Modularity-based Sparse Soft Graph Clustering (Supplementary Material)

Alexandre Hollocou
INRIA, Paris

Thomas Bonald
Telecom Paristech

Marc Lelarge
INRIA & ENS, Paris

A Proof of Theorem 4.1

Theorem 4.1. *The function J is convex if and only if the second lowest eigenvalue λ_2 of the normalized Laplacian $\boldsymbol{\mathcal{L}}$ verifies $\lambda_2 \geq 1$.*

Proof. The cost function J can be written as $J(\boldsymbol{p}) = \frac{1}{w}\sum_k \tilde{J}(\boldsymbol{p}_{\cdot k})$ where $\tilde{J}(\boldsymbol{q}) = \sum_i \sum_j (W_{ij} - w_i w_j / w) q_i q_j$. So J is convex if and only if $\tilde{J}$ is convex. Moreover, the function $\tilde{J}$ is convex iff its hessian matrix $\boldsymbol{H}$ is semi-definite positive. An expression of $\boldsymbol{H}$ is given by $\boldsymbol{H} = -2\left(\boldsymbol{W} - \frac{\boldsymbol{w}\boldsymbol{w}^T}{w}\right)$, where $\boldsymbol{w}$ denotes the vector $(w_i)_{i\in V}$.
Let $\boldsymbol{x}$ be a vector of $\mathbb{R}^n$, and $\boldsymbol{y} = \boldsymbol{D}^{1/2}\boldsymbol{x}$. Then we have:

$$\begin{aligned}\boldsymbol{x}^T\boldsymbol{H}\boldsymbol{x} &= -2\left(\boldsymbol{y}^T\boldsymbol{D}^{-1/2}\boldsymbol{W}\boldsymbol{D}^{-1/2}\boldsymbol{y} - \|\boldsymbol{u}_1^T\boldsymbol{y}\|^2\right)\\ &= 2(\boldsymbol{y}^T(\boldsymbol{\mathcal{L}} - \boldsymbol{I})\boldsymbol{y} + \|\boldsymbol{u}_1^T\boldsymbol{y}\|^2)\end{aligned}$$

where $\boldsymbol{u}_1 = (1/\sqrt{w})\boldsymbol{D}^{-1/2}\boldsymbol{w}$. It is easy to verify that $\boldsymbol{u}_1$ is a normalized eigenvector associated with the first eigenvalue $\lambda_1 = 0$ of the normalized Laplacian $\boldsymbol{\mathcal{L}}$. We use $\boldsymbol{u}_1, \ldots, \boldsymbol{u}_n$ to denote the orthonormal basis of eigenvectors corresponding to the eigenvalues $(\lambda_1, \ldots, \lambda_n)$. Thus, we can write:

$$\begin{aligned}\boldsymbol{x}^T\boldsymbol{H}\boldsymbol{x} &= 2(\sum_{k=1}^{n}(\lambda_k - 1)\|\boldsymbol{u}_k^T\boldsymbol{y}\|^2 + \|\boldsymbol{u}_1^T\boldsymbol{y}\|^2)\\ &= 2(\sum_{k=2}^{n}(\lambda_k - 1)\|\boldsymbol{u}_k^T\boldsymbol{y}\|^2)\end{aligned}$$

We see that $\boldsymbol{x}^T\boldsymbol{H}\boldsymbol{x} \geq 0$ for all $\boldsymbol{x}$ if and only if $1 \geq \lambda_2 \geq \ldots \geq \lambda_n$, which concludes the proof. $\square$

B Proof of Proposition 4.2

Proposition 4.2. *If the graph does not contain any self loops, i.e. if $W_{ii} = 0$ for all node i, the function $\boldsymbol{p} \mapsto J(\boldsymbol{p})$ is convex with respect to variable $\boldsymbol{p}_{i\cdot}$ for all $i \in V$.*

