A review on graph optimization and algorithmic frameworks

Alessandro Benfenati, Emilie Chouzenoux, Laurent Duval, Jean-Christophe Pesquet, Aurélie Pirayre

A review on graph optimization and algorithmic frameworks

Alessandro Benfenati, Emilie Chouzenoux, Laurent Duval,
Jean-Christophe Pesquet, Aurelie Pirayre

October 19, 2018

1 Introduction

In this report, we make a review of optimization problems involving graphs and state-of-the-art algorithms to solve them. First, we present a set of discrete optimization problems and resolution methods for edge selection problems. Then, we address the matrix optimization problems involved in the estimation of precision or covariance matrices given observations from multivariate Gaussian distribution.

2 Discrete optimization methods for graph edge selection

2.1 Introduction and notation

Let us consider a graph $\mathcal{G}$ composed of two objects: nodes (or vertices) and edges (or arcs), which tie nodes together. The set of nodes (corresponding to genes) is denoted by $\mathcal{V} = \{v_1, \ldots, v_G\}$. We introduce $\mathbb{V} = \{1, \ldots, G\}$ as the set of node indices. The set $\mathcal{E}$ refers to the set of edges, corresponding to plausible interaction between nodes. An edge between nodes i and j is labeled by $e_{i,j}$. From these data, nodes of the graph $\mathcal{G}$ can be multi-valued i.e. each node v_i is valued by the vector $\mathbf{m}_i$. The associated unweighted adjacency matrix[1] is denoted by $\mathbf{W}_{\mathcal{V}} = \mathbf{1}$, where $\mathbf{1}$ refers to a matrix of size $G \times G$ full of 1. We will denote the graph $\mathcal{G}(\mathcal{V}, \mathcal{E}; \omega)$, where $\mathcal{V}$ is the set of nodes and $\mathcal{E}$ the set of edges. From this multi-valued graph on nodes, the inference consists in recovering true regulatory links between genes. The resulting set of true links is denoted by $\mathcal{E}^*$ and the underlying graph $\mathcal{G}^*$. While some methods propose to directly infer $\mathcal{G}^*$ from $\mathcal{G}_{\mathcal{V}}$, some others require two steps:

Firstly node-node interaction scores are computed leading to a node-node interaction matrix $\mathbf{W}_{\mathcal{E}} \in \mathbb{R}^{G \times G}$, where the element $\omega_{i,j}$ of $\mathbf{W}_{\mathcal{E}}$ is a weight reflecting the strength of the interaction between node i and j. The node-node interaction matrix allows to define the graph $\mathcal{G}_{\mathcal{E}}$ where nodes are non-valued while edges $e_{i,j}$ are weighted by the element $\omega_{i,j}$ of the matrix $\mathbf{W}_{\mathcal{E}}$. In such a case, the matrix $\mathbf{W}_{\mathcal{E}}$ defines the adjacency matrix of the graph $\mathcal{G}_{\mathcal{E}}$. As nodes and edges are the same in $\mathcal{G}_{\mathcal{V}}$ and $\mathcal{G}_{\mathcal{E}}$, we can use the same notation for their respective sets of nodes and edges.

From the fully-connected and weighted network $\mathcal{G}_{\mathcal{E}}$, an edge selection is then performed to recover $\mathcal{E}^*$ by retaining edges having relevant weights only, ideally corresponding to true

[1] Matrix encoding the graph structure by setting elements to 1 when an edge is present in the graph and 0 otherwise.

regulatory relationships. This edge selection task is classically performed by removing all edges whose weights $\omega_{i,j}$ (possibly their absolute value) are lower than a threshold λ. The aim of this section is to discuss several methods based on discrete optimization tools that allow to perform this task efficiently.

2.2 Tresholding approach

Let us define, for each edge $e_{i,j} \in \mathcal{E}$ with weight $\omega_{i,j}$, a binary label $x_{i,j}$ of edge presence such that:

$$\forall (i,j) \in \mathbb{V}^2 \quad \text{and} \quad j > i, \qquad x_{i,j} = \begin{cases} 1 & \text{if } e_{i,j} \in \mathcal{E}^*, \\ 0 & \text{otherwise.} \end{cases} \tag{1}$$

Each label $x_{i,j}$ indicates the presence or the absence of the edge $e_{i,j}$ in the final graph $\mathcal{G}^*$. Performing a classical thresholding to select relevant edges is equivalent to defining an optimal edge labeling $x^*_{i,j}$ such that:

$$\forall (i,j) \in \mathbb{V}^2 \quad \text{and} \quad j > i, \qquad x^*_{i,j} = \begin{cases} 1 & \text{if } \omega_{i,j} > \lambda, \\ 0 & \text{otherwise.} \end{cases} \tag{2}$$

This optimal edge labeling $\boldsymbol{x}^*$ can be obtained by solving a simple regularized optimization problem:

$$\underset{\boldsymbol{x} \in \{0,1\}^E}{\text{maximize}} \quad \sum_{\substack{(i,j) \in \mathbb{V}^2 \\ j > i}} \omega_{i,j}\, x_{i,j} \quad + \quad \lambda\,(1 - x_{i,j}), \tag{3}$$

where E is the number of edges and equals $G(G-1)/2$ as the graph $\mathcal{G}$ is supposed undirected and λ the regularization parameter. The first term alone would select all edges. The second term restricts this selection to those with weights larger than λ. Hence, the threshold parameter λ in classical thresholding becomes a regularization parameter.

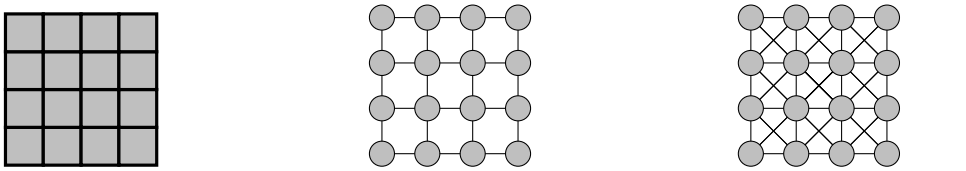
2.3 Maximal flow for discrete optimization

A classical problem encountered in computer vision is image segmentation, which aims at partitioning an image (set of pixels) into multiple objects (subsets of pixels) sharing the same characteristics. In other words, image segmentation aims at assigning a label to each pixel in an image. Pixels sharing certain characteristics (intensity, color, texture, etc.) are assigned to the same label. A large number of approaches exists under various frameworks: thresholding and clustering, variational methods, graph partitioning methods, to name a few. Graph partitioning methods encompass several approaches such as: normalized cuts [21], random walker [13], minimum spanning tree [18] or minimum cut [9], for instance. We now focus on minimum cut models and algorithms proposed to solve them.

An image can be seen as a structured graph, where pixels of the image are associated to nodes. A node is classically linked by edges to its four nearest neighbors, corresponding to its four "nearest" pixels in the cardinal directions. A variety of more complex graphs exists and allows connections with more "nearest" neighbors such as an eight nearest neighbors structure, for instance. Fig. 1 illustrates possible graph constructions from an image.

