A Doubly Nonnegative Relaxation for Modularity Density Maximization

Yoichi Izunaga[a,*], Tomomi Matsui[b], Yoshitsugu Yamamoto[c]

[a] *Faculty of Business Sciences, University of Tsukuba*
3-29-1 Otsuka, Bunkyo-ku, Tokyo 112-0012, Japan
[b] *Graduate School of Decision Science and Technology, Tokyo Institute of Technology*
2-12-1 Ookayama, Meguro-ku, Tokyo 152-8552, Japan
[c] *Faculty of Engineering, Shizuoka University*
3-5-1 Johoku, Hamamatsu-shi, Shizuoka 432-8561, Japan

Abstract

Modularity proposed by Newman and Girvan is the most commonly used measure when the nodes of a network are grouped into internally tightly and externally loosely connected communities. However, some drawbacks have been pointed out, among which is resolution limit degeneracy: being inclined to leave small communities unidentified. To overcome this drawback, Li *et al.* have proposed a new measure called modularity density. In this paper, we propose an equivalent formulation of the modularity density maximization as a variant of semidefinite programming, and demonstrate that its relaxation problem provides a good upper bound on the optimal modularity density. We also propose a lower bounding algorithm based on a combination of spectral heuristics and dynamic programming.

Keywords: Community detection, Modularity density, 0-1 semidefinite programming, Doubly nonnegative relaxation

1. Introduction

One of the most important issues in the network analysis is to identify community structure. Community is a set of nodes which are internally tightly connected while externally loosely connected. In the course of investigating the issue, Newman and Girvan [1], and Newman [2] proposed a novel measure called modularity, and suggested an advantage of maximizing the measure. Though the modularity maximization is now one of the central subjects of research, Fortunato and Barthélemy [3] brought up resolution limit as one of its drawbacks. Resolution limit refers to the sensitivity of the modularity to the total number of edges in the graph, which leaves small communities not identified and hidden inside larger ones. This feature narrows the application of the modularity maximization since most of real-world networks can contain communities with different scales. To overcome the resolution limit, there have been extensive studies so far [4, 5, 6, 7], and recently, Li *et al.* [8] have proposed a new measure, which is called modularity density, and showed that maximizing the modularity density is formulated as a nonlinear binary optimization.

As for the mathematical optimization approaches for the modularity density maximization, Costa [9] has presented some mixed integer linear programming formulations, MILP for short, which enables an application of general-purpose optimizers, e.g., CPLEX, Gurobi and Xpress, to the problem. However, the number of communities must be fixed in advance, and a difficult auxiliary problem need be solved in their formulations. More recently, a hierarchical divisive heuristics has been proposed by Costa *et al.* [10] to obtain a good lower bound on the modularity density.

*Corresponding author
Email address: izunaga@gssm.otsuka.tsukuba.ac.jp (Yoichi Izunaga)

In this paper, for the modularity density maximization, we give a new formulation based on a variant of semidefinite programming called 0-1SDP. The advantage of this formulation is twofold: it does not require the number of communities be known, and its size is independent of the number of edges of the graph in contrast to MILP formulations. In order to obtain an upper bound on the modularity density, we propose to relax 0-1SDP to a semidefinite programming problem with nonnegative constraints, hence the relaxation problem can be solved in polynomial time. Moreover, we develop a method based on a combination of spectral heuristics and dynamic programming to construct a feasible solution from a solution obtained by the relaxation problem.

This paper is organized as follows. In Sections 2 and 3, giving the nonlinear binary programming formulation of the modularity density maximization, we review some of its properties. In Section 4, we present 0-1SDP formulation for the modularity density maximization and show the equivalence between both problems. In Section 5, we propose a method to solve a doubly nonnegative relaxation problem of 0-1SDP, and in Section 6 we explain a heuristic algorithm to make a feasible solution from the solution of the relaxation problem. We report the computational experiments in Section 7, and then give some conclusions and further research topics in Section 8.

2. Definitions and Notation

Let $G = (V, E)$ be an undirected graph with the set V of n nodes and the set E of m edges. We assume that V has at least two nodes. We say that $\Pi = \{ C_1, C_2, \ldots, C_k \}$ is a *partition* of V if $V = \cup_{p=1}^{k} C_p$, $C_p \cap C_q = \emptyset$ for any distinct pair p and q, and $C_p \neq \emptyset$ for any p. Each member C_p of a partition is called a *community*. We denote the set of edges that have one end-node in C and the other end-node in C' by $E(C, C')$. When $C = C'$, we abbreviate $E(C, C')$ to $E(C)$. Then *modularity density*, denoted by $D(\Pi)$, for a partition Π is defined as

$$D(\Pi) = \sum_{C \in \Pi} \left(\frac{2|E(C)| - \sum_{C' \in \Pi \setminus \{C\}} |E(C, C')|}{|C|} \right),$$

where $|\cdot|$ denotes the cardinality of the corresponding set. We refer to each term of the above summation as the *contribution* of community C to the modularity density.

Modularity density maximization problem, MD for short, is to find a partition Π of V that maximizes the modularity density $D(\Pi)$, that is

$$\text{MD} \quad \left| \begin{array}{ll} \text{maximize} & \displaystyle\sum_{C \in \Pi} \left(\frac{2|E(C)| - \sum_{C' \in \Pi \setminus \{C\}} |E(C, C')|}{|C|} \right) \\ \text{subject to} & \Pi \text{ is a partition of } V. \end{array} \right.$$