Proof. Let i be a given node. We have for all $k, k' \in [\![1, n]\!]$,

$$\begin{aligned}\frac{\partial J}{\partial p_{ik}} &= -\frac{2}{w}\sum_{j\in V}(W_{ij} - w_i w_j / w) p_{jk},\\ \frac{\partial J}{\partial p_{ik}\partial p_{ik'}} &= \frac{2}{w}(w_i^2/w - W_{ii})\delta(k, k'),\end{aligned}$$

where $\delta(k, k') = 1$ if $k = k'$ and 0 otherwise. If $W_{ii} = 0$, the hessian matrix $\boldsymbol{H}_i$ of J with respect to $\boldsymbol{p}_{i\cdot}$ can be written $\boldsymbol{H}_i = 2(w_i/w)^2\boldsymbol{I}$. It is thus definite positive, which proves the convexity of J in $\boldsymbol{p}_{i\cdot}$. $\square$

C Proof of Theorem 4.3

Theorem 4.3. *The soft modularity objective function $Q(\boldsymbol{p})$ is non-decreasing under the update rule*

$$\boldsymbol{p}_{i\cdot} \leftarrow \pi_{\mathcal{Y}}\left(\boldsymbol{p}_{i\cdot} + \frac{2t}{w}\sum_{j\in V}\left(W_{ij} - \frac{w_i w_j}{w}\right)\boldsymbol{p}_{j\cdot}\right)$$

for all node $i \in V$ if the step size t verifies $t < (w/w_i)^2$. $Q(\boldsymbol{p})$ is invariant under all these update rules iff $\boldsymbol{p}_{i\cdot}$ minimizes $Q(\boldsymbol{p})$ with fixed $\boldsymbol{p}_{j\cdot}$, $j \neq i$, for all node $i \in V$.

Proof. Let i be a given node of V and $\boldsymbol{p} \in \mathcal{X}$ be a feasible solution of the relaxation of the soft modularity optimization problem. We define $\boldsymbol{p}^+$ with $\boldsymbol{p}_{i\cdot}^+ = \pi_{\mathcal{Y}}\left[\boldsymbol{p}_{i\cdot} + \frac{2t}{w}\sum_{j\in V}\left(W_{ij} - \frac{w_i w_j}{w}\right)p_{j\cdot}\right]$, and $\boldsymbol{p}_{j\cdot}^+ = \boldsymbol{p}_{j\cdot}$ for all $j \neq i$.

The objective function $J(\boldsymbol{p}) = -Q(\boldsymbol{p})$ is quadratic in p_{ik}, $k \in [\![1, n]\!]$, so we have:

$$\begin{aligned}J(\boldsymbol{p}^+) &= J(\boldsymbol{p}) + \nabla_i J(\boldsymbol{p})^T(\boldsymbol{p}_{i\cdot}{}^+ - \boldsymbol{p}_{i\cdot})\\ &\quad + \frac{1}{2}(\boldsymbol{p}_{i\cdot}{}^+ - \boldsymbol{p}_{i\cdot})\boldsymbol{H}_i(\boldsymbol{p}_{i\cdot}{}^+ - \boldsymbol{p}_{i\cdot}) \qquad (1)\\ &= J(\boldsymbol{p}) - t\nabla_i J(\boldsymbol{p})^T\boldsymbol{G}_i(\boldsymbol{p}) + \left(\frac{w_i t}{w}\right)^2\|\boldsymbol{G}_i(\boldsymbol{p})\|^2\end{aligned}$$

where $\nabla_i J(\boldsymbol{p})$ is the gradient of J with respect to $\boldsymbol{p}_{i\cdot}$, $\boldsymbol{H}_i$ is the hessian matrix of J with respect to $\boldsymbol{p}_{i\cdot}$ whose expression was given in the proof of Proposition 4.2, and $\boldsymbol{G}_i(\boldsymbol{p}) = (\boldsymbol{p}_{i\cdot} - \boldsymbol{p}_{i\cdot}{}^{+})/t$.