Weights $\omega_{i,j}$ can be defined for each edge $e_{i,j}$, and are commonly related to pixel intensities I_i and I_j [22]:

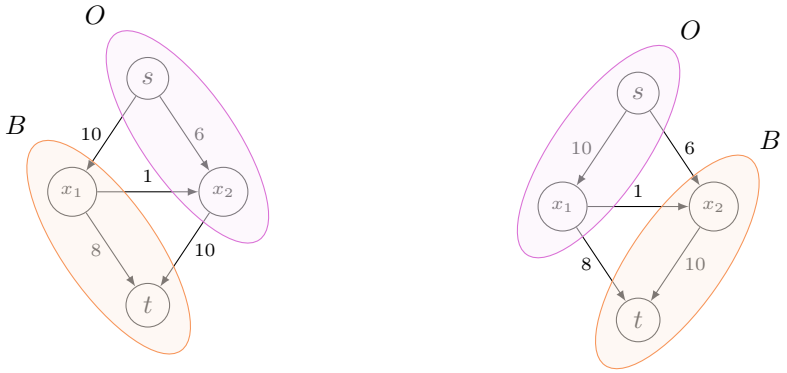
$$\omega_{i,j} = \exp(-\beta (I_i - I_j)^2). \tag{4}$$



(a) Image with 4×4 pixels. (b) 4-connected pixel graph. (c) 8-connected pixel graph.

Figure 1: Graph representations of a 4×4 image

From the graph represented in Fig. 1b, two special nodes are added: the source s — node without entering edges — and the sink t — node without leaving edges. The new generated graph is called a flow network $\mathcal{G}_f$ (or transportation network), and edge weights are called capacities. In a flow network $\mathcal{G}_f$, a cut is defined as a node partition into two disjoint subsets O and B such that $s \in O$ and $t \in B$ (subsets names borrow from image processing, denoting object and background). The capacity of a cut is obtained by summing the capacities of edges crossing the cut. As Fig. 2 illustrates with a toy example, the minimum cut problem is thus to find an $s - t$ cut in $\mathcal{G}_f$ that minimizes the cut capacity.



(a) Cut capacity = 21. (b) Cut capacity = 15.

Figure 2: Cuts in a transportation network:(a) Arbitrary cut and (b) minimum cut in a flow network $\mathcal{G}_f$. The cut leads to a node partitioning such that the source s belongs to a subset O and the sink t to a subset B, for instance. The cut capacity is obtained by summing the weights of edges crossing the cut. Finding the minimal cut capacity solved the minimum cut problem.

From a mathematical viewpoint, the minimum cut problem can be viewed as a discrete optimization problem. Indeed, this problem aims at finding a label variable x_i for each node v_i, where the label reflects the class the node belongs to. As the basic problem implies two classes only, x_i is a binary variable taking 1 or 0 values. Two nodes are linked by an edge with a capacity $\omega_{i,j} \geq 0$. These capacities reflects pixel similarity in terms of intensity, color or texture, for instance, and drive the partitioning. For seeded image segmentation [4], a constraint is added on specific nodes s and t such that s belongs to one class and t to the other class. This constraint is equivalent to fixing x_s, the label of s, to 1 and to fixing x_t, the label of t, to 0. Hence, the minimum cut problem is thus simply formulated as the minimization of a discrete

energy function:

$$\begin{aligned} &\underset{\boldsymbol{x}}{\text{minimize}} \quad \sum_{(i,j)\in\mathbb{V}^2} \omega_{i,j}\,|x_i - x_j|, \\ &\text{subject to } x_s = 1 \text{ and } x_t = 0. \end{aligned} \tag{5}$$

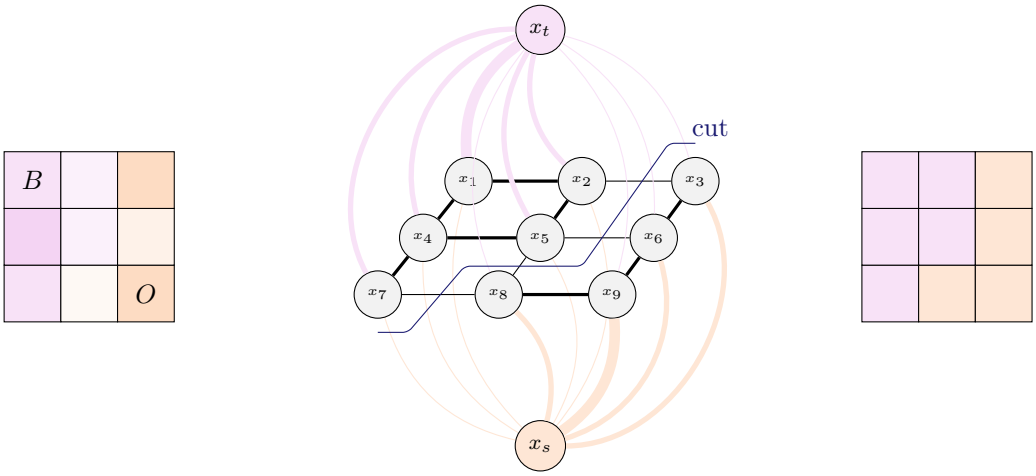
It has been proved in [9] that a dual problem exists and consists in maximizing a flow from s to t in $\mathcal{G}_f$. The duality minimum cut/maximal flow is exploited for image segmentation in an approach called "Graph Cuts" in the computer vision community. In a transportation network $\mathcal{G}_f$, a flow is a function assigning to each edge a value under two conditions. The first one is a capacity constraint: $f(e_{i,j}) \leq \omega_{i,j}$: for a given edge $e_{i,j}$, the assigned value of the flow $f(e_{i,j})$ is lower or equals to the edge capacity $\omega_{i,j}$. When $f(e_{i,j}) = \omega_{i,j}$, the edge is said saturated. The second condition refers to a divergence-free constraint: $f_e(v_i) = f_l(v_i)$. The sum of the flow entering each node v_i, and denoted by $f_e(v_i)$, is equal to $f_l(v_i)$, the sum of the flow leaving the node v_i. Hence, the problem consists in finding the maximal flow that going from s to t under the two mentioned constraints [9]. The resulting maximum flow value is equal to the capacity of the minimum cut. A large number of maximal flow algorithms have been proposed to solve the minimum cut problem [7, 12, 3].

Let us now discuss on how Graph cuts can lead to a segmented image. Suppose that we aim at partitioning an image into two groups of pixels according to their intensities such that one group is related to the background B and the other group to an object O in the image. Thus, each pixel node v_i can be labeled by x_i and can either take 0 or 1 valuation. A node label of one corresponds to a pixel belonging to the object while a label of zero is for a pixel belonging to the background. Now, let the transportation network $\mathcal{G}_f$ be the one that links all pixel nodes to s and t with infinite weights. Capacities between pixel nodes are weights $\omega_{i,j}$ defined as in 4, for instance. The source s is labeled by 1 (the reference label for the object) and the sink t by 0 (and is the reference label for the background). Looking for a minimum cut in such a graph is the same as finding a maximal flow. The maximal flow computation leads to saturated edges. Nodes v_i reaching node s without encountering saturated edges will be labeled by 1 as it is the label of s. Similarly, nodes v_i reaching node t *via* non-saturated edges will be labeled by 0, the label value of t. Resultantly, a label is affected at each node and reflect the groups of pixels it belongs to: nodes labeled by 1 encode pixels belonging to the object while nodes labeled by 0 encode pixels belonging to the background. Fig. 3 illustrates image segmentation with Graph Cuts.

