数理解析研究所講究録
第1981巻 2016年 84-97

A DOUBLY NONNEGATIVE RELAXATION FOR MODULARITY DENSITY MAXIMIZATION

YOICHI IZUNAGA, TOMOMI MATSUI, AND YOSHITSUGU YAMAMOTO

ABSTRACT. Modularity proposed by Newman and Girvan is the most commonly used measure when the nodes of a graph are grouped into communities consisting of tightly connected nodes. However, some authors pointed out drawbacks of the modularity, the main issue of which is resolution limit. 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. To overcome this drawback, Li *et al.* have proposed a new measure called modularity density. In this paper, introducing a variant of a semidefinite programming called 0-1SDP, we show that the modularity density maximization can be modeled as 0-1SDP equivalently. Then we solve a relaxation problem of 0-1SDP to obtain an upper bound on the modularity density, and propose a lower bounding algorithm based on the combination of spectral heuristics and dynamic programming.

1. INTRODUCTION

Network analysis has received a growing attention. One of the most important issues in the network analysis is to find a meaningful structure, which often addresses identifying or detecting community structure. Here communities are the sets of nodes consisting of tightly connected nodes, but loosely connected each other. Community detection is applied to analyze the underlying relationship of diverse networks such as the social network, the biological network, the world-wide-web, and VLSI design.

A variety of approaches to detect communities has been proposed by several researchers, then a novel measure, called modularity, is proposed by Newman and Girvan [24]. The modularity was originally used as a stopping criterion of the hierarchical divisive algorithm [24], and then Newman [23] suggested the alternative approach of maximizing the modularity directly since a high value of the modularity represents a good community structure. The NP-hardness of the modularity maximization problem shown by Brandes *et al.* [6] turned researchers' attention to heuristic algorithms resulting in several efficient heuristics, while exact algorithms have been proposed only in a few papers [2, 29, 3].

The modularity maximization became one of the central subjects of research, but some authors pointed out its drawbacks, the main issue of which is *resolution limit.* Resolution limit, reported in Fortunato and Barthélemy [11], 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 narrows the application area 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 [4, 22, 13, 28]. Recently, Li *et al.* [17] have proposed a new measure for community detection, which is called modularity density, and the problem of maximizing the modularity density can be straightforwardly formulated as a nonlinear binary programming.

As for the mathematical optimization approaches for the modularity density maximization, Costa [7] has presented some *mixed integer linear programming* formulations, MILP for short, which enables an application of general-purpose solvers, 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.* [8] 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 a semidefinite programming called 0-1SDP. One of the advantages of this formulation is that the size of the problem is independent of the number of edges of the graph. In order to obtain an upper bound on the modularity density, we propose to relax 0-1SDP to a semidefinite programming problem with non-negative constraints. The relaxation problem obtained can be solved in polynomial time, and also does not require the number of communities in contrast to MILP formulations. 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 definition of the modularity density and a nonlinear binary programming formulation of the modularity density maximization, we review some properties of the modularity density. 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 non-negative relaxation problem of 0-1SDP, and in Section 6 we explain a heuristic algorithm which constructs a feasible solution by means of the solution of the relaxation problem. In Section 7, we report the computational experiments. Finally, we give some conclusions and further research 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 Π is defined as

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

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

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

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

A nonlinear binary programming formulation for MD has been proposed in Li *et al.* [17]. Although the optimal number of communities is a priori unknown similarly to the modularity maximization problem, 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 formulation:

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

where a_{ij} is the (i, j) element of the adjacency matrix A of graph G, and d_i is the degree of node $i \in V$. 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 [9]. Another approach is a branch-and-bound algorithm [5, 16] in global optimization area.

3. Some Properties of Modularity Density

In this section, we give some properties of the modularity density. Now suppose that there exist several isolated nodes in a graph G. After removing the isolated nodes from G, we find a partition $\Pi^\star$ that maximizes the modularity density on the reduced graph of G. If the contribution of a community is non-negative for any community of $\Pi^\star$, 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 [7], 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.

