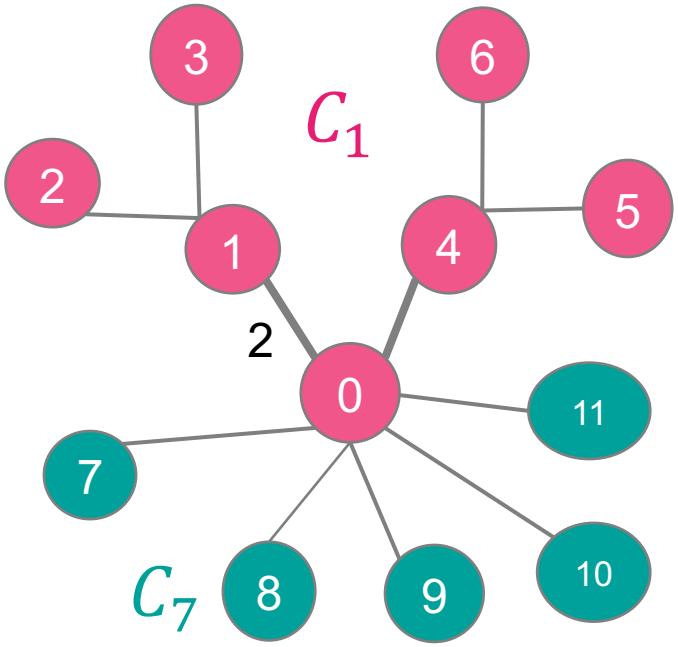
```
function AggregateGraph(Graph G, Partition P)
   V ← P
   E ← {(C, D) | (u, v) ∈ E(G), u∈C∈ P, v ∈D ∈P}
   return Graph(V, E)
end function
```

```
function SingletonPartition(Graph G)
   return {{v} | v ∈ V (G)}
end function
```

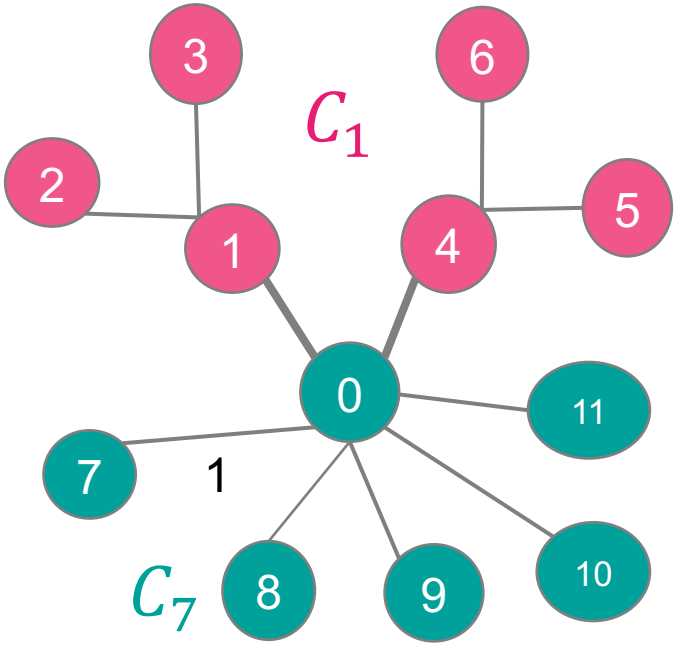


- After all nodes are assigned to communities, each communities are assumed as a node

