# A Doubly Nonnegative Relaxation for Modularity Density Maximization

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

[a] *Information Systems Research Division, The Institute of Behavioral Sciences*
*2-9 Ichigayahonmura-cho, Shinjuku-ku, Tokyo 162-0845, 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 the combination of spectral heuristics and dynamic programming.

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

*Corresponding author
*Email address:* yizunaga@ibs.or.jp (Yoichi Izunaga )

## 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, a novel measure called modularity was proposed by Newman and Girvan [22], and Newman [21] suggested an advantage of maximizing the modularity. Though the modularity maximization is now one of the central subjects of research, Fortunato and Barthélemy [9] 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 may contain communities with different scales. To overcome the resolution limit, there have been extensive studies so far [2, 20, 11, 26], and recently, Li *et al.* [15] 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 [4] 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.* [5] to obtain a good lower bound on the modularity density.

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 pro-

gramming problem with nonnegative constraints, hence the resulting problem can be solved in polynomial time. Moreover, we develop a method based on the combination of spectral heuristics and dynamic programming to construct a feasible solution from the 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 $E(C)$ for the sake of simplicity. Then *modularity density*, denoted by $D(\Pi)$, for a partition $\Pi$ 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 $\Pi$ 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.* [15]. 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:

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

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 belongs to exactly one community, and the second set of constraints imposes that each community should be a nonempty subset of $V$. The objective function in this problem is the sum of fractional functions with a quadratic numerator and a linear denominator. One of the widely used solution approaches for the problem of this kind is a parametric algorithm by Dinkelbach [6]. Another approach is a branch-and-bound algorithm [3, 14] 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 [4], 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 propostion.

**Proposition 3.2** (Costa [4], 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, which is based on MILP formulation according to Costa [4]. 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 Fartet inequalities [8]:

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

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 $\alpha_p$ 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 $\gamma_{ip}$ to replace $\alpha_p x_{ip}$, and make use of the following McCormick inequalities [17]:

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

where $L_\alpha$ and $U_\alpha$ are lower and upper bounds of $\alpha_p$, respectively. From the above discussion, MILP formulation is given as

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

From the inequality (2) and Proposition 3.2, a possible value of $L_\alpha$ is attained when the corresponding community consists of only two nodes with the largest degree, thus $L_\alpha$ 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 $\alpha_p$, we need to solve the following auxiliary problem:

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

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 [1], 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.

*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 [23].

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*

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

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

$$X \in \{0, 1\}^{n \times \ell}, \tag{5}$$

$$Z = X(X^\top X)^{-1} X^\top, \tag{6}$$

*for some positive integer $\ell$.*

*Proof.* Let $Z$ be a feasible solution of 0-1SDP, and 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 consider the $(i_1, j)$ element, denoted by $Z^2_{i_1 j}$, of the matrix $Z^2$, which is given as

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

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 (5), 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 (5). □

We say that the matrix $X = (x_{ip})$ which satisfies the conditions (3), (4) and (5) is an *incidence matrix* of a partition $\Pi = \{ C_1, C_2, \ldots, C_k \}$, that is, each member $C_p$ of $\Pi$ 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 $X$ representing a partition $\Pi$ which satisfies $Z = X(X^\top X)^{-1} X^\top$ due to Lemma 4.1. Since the objective function in the problem MD is written as $\mathrm{Tr}((2A - D)Z)$ by means of the matrix which satisfies (6), 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 $\Pi$ 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 (6). Moreover, it is symmetric, nonnegative and idempotent due to (6), hence $\hat{Z}$ is feasible to 0-1SDP. Then we have

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

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 $\Pi$ 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 $\lambda_i$ 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 $\lambda_i$ 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 $\mathcal{N}_n \cap \mathcal{S}_n^+$:

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

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 $\overline{\text{DNN}}$.

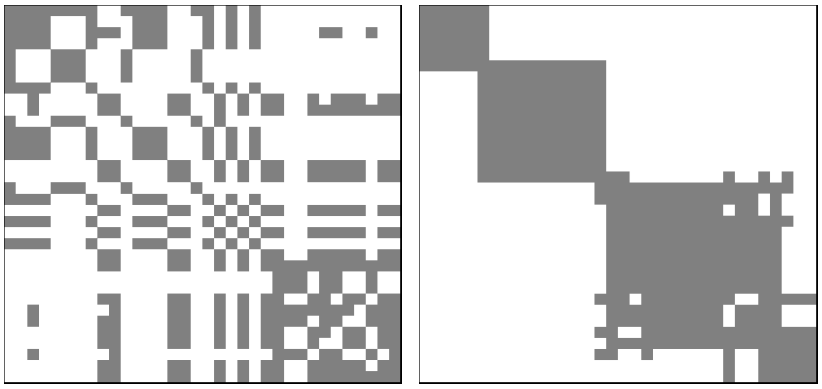
## 6. Heuristics based on Dynamic Programming

In this section, we will develop an algorithm to construct a feasible solution for MD from the solution obtained by solving 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 [7], 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 simultaneously permute the rows and columns of the matrix $Z^*$ consistent with this order. Take a benchmark instance Karate for example, we show an optimal solution $Z^*$ of DNN and the matrix obtained in the manner described above in Figures 1a and 1b. The color-code in the figures indicates



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

Figure 1: Emergence of block diagonal structure.