Proposition 3.2 (Costa [7], 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 $n - 2(|\Pi^\star| - 1)$, i.e.,*

$$2 \leq |C| \leq n - 2(|\Pi^\star| - 1) \text{ for any } C \in \Pi^\star.$$

4. Formulations

In this section, we first present a reformulation of the modularity density maximization, which is based on MILP formulation according to Costa [7]. Next, we show that modularity density maximization can be equivalently formulated as 0-1SDP, a variant of the semidefinite programming.

Hereafter, $\mathcal{S}_n, \mathcal{S}_n^+$ and $\mathcal{N}_n$ denote the set of $n \times n$ symmetric matrices, the positive semi-definite cone, and the symmetric non-negative cone, i.e.,

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

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* [10]:

$$y_{ijp} \leq x_{ip}, \quad y_{ijp} \leq x_{jp}, \quad x_{ip} + x_{jp} \leq y_{ijp} + 1 \quad \text{for } p \in T. \tag{4.1}$$

Note that the last inequality in (4.1) 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:

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

This equality constraint can be relaxed to

$$\alpha_p \leq \frac{4\sum_{\{i,j\}\in E} y_{ijp} - \sum_{i\in V} d_i x_{ip}}{\sum_{i\in V} x_{ip}}, \tag{4.2}$$

without overlooking an optimal solution due to the objective function. Since the denominator in (4.2) is positive, we can rewrite it as

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

Finally, to linearize the product $\alpha_p x_{ip}$, we then introduce a continuous variable γ_{ip} to replace $\alpha_p x_{ip}$ and make use of the following *McCormick inequalities* [19]:

$$\begin{aligned}
&L_\alpha x_{ip} \leq \gamma_{ip} \leq U_\alpha x_{ip} \quad \text{for } i \in V,\ p \in T, \\
&\alpha_p - U_\alpha(1 - x_{ip}) \leq \gamma_{ip} \leq \alpha_p - L_\alpha(1 - x_{ip}) \quad \text{for } i \in V,\ p \in T,
\end{aligned}$$

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

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

From the inequality (4.2) and Proposition 3.1, a valid lower bound on the variable α_p 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 the upper bound on α_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 is done for the purpose of comparing the solutions obtained by MILP formulation and 0-1SDP formulation introduced in Subsection 4.2, we solve the problem AP to optimality by Dinkelbach's parametric algorithm.

4.2. **0-1SDP formulation.** Let X denote the $n \times t$ matrix whose elements are the binary variables x_{ip} in the problem MD, i.e.,

$$
X = \begin{bmatrix}
x_{11} & x_{12} & \cdots & x_{1t} \\
x_{21} & x_{22} & \cdots & x_{2t} \\
\vdots & \vdots & \ddots & \vdots \\
x_{n1} & x_{n2} & \cdots & x_{nt}
\end{bmatrix}.
$$

The p-th column $(x_{1p}, x_{2p}, \ldots, x_{np})^\top$ of the matrix represents the incidence vector of the community C_p. Then the constraints of MD are concisely expressed as follows:

$$X\boldsymbol{e}_t = \boldsymbol{e}_n, \tag{4.3}$$

$$X^\top \boldsymbol{e}_n \geq \boldsymbol{e}_t, \tag{4.4}$$

$$X \in \{0,1\}^{n\times t}, \tag{4.5}$$

where $\boldsymbol{e}_k$ is the k-dimensional vector of 1's.

For a matrix X which satisfies (4.3), (4.4) and (4.5), we define the matrix Z as

$$Z = (z_{ij})_{i,j\in V} = X(X^\top X)^{-1}X^\top. \tag{4.6}$$

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 (4.3), (4.4) and (4.5). Then we can write the objective function in MD as $\mathrm{Tr}((2A - D)Z)$ by means of the matrix Z, where D is a diagonal matrix whose i-th diagonal element is the degree of node i, i.e., $D = \mathrm{Diag}(d_1, d_2, \ldots, d_n) \in \mathbb{R}^{n\times n}$. Clearly Z satisfies

$$Z\boldsymbol{e}_n = ZX\boldsymbol{e}_t = X\boldsymbol{e}_t = \boldsymbol{e}_n, \text{ and } \mathrm{Tr}(Z) = t$$

from (4.3) and (4.6). Moreover, it is symmetric and idempotent, i.e., $Z^2 = Z$ due to (4.6), hence Z is a orthogonal projection matrix. Now we consider the following problem, which we call 0-1SDP

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

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 [25].