Besides, by definition of $\pi_{\mathcal{Y}}$, and using the expression of $\nabla_i J(\boldsymbol{p})$ given in the proof of Proposition 4.2, we have

$$\begin{aligned}\boldsymbol{p}_{i\cdot}^{+} &= \arg\min_{\boldsymbol{q}\in\mathcal{Y}} \left\{\|\boldsymbol{q} - (\boldsymbol{p}_{i\cdot} - t\nabla_i J(\boldsymbol{p}))\|^2\right\} \\ &= \arg\min_{\boldsymbol{q}\in\mathcal{Y}} \left\{2t\nabla_i J(\boldsymbol{p})^T(\boldsymbol{q} - \boldsymbol{p}_{i\cdot}) + \|\boldsymbol{q} - \boldsymbol{p}_{i\cdot}\|^2\right\} \\ &= \arg\min_{\boldsymbol{q}\in\mathbb{R}^n} \left\{\nabla_i J(\boldsymbol{p})^T(\boldsymbol{q} - \boldsymbol{p}_{i\cdot}) + \frac{\|\boldsymbol{q} - \boldsymbol{p}_{i\cdot}\|^2}{2t} + h(\boldsymbol{q})\right\}\end{aligned} \tag{2}$$

where $h(\boldsymbol{q}) = 0$ if $\boldsymbol{q} \in \mathcal{Y}$, and $h(\boldsymbol{q}) = +\infty$ otherwise.

$\mathcal{Y}$ is a convex set, so h is a convex function. Note that h is non-differentiable. We use $\partial h(\boldsymbol{q})$ to denote the subdifferential of h at $\boldsymbol{q}$ i.e. the set of all its subgradients. Remember that a vector $\boldsymbol{v}$ is defined as a subgradient of h at $\boldsymbol{q}$ if it verifies, for all $\boldsymbol{q}'$, $h(\boldsymbol{q}') - h(\boldsymbol{q}) \geq \boldsymbol{v}^T(\boldsymbol{q}' - \boldsymbol{q})$.

Equation (2) can be written $\boldsymbol{p}_i^{+} = \arg\min_{\boldsymbol{q}} L(\boldsymbol{q})$, where L is a non-differentiable convex function. The optimality of $\boldsymbol{p}_{i\cdot}^{+}$ gives us $\boldsymbol{0} \in \partial L(\boldsymbol{p}_{i\cdot}^{+})$. Therefore, there exists a $\boldsymbol{v} \in \partial h(\boldsymbol{p}_{i\cdot}^{+})$, such that

$$\nabla_i J(\boldsymbol{p}) + \frac{1}{t}(\boldsymbol{p}_{i\cdot}{}^{+} - \boldsymbol{p}_{i\cdot}) + \boldsymbol{v} = \boldsymbol{0}.$$

Using this result in equation (1), we obtain

$$J(\boldsymbol{p}^{+}) = J(\boldsymbol{p}) + t\boldsymbol{v}^T\boldsymbol{G}_i(\boldsymbol{p}) - t\left(1 - t\left(\frac{w_i}{w}\right)^2\right)\|\boldsymbol{G}_i(\boldsymbol{p})\|^2.$$

We have $t\boldsymbol{v}^T\boldsymbol{G}_i(\boldsymbol{p}) = \boldsymbol{v}^T(\boldsymbol{p}_{i\cdot} - \boldsymbol{p}_{i\cdot}{}^{+}) \leq h(\boldsymbol{p}_{i\cdot}) - h(\boldsymbol{p}_{i\cdot}{}^{+})$ because $\boldsymbol{v} \in \partial h(\boldsymbol{p}_{i\cdot}^{+})$. Both $\boldsymbol{p}_{i\cdot}$ and $\boldsymbol{p}_{i\cdot}^{+}$ belongs to $\mathcal{Y}$, thus $\boldsymbol{v}^T\boldsymbol{G}_i(\boldsymbol{p}) \leq 0$.