A nonlinear binary programming formulation for MD has been proposed in Li *et al.* [8]. Although the optimal number of communities is a priori unknown, we suppose it is known for the time being, and denote it by t, and let T be the index set $\{ 1, 2, \ldots, t \}$. Introducing a binary variable x_{ip} indicating whether node i belongs to community C_p, we have the following nonlinear binary programming formulation:

$$\text{NLP} \quad \left| \begin{array}{lll} \text{maximize} & \displaystyle\sum_{p \in T} \left(\frac{2 \sum_{i \in V} \sum_{j \in V} a_{ij} x_{ip} x_{jp} - \sum_{i \in V} d_i x_{ip}}{\sum_{i \in V} x_{ip}} \right) & \\ \text{subject to} & \displaystyle\sum_{p \in T} x_{ip} = 1 & (i \in V), \\ & \displaystyle\sum_{i \in V} x_{ip} \geq 1 & (p \in T), \\ & x_{ip} \in \{0, 1\} & (i \in V, p \in T), \end{array} \right.$$

where a_{ij} is the (i, j) element of the adjacency matrix A of graph G, and d_i is the degree of node i. The first set of constraints states that each node should belong to exactly one community, and the second set

of constraints imposes that each community should be a nonempty subset of V. The objective function is the sum of fractional functions with a quadratic numerator and a linear denominator. Among widely used solution approaches for problems of this kind are a parametric algorithm by Dinkelbach [11] and a branch-and-bound algorithm [12, 13] in global optimization area.

3. Some Properties of Modularity Density

Now suppose that there exist several isolated nodes in a graph G. After removing them from G, we find a partition $\Pi^\star$ that maximizes the modularity density on the reduced graph of G. If the contribution of every community in $\Pi^\star$ is non-negative, then $\Pi^\star \cup \{\bar{C}\}$ is an optimal partition on the original graph G, where $\bar{C}$ consists of the isolated nodes once deleted. If there exist some communities with a negative contribution, then the contribution increases by adding the isolated nodes to these communities since the denominator of the contribution increases. Therefore we have the following lemma.

Lemma 3.1 (Costa [9], Lemma 1.)**.** *The isolated nodes can be assigned to communities a posteriori.*

Due to Lemma 3.1, we have only to consider graphs with no isolated nodes. Concerning the size of community we have the following proposition.

Proposition 3.2 (Costa [9], Proposition 1, Corollary 1, and Corollary 2.)**.** *Let $\Pi^\star$ be a partition with maximum modularity density, then the size of each community is between 2 and $n - 2(|\Pi^\star| - 1)$.*

4. Formulations

In this section, we first present a reformulation of the modularity density maximization based on MILP formulation according to Costa [9]. Next, we show that modularity density maximization can be equivalently formulated as 0-1SDP, a variant of semidefinite programming.

Hereafter we use following notations:

$$\begin{aligned}
\mathcal{S}_n &= \{\, Y \in \mathbb{R}^{n\times n} \mid Y = Y^\top \,\},\\
\mathcal{S}_n^+ &= \{\, Y \in \mathcal{S}_n \mid \boldsymbol{u}^\top Y \boldsymbol{u} \geq 0 \text{ for all } \boldsymbol{u} \in \mathbb{R}^n \,\},\\
\mathcal{N}_n &= \{\, Y = (y_{ij}) \in \mathcal{S}_n \mid y_{ij} \geq 0 \text{ for all } i, j \in \{1, 2, \ldots, n\} \,\},
\end{aligned}$$

the set of $n \times n$ symmetric matrices, the positive semidefinite cone, and the symmetric nonnegative cone, respectively. For a given vector $\boldsymbol{u}$, $\mathrm{Diag}(\boldsymbol{u})$ is the diagonal matrix with u_i as the i-th diagonal element, and $\mathrm{vec}(U)$ is the vector obtained by stacking columns of a given matrix U.

4.1. MILP formulation

We can rewrite the objective function in the problem MD as follows:

$$\sum_{p\in T} \left(\frac{4\sum_{\{i,j\}\in E} x_{ip}x_{jp} - \sum_{i\in V} d_i x_{ip}}{\sum_{i\in V} x_{ip}} \right),$$

due to the definition of adjacency matrix $A = (a_{ij})_{ij\in V}$. The quadratic term $x_{ip}x_{jp}$ can be linearized by replacing it with a new variable y_{ijp} and adding the following Fortet inequalities [14]:

$$y_{ijp} \leq x_{ip}, \quad y_{ijp} \leq x_{jp}, \quad x_{ip} + x_{jp} \leq y_{ijp} + 1 \quad \text{for } p \in T. \tag{1}$$

Note that the last inequality in (1) is redundant owing to the objective function of maximizing with respect to the variable y_{ijp}, hence can be omitted. Next, we introduce a continuous variable α_p defined as

$$\alpha_p = \frac{4\sum_{\{i,j\}\in E} y_{ijp} - \sum_{i\in V} d_i x_{ip}}{\sum_{i\in V} x_{ip}}.$$

Since the objective function will be replaced by $\sum_{p\in T}\alpha_p$ and is to be maximized, this equality constraint can be relaxed to the inequality constraints

$$\alpha_p \le \frac{4\sum_{\{i,j\}\in E} y_{ijp} - \sum_{i\in V} d_i x_{ip}}{\sum_{i\in V} x_{ip}}. \tag{2}$$

Due to the positivity of the denominator in (2), we can rewrite it as

$$\alpha_p \sum_{i\in V} x_{ip} \le 4 \sum_{\{i,j\}\in E} y_{ijp} - \sum_{i\in V} d_i x_{ip}.$$