From the discussion so far, we have seen that we can construct a feasible solution of 0-1SDP from a feasible solution of MD. Furthermore, we can also construct a feasible solution $X = (x_{ip})_{i\in V, p\in T}$ of MD satisfying (4.6) from a feasible solution of 0-1SDP.

Lemma 4.1. *For any feasible solution Z of 0-1SDP, we can construct a feasible solution X for MD which satisfies $Z = X(X^\top X)^{-1}X^\top$.*

Proof. Let Z be a feasible solution of 0-1SDP. Then it clearly satisfies the positive semi-definiteness due to the idempotence constraint. Then there exists an index $i_1 \in V$ which satisfies $z_{i_1 i_1} = \max\{ z_{ij} \mid i, j \in V \}$, 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

$$\sum_{j\in\mathcal{I}_1}\left(\frac{z_{i_1j}}{z_{i_1i_1}}\right)z_{i_1j}=\sum_{j\in\mathcal{I}_1}z_{i_1j},$$

which is rewritten as

$$\sum_{j\in\mathcal{I}_1}\left(\frac{z_{i_1j}}{z_{i_1i_1}}-1\right)z_{i_1j}=0.$$

From the non-negativity of z_{ij} and the maximality of $z_{i_1i_1}$, we have $(z_{i_1j}/z_{i_1i_1}-1)=0$, which implies $z_{i_1j}=z_{i_1i_1}$ for any $j\in\mathcal{I}_1$. Then we have

$$z_{i_1i_1}=z_{i_1j}=\frac{1}{|\mathcal{I}_1|}\quad\text{for any } j\in\mathcal{I}_1. \tag{4.7}$$

For an index $j\in\mathcal{I}_1$, we consider the (i_1,j) element, denoted by $Z^2_{i_1j}$, of the matrix Z^2, which is given as

$$Z^2_{i_1j}=\sum_{k\in V}z_{i_1k}z_{kj}=\sum_{k\in\mathcal{I}_1}z_{i_1k}z_{kj}=z_{i_1i_1}\left(\sum_{k\in\mathcal{I}_1}z_{kj}\right).$$

Note that the last equality is due to (4.7). The above equality, $Z^2=Z$ and (4.7) yield

$$z_{i_1i_1}=z_{i_1i_1}\left(\sum_{k\in\mathcal{I}_1}z_{kj}\right),$$

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 $z_{i_1i_1}$ 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

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

where P 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:

$$Z_{\bar{\mathcal{I}}_1}\boldsymbol{e}_{|\bar{\mathcal{I}}_1|}=\boldsymbol{e}_{|\bar{\mathcal{I}}_1|},\ \mathrm{Tr}(Z_{\bar{\mathcal{I}}_1})=t-1,\ Z^2_{\bar{\mathcal{I}}_1}=Z_{\bar{\mathcal{I}}_1}\text{ and }Z_{\bar{\mathcal{I}}_1}\in\mathcal{N}_{|\bar{\mathcal{I}}_1|}.$$

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

$$P^\top ZP=\begin{bmatrix}Z_{\mathcal{I}_1} & & & \\ & Z_{\mathcal{I}_2} & & \\ & & \ddots & \\ & & & Z_{\mathcal{I}_t}\end{bmatrix},$$

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 is clearly feasible for the problem MD and we can confirm $Z = X(X^\top X)^{-1}X^\top$ by simple calculation. □

The equivalence between MD and 0-1SDP directly follows the above lemma.