Finally, we obtain

$$J(\boldsymbol{p}^{+}) \leq J(\boldsymbol{p}) - t\left(1 - t\left(\frac{w_i}{w}\right)^2\right)\|\boldsymbol{G}_i(\boldsymbol{p})\|^2.$$

We have $Q(\boldsymbol{p}^{+}) \geq Q(\boldsymbol{p})$ if $t < (w/w_i)^2$. The inequality becomes an equality if and only if $\boldsymbol{G}_i(\boldsymbol{p}) = \boldsymbol{0}$, which gives us $\nabla_i J(\boldsymbol{p}) + \boldsymbol{v} = 0$. This is equivalent to say that $\boldsymbol{p}$ is an optimum of $J + h$ with respect to $\boldsymbol{p}_{i\cdot}$. □

D Proof of Theorem 4.4

Theorem 4.4. *The algorithm converges to a local maximum of the soft modularity function $\boldsymbol{p} \mapsto Q(\boldsymbol{p})$ which is a fixed point of the updates of Theorem 4.3.*

Proof. The sequence of matrices $\boldsymbol{p}$ built by the algorithm converges to a limit $\boldsymbol{p}^* \in \mathcal{Y}$, since the soft modularity Q is non-decreasing under each update, $\mathcal{Y}$ is a compact, and $\boldsymbol{p} \mapsto Q(\boldsymbol{p})$ is continuous and upper bounded. From the proof of Theorem 4.3, we have for all $i \in V$, $\boldsymbol{G}_i(\boldsymbol{p}) = 0$, which gives us $(\boldsymbol{p}_{i\cdot}^*)^+ = \boldsymbol{p}_{i\cdot}^*$, thus $\boldsymbol{p}^*$ is a fixed point for the update relative to node i. Moreover, we have for all node $i \in V$, $\nabla_i J(\boldsymbol{p}^*) + \boldsymbol{v} = 0$ for some $\boldsymbol{v} \in \partial h(\boldsymbol{p}_{i\cdot}^*)$, which proves that $\boldsymbol{p}^*$ is a local optimum of the function $\boldsymbol{p} \mapsto J(\boldsymbol{p}) = -Q(\boldsymbol{p})$. □

E Proof of Proposition 5.1

Proposition 5.1. *In order to compute $\pi_{\mathcal{Y}}(\hat{\boldsymbol{p}}_{i\cdot})$, we only need to sort the components $k \in \mathrm{supp}_i(\boldsymbol{p})$ of $\hat{\boldsymbol{p}}_{i\cdot}$ to determine ρ and θ. All components $k \notin \mathrm{supp}_i(\boldsymbol{p})$ of $\pi_{\mathcal{Y}}(\hat{\boldsymbol{p}}_{i\cdot})$ are set to zero.*

Proof. First note that

$$\sum_{k=1}^{n} \hat{p}_{ik} = \sum_k p_{ik} + \frac{2t}{w}\sum_j W_{ij}\left(\sum_k p_{ik} - \sum_k \bar{p}_k\right) = 1$$

since $\sum_k \sum_j \frac{w_j}{w} p_{jk} = \sum_j \frac{w_j}{w}\sum_k p_{jk} = 1$.

Let j_0 be defined by $j_0 = \max\{j : \mu_j \geq 0\}$. The function $j \mapsto \left(\sum_{r=1}^{j}\mu_r - 1\right)$ increases for $j \leq j_0$ and then decreases to 0 when $j = n$. In particular, this function is non-negative for $j \geq j_0$. This implies that $\rho \leq j_0$ and $\theta \geq 0$.

