Supplementary of *Active modules for multilayer weighted gene co-expression networks: a continuous optimization approach*

Dong Li, Shan He
School of Computer Science, The University of Birmingham, UK
Zhisong Pan, Guyu Hu
PLA University of Science and Technology, China

June 3, 2016

1 NP-hardness of problem

We copy the problem of active module identification in weighted gene co-expression network here,

Problem 1. *Given a complete undirected graph $G = (V, E)$, with vertex weight $\mathbf{z}_v \in R$ for each $v \in V$ and edge weights $W = [w_{ij}]$ for each edge (i, j), find a subgraph T of size k with large vertices weight $\sum_{i \in T} z_i$ and also edges weights $\sum_{i,j \in T} w_{ij}$.*

The graph in problem (1) can be summarized as a complete network with vertices weights and edges weights. The NP-hardness of problem (1) can be proved by considering finding heaviest subnetworks on the following four simplified cases:

- 1), Complete network with identical vertices weights but non-identical edges weights.
- 2), Incomplete network with identical vertices weights and edges weights.
- 3), Incomplete network with identical vertices weights but non-identical edge weights.
- 4), Incomplete network with identical edges weights but non-identical node weights.

Lemma 1.1. *All the cases above are NP-hard, thus problem (1) is NP-hard.*

Proof. A more simplified version of case 1) corresponds to a k-maximal spanning tree problem, which asks for a tree covers exactly k nodes with maximal edge weights. Here we ask for a complete subgraph instead of a subtree. The NP-hardness of edge-weighted k-cardinality tree problem has been proven in [1] and [2].

Case 2) can be viewed as a special case of case 3) which is also can be reduced to the well-known NP-complete problem k-clique. The NP-hardness proof is available in the Supplementary Text S1 of [3].

Case 4) corresponds to the Maximum-Weight Connected Subgraph Problem. The NP-hardness can be proven by reducing the well-known NP-complete problem MINIMUM COVER to the problem. The proof sketch is available in Supplementary information of [4].

Finding heaviest subnetworks on all above different networks are NP-hard and special case of problem (1), which means problem (1) is also NP-hard. □

For completeness, finding a heaviest subgraph on a complete graph with identical edges weights but non-identical vertices weights is a trivial task. Just sorting the vertices weights and picking the highest k ones is not NP-hard, thus we did not list this case above.

2 Piecewise root finding for Euclidean projection on elastic net

Here is the detailed steps to solve problem (1):

$$\Pi_C(\mathbf{y}) = \arg\min_{\mathbf{x}\in\mathbb{R}^n_+} \frac{1}{2}\|\mathbf{x}-\mathbf{g}\|_2^2 \; s.t. \; (1-\alpha)\|\mathbf{x}\|_1 + \alpha\|\mathbf{x}\|^2 \leq 1. \tag{1}$$

We have the optimal solution of (1) as follows:

$$x_i^* = max\left(0, \frac{g_i - \alpha\theta^*}{1+2(1-\alpha)\theta^*}\right), \tag{2}$$

where θ^* is the optimal lagrange multiplier.

The problem turns out to find a proper θ^* to satisfy $\alpha\|\mathbf{x}^*\|_1 + (1-\alpha)\|\mathbf{x}^*\|_2^2 = t$. Using the same strategy of Gong et al. [5], we formulate the problem of Euclidean projections on the elastic net constraint set as a root finding problem. Being slightly different from that in [5], we use the original form of elastic net penalty, i.e $f(\mathbf{x}) =$

$\alpha\|\mathbf{x}\|_1 + (1-\alpha)\|\mathbf{x}\|_2^2$, which makes it easy to control the module size by tuning the parameter α: the larger α leads to a smaller module.

$$\sum_{i=1}^{n}\left(\alpha max\big(0, \frac{g_i - \alpha\theta}{1+2(1-\alpha)\theta}\big) + (1-\alpha)\frac{max(0, g_i-\alpha\theta)^2}{(1+2(1-\alpha)\theta)^2}\right) = t. \tag{3}$$

The solution θ is then transformed to the root of following function,

$$\begin{aligned} &f_{en}(\theta) = t(1+2(1-\alpha)\theta)^2 - \\ &\sum_{i=1}^{n}\big((1+2(1-\alpha)\theta)max(0, g_i-\alpha\theta) + (1-\alpha)max(0, g_i-\alpha\theta)^2\big) \\ &= a_\theta\theta^2 + b_\theta\theta + c_\theta, \end{aligned} \tag{4}$$

where $a_\theta = (\alpha^3-\alpha^2)|R_{\mathbf{g},\theta}| - 4t(1-\alpha)^2$, $b_\theta = -\alpha^2|R_{\mathbf{g},\theta}| - 4t(1-\alpha)$ and $c_\theta = \alpha\sum_{i\in R_{\mathbf{g},\theta}} y_i + (1-\alpha)\sum_{i\in R_{\mathbf{g},\theta}} g_i^2 - t$, and $R_{\mathbf{g},\theta} = \{i|g_i \geq \theta\}$. $|R_{\mathbf{g},\theta}|$ is the cardinality of the set. The root finding procedure of (4) can be achieved by the following sequence $\{\theta_k\}$:

$$\theta_k = \frac{-b_{\theta_{k-1}} + \sqrt{b_{\theta_{k-1}}^2 - 4a_{\theta_{k-1}}c_{\theta_{k-1}}}}{2a_{\theta_{k-1}}}. \tag{5}$$

The algorithm for problem (1) is summarized as algorithm 1.