Theorem 4.2. *Solving the problem 0-1SDP is equivalent to finding an optimal solution of MD.*

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 the case of MILP formulation. These features make 0-1SDP more attractive than MILP formulation. The objective function in 0-1SDP is linear with respect to the matrix Z, but 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, which imposes the condition that each eigenvalue of Z, denoted by λ_i, is either 0 or 1. Then it would be a natural strategy to relax the constraint to a more tractable constraint. The first choice is to relax the binary constraint to $0 \leq \lambda_i \leq 1$, which 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. Since the optimal number t is unknown a priori, we further relax the constraint $\mathrm{Tr}(Z) = t$. Then we obtain the following relaxation problem over the *doubly non-negative 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 are solved efficiently, e.g., linear programming, second-order cone programming, and semidefinite programming problems. Indeed, the primal-dual-interior-point method solves the problems in polynomial time. On the other hand, since the doubly non-negative cone is not symmetric, we cannot directly apply the primal-dual-interior-point method to solve the problem DNN. Representing the doubly non-negative cone as a symmetric cone embedded in a higher dimension, we could apply the primal-dual-interior-point method to the embedded problem which is described as follows:

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

Although the above problem can be solved in polynomial time theoretically, we have to solve a quite large optimization problem over the positive semidefinite cone and it is too computationally expensive in practice. Nevertheless it is worthwhile to solve the problem DNN due to the fact that the doubly non-negative relaxation often provides significantly tight bound for some 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 the i-th row of Z is located on the (i,i) element for each $i \in V$, hence we have the following result.

Lemma 5.1. *The following inequalities are valid for 0-1SDP.*

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

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

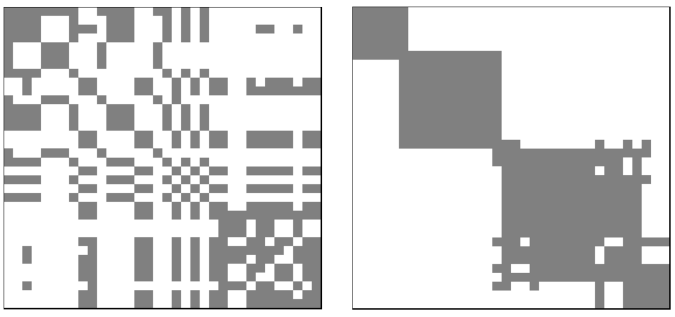
6. Heuristics based on Dynamic Programming

In this section, we will develop an algorithm to construct a feasible solution for MD from the solution obtained by solving the relaxation problems DNN and $\overline{\text{DNN}}$ presented in Section 5. The algorithm we propose is based on the combination of spectral heuristics and dynamic programming.

6.1. Permutation based on spectrum.

From the proof of Lemma 4.1, we have seen that an optimal solution of 0-1SDP forms a matrix with block diagonal structure by applying an appropriate simultaneous permutation of the rows and columns, and each block corresponds to a community. Unless otherwise stated, we refer to the simultaneous permutation of the rows and columns simply as a permutation. The optimal solution of the problem DNN or $\overline{\text{DNN}}$ is not necessarily transformed to a block diagonal matrix by any permutation since we relaxed some constraints in the problems. The solution however may provide a clue as to possibly a good solution of the original problem MD. Thus, it would be helpful to transform the solution matrix to a matrix which is close to a block diagonal one. To this end, we exploit the spectrum of the optimal solution.

Let $\lambda_1, \lambda_2, \ldots, \lambda_n \in \mathbb{R}$ be the eigenvalues of an optimal solution Z^* for the relaxation problem and $\boldsymbol{u}^1, \boldsymbol{u}^2, \ldots, \boldsymbol{u}^n \in \mathbb{R}^n$ be their corresponding eigenvectors. We assume that the eigenvalues are sorted in the non-increasing order, that is, $1 = \lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n \geq 0$. Focusing on the eigenvector $\boldsymbol{u}^2 = (u_1^2, u_2^2, \ldots, u_n^2)^\top$ corresponding to the second largest eigenvalue, we permute the rows and columns of the matrix Z^* consistent with the non-increasing order of values of components of $\boldsymbol{u}^2$. 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 1 (a) and (b). From these figures, we observe that the permuted



(a) Original matrix (b) Permuted matrix

Figure 1: Comparison with two matrices

matrix is considerably close to a block diagonal matrix. However, there is no theoretical validity of using the eigenvector corresponding to the second largest eigenvalue. Here still remains room for further research.