Finally, to linearize the product $\alpha_p x_{ip}$, we introduce a continuous variable γ_{ip} to replace $\alpha_p x_{ip}$, and make use of the following McCormick inequalities [15]:

$$\begin{aligned} &L_\alpha x_{ip} \le \gamma_{ip} \le U_\alpha x_{ip}, \\ &\alpha_p - U_\alpha(1-x_{ip}) \le \gamma_{ip} \le \alpha_p - L_\alpha(1-x_{ip}), \end{aligned}$$

where L_α and U_α are lower and upper bounds of α_p, respectively. From the above discussion, MILP formulation is given as

$$\text{MILP} \left| \begin{array}{lll} \text{maximize} & \displaystyle\sum_{p\in T} \alpha_p & \\ \text{subject to} & \displaystyle\sum_{p\in T} x_{ip} = 1 & (i \in V), \\ & 2 \le \displaystyle\sum_{i\in V} x_{ip} \le n - 2\,(t-1) & (p \in T), \\ & y_{ijp} \le x_{ip}, \quad y_{ijp} \le x_{jp} & (\{i,j\} \in E,\, p \in T), \\ & \displaystyle\sum_{i\in V} \gamma_{ip} \le 4 \sum_{\{i,j\}\in E} y_{ijp} - \sum_{i\in V} d_i x_{ip} & (p \in T), \\ & L_\alpha x_{ip} \le \gamma_{ip} \le U_\alpha x_{ip} & (i \in V,\, p \in T), \\ & \alpha_p - U_\alpha(1-x_{ip}) \le \gamma_{ip} \le \alpha_p - L_\alpha(1-x_{ip}) & (i \in V,\, p \in T), \\ & x_{ip} \in \{0,1\} & (i \in V,\, p \in T), \\ & y_{ijp} \in \mathbb{R} & (\{i,j\} \in E,\, p \in T), \\ & L_\alpha \le \alpha_p \le U_\alpha & (p \in T), \\ & \gamma_{ip} \in \mathbb{R} & (i \in V,\, p \in T). \end{array} \right.$$

From the inequality (2) and Proposition 3.2, a possible value of L_α is attained when the corresponding community consists of only two nodes with the largest degree, thus L_α is given as $L_\alpha = -(d_{\max_1} + d_{\max_2})/2$ where $d_{\max_1}$ and $d_{\max_2}$ are the largest and the second largest degrees, respectively. On the other hand, in order to obtain an upper bound on α_p, we need to solve the following auxiliary problem:

$$\text{AP} \left| \begin{array}{ll} \text{maximize} & \dfrac{4\sum_{\{i,j\}\in E} x_i x_j - \sum_{i\in V} d_i x_i}{\sum_{i\in V} x_i} \\ \text{subject to} & 2 \le \displaystyle\sum_{i\in V} x_i \le n - 2(t-1), \\ & x_i \in \{0,1\} \quad (i \in V). \end{array} \right.$$

The problem AP is a problem of maximizing a nonlinear objective function with binary variables, thus difficult to optimize globally. Using a nonlinear programming solver SCIP [16], Costa solved the continuous relaxation problem of AP. In our experiment which aims to compare the solutions obtained by MILP formulation and 0-1SDP formulation, we solve the problem AP to optimality by Dinkelbach's parametric algorithm to avoid the effect of inaccuracy in U_α.

4.2. 0-1SDP formulation

Now we consider the following problem:

$$\text{0-1SDP} \quad \left| \begin{array}{ll} \text{maximize} & \mathrm{Tr}((2A - D)Z) \\ \text{subject to} & Z\boldsymbol{e}_n = \boldsymbol{e}_n, \\ & Z^2 = Z, \\ & Z \in \mathcal{N}_n, \end{array} \right.$$

where D is a diagonal matrix whose i-th diagonal entry is the degree d_i of node i, i.e., $D = \mathrm{Diag}(d_1, d_2, \ldots, d_n) \in \mathbb{R}^{n\times n}$, and $\boldsymbol{e}_n$ is the n-dimensional vector of 1's. We call the problem 0-1SDP because Peng and Xia stated "we call it 0-1SDP owing to the similarity of the constraint $Z^2 = Z$ to the classical 0-1 requirement in integer programming" in [17].

Henceforth we will show the equivalence between the problems MD and 0-1SDP.

Lemma 4.1. *For any feasible solution $Z = (z_{ij})$ of 0-1SDP, there is a matrix X satisfying*

$$X\boldsymbol{e}_\ell = \boldsymbol{e}_n, \tag{3}$$

$$X^\top \boldsymbol{e}_n \geq \boldsymbol{e}_\ell, \tag{4}$$

$$Z = X(X^\top X)^{-1}X^\top, \tag{5}$$

$$X \in \{0,1\}^{n\times \ell}, \tag{6}$$

for some positive integer ℓ.

Proof. Let Z be a feasible solution of 0-1SDP. It is clearly positive semi-definite due to the idempotence constraint $Z^2 = Z$. Then $\max\{\, z_{ij} \mid i, j \in V \,\}$ is attained at a diagonal element, say $z_{i_1 i_1}$, which is positive owing to the non-negativity of Z and the constraint $Z\boldsymbol{e}_n = \boldsymbol{e}_n$. Let us define the index set $\mathcal{I}_1 = \{\, j \in V \mid z_{i_1 j} > 0 \,\}$, then we readily obtain the following equalities

$$\sum_{j\in\mathcal{I}_1} (z_{i_1 j})^2 = z_{i_1 i_1} \text{ and } \sum_{j\in\mathcal{I}_1} z_{i_1 j} = 1,$$

due to the constraints $Z^2 = Z$, $Z\boldsymbol{e}_n = \boldsymbol{e}_n$ and the symmetry of Z. Since $z_{i_1 i_1}$ is positive, the first equality reduces to

$$\sum_{j\in\mathcal{I}_1} \left(\frac{z_{i_1 j}}{z_{i_1 i_1}}\right) z_{i_1 j} = 1.$$

By using the second equality, this yields

$$\sum_{j\in\mathcal{I}_1} \left(\frac{z_{i_1 j}}{z_{i_1 i_1}}\right) z_{i_1 j} = \sum_{j\in\mathcal{I}_1} z_{i_1 j},$$

which is rewritten as

$$\sum_{j\in\mathcal{I}_1} \left(\frac{z_{i_1 j}}{z_{i_1 i_1}} - 1\right) z_{i_1 j} = 0.$$