The energy function to be minimized in 5 is one of the many possible energy function that can be solved using Graph Cuts. The generic formulation of the energy function $E(\boldsymbol{x})$ to be minimized *via* Graph Cuts for pixel-labeling problem takes the following form [14]:

$$E(\boldsymbol{x}) \quad = \quad \sum_{i\in\mathbb{V}} D_i(x_i) \quad + \quad \sum_{\substack{(i,j)\in\mathbb{V}^2 \\ j>i}} V_{i,j}(x_i, x_j), \tag{6}$$

where the first term is a data fidelity term derived from observations and reflects the cost to assign the label x_i to the node v_i (pixel p_i). The second term is a pairwise penalization term promoting spatial smoothness and encodes the cost to assign labels x_i and x_j to the nodes v_i and v_j (pixels p_i and p_j), respectively.



(a) Initial image with seeds. *(b) Cut in the flow network $\mathcal{G}_f$.* *(c) Segmented image.*

Figure 3: Image segmentation with Graph Cuts: A flow network $\mathcal{G}_f$ is constructed from the graph of the initial image (a): all pixel nodes are linked to a source s and a sink t. Some pixel nodes, called seeds, are pre-labeled either by O or B such that these nodes are associated to the object or the background in the segmented image. After computing a maximum flow in $\mathcal{G}_f$ (b), a node is labeled by the label of x_s whether the node can be reached from the source s through non-saturated edges (thick edges) or by the label of x_t in the contrary case. The final labeling $\boldsymbol{x}^*$ leads to the segmented image (c).

2.4 Random walker for multi-class and relaxed optimization

As mentioned in the previous subsection, image segmentation is a frequently encountered problem in computer vision. While extensions of Graph Cuts for multi-label problems can be used, another algorithm called random walker provides an alternative. Random walker is a semi-supervised graph partitioning algorithm. Based on a network $\mathcal{G}$, valued on its edges by weights $\omega_{i,j}$, and composed of a set $\mathcal{V}$ of G nodes, let us define by $\mathcal{V}_M$ a subset of K pre-labeled (marked/seeded) nodes. We can thus define the complementary subset of unlabeled nodes $\mathcal{V}_U$. Knowing the label of the nodes in $\mathcal{V}_M$, the random walker algorithm assigns a label to the remaining nodes in $\mathcal{V}_U$.

In more details, let $K \in \mathbb{N}$ be the number of possible label values of nodes from $\mathcal{V}_M$. In addition, let $y_i \in \{1, \ldots, K\}$ be the label variable for node v_i and $\boldsymbol{y} \in \mathbb{N}^G$ be the vector gathering the label variables of the G nodes. Defining a cost function $E(\boldsymbol{y})$ as follow

$$E(\boldsymbol{y}) = \sum_{(i,j)\in\mathbb{V}^2} \omega_{i,j}(y_i - y_j)^2, \tag{7}$$

the random walker algorithm solves the following constrained minimization problem:

$$\begin{aligned} &\underset{\boldsymbol{y}}{\text{minimize}} \quad E(\boldsymbol{y}), \\ &\text{subject to } y_i = k, \quad \forall v_i \in \mathcal{V}_M. \end{aligned} \tag{8}$$

In [13], the cost function $E(\boldsymbol{y})$ in 8 can be re-expressed as a combinatorial formulation of the

Dirichlet integral:

$$E(\boldsymbol{y}) = \sum_{(i,j)\in\mathbb{V}^2} \omega_{i,j}(y_i - y_j)^2 = \boldsymbol{y}^\top L\, \boldsymbol{y},$$

where L is the combinatorial Laplacian matrix of the graph $\mathcal{G}$, defined as:

$$L_{i,j} = \begin{cases} d_i & \text{if } i = j, \\ -\omega_{i,j} & \text{if } v_i \text{ and } v_j \text{ are adjacent nodes,} \\ 0 & \text{otherwise,} \end{cases} \tag{9}$$

with d_i the degree of node v_i. Taking into account the constraint on pre-labeled nodes in 8, only the labels of unseeded nodes have to be determined, and the energy to be minimized in Problem 8 can be decomposed into:

$$E(\boldsymbol{y}_U) = \begin{bmatrix} \boldsymbol{y}_M^\top & \boldsymbol{y}_U^\top \end{bmatrix} \begin{bmatrix} L_M & B \\ B^\top & L_U \end{bmatrix} \begin{bmatrix} \boldsymbol{y}_M \\ \boldsymbol{y}_U \end{bmatrix} = \boldsymbol{y}_M^\top L_M \boldsymbol{y}_M + 2\boldsymbol{y}_U^\top B^\top \boldsymbol{y}_M + \boldsymbol{y}_U^\top L_U \boldsymbol{y}_U, \tag{10}$$

where, in this context, $\boldsymbol{y}_M$ and $\boldsymbol{y}_U$ correspond to probability vectors of seeded and unseeded nodes, respectively. The unique critical point is obtained by differentiating the energy $E(\boldsymbol{y}_U)$ with respect to $\boldsymbol{y}_U$

$$\frac{\partial E(\boldsymbol{y}_U)}{\partial \boldsymbol{y}_U} = B^\top \boldsymbol{y}_M + L_U \boldsymbol{y}_U \tag{11}$$

thus yielding

$$L_U \boldsymbol{y}_U = -B^\top \boldsymbol{y}_M, \tag{12}$$

which is a system of linear equations with $|\mathcal{V}_U|$ unknowns. Note that if the graph is connected or if every connected component contains a seed, then 12 will be nonsingular and a unique solution will be found.

Let us define the set of labels for the seed nodes as a function $Q(v_i) = k$, for all $v_i \in \mathcal{V}_M$, where $k \in \{1, \ldots, K\}$. For each label k, a vector of markers $M^{(k)}$ of size $|\mathcal{V}_M|$ can thus be defined such that, for each node $v_i \in \mathcal{V}_M$

$$m_i^{(k)} = \begin{cases} 1 & \text{if } Q(v_i) = k, \\ 0 & \text{if } Q(v_i) \neq k. \end{cases} \tag{13}$$

The marker matrix $M = [M^{(1)}, \ldots, M^{(K)}]$ thus gathers all the vector of markers. By analogy, let us define the matrix $Y = [Y^{(1)}, \ldots, Y^{(K)}]$, where for all $k \in \{1, \ldots, K\}$, the vector $Y^{(k)}$ is of size $|\mathcal{V}_U|$. For each node $v_i \in \mathcal{V}_U$, the component $y_i^{(k)}$ denotes the probability for the node v_i to be assigned to the label k. Probabilities in Y are unknown and have to be computed. Based on Equation 12, they can be computed by solving the following system of linear equations:

$$L_U Y = -B^\top M. \tag{14}$$

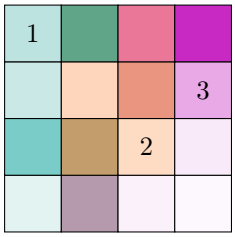
This strategy is equivalent to solving K binary-labeling sub-problems instead of solving a K-labeling problem. Nevertheless, dealing with probabilities enforces a sum-to-one constraint for each node $i \in \{1, \ldots, G\}$ i.e.