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-increasing order of components of $\boldsymbol{u}^2$. For the sake of notational simplicity, we renumber the nodes in V and denote the sequence $\bar{V}$ by $(1, 2, \ldots, n)$. Now, we try to find a partition with maximum modularity density of $\bar{V}$ under the constraint that each member consists of consecutive indices of $\bar{V}$. For this problem, we propose an algorithm using the dynamic programming. We define $q(k, l)$ 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 $(k, \ldots, l)$ of $\bar{V}$ to the modularity density when it is selected as a member of the partition. For any 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 consecutive subsequences. If we define $\mu(0) = 0$ for notational convenience, then $\mu(s)$ satisfies the recursive equation

$$\mu(s) = \max\{\, \mu(h) + q(h+1, s) \mid h \in \{0, 1, \ldots, s-1\} \,\}. \tag{6.1}$$

Owing to (6.1), 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 is given as follows.

Algorithm DP

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

Step 2 : Find the eigenvector $\boldsymbol{u}^2$ 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-increasing order of corresponding components in $\boldsymbol{u}^2$.

Step 3 : Set $\mu(0) := 0$.
`for` $s = 1$ `to` n `do`
Compute $\mu(s)$ according to (6.1).
`end-do`

7. Computational Experiments

To evaluate the lower and upper bounds obtained by our algorithm, we conducted the computational experiments. The experiments were performed 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 seven instances; Michael's strike dataset [20], Zachary's karate dataset [30], Gil-Mendieta and Schmidt's Mexico dataset [12], Michael and Massey's sawmill dataset [21], Lusseau's dolphins dataset [18], Hugo's Les Misérables dataset [14], and Krebs' books dataset [15]. Details of each instance are listed in Table 1, where the columns "LB*" and "$t^\star$" represent the known lower bound and the corresponding number of communities,

respectively. Since the first four instances in the table were solved to optimality in Costa [7], the optimal values and the optimal numbers of communities are listed. For the remaining instances, we list the lower bounds and the number of communities reported in Costa *et al.* [8] since the instances were not solved so far to our knowledge.

Table 1: Instances

ID	name	n	m	LB*	$t^\star$
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

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(UB – LB)/LB, and the computation time in seconds, respectively. For each instance, we observed that the predominant portion of the computation time was spent for solving a 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				$\overline{\text{DNN}}$			
	UB	LB	gap(%)	time(sec.)	UB	LB	gap(%)	time(sec.)
1	9.5808	8.8611	8.122	1.05	9.3049	8.8611	5.008	3.54
2	8.9548	7.8424	14.184	5.83	8.4141	7.8451	7.253	36.04
3	10.3151	8.5580	20.532	7.64	9.9570	8.5227	16.829	43.48
4	10.5048	7.0486	49.034	7.75	10.0311	7.3587	36.316	54.21
5	15.0218	9.8286	52.838	316.61	14.3552	11.4610	25.253	1681.81
6	28.0957	22.2680	35.279	658.28	27.4276	23.3416	17.505	7018.03
7	26.5387	20.2470	31.075	4626.11	24.7749	20.3150	21.953	60437.45