From the non-negativity of z_{ij} and the maximality of $z_{i_1 i_1}$, we have $(z_{i_1 j}/z_{i_1 i_1} - 1) = 0$, which implies $z_{i_1 j} = z_{i_1 i_1}$ for any $j \in \mathcal{I}_1$. Then we have

$$z_{i_1 i_1} = z_{i_1 j} = \frac{1}{|\mathcal{I}_1|} \quad \text{for any } j \in \mathcal{I}_1. \tag{7}$$

For an index $j \in \mathcal{I}_1$, we denote the (i_1, j) element of the matrix Z^2 by $Z^2_{i_1 j}$, then it is given

$$Z^2_{i_1 j} = \sum_{k\in V} z_{i_1 k} z_{kj} = \sum_{k\in\mathcal{I}_1} z_{i_1 k} z_{kj} = z_{i_1 i_1} \left(\sum_{k\in\mathcal{I}_1} z_{kj}\right).$$

Note that the last equality is due to (7). The above equality, $Z^2 = Z$ and (7) yield

$$z_{i_1 i_1} = z_{i_1 i_1} \left(\sum_{k \in \mathcal{I}_1} z_{kj} \right),$$

which is reduced to

$$1 = \sum_{k \in \mathcal{I}_1} z_{kj}.$$

This implies that $z_{jk} = 1/|\mathcal{I}_1|$ for any $j, k \in \mathcal{I}_1$ owing to the maximality of $z_{i_1 i_1}$ and the constraint $Z\boldsymbol{e}_n = \boldsymbol{e}_n$. Denote the sub-matrix $(z_{ij})_{i,j \in \mathcal{I}_1}$ by $Z_{\mathcal{I}_1}$. By permuting the rows and columns of Z simultaneously, we obtain

$$P_1^\top Z P_1 = \begin{bmatrix} Z_{\mathcal{I}_1} & O \\ O & Z_{\bar{\mathcal{I}}_1} \end{bmatrix}$$

where P_1 is an appropriate permutation matrix, and $\bar{\mathcal{I}}_1 = V \setminus \mathcal{I}_1$. Then it is clear that the sub-matrix $Z_{\bar{\mathcal{I}}_1}$ satisfies the following:

$$Z_{\bar{\mathcal{I}}_1} \boldsymbol{e}_{|\bar{\mathcal{I}}_1|} = \boldsymbol{e}_{|\bar{\mathcal{I}}_1|},\ Z_{\bar{\mathcal{I}}_1}^2 = Z_{\bar{\mathcal{I}}_1} \text{ and } Z_{\bar{\mathcal{I}}_1} \in \mathcal{N}_{|\bar{\mathcal{I}}_1|}.$$

Thus repeating the process described above, we can convert Z to a block diagonal matrix as follows:

$$(P_{\ell-1}^\top \cdots P_2^\top P_1^\top) Z (P_1 P_2 \cdots P_{\ell-1}) = \begin{bmatrix} Z_{\mathcal{I}_1} & & & \\ & Z_{\mathcal{I}_2} & & \\ & & \ddots & \\ & & & Z_{\mathcal{I}_\ell} \end{bmatrix},$$

where each block diagonal element $Z_{\mathcal{I}_p}$ is the $|\mathcal{I}_p| \times |\mathcal{I}_p|$ matrix whose elements are all $1/|\mathcal{I}_p|$. Now, we construct a matrix $X = (x_{ip})$ such that

$$x_{ip} = \begin{cases} 1 & (\text{if } i \in \mathcal{I}_p), \\ 0 & (\text{otherwise}), \end{cases}$$

then X clearly satisfies (3), (4) and (6), and we can confirm $Z = X(X^\top X)^{-1} X^\top$ by a simple calculation. Note that the inverse of $X^\top X$ exists since the matrix $X^\top X$ is a nonsingular diagonal matrix whose diagonal entry is the number of 1's of each column of X by (3), (4) and (6). □

We say that the matrix $X = (x_{ip})$ which satisfies the conditions (3), (4) and (6) is an *incidence matrix* of a partition $\Pi = \{ C_1, C_2, \ldots, C_k \}$, that is, each member C_p of Π is represented as $C_p = \{ i \in V \mid x_{ip} = 1 \}$.

Theorem 4.2. *The problem 0-1SDP is equivalent to the problem MD.*

Proof. Let Z be an optimal solution of 0-1SDP, then we obtain an incidence matrix X which satisfies $Z = X(X^\top X)^{-1} X^\top$ due to Lemma 4.1 and represents a partition Π. Since the objective function in the problem MD is written as $\mathrm{Tr}((2A - D)Z)$ by means of the matrix which satisfies (5), we have the following inequality

$$\begin{aligned} \mathrm{Tr}((2A - D)Z) &= \mathrm{Tr}((2A - D)X(X^\top X)^{-1} X^\top) \\ &= \sum_{C \in \Pi} \left(\frac{2|E(C)| - \sum_{C' \in \Pi \setminus \{C\}} |E(C, C')|}{|C|} \right) \\ &\leq \omega(\mathrm{MD}), \end{aligned} \tag{8}$$

where $\omega(\mathrm{MD})$ denotes the optimal value of problem MD.

Next, we show the optimality of the partition Π obtained from Z for the problem MD. For an optimal solution $\hat{\Pi}$ of MD, let $\hat{X}$ be an incidence matrix corresponding to $\hat{\Pi}$, and let $\hat{Z} = \hat{X}(\hat{X}^\top \hat{X})^{-1}\hat{X}^\top$. Clearly $\hat{Z}$ satisfies

$$\hat{Z}\boldsymbol{e}_n = \hat{Z}\hat{X}\boldsymbol{e}_t = \hat{X}\boldsymbol{e}_t = \boldsymbol{e}_n$$

from (3) and (5). Moreover, it is symmetric, nonnegative and idempotent due to (5), hence $\hat{Z}$ is feasible to 0-1SDP. Then we have

$$\begin{aligned}
\omega(\mathrm{MD}) &= \sum_{C \in \hat{\Pi}} \left(\frac{2|E(C)| - \sum_{C' \in \hat{\Pi} \setminus \{C\}} |E(C, C')|}{|C|} \right) \\
&= \mathrm{Tr}((2A - D)\hat{X}(\hat{X}^\top \hat{X})^{-1}\hat{X}^\top) \\
&= \mathrm{Tr}((2A - D)\hat{Z}) \\
&\leq \mathrm{Tr}((2A - D)Z) = \omega(\text{0-1SDP}). \qquad (9)
\end{aligned}$$