Disconnection problem of Louvain algorithm



$$\Delta H(0 \rightarrow C_1) = 2{\times}2 - 6\gamma$$

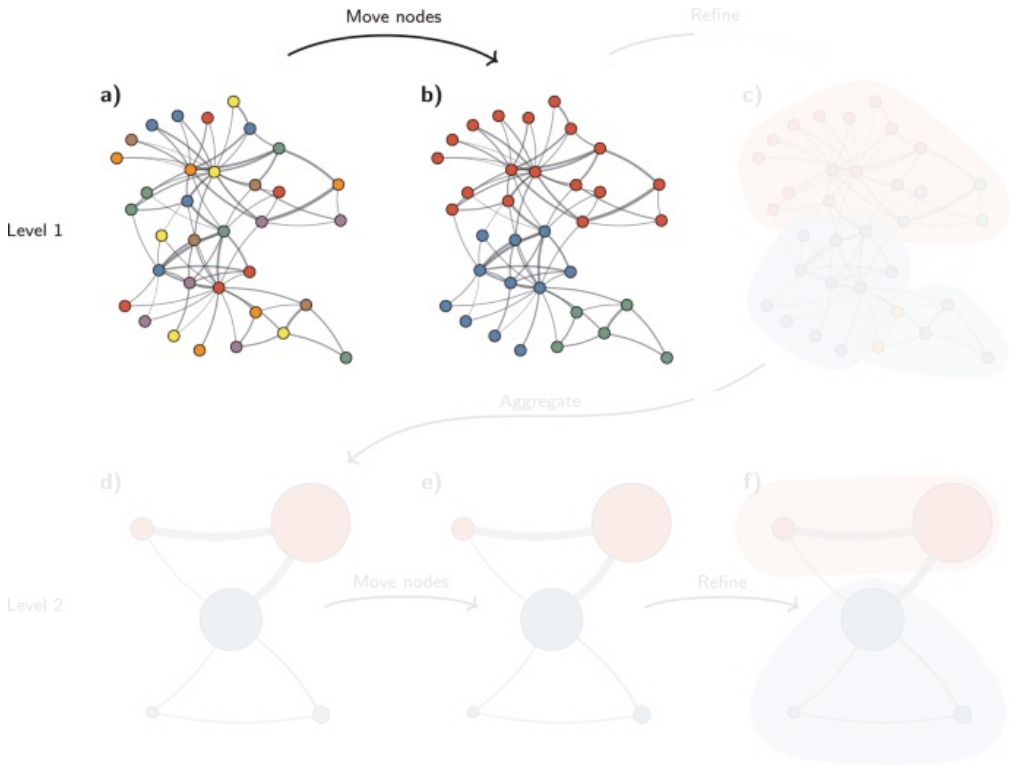


$$H(0 \rightarrow C_7) = 5 - 5\gamma$$

Leiden algorithm

```
function Leiden(Graph G, Partition P)
    do
        P ← MoveNodesFast(G, P)
        done ← |P| = |V (G)|
        if not done then
            P_refined ← RefinePartition(G, P)
            G ← AggregateGraph(G, P_refined)
            P ← {{v | v ⊆ C, v ∈ V (G)} | C ∈ P}
        end if
     while not done
     return flat* (P)
end function
```

```
function MoveNodesFast(Graph G, Partition P)
    Q ← Queue(V (G))
    do
        v ← Q.remove()
        C'← argmax_{C∈P∪∅} ΔH_P(v→ C)
        if ΔH_P(v→ C') > 0 then
            v→ C'
            N ← {u | (u, v) ∈ E(G), u /∈ C' }
            Q.add(N − Q)
        end if
    while Q ≠ ∅ .
    return P
end function
```