$$\forall i \in \{1, \ldots, G\}, \qquad \sum_{k=1}^{K} y_i^{(k)} = 1. \tag{15}$$

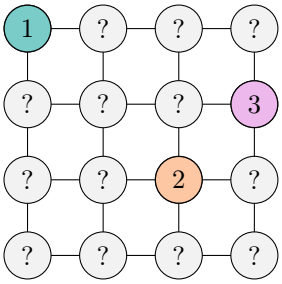
This implies that only $K-1$ systems of linear equations must be solved. Once probabilities at each node and for each label are computed, a final labeling has to be assigned. For this purpose, the label given by the maximal probability is assigned to each node:

$$\forall i \in \{1,\ldots,G\}, \qquad y_i^* = \arg \max_{k \in \{1,\ldots,K\}} \; y_i^{(k)}. \tag{16}$$

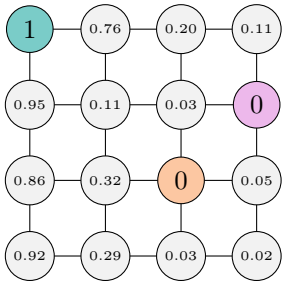
Fig. 4 illustrates how the random walker algorithm can segment an image in 3 classes. It can be interesting to view how the random walker algorithm assigns an unseeded pixel to a label. Indeed, an elegant analogy can be draw. Given a weighted graph, if a a random walker leaving the pixel is most likely to first reach a seed bearing label s, assign the pixel to label s.



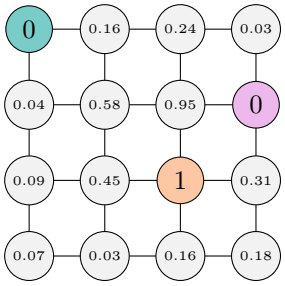
(a) Image with seeds.



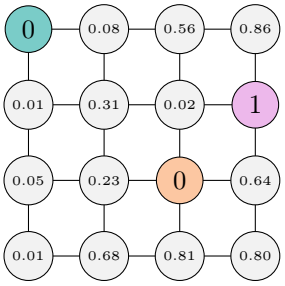
(b) 3-labeling problem.



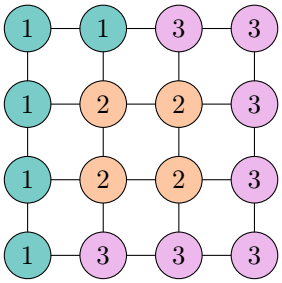
(c) First sub-problem.



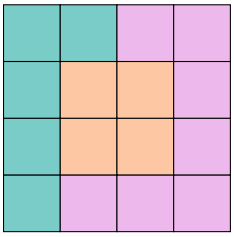
(d) Second sub-problem.



(e) Third sub-problem.



(f) Random walker output.



(g) Segmented image.

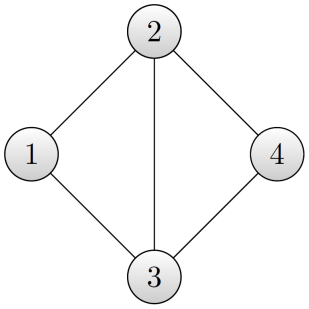
Figure 4: Image segmentation with random walker: On the initial image to be segmented (a), three labels valued by 1,2 and 3 are defined. The graph representation of the image is represented in (b), where edges are valued by weights from a function of the intensity gradient. The multi-labeling problem in (b) is decoupled into 3 sub-problems from (c) to (e), where for each of them, the label of the corresponding markers is set to 1 while keeping the others equal to 0. Assignment probabilities are then computed for the unlabeled nodes. They correspond to the probability that a random walker, starting at each node first reaches the pre-labeled node currently set to unity. The final graph partitioning in (f) is obtained by assigning to each node the label that corresponds to its greatest probability, yielding the segmented image (g).

3 Graph inference with matrix optimization methods

3.1 Problem statement

The problem of covariance selection was originally studied in [5], and it has strong connection with the problem of graph estimation. Let suppose to have $1, \ldots, p$ nodes, some of them connected

by a link, or *edge*: one can represents the connection between nodes by a matrix C, in which the element $c_{ij} \neq 0$ if the node i and the node j are connected. It is obvious that the diagonal elements c_{ii} are always different from zero, since they represent a loop around the i–node. An edge between two nodes i and j is absent (i.e. $c_{ij} = 0$) if these two nodes are conditionally independent given all the other nodes.



(a) A graph

$$\begin{pmatrix} 1.0204 & -0.1020 & -0.1020 & 0 \\ -0.1020 & 1.0204 & 0.0204 & -0.1020 \\ -0.1020 & 0.0204 & 1.0204 & -0.1020 \\ 0 & -0.1020 & -0.1020 & 1.0204 \end{pmatrix}$$

(b) The matrix describing the graph.

Figure 5: The edges between nodes of the graph on the left are described in the matrix C on the right: the node 1 and 2 are linked, hence $c_{12} = c_{21} \neq 0$. This example is taken from [16].

Let consider now a multivalued random variable $\mathcal{X} = \{X_1, X_2, \ldots, X_p\}$, with Gaussian distribution

$$\mathcal{N}(0, \Sigma) = \frac{1}{\sqrt{(2\pi)^p |\Sigma|}} \exp\left(-\frac{1}{2} \left\langle \mathcal{X}, \Sigma^{-1} \mathcal{X} \right\rangle\right) \tag{17}$$

where Σ is the matrix of covariances ($\Sigma_{ij} = \mathrm{Cov}(X_i, X_j)$) and $|\cdot|$ denotes the determinant of a matrix.
Let suppose to be given of n realizations $X^1, X^2, \ldots, X^n$ of $\mathcal{X}$, and furthermore the covariance matrix is unknown. The classical way to obtain an estimation of Σ is to maximize the log–likelihood of the products of the Gaussian distribution computed in the n realizations:

$$\hat{\Sigma} = \arg\max_{A \succ 0} \log\left(\prod_{i=1}^{n} \frac{1}{\sqrt{(2\pi)^p |A|}} \exp\left(-\frac{1}{2} \left\langle X^i, A^{-1} X^i \right\rangle\right)\right)$$

Considering the empirical covariance matrix $S = \frac{1}{n} \sum_{i=1}^{n} \left\langle X^i, X^i \right\rangle$, performing the computation leads to the maximization problem

$$\hat{C} = \arg\max \log |C| - \mathrm{trace}(SC)$$

where actually the matrix C is the *inverse* of the estimated covariance matrix. C is the precision (or concentration) matrix which describes the conditional independences between the component of $\mathcal{X}$: $c_{ij} = 0$ means that the two random variables X_i and X_j are conditionally independent given all other variables (see [26]).

In various applications, one can consider the i–th node of a graph as the realization of a Gaussian random vaiable X_i with distribution $\mathcal{N}(0, \sigma_i^2)$; when the mean is not zero, i.e. $X_i \sim \mathcal{N}(\mu, \sigma^2)$, a linear shift $X_i - \mu$ clearly puts the discussion in the mentioned case. It is natural hence to consider a multivalued random variable $\mathcal{X}$ with a Gaussian distribution $\mathcal{N}(0, \Sigma)$: each component of $\mathcal{X}$ represents a node on the graph. Hence, *the problem of estimate the concentration matrix*

of a graph is strictly linked to the problem of the covariance matrix estimation. See [24, 15, 8] for more statistical proprieties of this two connected problems.