Now, for $k \notin \mathrm{supp}_i(\boldsymbol{p})$, we have $\hat{p}_{ik} = -\frac{2tw_i}{w}\bar{p}_k \leq 0$ so that the k-th component of $\pi_{\mathcal{Y}}(\hat{\boldsymbol{p}}_{i\cdot})$ will be zero since $\theta \geq 0$. Moreover, since $\rho \leq j_0$, the value of $\hat{p}_{ik}$ is not used for the determination of ρ and θ. □

F Proof of Proposition 6.1

Proposition 6.1. *Let $\boldsymbol{p} \in \mathcal{X}$ be a membership matrix. If $\boldsymbol{p} \in \{0,1\}^{n^2}$, if the hypothesis (H) is verified, and if $t > w/\delta$ where*

$$\delta = \min_{\substack{C,C'\subset V\\ i\in V\\ C\neq C'}} \left|\left(w_i(C) - \frac{w_i \mathrm{Vol}(C)}{w}\right) - \left(w_i(C') - \frac{w_i \mathrm{Vol}(C')}{w}\right)\right|,$$

then the update rule of Theorem 4.3 for node $i \in V$ reduces to $\forall k$, $p_{ik} \leftarrow 1$ if $k = \arg\max_{l: j\in C_l, j\sim i}\left[w_i(C_l) - w_i\frac{\mathrm{Vol}(C_l)}{w}\right]$, and $p_{ik} \leftarrow 0$ otherwise, where C_k denotes the k^{th} cluster defined by $\boldsymbol{p}$.

Proof. Given $\boldsymbol{p} \in \mathcal{X} \cap \{0,1\}^{n^2}$ and $i \in V$, we use $\boldsymbol{p}^{+}$ to denote the vector obtained by applying the update rule of Theorem 4.3 for node i. We have, for all $k \in [\![1, n]\!]$, $p_{ik}^{+} = \max(\hat{p}_{ik} - \lambda, 0)$ where $\hat{p}_{ik} = p_{ik} + \frac{2t}{w}[w_i(C_k) - w_i\mathrm{Vol}(C_k)/w]$ and λ is chosen so that $\sum_k p_{ik}^{+} = 1$.

Let $k^* = \arg\max_k[w_i(C_k) - w_i\mathrm{Vol}(C_k)/w]$. We assume that (H) is verified and that $t > w/\delta$. Thus, for all $k \in [\![1, n]\!]$ s.t. $k \neq k^*$,

$$\begin{aligned}\hat{p}_{ik^*} - \hat{p}_{ik} &\geq p_{ik^*} - p_{ik} + \frac{2t}{w}\delta \\ &\geq -1 + \frac{2t}{w}\delta > 1. \end{aligned} \tag{3}$$

In particular, this implies $p^+_{ik^*} > p^+_{ik}$. Now, note that if we had $p^+_{ik} > 0$ for a certain $k \neq k^*$, we would have $p^+_{ik^*} \in (0, 1]$, $p^+_{ik} \in (0, 1]$, and therefore $p^+_{ik^*} - p^+_{ik} < 1$. Yet, we would also have $p^+_{ik^*} - p^+_{ik} = (\hat{p}_{ik^*} - \lambda) - (\hat{p}_{ik} - \lambda) = \hat{p}_{ik^*} - \hat{p}_{ik}$, which leads to a contradiction with (3).

Therefore, we have $\forall k \neq k^*$, $p^+_{ik} = 0$, which immediately gives us $p^+_{ik^*} = 1$.

Finally, we see have seen in our simplification of the projection onto the probability simplex that the arg max in the definition of k^* can be taken over the communities in the neighborhood of i. □

G Proposition 6.2

Proposition 6.2. *The update performed by* ***LouvainUpdate***(i) *is equivalent to transferring node i to the cluster C^*_k such that:*

$$k^* = \underset{k \in \mathrm{NeiCom}(i)}{\arg\max} \left[w_i(C_k) - w_i \frac{\mathrm{Vol}(C_k)}{w} \right]$$

Proof. The variation of *hard* modularity when node $i \in V$ joins cluster C_k can be written as

$$\Delta Q(i \to C_k) = \frac{1}{w}\left[2\left(w_i(C_k) - w_i\frac{\mathrm{Vol}(C_k)}{w}\right) - \frac{w_i^2}{w}\right]. \tag{4}$$

Therefore, $\arg\max_k \Delta Q(i \to C_k) = \arg\max_k \left(w_i(C_k) - \frac{w_i \mathrm{Vol}(C_k)}{w}\right)$. □

H Algorithm pseudo-code

We give the pseudo-code of the algorithm as working python code in in Algorithm 1. The algorithm only requires one parameter, the learning rate `lr`, which corresponds to the $\frac{2t}{w}$ factor in the previous section.