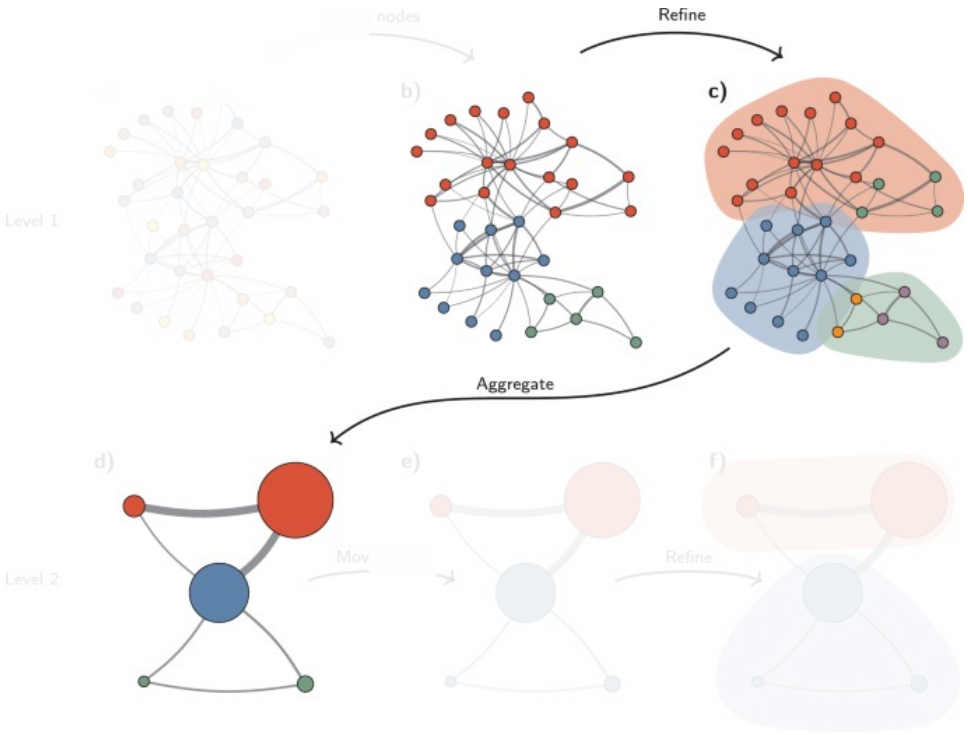
- Make sure to visit all nodes by using queue

Leiden algorithm

```
function Leiden(Graph G, Partition P)
    do
        P ← MoveNodesFast(G, P)
        done ← |P| = |V (G)|
        if not done then
            P_refined ← RefinePartition(G, P)
            G ← AggregateGraph(G, P_refined)
            P ← {{v | v ⊆ C, v ∈ V (G)} | C ∈ P}
        end if
    while not done
    return flat* (P)
end function
```

```
function RefinePartition(Graph G, Partition P)
    P_refined ← SingletonPartition(G)
    for C ∈ P do
        P_refined ← MergeNodesSubset(G, P_refined, C)
    end for
    return P_refined
end function
```

```
function AggregateGraph(Graph G, Partition P)
    V ← P
    E ← {(C, D) | (u, v) ∈ E(G), u∈C∈ P,v ∈ D ∈P}
    return Graph(V, E)
end function
```



- Divide partitions into subsets (Refinement)
- Aggregation is based on $P_{refined}$,but these subsets will be assigned to the same community

Code availability

https://github.com/Io-Saito/ProteinClustering MIT License

Requirement

Python >= 3.7

TODO

Interface for .rda

References

[1] Genetic Disruption of WASHC4 Drives Endo-lysosomal Dysfunction and Cognitive-Movement Impairments in Mice and Humans. Courtland J.L., Bradshaw T.W.A., Waitt G., Soderblom E., Ho T., Rajab A., Vancini R., Kim I.H., Soderling S.H. (2021). eLife; 10:e61590 doi: 10.7554/eLife.61590

[2] From Louvain to Leiden: guaranteeing well-connected communities. Traag, V.A., Waltman. L., Van Eck, N.-J. (2018). Scientific reports, 9(1), 5233. 10.1038/s41598-019-41695-z

[3] Network Enhancement as a general method to denoise weighted biological networks. Wang B., Pourshafeie A., Zitnik M., Zhu J., Bustamante C.D., Batzoglou S., Leskovec J. (2018). Nature Communications, 9, 3108. 10.1038/s41467-018-05469-x

Appendix

Refine partition

function MergeNodesSubset(Graph G, Partition P, Subset S)
  R = {$v \mid v \in S, E(v, S - v) \geq \gamma||v|| \cdot (||S|| - ||v||)$}
  for $v \in R$ **do**
    if v in singleton community **then**
      $T \leftarrow \{C \mid C \in P, C \subseteq S, E(C, S-C) \geq \gamma||C|| \cdot (||S|| - ||C||)\}$

$$\Pr(C' = C) \sim \begin{cases} \exp\left(\frac{1}{\theta}\Delta H_P(v \to C)\right) & \text{if } \Delta H_P(v \to C) \geq 0 \\ 0 & \text{otherwise} \end{cases} \quad \text{for } C \in T$$

      $v \rightarrow C'$
    end if
  end for
  return P
end function