Another way to get to the same maximization problem is to consider the *Bregman divergence*: given a strictly convex function g defined on the set of matrices, the Bregman divergence of g at $B \in \mathcal{M}_{n\times m}(\mathbb{R})$ is

$$D(A\|B) = g(A) - g(B) - \langle \nabla g(B), A - B\rangle, \quad A \in \mathcal{M}_{n\times m}(\mathbb{R})$$

Considering the log–determinant barrier function defined on $\mathcal{S}_{++}$

$$g(A) = \begin{cases} -\log|A| & \text{if } A \succ 0 \\ \\ +\infty & \text{otherwise} \end{cases}$$

Then, the problem is estimating C given an initial approximation of it $\tilde{C}$ by minimizing the Bregman divergence of g at $\tilde{C}$:

$$\hat{C} = \arg\min_{C\succ 0} -\log|C| + \operatorname{trace}(C\tilde{C}^{-1})$$

and considering $\tilde{C}^{-1} = S$, where S is the empirical covariance matrix, one is led to the minimization (or maximization, by a simple switch of sign) problem.

3.2 State-pf-the-art approaches

Solving simply $\arg\max \log|C| - \operatorname{trace}(SC)$ leads to the obvious solution $\hat{C} = S^{-1}$, which in case of small n is not a reliable estimation of Σ; moreover, taking a look at the graph model, on which the aim of the present survey is focused, one wants C to be *sparse*, i.e. the number of edges must be very small. In order to enforce this characteristic on the computed solution, an ℓ_0 regularization term is added to the objective function appearing in the maximization problem, but unfortunately this term makes the problem non–convex. A little trick to overcome this difficult is to take the convexification of the ℓ_0 term, i.e. the ℓ_1 norm which however promotes sparsity in the final solution. Hence, the problem to be solved is

$$\hat{C} = \arg\max_{C\succ 0} \log|C| - \operatorname{trace}(SC) - \mu\|C\|_1 \tag{18}$$

which is called LASSO estimation in the literature.

In the past years, different algorithms with different approaches were proposed: in this survey the starting point are the works [11] and [1], which are based on a dual formulation of the problem. Trying to follow the chronological appearance, the projected subgradient method [6] and the application [26] of a Vandenberghe's algorithm [23] are presented. The last algorithm shown is the application of the Alternating Direction of Multipliers Method (ADMM) presented in [20, 2].

3.2.1 Block Coordinate Descent Algorithm

The main idea in [1] consists in substituting the ℓ_1 norm appearing in (18) with its conjugate function

$$\|C\|_1 = \sup_U \langle U, C\rangle - \|U\|_1^* = \max_{\|U\|_\infty \le 1} \operatorname{trace}(UC)$$

getting a the primal–dual formulation

$$\max_{C\succ 0}\min_{\|U\|_\infty\leq\mu}\log|C|-\operatorname{trace}(C(S+U)) \tag{19}$$

Exchanging he max and the min, and imposing the first order conditions related to C, ones obtains that

$$-C^{-1}-(S+U)=0\leftrightarrow C=(S+U)^{-1}$$

moreover, setting $W=U+S$, the original problem becomes

$$\hat{\Sigma}=\operatorname{argmax}_{\|W-S\|_\infty\leq\mu}\log|W| \tag{20}$$

In [1, Thh 1] it is proved that for any μ the solution of (20) exists and its unique, and moreover it as bounded eigenvalues. In order to get a full insight into the algorithm, let consider a block symmetric matrix M:

$$M=\begin{pmatrix} A & B \\ B^\top & D\end{pmatrix}.$$

By using the Shur complement, the determinant of such matrix is easily computed:

$$|M|=|D|\cdot|A-B^\top D^{-1}B|$$

Let now adopt the following notation:

- ⋆ let $W_{\dot{p}\dot{p}}$ be the submatrix of W from which the p–column and the p–th row were removed;
- ⋆ let w_p be the p–th row (which is identical to the p–th column, since W is symmetric), with the p–th element suppressed;
- ⋆ as usual, let W_{pp} the element of W with p index of row and column.

Thus, we can write

$$W=\begin{pmatrix} W_{\dot{p}\dot{p}} & w_p \\ w_p^\top & W_{pp}\end{pmatrix}$$

The Shur complements then tells that[2]

$$|W|=|W_{\dot{p}\dot{p}}|\cdot|W_{pp}-w_p^\top W_{\dot{p}\dot{p}}^{-1}w_p|$$

and since $|W_{pp}-w_p^\top W_{\dot{p}\dot{p}}^{-1}w_p|$ is just a real number, maximizing $\log|W|$ corresponds to minimize $w_p^\top W_{\dot{p}\dot{p}}^{-1}w_p$. The procedure shown consists in considering the j–th column of W, performing the minimization procedure

$$\hat{y}=\arg\min_y y^\top W_{\dot{j}\dot{j}}^{-1}y\quad\text{s.t. }\|y-S_j\|_\infty\leq\mu.$$

After a sweep on all the columns, the convergence condition is checked.

The regularization parameter is chosen as in the following formula:

$$\mu_\alpha=\max_{i>j}(\sigma_i,\sigma_j)\frac{t_{n-2}(\alpha/2p^2)}{\sqrt{n-2+t_{n-2}^2(\alpha/2p^2))}}$$

[2] W.

Algorithm 1 Banerjee, 2008

Choose μ, set $W^{(0)} = S + \mu\,\mathrm{I}$;
for $k = 0, 1, 2, \ldots$ **do**
  for $j = 0, 1, 2, \ldots, p$ **do**
    Let $W^{(j-1)}$ the current iterate. Solve

$$\hat{y} = \arg\min_{y} y^{\top} W_{\dot{j}\dot{j}}^{-1} y \quad \text{s.t. } \|y - S_j\|_{\infty} \leq \mu. \tag{21}$$

    $W^{(j)}$ is $W^{(j-1)}$ with column/row $\hat{y}$
  end for
  Set $W^{(0)} = W^{(p)}$
  Check convergence condition

$$\mathrm{trace}(\left(W^{(0)}\right)^{-1} S) - p + \mu \left\| \left(W^{(0)}\right)^{-1} \right\|_1 \leq \varepsilon$$

end for

where $t_{n-2}(\alpha)$ denotes the $(100-\alpha)\%$ point of the Student's distribution for $n-2$ dof. With such a parameter, one has that

$$P\left(\exists k \in \{1, \ldots, p\} | C_k^{\mu} \nsubseteq C_k\right) \leq \alpha$$

The convergence results for 1 is shown in [1, Thh 3], where it is also assured that the Ws computed during the procedure are all definite positive. A very important fact about the regularization parameter is that for any $k \in \{1, \ldots, p\}$, if $\mu \geq |S_{kj}|$ for all $j \neq k$, then one can show that column and row k of the solution of (20) are zero, excluding the diagonal element. The implications of this result consist in observing that, for a given second moment matrix S, if μ is chosen such that the condition on S is met for some column k, then the k–th variable is estimated to be independent of all other variables in the distribution. In particular, if $\mu \geq |S_{kj}|$ for all $k > j$, then all variables are estimated to be pairwise independent.

