A Convex Formulation of Modularity Maximization for Community Detection

Emprise Y. K. Chan **Dit-Yan Yeung**
Department of Computer Science and Engineering
Hong Kong University of Science and Technology
Hong Kong, China
{emprisec,dyyeung}@cse.ust.hk

Abstract

Complex networks pervade in diverse areas ranging from the natural world to the engineered world and from traditional application domains to new and emerging domains, including web-based social networks. Of crucial importance to the understanding of many network phenomena, dynamics and functions is the study of network structural properties. One important type of network structure is known as community structure which refers to the existence of communities that are tightly knit local groups with relatively dense connections among their members. Community detection is the problem of detecting these communities automatically. In this paper, based on the modularity measure proposed previously for community detection, we first propose a reformulation of an optimization problem for the 2-partition problem. Based on this new formulation, we can extend it naturally for tackling the general k-partition problem directly without having to tackle multiple 2-partition subproblems like what other methods do. We then propose a convex relaxation scheme to give an iterative algorithm which solves a simple quadratic program in each iteration. We empirically compare our method with some related methods and find that our method is both scalable and competitive in performance via maintaining a good tradeoff between efficiency and quality.

1 Introduction

Many complex systems that consist of a large number of interacting entities or individuals are best described as networks. Complex networks can be found everywhere. Broadly speaking there are three major types of complex networks, namely, social networks, physical networks and biological networks. Social networks include Web-based communities, coauthorship and citation networks of researchers, social services networks, terrorist networks, etc.; physical networks include electric power distribution networks, Internet and the World Wide Web, telecommunication networks, sensor networks, highway transportation networks, etc.; and biological networks include neural networks in human and animal brains, networks of metabolic pathways, genetic regulatory networks, food webs, and so on. Many of the networks found in these applications are gigantic in size and also complex in both structure and dynamics.

Since network structural properties often affect network functions and behavioral characteristics, studying the topology or structure of complex networks is among the most important facets of network science [Newman *et al.*, 2006]. One important type of network structure is the so-called community structure [Girvan and Newman, 2002]. In many networks, there exist *communities* (or called clusters or network motifs) which correspond to relatively densely connected sets of vertices sharing some common properties. For example, they may correspond to emerging communities or political groups in a social network, congested areas in a telecommunication network, pathways between interacting constituents in a metabolic network, or hot spots in a disease or rumor spreading network. Identifying or detecting the communities in a network, called *community detection*, is to identify the clusters and possibly also their hierarchical organization using primarily local information about the relationships between individuals.

A principled approach to the study of complex networks is based on graph theory by representing a complex network as a graph, so that the relationships between the interacting entities correspond to the edges between vertices in the graph. For example, the presence of an edge between two vertices in a social network may represent friendship or partnership between two persons. There have been two relatively independent lines of research on several variants of this problem. One line of research has been pursued by computer scientists under the general name of graph partitioning [Kernighan and Lin, 1970; Shi and Malik, 1997]. The problem is to partition the vertices of a graph into groups such that the number of edges lying between the groups is minimal. This problem is fundamental to many parallel computing and circuit layout applications. The second line of research, sometimes referred to as hierarchical clustering or community structure detection, has mainly been pursued by sociologists and more recently by physicists, mathematicians and biologists. Unlike the first line of research, many applications of the second line have been focused on social and biological networks although possible applications in other areas have also drawn the attention of many researchers.

If there exists community structure in a network, we expect the intra-community edges to be significantly denser than the inter-community edges. In other words, when detecting the existence of communities, we search for structure that maximizes the number of intra-community edges while minimizing the number of inter-community edges. To provide a quantitative measure for characterizing the existence of community structure, the modularity measure was proposed in [Newman and Girvan, 2004] by measuring how different the network is when compared with one with edges placed at random according to the degree distribution. The modularity values range from -0.5 to 1 [Brandes *et al.*, 2008], with a higher positive value indicating a stronger support for community structure in the network. However, finding a partition with the highest modularity is NP-hard, making it necessary to rely on approximation methods especially when the network is large.

Most existing methods are hierarchical algorithms based on either the agglomerative or divisive approach. The agglomerative approach merges clusters repeatedly. An agglomerative method based on the modularity measure is called CNM [Clauset *et al.*, 2004]. On the other hand, the divisive approach tackles multiple 2-partition problems repeatedly. One popular divisive method based on the modularity measure uses an eigenvector-based algorithm to tackle each 2-partition problem by splitting a cluster based on the leading eigenvector of some matrix [Newman, 2006]. After applying the hierarchical algorithms, postprocessing methods such as the Kernighan-Lin algorithm [Kernighan and Lin, 1970] and greedy vertex move [Noack and Rotta, 2009] are usually applied to further improve the modularity value obtained.

Modularity maximization may also be formulated as an optimization or mathematical programming problem. It has been proved that the problem is an integer linear program (ILP) with $O(n^2)$ variables [Brandes *et al.*, 2008]. If the method tackles the problem via tackling multiple 2-partition modularity maximization subproblems, each 2-partition subproblem can be formulated as an integer quadratic program (IQP) with $O(n)$ variables [Agarwal and Kempe, 2008].