The last inequality is due to the optimality of Z for the problem 0-1SDP. By the inequalities (8) and (9), we conclude that $\omega(\mathrm{MD}) = \omega(\text{0-1SDP})$, which implies the optimality of the partition Π obtained from Z for the problem MD. □

Note that the size of 0-1SDP depends on neither the number of edges nor the number of communities. Moreover, we need not solve the auxiliary problem unlike MILP formulation. These features make 0-1SDP more attractive than MILP formulation. Although the objective function in 0-1SDP is linear with respect to the matrix Z, the idempotence constraint makes the problem difficult. We will discuss how to deal with this difficult part in the next section.

5. Doubly Nonnegative Relaxation

As stated in Subsection 4.2, what makes 0-1SDP difficult to solve is the idempotence constraint. It requires every eigenvalue λ_i of Z to be either 0 or 1. Relaxing it to $0 \leq \lambda_i \leq 1$ would be the first choice to consider. This constraint is expressed as $Z \in \mathcal{S}_n^+$ and $I - Z \in \mathcal{S}_n^+$, where I is the identity matrix. The latter constraint $I - Z \in \mathcal{S}_n^+$ which represents the upper bound constraint of λ_i is redundant owing to the presence of $Z \in \mathcal{N}_n$ and $Z\boldsymbol{e}_n = \boldsymbol{e}_n$, hence can be omitted. Then we obtain the following relaxation problem over the doubly nonnegative cone $\mathcal{N}_n \cap \mathcal{S}_n^+$:

$$\mathrm{DNN} \quad \left| \begin{array}{ll} \text{maximize} & \mathrm{Tr}((2A - D)Z) \\ \text{subject to} & Z\boldsymbol{e}_n = \boldsymbol{e}_n, \\ & Z \in \mathcal{N}_n \cap \mathcal{S}_n^+. \end{array} \right.$$

The optimization problems over a symmetric cone e.g., linear programming, second-order cone programming, and semidefinite programming problems, are now solved efficiently. Indeed, the primal-dual interior point method solves the problems in polynomial time. On the other hand, due to the asymmetry of doubly nonnegative cone we cannot directly apply the primal-dual interior point method to solve the problem DNN. However representing the doubly nonnegative cone as a symmetric cone embedded in a higher dimensional space, we could apply the primal-dual interior point method to the embedded problem:

$$\left| \begin{array}{ll} \text{maximize} & \mathrm{Tr}((2A - D)Z) \\ \text{subject to} & Z\boldsymbol{e}_n = \boldsymbol{e}_n, \\ & \begin{bmatrix} Z & O \\ O & \mathrm{Diag}(\mathrm{vec}(Z)) \end{bmatrix} \in \mathcal{S}_{n+n^2}^+. \end{array} \right.$$

Although the above problem can be solved in polynomial time in theory, it is a quite large problem over the positive semidefinite cone of high dimension, hence computationally very expensive to solve in practice. Nevertheless it should be worthwhile to solve the problem DNN because the doubly nonnegative relaxation is often observed to provide significantly tight bound for several combinatorial optimization problems.

Now, we introduce valid inequalities for 0-1SDP in order to strengthen the bound obtained by the relaxation problem. From the proof of Lemma 4.1, any feasible solution Z of 0-1SDP can be transformed to a block diagonal matrix. It is easy to see that the maximum value in each row of Z is located on the diagonal element, hence we have the following valid inequality.

Lemma 5.1. *The inequalities*

$$z_{ii} \geq z_{ij} \quad \textit{for } i,\ j \in V$$

are valid for 0-1SDP.

We denote the problem DNN with the above valid inequalities added by $\overline{\text{DNN}}$.

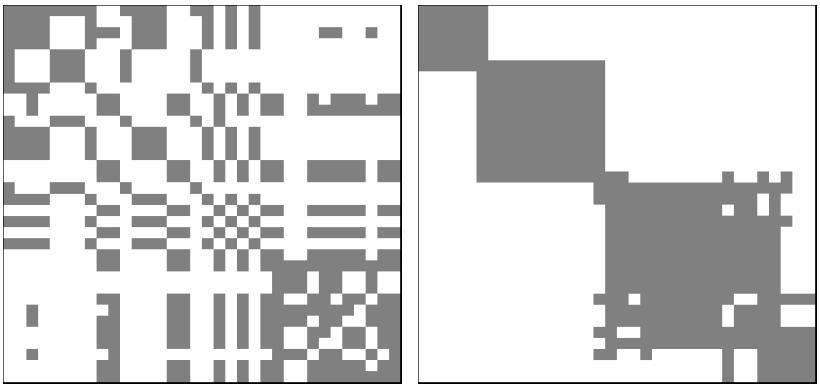
6. Heuristics based on Dynamic Programming

In this section, we will develop an algorithm to construct a feasible solution for MD from the solution obtained by solving either DNN or $\overline{\text{DNN}}$ with the valid inequalities presented in Lemma 5.1. The algorithm is based on the combination of spectral heuristics and dynamic programming.

6.1. Permutation based on spectrum

As we have seen in the proof of Lemma 4.1, an optimal solution of 0-1SDP is a block diagonal matrix each of whose blocks corresponds to a community. An optimal solution of the problem DNN or $\overline{\text{DNN}}$ is not necessarily transformed to a block diagonal matrix by any permutation, however the solution may provide a useful clue as to possibly a good solution of the original problem MD. Thus, it would be helpful to try to diagonalize the solution matrix. To this end, we exploit the spectrum of the optimal solution. Our spectral approach is inspired by the method by Fiedler [18], which provides a bipartition of the node set according to the Fiedler vector, the eigenvector corresponding to the second smallest eigenvalue of Laplacian of the graph.