Setting $Q \equiv \sqrt{W_{\dot{j}\dot{j}}^{(j-1)}}$, $b = \frac{1}{2} Q^{-1} S_j$, taking the dual of (21) is equivalent to solve

$$\min_{\theta} \|Q\theta - b\|_2^2 + \mu \|\theta\|_1$$

which is a LASSO formulation. Hence, the coordinate descent procedure can be viewed as a sequence of LASSO problems. This approach is very similar to the one of [17], but two differences are evident:

- ⋆ a multiple of the identity matrix is added to the empirical covariance matrix, in order to obtain the uniqueness of the solution without any other requirements;
- ⋆ the matrix $W_{\dot{j}\dot{j}}^{(j-1)}$ is never a minor of S, it is updated at each inner step.

Using Nesterov's method [19], a theoretical complexity bound of $\mathcal{O}(p^{4.5}\varepsilon)$ is proved, where ε is the desired tolerance.

3.2.2 LASSO estimation on the dual problem

In [1] it is shown the connection between the dual formulation of (18) and a LASSO formulation, pursuing the solution of the dual problem with a coordinate descent algorithm. In [11], instead, the LASSO problem is solved in a very cheap way. In fact, the minimization problem to be solved is

$$\min_{\theta} \frac{1}{2}\|W_{\dot{p}\dot{p}}^{1/2}\theta - b\|_2^2 + \mu\|\theta\|_1 \tag{22}$$

with the minor difference $b = W_{\dot{p}\dot{p}}^{-1/2}S_p$. It is proved, with some elementary computation, that if θ solves (22), than $\beta = W_{\dot{p}\dot{p}}\theta$ solves also $\arg\min_y y^\top W_{\dot{p}\dot{p}}^{-1} y$ s.t. $\|y - S_p\| \geq \mu$s. The main point of [11] is that (22) looks like a LASSO estimation. In fact, if $W_{\dot{p}\dot{p}} = S_{\dot{p}\dot{p}}$, then the solution θ_p is exactly the LASSO estimate of the p–th variable on the others, hence it recalls the [17] procedure. But since the matrix W is updating, due the sweep on the columns, it is not the case. The general procedure is depicted in Algorithm 2. The inner problem (23) can be easily

Algorithm 2 Friedman, 2008

1: Choose μ, set $W = S + \mu\,\mathrm{I}$;
2: **for** $k = 0, 1, 2, \ldots$ **do**
3: **for** $j = 0, 1, 2, \ldots, p$ **do**
4: Consider $W_{\dot{p}\dot{p}}$ and the relative S_p. Solve

$$\hat{\theta} = \arg\min_{\theta} \frac{1}{2}\|W_{\dot{p}\dot{p}}^{1/2}\theta - b\|_2^2 + \mu\|\theta\|_1 \tag{23}$$

5: $\hat{\beta} = W_{\dot{p}\dot{p}}\hat{\theta}$
6: Fill W with the p–th row and p–th column using $\hat{\beta}$
7: **end for**
8: Check convergence condition
9: **end for**

solved by coordinate descent [10, 25]: in fact, setting for sake of clearness $V = W_{\dot{p}\dot{p}}$ and $u = S_p$, the solution is computed componentwise as

$$\theta_j = \frac{1}{V_{jj}}\mathrm{Soft}\left(u_j - \sum_{k\neq j} V_{kj}\theta_k, \mu\right)$$

where $\mathrm{Soft}\,(x, \tau) = \mathrm{sign}(x)\max\{|x| - \tau, 0\}$ is the soft threshold operator. The interest lies in computing the concentration matrix $C = W^{-1}$ (note that the dual problem estimates the covariance matrix): it can be easily done by some simply calculation:

$$WC = \mathrm{I} \Leftrightarrow \begin{pmatrix} W_{\dot{p}\dot{p}} & w_p \\ w_p^\top & W_{pp} \end{pmatrix}\begin{pmatrix} C_{\dot{p}\dot{p}} & c_p \\ c_p^\top & C_{pp} \end{pmatrix} = \begin{pmatrix} \mathrm{I} & 0 \\ 0^\top & 1 \end{pmatrix}$$

hence

$$\begin{cases} W_{\dot{p}\dot{p}}c_p + w_p C_{pp} = 0 \\ w_p^\top c_p + W_{pp}C_{pp} = 1 \end{cases} \rightarrow \begin{cases} c_p = -W_{\dot{p}\dot{p}}^{-1}w_p C_{pp} \\ C_{pp} = 1/\left(W_{pp} - w_p^\top W_{\dot{p}\dot{p}}^{-1}w_p\right) \end{cases} \rightarrow \left(\hat{\theta} = W_{\dot{p}\dot{p}}^{-1}w_p\right) \rightarrow \begin{cases} c_p = -\hat{\theta}C_{pp} \\ C_{pp} = 1/\left(W_{pp} - w_p^\top\hat{\theta}\right) \end{cases}$$

All these computations can be done during each step of the procedure, but the cheapest way is to store the valued for $\hat{\theta}$ and then compute the inverse matrix when convergence is achieved.

3.2.3 Gradient Projection approach

In the paper [6], the projected gradient method is applied: suppose that $g \in \mathcal{C}^1$, then an iterative procedure to solve

$$x = \arg\max_{x \in \Omega} g(x)$$

where Ω represents a set of constraints, is

$$x^{k+1} = \Pi_\Omega \left(x^k + t \nabla g(x^k) \right),$$

where t is the step size. In order to apply this procedure to (18), let rewrite it as

$$-\log|C| + \operatorname{trace}(SC) + \sum_{ij} \mu_{ij}|c_{ij}|$$

where in this formulation the regularization parameter is a matrix; one can recover the previous cases imposing $\mu_{ij} = \mu \quad \forall i, j$. Imposing a new constrain $Z = C$, the Lagrangian formulation of such problem is

$$-\log|C| + \operatorname{trace}(SC) + \sum_{ij} \mu_{ij}|Z_{ij}| + \operatorname{trace}(W(C - Z))$$

being W is the Lagrangian parameter. Since the Lagrangian is separable, imposing first order conditions on C the dual formulation of the problem becomes thus

$$\max \log|W + S| \quad \text{s.t. } \|W_{ij}\|_1 \leq \mu_{ij}.$$

The desired constraint on the positive definiteness on the solution is assured by the log det function, which acts as a barrier function on the set $\mathcal{S}_{++}$. The dual formulation of the problem via the Lagrangian enables to prove the boundness from below and the uniqueness of the solution as long as $\mu_{ij} > 0$ for $i \neq j$ and $\mu_{ii} \geq 0$ [6, Lemma 1].
The procedure is shown in Algorithm 3. In this particular case, the parameter t is easily computed by a line search based on the second order approximation of the objective function: if the increase is not sufficient, the step size is decreased.
This algorithm is easily extended in presence of block structures in the concentration matrix: the only difference between this block algorithm and Algorithm 3 regards the parameter t. Introducing the block structures does not allow to compute it by a line search, hence an Armijo procedure is adopted.