Algorithm 1 Euclidean projections on elastic net

Input: $\mathbf{y} \in \mathbb{R}^n$, $0 < \alpha < 1$, $t > 1$ and $0 < \theta_0 < y_n$

Output: $\mathbf{x}$

1: **repeat**
2: Update θ by (5)
3: **until** reach maximal iterations or $f_{en}(\theta_k) == 0$
4: set $\mathbf{x}$ according to (2).

For the optimization procedure in a two-layer network, each single layer module identification share the same algorithm as we adopt the alternating optimization strategy.

3 Tables and figures

Figure S1, S1 and S3 show the GO terms of corresponding modules.

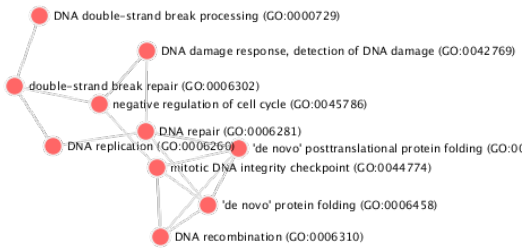

Figure S1: GO terms network of the identified module at 24h

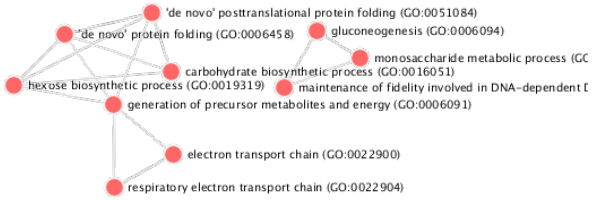

Figure S2: GO terms network of the identified module at 48h

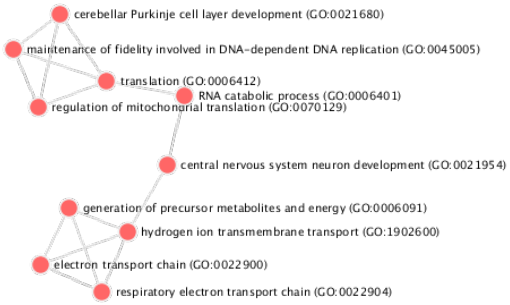

Figure S3: GO terms network of the identified module at 72h

4 Case study

Here is a case study about how to use the package to generate a synthetic network with 10000 nodes which contains a single module. And then we use a grid search method to select optimal parameters for active module identification algorithm.

```
n=10000
k=100
theta = 0.5
pp <- networkSimulation(n,k,theta)
moduleid <- pp[[3]]
aset <- seq(0.1,0.9,by=0.1)
lambdaset <-  2^seq(-5,5)
#using a grid search to select lambda and alpha
gridFscore <- matrix(0, nrow=length(lambdaset), ncol= ength(aset))
for (j in 1:length(lambdaset)) {
        for (k in 1:length(aset)) {
                x <- moduleIdentificationGPFixSS(pp[[1]],pp[[2]],rep(1/n,n),maxiter=500,
                        a=aset[k],lambda = lambdaset[j])
                predictedid<-which(x[[2]]!=0)
                recall <-  length(intersect(predictedid,moduleid))/length(moduleid)
                precise <-  length(intersect(predictedid,moduleid))/length(predictedid)
                print(2*precise*recall/(precise+recall))
                gridFscore[j,k] <- 2*precise*recall/(precise+recall)
        }
}
```

Here is a case study about how to use the package to select the proper parameter so that the desired module size can be archived. When fixing $\lambda = 0.01$, we use a binary search method to select α for elastic net penalty which control the sparsity of the module.

```
# binary search parameter to fix module size to 100~200
abegin = 0.01
aend = 0.9
maxsize = 200
minsize = 100
for (i in 1:100) {
        x <- moduleIdentificationGPFixSS(W,z,rep(1/n,n),a=(abegin+aend)/2,lambda=0.001,maxiter=500)
        predictedid<-which(x[[2]]!=0)
        if(length(predictedid) > maxsize){
                abegin = (abegin+aend)/2
        }else if (length(predictedid) < minsize){
                aend = (abegin+aend)/2
        }else
                break
}
```

More details about functions and case studies can be found at the package website[1].

References

[1] Matteo Fischetti, Horst W Hamacher, Kurt Jørnsten, and Francesco Maffioli. Weighted k-cardinality trees: Complexity and polyhedral structure. *Networks*, 24(1):11–21, 1994.

[2] Gerhard J Woeginger. Computing maximum valued regions. *Acta Cybern.*, 10(4):303–315, 1992.

[1] https://github.com/fairmiracle/AMOUNTAIN

[3] Wenyuan Li, Chun-Chi Liu, Tong Zhang, Haifeng Li, Michael S Waterman, and Xianghong Jasmine Zhou. Integrative analysis of many weighted co-expression networks using tensor computation. *PLoS Comput Biol*, 7(6):e1001106, 2011.

[4] Trey Ideker, Owen Ozier, Benno Schwikowski, and Andrew F Siegel. Discovering regulatory and signalling circuits in molecular interaction networks. *Bioinformatics*, 18(suppl 1):S233–S240, 2002.

[5] Pinghua Gong, Kun Gai, and Changshui Zhang. Efficient euclidean projections via piecewise root finding and its application in gradient projection. *Neurocomputing*, 74(17):2754–2766, 2011.