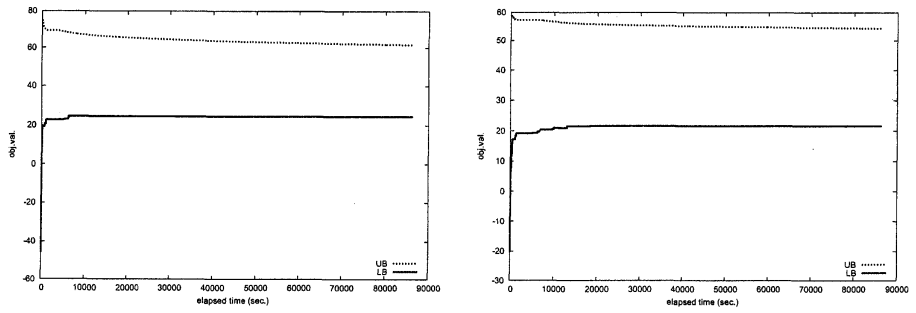
From Table 2, we observe that the upper bounds UB provided by $\overline{\text{DNN}}$ are tighter than those provided by DNN for all instances, which indicates the effectiveness of the valid inequalities in Lemma 5.1. Moreover, we also confirm that the lower bounds obtained for $\overline{\text{DNN}}$ are equal to or larger than those obtained for DNN for all instances except Mexico (ID=3). However, 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 4600 seconds, whereas $\overline{\text{DNN}}$ takes over 60000 seconds, approximately 16 hours.

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

Next, we solve the problem MILP by branch-and-bound algorithm in Gurobi 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 a time limit of 86400 seconds, 24 hours, on the computation time of the branch-and-bound algorithm.

In Table 3, we see 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. Especially, for the instances Les Misérables (ID=6) and Books (ID=7), a quite large duality gap remains. Figures 2 (a) and (b) show the upper and lower bounds vs. the elapsed time. From the figures, we observe the following behavior: (i) the branch-and-bound algorithm



(a) Instance Les Misérables (b) Instance Books

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

gives a good lower bound in early stage, and (ii) the improvement of upper bound rarely occurs throughout the computation. Owing to the latter of these, the duality gap still remains large even though a good feasible solution has been found. This suggests that deriving a tight upper bound enables us to estimate the accuracy of an incumbent solution obtained by the branch-and-bound algorithm more precisely.

8. Conclusion

In this paper, we presented 0-1SDP formulation originally introduced by Peng and Xia [25] for *minimum sum-of-squares clustering problem*, and showed that the equivalence between

the problem 0-1SDP and the modularity density maximization. Then, we proposed a method to solve a doubly non-negative relaxation of the problem 0-1SDP in order to obtain an upper bound on the modularity density. The advantage of the relaxation problem is twofold: the problem does not require the number of communities be known, and the size of the problem depends on neither the number of communities nor the number of edges. In addition, we developed a lower bounding algorithm based on the combination of spectral heuristics and dynamic programming.

The relaxation problem for our formulation was numerically compared to MILP formulation, and the results showed that the upper bounds provided by our formulation are competitive.

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 [26], and Tarjan [27].

References

[1] T. Achterberg. SCIP: Solving constraint integer programs. *Mathematical Programming Computation*, 1(1):1–41, 2009.

[2] G. Agarwal and D. Kempe. Modularity-maximizing graph communities via mathematical programming. *The European Physical Journal B*, 66(3):409–418, 2008.

[3] D. Aloise, S. Cafieri, G. Caporossi, P. Hansen, S. Perron, and L. Liberti. Column generation algorithms for exact modularity maximization in networks. *Physical Review E*, 82(4):046112, 2010.

[4] A. Arenas, A. Fernández, and S. Gómez. Analysis of the structure of complex networks at different resolution levels. *New Journal of Physics*, 10(5):053039, 2008.

[5] H.P. Benson. Global optimization algorithm for the nonlinear sum of ratios problem. *Journal of Optimization Theory and Applications*, 112(1):1–29, 2002.

[6] U. Brandes, D. Delling, M. Gaertler, R. Gorke, M. Hoefer, Z. Nikoloski, and D. Wagner. On modularity clustering. *IEEE Transactions on Knowledge and Data Engineering*, 20(2):172–188, 2008.

[7] A. Costa. MILP formulations for the modularity density maximization problem. *European Journal of Operational Research*, 245(1):14–21, 2015.