3.2.4 Interior Point Method

In [26] the covariance selection problem is solved by considering the original formulation (18) with the regularization of the off–diagonal elements of C:

$$\min_{C \succ 0} \operatorname{trace}(SC) - \log(|C|) \quad \text{s.t} \sum_{i \neq j} \|c_{ij}\|_{1,\text{off}} < t. \tag{24}$$

(which can be rewritten as $\operatorname{trace}(\cdot) - \log|\cdot| + \lambda\|\cdot\|_{1,\text{off}}$). The proposal is to employ the method developed in [23], which aims to solved

$$\begin{aligned} \min_{x \in \mathbb{R}^n} \quad & \langle b, x \rangle - \log|G(x)| \\ & \text{s.t. } G(x) \succ 0 \\ & \text{s.t. } F(x) \succ 0 \end{aligned}$$

Algorithm 3 Duchi, 2008

Choose μ, $B_\mu = \{W | W_{ij} < \mu_{ij}|\}$, set $W \in B_\mu$ and $W + S \succ 0$, $W_{ii} = \mu_{ii}$, set tolerance ε;
for $k = 0, 1, 2, \dots$ **do**
  $G = (S + W)^{-1}$
  Impose constraints on the gradient in order to satisfy the constraints

$$\begin{aligned} G_{ii} &= 0 \\ G_{ij} &= 0 \text{ for all } W_{ij} = \mu_{ij}, G_{ij} > 0 \\ G_{ij} &= 0 \text{ for all } W_{ij} = -\mu_{ij}, G_{ij} < 0 \end{aligned}$$

  Line search for t:

$$t = \frac{\text{trace}((S+W)^{-1}G}{)}\text{trace}((S+W)^{-1}G(S+W)^{-1}G)$$

  while $\log|S + \Pi_{B_\mu}(W + tG)| < \log|S + W|$ **do**
    $t \leftarrow t/2$
  end while
  $W = \Pi_{B_\mu}(W + tG)$
  $C = (S + W)^{-1}$
  Compute duality gap

$$\eta = \text{trace}(SC) + \sum_{ij} \mu_{ij}|C_i j| - n$$

  Check convergence $\eta < \varepsilon$
end for

where $F(x) = F_0 + x_1F_1(x) + x_2F_2(x) + \cdots + x_nF_n(x)$, $G(x) = G_0 + x_1G_1(x) + x_2G_2(x) + \cdots + x_nG_n(x)$ and $G_i \succ 0$, $F_i \succ 0$ o all i. Writing (24) as

$$\begin{aligned} \min_C \quad & 2\sum_{i<j} s_{ij}c_{ij} + \sum_i s_{ii}c_{ii} - \log\left|\sum_i c_{ii}I^i + \sum_{i<j} c_{ij}I^{ij}\right| \\ & \text{s.t. } \sum_i c_{ii}I^i + \sum_{i<j} c_{ij}I^{ij} \succ 0 \\ & \text{s.t. } t - 2\sum_{i<j} \text{sign}(c_{ij})c_{ij} \geq 0, \quad \text{sign}(c_{ij})c_{ij} \geq 0 \end{aligned}$$

where I^i is a matrix full of zeros except the diagonal element in i–th position which is equal to 1, I^{ij} is a matrix full of of zeros except the elements in position i, j and j, i which are equal to 1. Let $\hat{C}$ the LASSO estimator of the concentration matrix: the following result holds.
If $\sqrt{n}\lambda \to \lambda_0$ for $n \to \infty$, the LASSO–type estimator is such that

$$\sqrt{n}(C - \hat{C}) \to \arg\min_{U=U'} V$$

where

$$V = \text{trace}(U\Sigma U\Sigma) + \text{trace}(UW) + \lambda_0 \sum i \neq j \, (u_{ij}\text{sign}(c_{ij})I(c_{ij} \neq 0) + |u_{ij}|I(c_{ij} = 0))$$

in which W is a random symmetric $p \times p$ matrix such that $\text{vec}(W) \sim \mathcal{N}(0, \Lambda)$, where Λ is such that

$$\text{Cov}\,(w_{ij}, w_{kl}) = \text{Cov}\,(X_i X_j, X_k X_l)$$

Moreover, Lemma 3 in [26] states that the proposed algorithm always converges and furthermore converges to a solution of (24). Regarding the selection of the penalty parameter, the author propose to employ a Bayesian information criterion, but is not clear how they calculate it.

An interesting result lies in [26, Lemma 4]: it states that by considering the second order approximation of (24), the solution of the penalized problem can be approximated by the solution of

$$\min_C \text{trace}((C - S^{-1})S(C - S^{-1})S) + \lambda\|C\|_1 \tag{25}$$

Recalling the method proposed in [17], let consider the matrix Θ in which each column j is the solution of the j–th subproblem; let moreover be $\Theta_{ii} = 1$. Then Θ is the unconstrained solution of (25) over the $p \times p$ matrices with diagonal elements fixed to 1.

3.2.5 Alternating Direction of Multipliers algorithm

A very famous and performing algorithm for convex problem is the Alternating Direction of Multipliers Method (ADMM), which allows to remove the constraints by using the Lagrangian multiplier and to compute with a Gauss–Siedel strategy the variables coming into play. Given a general problem

$$\min f(x) \quad \text{s.t. } g(x) = b$$

the ADMM version of it can be written as

$$\min f(x) + \langle \lambda, g(x) - b \rangle + \frac{1}{2\gamma_k}\|g(x) - b\|_2^2$$

with a more compact form (adding and subtracting suitable terms) $\min f(x) + \frac{1}{2\gamma_k}\|g(x) - b + \lambda\|_2^2$. Let's recall the problem to be solved

$$\min_{C \in \mathcal{S}_{++}} -\log|C| + \text{trace}(SC) + \mu\|C\|_1$$

and consider a new constraint: $C \in \mathcal{C} = \{C \in \mathcal{S}_+ | C \succeq \frac{\alpha}{2}\}$, with $\alpha = \frac{1}{\|S\| + n\mu}$ (see [20]). This new constraint allows the Lipschitz constant of gradient of $\log|\cdot|$ to be $\frac{1}{\alpha^2}$. The new formulation of the problem is hence

$$\min_{C \in \mathcal{C}, Y \in \mathcal{C}} \quad -\log|C| + \text{trace}(SC) + \mu\|Y\|_1 \tag{26}$$

$$\text{s.t. } C = Y \tag{27}$$

$$\tag{28}$$

Some preliminary observations:

- ⋆ the sequence γ_k has to be decreasing;
- ⋆ the limit for the Lipschitz constant is α^{-2}: hence it could be very large. γ_k can ot be chosen smaller that α^2, otherwise the solution is hard to reach;

⋆ the request for the Lipschitz–ness of the gradient is due to the fact that the rate of convergence is

$$Y^k - C^* \leq \frac{\|C^0 - C^*\|^2}{2\mu k}$$

and the number of iteration for reaching a give tolerance of ε is $\mathcal{O}\left(\frac{1}{\varepsilon}\right)$.