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. The figures show that the solution matrix inherits a block diagonal structure.

### *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 $\boldsymbol{u}$. 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 obtained by our algorithm, we conducted computational experiments on a computer with Intel Core i7, 3.70 GHz

processor and 32.0 GB of memory. Using SeDuMi 1.2 as an SDP solver, we implemented the algorithm in MATLAB 2010. In the experiments, we used instances from the following seven datasets: Michael's Strike [18], Zachary's Karate [27], Gil-Mendieta and Schmidt's Mexico [10], Michael and Massey's Sawmill [19], Lusseau's Dolphins [16], Hugo's Les Misérables [12], and Krebs' Books [13]. Details of each instance are given in Table 1, where the columns "$\mathrm{LB}^\star$" and "$t^\star$" represent the known best lower bound and the corresponding number of communities, respectively. We list the optimal values and the optimal

Table 1: Instances.

| ID | name | $n$ | $m$ | $\mathrm{LB}^\star$ | $t^\star$ |
|---|---|---|---|---|---|
| 1 | Strike | 24 | 38 | 8.8611 | 4 |
| 2 | Karate | 34 | 78 | 7.8451 | 3 |
| 3 | Mexico | 35 | 117 | 8.7180 | 3 |
| 4 | Sawmill | 36 | 62 | 8.6233 | 4 |
| 5 | Dolphins | 62 | 159 | 12.1252 | 5 |
| 6 | Les Misérables | 77 | 254 | 24.5339 | 9 |
| 7 | Books | 105 | 441 | 21.9652 | 7 |

numbers of communities reported in Costa [4] for the first four instances, and the lower bounds and the number of communities reported in Costa *et al.* [5] for the rest.

Table 2 shows 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. 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 2: Computational results of our algorithm.

| ID | DNN | | | |
|---|---|---|---|---|
| | UB | LB | gap(%) | time(sec.) |
| 1 | 9.5808 | 8.8611 | 8.122 | 1.05 |
| 2 | 8.9548 | 7.8424 | 14.184 | 5.83 |
| 3 | 10.3151 | 8.5580 | 20.532 | 7.64 |
| 4 | 10.5048 | 7.0486 | 49.034 | 7.75 |
| 5 | 15.0218 | 9.8286 | 52.838 | 316.61 |
| 6 | 28.0957 | 22.2680 | 35.279 | 658.28 |
| 7 | 26.5387 | 20.2470 | 31.075 | 4626.11 |
| ID | $\overline{\text{DNN}}$ | | | |
| | UB | LB | gap(%) | time(sec.) |
| 1 | 9.3049 | 8.8611 | 5.008 | 3.54 |
| 2 | 8.4141 | 7.8451 | 7.253 | 36.04 |
| 3 | 9.9570 | 8.5227 | 16.829 | 43.48 |
| 4 | 10.0311 | 7.3587 | 36.316 | 54.21 |
| 5 | 14.3552 | 11.4610 | 25.253 | 1681.81 |
| 6 | 27.4276 | 23.3416 | 17.505 | 7018.03 |
| 7 | 24.7749 | 20.3150 | 21.953 | 60437.45 |

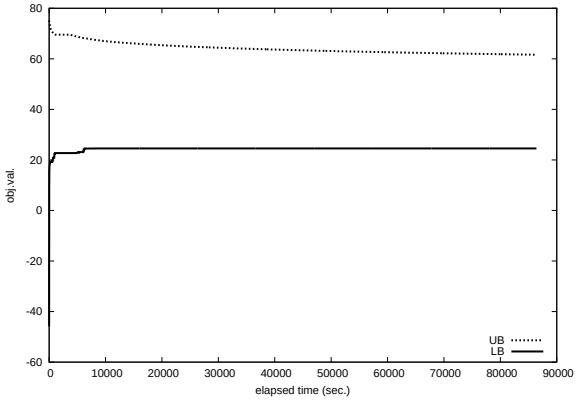
Table 2 shows that $\overline{\text{DNN}}$ provides a tighter upper bound than DNN does for all instances, which indicates the effectiveness of the valid inequalities we proposed in Lemma 5.1. Moreover, we also see that the lower bounds obtained for $\overline{\text{DNN}}$ are equal to or larger than those obtained for DNN for all instances except Mexico (ID=3). On the other hand, solving the problem $\overline{\text{DNN}}$ requires a rather long computation time compared with solving DNN as the instance size grows. To be specific, for the instance Books (ID=7), DNN takes approximately 4,600 seconds, whereas $\overline{\text{DNN}}$ takes over 60,000 seconds, approximately 16 hours.