The algorithm stores two variables `p` and `avg_p`. The variable `p` corresponds to the membership matrix $\boldsymbol{p}$, i.e. the output of our algorithm, and is stored as a dictionary of dictionary, so that each `p`$[i][k]$ corresponds to a positive coefficient of $\boldsymbol{p}$, p_{ik}. The variable `avg_p` is a dictionary used to store the average cluster membership vector $\bar{\boldsymbol{p}}$, so that `avg_p`$[k]$ corresponds to $\bar{p}_k$.

Algorithm 1 Soft-modularity optimization

Require: `nodes`, `edges`, `degree`, `w`, `lr` (learning rate)

Initialization

```
p = dict()
avg_p = dict()
for node in nodes:
    p[node] = {node: 1.}
    avg_p[node] = (1./w) * degree[node]
```

One epoch (update the membership matrix `p`)

```
for node in nodes:
    new_p = dict()
    # Gradient descent step
    for com in p[node]:
        new_p[com] = p[node][com]
    for neighbor in edges[node]:
        weight = edges[node][neighbor]
        for com in p[neighbor]:
            if com not in new_p:
                new_p[com] = 0.
            new_p[com] += lr * weight * p[neighbor][com]
    for com in new_p:
        new_p[com] -= lr * degree[node] * avg_p[com]
    # Projection step
    new_p = project(new_p)
    # Updating average membership vector
    for com in p[node]:
        avg_p[com] -= (1./w) * degree[node] * p[node][com]
    p[node] = dict()
    for com in new_p:
        avg_p[com] += (1./w) * degree[node] * new_p[com]
        p[node][com] = new_p[com]
```

Sub-routine `project`

```
def project(in_dict):
    # Sort the values of in_dict in decreasing order
    values = sorted(in_dict.values(), reverse=True)
    # Find the value of lambda
    cum_sum = 0.; lamb = 0.
    i = 1
    for val in values:
        cum_sum += val
        new_lamb = (1. / i) * (cum_sum - 1.)
        if val - new_lamb <= 0.:
            break
        else:
            lamb = new_lamb
            i += 1
    # Create the output dictionary
    out_dict = dict()
    for key in in_dict:
        out_dict[key] = max(in_dict[key] - lamb, 0)
    return out_dict
```

The graph is given to the algorithm as four variables: `nodes`, `edges`, `degree` and `w`. The variable `nodes` contains the list of the nodes. The variable `edges` is a dictionary of dictionary, where `edges`$[i][j]$ contains the weight W_{ij}. The variable `degree` is a dictionary containing the node degrees, and `w` is the total weight of the graph.

One epoch in our algorithm corresponds to the application of all the update rules of Theorem 4.3, each update rule corresponding to the update of the membership probabilities of one node. Several strategies can be adopted regarding the choice of the number of epochs. A fixed number of epoches can be given in advance as an additional parameter to the algorithm, or the soft-modularity can be computed at the end of each epoch, and be used as a stopping criterion. For instance, we can stop the algorithm when the modularity increase becomes smaller than a given precision factore $\epsilon > 0$. This later strategy is the equivalent of the strategy used by the Louvain algorithm for the *hard* modularity problem. In practice, in some cases, it proves more efficient to consider the more general update rule

$$\boldsymbol{p}_{i\cdot} \leftarrow \pi_{\mathcal{Y}} \left(b\boldsymbol{p}_{i\cdot} + t' \sum_{j \sim i} W_{ij} (\boldsymbol{p}_{j\cdot} - \bar{\boldsymbol{p}}) \right)$$

where $b \geq 0$ is a bias parameter. In future work, we would like to understand the impact of a $b \neq 1$ on the solution.