Now let $\boldsymbol{u} = (u_1, u_2, \ldots, u_n)^\top \in \mathbb{R}^n$ be the eigenvector corresponding to the second largest eigenvalue of an optimal solution Z^* for the relaxation problem. We rearrange its components in the non-decreasing order $u_{\sigma(1)} \leq u_{\sigma(2)} \leq \cdots \leq u_{\sigma(n)}$, and permute simultaneously the rows and columns of the matrix Z^* consistent with this order. Take a benchmark instance Karate for example, Figures 1a and 1b show an optimal solution Z^* of DNN and the matrix obtained in the manner described above. They show that the solution matrix inherits a block diagonal structure.



(a) Z^* before permutation (b) Z^* after permutation

Figure 1: Emergence of block diagonal structure. The color-code in the figures indicates the magnitude of elements: elements whose value is greater than or equal to $1/n$ are marked with gray, while other elements (with value of less than $1/n$) are marked with white.

6.2. Dynamic programming

Next, we discuss how to construct a feasible solution of MD from the permuted matrix. Let $\bar{V}$ be a sequence of nodes consistent with the non-decreasing order of components of the eigenvector $\boldsymbol{u}$. For the sake of notational simplicity, we renumber the nodes of V so that $\bar{V} = (1, 2, \ldots, n)$. Now, we try to find a partition with maximum modularity density of $\bar{V}$ under the constraint that each member of the partition consists of consecutive indices of $\bar{V}$. We propose an algorithm using the dynamic programming for this problem. Let $q(k, l)$ be defined by

$$q(k,l) = \frac{2\sum_{i=k}^{l}\sum_{j=k}^{l} a_{ij} - \sum_{i=k}^{l} d_i}{l-(k-1)}$$

for k and l of $\bar{V}$ with $k \leq l$. The value $q(k, l)$ represents the contribution of the community $(k, k+1, \ldots, l)$ of $\bar{V}$ to the modularity density when it is selected as a member of the partition. For each index s of $\bar{V}$, let $\mu(s)$ be the maximum modularity density that is achieved by partitioning the sequence $(1, \ldots, s)$ into several subsequences of consecutive indices. If we define $\mu(0) = 0$ for notational convenience, then $\mu(s)$ satisfies the recursive equation

$$\mu(s) = \max\{\, \mu(h) + q(h+1, s) \mid h \in \{0, 1, \ldots, s-1\}\,\}. \tag{10}$$

Owing to (10), starting from $\mu(1) = q(1, 1)$, we first obtain $\mu(2) = \max\{\, q(1, 2),\ \mu(1) + q(2, 2)\,\}$, then obtain $\mu(3)$ by means of $\mu(1)$ and $\mu(2)$, and so on. Our algorithm DP is given as follows.

Algorithm DP

Step 1 : Solve the relaxation problem to obtain an optimal solution Z^*.

Step 2 : Find the eigenvector $\boldsymbol{u}$ corresponding to the second largest eigenvalue of Z^*.
Let $\bar{V} = (1, 2, \ldots, n)$ be a sequence of nodes obtained by rearranging them in non-decreasing order of corresponding components in $\boldsymbol{u}$.

Step 3 : Set $\mu(0) := 0$.
`for` $s = 1$ `to` n `do`
  Compute $\mu(s)$ according to (10).
`end-do`

7. Computational Experiments

To evaluate the lower and upper bounds provided by our algorithm, we conducted computational experiments on a computer with Intel Core i7, 3.40 GHz processor and 8.0 GB of memory. The algorithm is implemented in Python 3.6.1, calling the Python API of MOSEK 8.0 as SDP solver. We used 18 instances available on the Pajek repository [19]. Details of each instance are given in the first six columns in Table 1, where the columns "OPT" and "$t^\star$" represent the optimal value and the optimal number of communities, respectively. These optimal values are reported in Costa [9] and Sato and Izunaga [20]. The other columns in Table 1 show the computational results of the algorithm described in Subsection 6.2, where the columns "UB", "LB", "gap" and "time" represent the optimal value of the relaxation problem, the lower bound obtained by the algorithm DP, the duality gap defined by $100(\mathrm{UB} - \mathrm{LB})/\mathrm{LB}$, and the computation time in seconds, respectively. Note that the largest lower bounds for each instance are bold-faced. We observed for each instance that the predominant portion of the computation time was spent for solving the relaxation problem, and the remaining parts of the algorithm require a fraction of time, specifically less than one second.

Table 1 shows that $\overline{\mathrm{DNN}}$ provides a tighter upper bound than DNN does for all instances, which indicates the effectiveness of the valid inequalities we proposed in Lemma 5.1. Moreover, we also see that the lower

Table 1: Computational results of our algorithm. The symbol "NA" means that the optimal solution is unknown.