[8] A. Costa, S. Kushnarev, L. Liberti, and Z. Sun. Divisive heuristic for modularity density maximization. Optimization Online, 2015. `http://www.optimization-online.org/DB_HTML/2015/09/5116.html`.

[9] W. Dinkelbach. On nonlinear fractional programming. *Management Science*, 13(7):492–498, 1967.

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

[11] S. Fortunato and M. Barthélemy. Resolution limit in community detection. *Proceedings of the National Academy of Sciences*, 104(1):36–41, 2007.

[12] J. Gil-Mendieta and S. Schmidt. The political network in mexico. *Social Networks*, 18(4):355–381, 1996.

[13] C. Granell, S. Gomez, and A. Arenas. Hierarchical multiresolution method to overcome the resolution limit in complex networks. *International Journal of Bifurcation and Chaos*, 22(7):1250171, 2012.

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

[15] V Krebs. `http://www.orgnet.com/`(last visited october 13, 2015.).

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

[17] Z. Li, S. Zhang, R.S. Wang, X.S. Zhang, and L. Chen. Quantitative function for community detection. *Physical Review E*, 77(3):036109, 2008.

[18] D. Lusseau, K. Schneider, O.J. Boisseau, P. Haase, E. Slooten, and S.M. Dawson. The bottlenose dolphin community of doubtful sound features a large proportion of long-lasting associations. *Behavioral Ecology and Sociobiology*, 54(4):396–405, 2003.

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

[20] J.H. Michael. Labor dispute reconciliation in a forest products manufacturing facility. *Forest Products Journal*, 47(11/12):41–45, 1997.

[21] J.H. Michael and J.G. Massey. Modeling the communication network in a sawmill. *Forest Products Journal*, 47(9):25–30, 1997.

[22] S. Muff, F. Rao, and A. Caflisch. Local modularity measure for network clusterizations. *Physical Review E*, 72(5):056107, 2005.

[23] M.E. Newman. Fast algorithm for detecting community structure in networks. *Physical Review E*, 69(6):066133, 2004.

[24] M.E. Newman and M. Girvan. Finding and evaluating community structure in networks. *Physical Review E*, 69(2):026113, 2004.

[25] J. Peng and Y. Xia. A new theoretical framework for k-means-type clustering. In *Studies in Fuzziness and Soft Computing*, volume 180, chapter Foundations and Advances in Data Mining, pages 79–96. Springer, 2005.

[26] R.W.H. Sargent and A.W. Westerberg. Speed-up in chemical engineering design. *Transactions of the Institution of Chemical Engineers*, 42:190–197, 1964.

[27] R. Tarjan. Depth-first search and linear graph algorithms. *SIAM Journal on Computing*, 1(2):146–160, 1972.

[28] V.A. Traag, P. Van Dooren, and Y. Nesterov. Narrow scope for resolution-limit-free community detection. *Physical Review E*, 84(1):016114, 2011.

[29] G. Xu, S. Tsoka, and L.G. Papageorgiou. Finding community structures in complex networks using mixed integer optimisation. *The European Physical Journal B*, 60(2):231–239, 2007.

[30] W.W. Zachary. An information flow model for conflict and fission in small groups. *Journal of Anthropological Research*, 33(4):452–473, 1977.

(Y. Izunaga) Graduate School of Systems and Information Engineering, University of Tsukuba, Tsukuba, Ibaraki 305-8573, Japan
E-mail address: s1130131@sk.tsukuba.ac.jp

(T. Matsui) Department of Social Engineering, Graduate School of Decision Science and Technology, Tokyo Institute of Technology, Meguro-ku, Tokyo 152-8550, Japan
E-mail address: matsui.t.af@m.titech.ac.jp

(Y. Yamamoto) Faculty of Engineering, Information and Systems, University of Tsukuba, Tsukuba, Ibaraki 305-8573, Japan
E-mail address: yamamoto@sk.tsukuba.ac.jp