The procedure is depicted in Algorithm 4

Algorithm 4 Scheinberg, 2010

Choose $\gamma_0, \lambda_0, C^0, Y^0$;
for $k = 0, 1, 2, \dots$ **do**
  Set $\gamma_{k+1} \leq \gamma_k$
  Compute C^{k+1}

$$C^{k+1} \leftarrow \arg\min_{C \in \mathcal{C}} -\log|C| + \operatorname{trace}(CS) + \frac{1}{2\gamma_k}\|C - \left(Y^k + \lambda^k\right)\|_2^2$$

  if $\|C^{k+1}\|_1 > \|C^k\|_1 + \frac{1}{2\gamma_k}\|C^{k+1} - \left(Y^k + \lambda^k\right)\|_2^2$ **then**
    $C^{k+1} = Y^k$
  Compute Y^{k+1}

$$Y^{k+1} \leftarrow \arg\min_{Y} \|Y\|_1 + \frac{1}{2\gamma_k}\|C^{k+1} - \left(Y + \lambda^k\right)\|_2^2$$

  $\lambda^{k+1} \leftarrow \frac{1}{C^{k+1}} - \frac{\left(C^{k+1} - Y^{k+1}\right)}{\lambda^k}$
  Check convergence
end for

Remark: In computing Y^{k+1} the constraint $Y \in \mathcal{C}$ is not included since the computation of it is very hard: since the procedure has to force $X = Y$, it exists $k > \overline{k}$ such that $X^{\overline{k}} = Y^{\overline{k}}$, then $Y^{\overline{k}} \in \mathcal{C}$.
Even if the algorithm looks quite complicated, the solutions of the inner problems are easy to compute. Let us consider the first order condition for the first one:

$$\frac{1}{\gamma_k}C - \frac{1}{C} = \frac{1}{\gamma_k}\left(Y^k + \lambda^k - S\right);$$

since $C \in \mathcal{C}$, one can compute the SVD of the rsh term of the previous equation

$$\frac{1}{\gamma_k}\left(Y^k + \lambda^k - S\right) = QDQ^\top$$

where $Q^\top Q = QQ^\top = \mathrm{I}$; thus

$$Q^\top\left(\frac{1}{\gamma_k}C - \frac{1}{C}\right)Q = (Q^\top CQ \equiv \tilde{C}) = \frac{1}{\gamma_k}\tilde{C} - \tilde{C}^{-1} = D.$$

Solving the second degree equation, the diagonal elements of $\tilde{C}$ are

$$\tilde{C}_{ii} = \frac{\gamma_k d_{ii} + \sqrt{\gamma_k^2 d_{ii} + 4\gamma_k}}{2}$$

and performing the multiplication by Q and $Q^\top$ $C = Q\tilde{C}Q^\top$.

The computation of Y^{k+1} involves a simple soft thresholding, thanks to the fact that the constraint on $\mathcal{C}$ is not included:

$$Y^{k+1} = \mathrm{Soft}_{\gamma_k}\left(C^{k+1} + \lambda_k\right).$$

References

[1] Onureena Banerjee, Laurent El Ghaoui, and John Lafferty. Model selection through sparse maximum likelihood estimation. *Journal of Machine Learning Research*, 2008.

[2] Stephen Boyd, Neal Parikh, Eric Chu, Borja Peleato, and Jonathan Eckstein. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Found. Trends Mach. Learn.*, 3(1):1–122, jan 2011.

[3] Y. Boykov and V. Kolmogorov. An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. *IEEE Trans. Pattern Anal. Mach. Intell.*, 26(9):1124–1137, Sep. 2004.

[4] Yuri Boykov and Marie-Pierre Jolly. Interactive organ segmentation using graph cuts. In *Proc. Medical Image Computing and Computer-Assisted Intervention Conf.*, pages 276–286. Springer, 2000.

[5] A Dempster. Covariance selection. *Biometrics*, 28:157–175, 1972.

[6] J. Duchi, S. Gould, and D. Koller. Projected Subgradient Methods for Learning Sparse Gaussians. In *Proceedings of the Twenty-fourth Conference on Uncertainty in AI (UAI)*, 2008.

[7] Jack Edmonds and Richard M. Karp. Theoretical improvements in algorithmic efficiency for network flow problems. *J. ACM*, 19(2):248–264, Apr. 1972.

[8] D M Edwards. *Introduction to graphical modeling*. Springer, 2000.

[9] L. R. Ford, Jr. and D. R. Fulkerson. Maximal flow through a network. *Canad. J. Math.*, 8:399–404, Jan. 1956.

[10] Jerome Friedman, Trevor Hastie, Holger HÃ¶fling, and Robert Tibshirani. Pathwise coordinate optimization. Technical report, Annals of Applied Statistics, 2007.

[11] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. Sparse inverse covariance estimation with the graphical lasso. *Biostatistics*, 9(3):432–441, jul 2008.

[12] A. V. Goldberg and R. E. Tarjan. A new approach to the maximum flow problem. In *Proc. ACM Symp. Theor. Comput.*, pages 136–146, Berkeley, CA, USA, May 28-30, 1986.

[13] L. Grady. Random walks for image segmentation. *IEEE Trans. Pattern Anal. Mach. Intell.*, 28(11):1768–1783, Nov. 2006.

[14] V. Kolmogorov and R. Zabih. What energy functions can be minimized via graph cuts? *IEEE Trans. Pattern Anal. Mach. Intell.*, 26(2):147–159, Feb. 2004.

[15] Steffen L. Lauritzen. *Graphical Models*. Oxford University Press, 1996.

[16] Nicolai Meinshausen. A note on the lasso for gaussian graphical model selection. *Statistics & Probability Letters*, 78(7):880 – 884, 2008.

[17] Nicolai Meinshausen and Peter Bühlmann. High dimensional graphs and variable selection with the lasso. *ANNALS OF STATISTICS*, 34(3):1436–1462, 2006.

[18] Fernand Meyer. Minimum spanning forests for morphological segmentation. In Jean Serra and Pierre Soille, editors, *Mathematical Morphology and Its Applications to Image Processing*, pages 77–84. Springer, Dordrecht, 1994.

[19] Yu Nesterov. Smooth minimization of non-smooth functions. *Math. Program.*, 103(1):127–152, may 2005.

[20] Katya Scheinberg, Shiqian Ma, and Donald Goldfarb. Sparse inverse covariance selection via alternating linearization methods. In J. D. Lafferty, C. K. I. Williams, J. Shawe-Taylor, R. S. Zemel, and A. Culotta, editors, *Advances in Neural Information Processing Systems 23*, pages 2101–2109. Curran Associates, Inc., 2010.

[21] Jianbo Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Trans. Pattern Anal. Mach. Intell.*, 22(8):888–905, 2000.

[22] M. Unger, T. Pock, W. Trobin, D. Cremers, and H. Bischof. TVSeg - interactive total variation based image segmentation. In *Proc. Brit. Machine Vis. Conf.*, pages 40.1–40.10, Leeds, UK, Sep. 1-4, 2008. British Machine Vision Association and Society for Pattern Recognition.

[23] Lieven Vandenberghe, Stephen Boyd, and Shao po Wu. Determinant maximization with linear matrix inequality constraints. *SIAM Journal on Matrix Analysis and Applications*, 19:499–533, 1998.

[24] Joe Whittaker. *Graphical Models in Applied Multivariate Statistics*. Wiley Publishing, 2009.

[25] Tong T. Wu and Kenneth Lange. Coordinate Descent Algorithms for Lasso Penalized Regression. *The Annals of Applied Statistics*, 2(1):224–244, 2008.

[26] Ming Yuan and Yi Lin. Model selection and estimation in the gaussian graphical model. *Biometrika*, 94(1):19–35, 2007.