Instance						DNN				$\overline{\text{DNN}}$			
ID	name	n	m	OPT	t^*	UB	LB	gap(%)	time(sec.)	UB	LB	gap(%)	time(sec.)
1	Strike	24	38	8.8611	4	9.5808	**8.8611**	8.122	0.43	9.3049	**8.8611**	5.008	0.61
2	Galesburg F	31	63	8.2857	3	9.4170	**8.2857**	13.653	0.35	9.0781	**8.2857**	9.563	1.34
3	Galesburg D	31	67	6.9269	3	8.3980	**6.8269**	23.013	0.34	8.1511	**6.8269**	19.396	1.60
4	Karate	34	78	7.8451	3	8.9548	7.8424	14.184	0.46	8.4822	**7.8451**	8.121	2.02
5	Korea 1	35	69	10.9667	5	12.3216	10.0667	22.400	0.46	11.9196	**10.9667**	8.690	2.07
6	Korea 2	35	84	11.1430	5	12.5689	**11.1430**	12.797	0.41	12.1114	**11.1430**	8.691	2.35
7	Mexico	35	117	8.7180	3	10.3151	**8.5580**	20.532	0.45	9.9570	8.5227	16.829	2.17
8	Sawmill	36	62	8.6233	4	10.5048	7.0486	49.034	0.46	10.0718	**7.3587**	36.869	2.38
9	Dolphins small	40	70	13.0519	8	15.2480	**12.5520**	21.479	0.64	14.7963	10.5208	40.638	3.59
10	Journal index	40	189	17.8000	4	20.1446	17.5500	14.784	0.61	19.3510	**17.6929**	9.371	3.70
11	Graph	60	114	9.7524	7	12.5626	6.9168	81.626	2.50	12.1532	**7.6716**	58.419	20.61
12	Dolphins	62	159	12.1252	5	15.0218	9.8286	52.838	2.90	14.3559	**11.4610**	25.258	22.13
13	Les Misérables	77	254	24.5474	8	28.0957	22.2680	26.171	5.83	27.4276	**23.3416**	17.505	71.95
14	A00 main	83	125	13.4825	12	16.7035	**11.9534**	39.738	8.70	16.1569	10.4254	54.977	104.07
15	Protein p53	104	226	13.2143	9	17.4832	10.1545	72.173	23.10	16.3994	**10.7040**	53.209	326.97
16	Books	105	441	21.9652	7	26.5387	20.2470	31.075	21.19	24.7788	**20.3150**	21.973	357.51
17	Adjnoun	112	425	NA	NA	10.2341	**7.6514**	33.755	29.34	9.6807	**7.6514**	26.523	527.95
18	Football	115	613	NA	NA	46.5359	35.6921	30.382	30.40	45.9165	**43.2564**	6.150	672.10

bounds obtained for $\overline{\text{DNN}}$ are equal to or larger than those obtained for DNN for 15 instances out of whole of instances. Particularly, the result for the instance Football (ID=18), which has not been solved to optimality so far, is remarkable; our algorithm gives a solution with a quite small duality gap in around ten minutes.

Next, we solved the problem MILP by the branch-and-bound algorithm in Gurobi Optimizer 7.5.1 to compare the quality of the solutions obtained by our algorithm for 0-1SDP formulation. We set the number of communities required as input of MILP to the optimal number t^*. Note here that, for each instance whose optimal solution is unknown, we set it to the number of communities corresponding to the best-known heuristic solution, which is listed in parentheses in the column "$t^\star$" of Table 2. In the experiment, we impose the time limit of 3,600 seconds, i.e., one hour, on the computation time of the branch-and-bound algorithm. The computational results are given in Table 2, which shows that the branch-and-bound

Table 2: Computational results of the branch-and-bound algorithm for MILP formulation. The symbol "NA" means that the optimal solution is unknown.

Instance						MILP			
ID	name	n	m	OPT	t^*	UB	LB	gap(%)	time(sec.)
1	Strike	24	38	8.8611	4	8.8611	8.8611	0	0.42
2	Galesburg F	31	63	8.2857	3	8.2857	8.2857	0	0.37
3	Galesburg D	31	67	6.9269	3	6.9269	6.9269	0	1.03
4	Karate	34	78	7.8451	3	7.8451	7.8451	0	0.39
5	Korea 1	35	69	10.9667	5	10.9667	10.9667	0	30.29
6	Korea 2	35	84	11.1430	5	11.1430	11.1430	0	130.43
7	Mexico	35	117	8.7180	3	8.7180	8.7180	0	8.10
8	Sawmill	36	62	8.6233	4	8.6233	8.6233	0	4.54
9	Dolphins small	40	70	13.0519	8	13.0519	13.0519	0	2187.18
10	Journal index	40	189	17.8000	4	17.8000	17.8000	0	565.65
11	Graph	60	114	9.7524	7	19.9640	9.7523	104.70	3600.00
12	Dolphins	62	159	12.1252	5	19.5742	12.1252	61.43	3600.00
13	Les Misérables	77	254	24.5474	8	55.8069	23.9430	133.08	3600.00
14	A00 main	83	125	13.4825	12	28.0248	13.1982	112.33	3600.00
15	Protein p53	104	226	13.2143	9	41.6739	12.4543	234.61	3600.00
16	Books	105	441	21.9652	7	56.4203	20.9333	169.52	3600.00
17	Adjnoun	112	425	NA	(2)	7.8250	7.8250	0	49.63
18	Football	115	613	NA	(10)	97.8350	22.6928	331.12	3600.00

algorithm solves the instances up to 40 nodes to optimality, while it run out the time limit with a quite large duality gap left behind except the instance Adjnoun (ID=17). We sepculate that the short computation for the instance Adjnoun (ID=17) is due to the small size of the number of communities.

As example of the behavior of the branch-and-bound algorithm in Gurobi Optimizer, we give Figure 2

which shows the upper and lower bounds vs. the elapsed time for the instance Dolphins (ID=12).

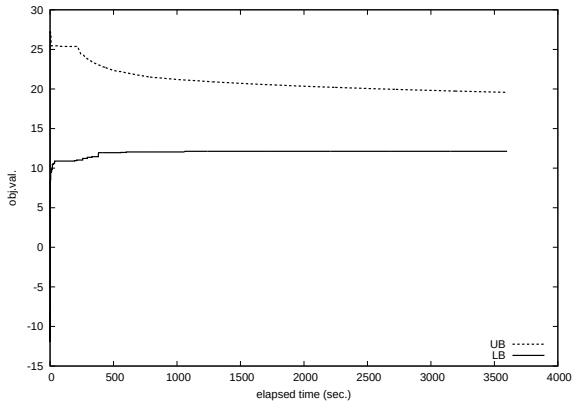


Figure 2: The upper and lower bounds vs. elapsed time in the branch-and-bound.