Next, we solved the problem MILP by the branch-and-bound algorithm in Gurobi Optimizer 6.0.0 to compare the quality of the solutions obtained by our algorithm for 0-1SDP formulation. The computational results are given in Table 3. In our experiments, we impose the time limit of 86,400 seconds, i.e., 24 hours, on the computation time of the branch-and-bound algorithm.

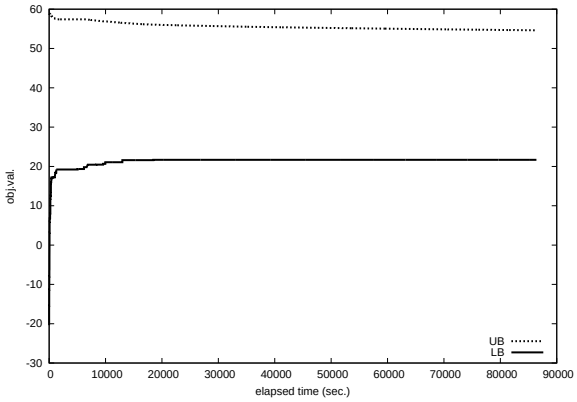
Table 3: Computational results of the branch-and-bound algorithm for MILP formulation.

| ID | MILP | | | |
|---|---|---|---|---|
| | UB | LB | gap(%) | time(sec.) |
| 1 | 8.8611 | 8.8611 | 0 | 0.50 |
| 2 | 7.8451 | 7.8451 | 0 | 0.74 |
| 3 | 8.7180 | 8.7180 | 0 | 7.84 |
| 4 | 8.6233 | 8.6233 | 0 | 6.10 |
| 5 | 13.8466 | 12.1252 | 14.196 | 86400.00 |
| 6 | 61.6302 | 24.5339 | 151.204 | 86400.00 |
| 7 | 54.6234 | 21.6803 | 151.949 | 86400.00 |

Table 3 shows that the first four instances are solved to optimality within a short computation time, while the remaining instances cannot be solved within the time limit we set. Especially, for the instances Les Misérables (ID=6) and Books (ID=7), a quite large duality gap still remained. Figures 2a and 2b show the upper and lower bounds vs. the elapsed time for these instances. From




(a) Instance Les Misérables




(b) Instance Books

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

the figures, we observe that (i) the branch-and-bound algorithm gives a good lower bound in an 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 [23] 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 the 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 [24], and Tarjan [25].

240 **Acknowledgment**

The authors thank Amy N. Langville, College of Charleston for drawing their attention to Fiedler's work [7].

## References


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

[2] Arenas, A., Fernández, A., & Gómez, S. (2008). Analysis of the structure of complex networks at different resolution levels. *New Journal of Physics*, *10*. doi:10.1088/1367-2630/10/5/053039.

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

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

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

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

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

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

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

[10] Gil-Mendieta, J., & Schmidt, S. (1996). The political network in mexico. *Social Networks*, *18*, 355–381. doi:10.1016/0378-8733(95)00281-2.

[11] Granell, C., Gomez, S., & Arenas, A. (2012). Hierarchical multiresolution method to overcome the resolution limit in complex networks. *International Journal of Bifurcation and Chaos*, *22*. doi:10.1142/S0218127412501714.

[12] Knuth, D. (1993). *The Stanford GraphBase: A platform for combinatorial computing* volume 37. Addison-Wesley Reading.

[13] Krebs, V. URL: http://www.orgnet.com/ (last visited April 19, 2016).

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

[15] Li, Z., Zhang, S., Wang, R., Zhang, X., & Chen, L. (2008). Quantitative function for community detection. *Physical Review E*, *77*. doi:10.1103/PhysRevE.77.036109.

[16] Lusseau, D., Schneider, K., Boisseau, O., Haase, P., Slooten, E., & Dawson, S. (2003). The bottlenose dolphin community of doubtful sound features a large proportion of long-lasting associations. *Behavioral Ecology and Sociobiology*, *54*, 396–405. doi:10.1007/s00265-003-0651-y.

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

[18] Michael, J. (1997). Labor dispute reconciliation in a forest products manufacturing facility. *Forest Products Journal*, *47*, 41–45.

[19] Michael, J., & Massey, J. (1997). Modeling the communication network in a sawmill. *Forest Products Journal*, *47*, 25–30.

[20] Muff, S., Rao, F., & Caflisch, A. (2005). Local modularity measure for network clusterizations. *Physical Review E*, *72*. doi:10.1103/PhysRevE.72.056107.

[21] Newman, M. (2004). Fast algorithm for detecting community structure in networks. *Physical Review E*, *69*. doi:10.1103/PhysRevE.69.066133.

[22] Newman, M., & Girvan, M. (2004). Finding and evaluating community structure in networks. *Physical Review E*, *69*. doi:10.1103/PhysRevE.69.026113.

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

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

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

[26] Traag, V., Van Dooren, P., & Nesterov, Y. (2011). Narrow scope for resolution-limit-free community detection. *Physical Review E*, *84*. doi:10.1103/PhysRevE.84.016114.

[27] Zachary, W. (1977). An information flow model for conflict and fission in small groups. *Journal of Anthropological Research*, *33*, 452–473.