From the figure, we observe that (i) the branch-and-bound algorithm gives a good lower bound in a fairly early stage, and (ii) the improvement of upper bound rarely occurs throughout the computation. Owing to the latter, the duality gap won't shrink even after a good feasible solution has been found. This disrupts the early termination of the algorithm. Therefore the branch-and-bound algorithm should be significantly improved if combined with a better method for a tight upper bound such as DNN and $\overline{\text{DNN}}$.

8. Conclusion

In this paper, we presented 0-1SDP formulation which was originally introduced by Peng and Xia [17] for minimum sum-of-squares clustering problem, and showed that the equivalence between the problem 0-1SDP and the modularity density maximization. The problem 0-1SDP has the big advantage that its size is not dependent on the number of edges or the number of communities. Then, we proposed to solve a doubly nonnegative relaxation of the problem 0-1SDP in order to obtain an upper bound on the optimal modularity density. In addition, we developed a lower bounding algorithm based on a combination of spectral heuristics and dynamic programming.

Our formulation was numerically compared to MILP formulation, and the results showed that the upper bounds provided by our formulation are much tighter than those provided by MILP formulation.

We observed that the solution matrix showed a block-diagonal-like structure when its rows and columns are rearranged simultaneously in accordance with the magnitude of the components of the eigenvector corresponding to the second largest eigenvalue. Theoretical study should be carried out about whether this procedure functions effectively, and why if it does. The alternative to form a block-diagonal-like matrix is to develop a heuristics based on the numerical linear algebraic computation such as the algorithms of Sargent and Westerberg [21], and Tarjan [22].

Acknowledgment

The authors thank Amy N. Langville, College of Charleston for drawing their attention to Fiedler's work [18].

References

[1] M. E. Newman, M. Girvan, Finding and evaluating community structure in networks, Physical Review E 69 (2). doi:10.1103/PhysRevE.69.026113.

[2] M. E. Newman, Fast algorithm for detecting community structure in networks, Physical Review E 69 (6). doi:10.1103/PhysRevE.69.066133.

[3] S. Fortunato, M. Barthélemy, Resolution limit in community detection, Proceedings of the National Academy of Sciences 104 (1) (2007) 36–41. doi:10.1073/pnas.0605965104.

[4] A. Arenas, A. Fernández, S. Gómez, Analysis of the structure of complex networks at different resolution levels, New Journal of Physics 10 (5). doi:10.1088/1367-2630/10/5/053039.

[5] S. Muff, F. Rao, A. Caflisch, Local modularity measure for network clusterizations, Physical Review E 72 (5). doi:10.1103/PhysRevE.72.056107.

[6] C. Granell, S. Gomez, A. Arenas, Hierarchical multiresolution method to overcome the resolution limit in complex networks, International Journal of Bifurcation and Chaos 22 (7). doi:10.1142/S0218127412501714.

[7] V. A. Traag, P. Van Dooren, Y. Nesterov, Narrow scope for resolution-limit-free community detection, Physical Review E 84 (1). doi:10.1103/PhysRevE.84.016114.

[8] Z. Li, S. Zhang, R.-S. Wang, X.-S. Zhang, L. Chen, Quantitative function for community detection, Physical Review E 77 (3). doi:10.1103/PhysRevE.77.036109.

[9] A. Costa, MILP formulations for the modularity density maximization problem, European Journal of Operational Research 245 (1) (2015) 14–21. doi:10.1016/j.ejor.2015.03.012.

[10] A. Costa, S. Kushnarev, L. Liberti, Z. Sun, Divisive heuristic for modularity density maximization, Computers & Operations Research 71 (2016) 100–109. doi:10.1016/j.cor.2016.01.009.

[11] W. Dinkelbach, On nonlinear fractional programming, Management Science 13 (7) (1967) 492–498. doi:10.1287/mnsc.13.7.492.

[12] H. P. Benson, Global optimization algorithm for the nonlinear sum of ratios problem, Journal of Optimization Theory and Applications 112 (1) (2002) 1–29. doi:10.1023/A:1013072027218.

[13] T. Kuno, A branch-and-bound algorithm for maximizing the sum of several linear ratios, Journal of Global Optimization 22 (1-4) (2002) 155–174. doi:10.1023/A:1013807129844.

[14] R. Fortet, Applications de lalgebre de boole en recherche opérationelle, Revue Française de Recherche Opérationelle 4 (14) (1960) 17–26.

[15] G. P. McCormick, Computability of global solutions to factorable nonconvex programs: Part I convex underestimating problems, Mathematical Programming 10 (1) (1976) 147–175. doi:10.1007/BF01580665.

[16] T. Achterberg, SCIP: Solving constraint integer programs, Mathematical Programming Computation 1 (1) (2009) 1–41. doi:10.1007/s12532-008-0001-1.

[17] J. Peng, Y. Xia, A new theoretical framework for k-means-type clustering, in: W. Chu, T. Lin (Eds.), Foundations and Advances in Data Mining, Vol. 180 of Studies in Fuzziness and Soft Computing, Springer, 2005, pp. 79–96. doi:10.1007/11362197_4.

[18] M. Fiedler, A property of eigenvectors of nonnegative symmetric matrices and its applications to graph theory, Czechoslovak Mathematical Journal 25 (4) (1975) 619–633.

[19] V. Batagelj, A. Mrvar, Pajek datasets (2006).
URL http://vlado.fmf.uni-lj.si/pub/networks/data/

[20] K. Sato, Y. Izunaga, An enhanced MILP-based branch-and-price approach to modularity density maximization on graphs, Computers & Operations Researchdoi:10.1016/j.cor.2018.01.012.

[21] R. Sargent, A. Westerberg, Speed-up in chemical engineering design, Transactions of the Institution of Chemical Engineers 42 (1964) 190–197.

[22] R. Tarjan, Depth-first search and linear graph algorithms, SIAM Journal on Computing 1 (2) (1972) 146–160. doi:10.1137/0201010.