The focus of this paper is to study the community detection problem based on a principled optimization approach. We first propose a reformulation of an optimization problem for the 2-partition problem. Based on this new formulation, we can extend it naturally for a formulation representing the general k-partition problem directly as an IQP with kn variables without having to tackle multiple 2-partition subproblems. We then propose a convex relaxation scheme based on the k-partition problem formulation to give an iterative algorithm which solves a simple quadratic program (QP) in each iteration. Empirical studies show that our method is scalable and gives competitive results. Compared with other methods based on mathematical programming, our method is much faster and more scalable. In addition, with respect to the modularity obtained, it outperforms other methods which require postprocessing to obtain reasonable results.

The rest of this paper is organized as follows. In the next section, we will review the modularity measure and previous work on the 2-partition problem. Section 3 will present our extension for the k-partition problem and details of the algorithm. We will present both 2-partition and k-partition experiments in Section 4 and compare our method with several related methods. Some implementation details for dealing with large networks will be discussed in Section 5. Finally, we will give some concluding remarks in Section 6.

2 Previous Work

We represent each network by an undirected connected graph $G = (V, E)$, where $V = \{v_i\}$ and $E = \{\{v_i, v_j\}\}$ denote the vertex set and edge set, respectively, and $n = |V|$ and $m = |E|$ the corresponding cardinalities. The edge set E can also be represented by a symmetric $n \times n$ adjacency matrix $\mathbf{A}$ where each element A_{ij} in $\mathbf{A}$ is equal to 1 if $\{v_i, v_j\} \in E$ and 0 otherwise. Let $\mathcal{P}_k = \{C_1, \ldots, C_k\}$ denote a complete partition of V into k disjoint clusters C_i each of which is a nonempty set of vertices. In the sequel, we may drop the subscript k in $\mathcal{P}_k$ for notational simplicity if its value is irrelevant in the context. Also, we let $E(C_i)$ denote the number of intra-cluster edges within cluster C_i.

Modularity was first proposed in [Newman and Girvan, 2004] as a performance measure for the quality of the community structure (i.e., partition) found by a community detection or clustering algorithm. There are several different but equivalent forms for expressing the modularity $q(\mathcal{P})$ of a partition $\mathcal{P}$. One convenient form is expressed based on the cluster structure [Brandes *et al.*, 2008]:

$$q(\mathcal{P}) = \sum_{C_j \in \mathcal{P}} \left[\frac{E(C_j)}{m} - \left(\frac{\sum_{v_i \in C_j} \deg(v_i)}{2m} \right)^2 \right], \quad (1)$$

where $\deg(v_i) = \sum_j A_{ij}$ is the degree of v_i. To maximize $q(\mathcal{P})$, it would be desirable to maximize the first term and minimize the second term inside the outermost summation in (1). To maximize the first term, each cluster should contain as many edges as possible. To minimize the second term, the graph should be split into many clusters each with a small total degree. However, these two criteria cannot be achieved independently, showing the necessity of a tradeoff in the optimization problem. It has been shown that maximizing the modularity can exhibit undesirable behavior such as nonlocality [Brandes *et al.*, 2008].

Instead of expressing $q(\mathcal{P})$ in terms of the cluster structure, we may also expand the outermost summation in (1) to express it in terms of the vertices directly. Let us introduce an extra variable d_{ij} for each pair of vertices v_i and v_j such that $d_{ij} = 1$ if v_i and v_j belong to the same cluster and $d_{ij} = 0$ otherwise. The modularity is now expressed as follows:

$$q(\mathcal{P}) = \frac{1}{2m} \sum_{ij} \left(A_{ij} - \frac{\deg(v_i)\deg(v_j)}{2m} \right) d_{ij}. \quad (2)$$

This alternative form is informative in revealing a different interpretation of the modularity measure. Inside the summation in (2), the first term A_{ij} is the *actual* number of edges between v_i and v_j and the second term $\deg(v_i)\deg(v_j)/2m$ is the *expected* number of edges between them if edges are placed randomly according to the degrees. To a certain extent, the second term plays the role of a "null hypothesis"

against which the first term is "tested". The modularity $q(\mathcal{P})$ is obtained by summing the difference of the two terms above over all pairs of vertices that belong to the same cluster in $\mathcal{P}$.

Given a partition $\mathcal{P}$ and hence the relationship between vertices, we can compute the modularity $q(\mathcal{P})$ easily as in (2). The problem of maximizing the modularity can be formulated as an ILP to obtain a partition $\mathcal{P}$, via the d_{ij}, that maximizes $q(\mathcal{P})$. Note that since $d_{ij} = d_{ji}$ and $d_{ii} = 1$, it suffices to introduce only $\binom{n}{2} = O(n^2)$ optimization variables d_{ij} for $i < j$. To ensure consistency of the variables d_{ij} in representing a valid partition $\mathcal{P}$, $\binom{n}{3} = O(n^3)$ constraints are introduced to enforce the reflexivity, symmetry and transitivity properties. However, this discrete optimization problem is known to be NP-complete [Brandes *et al.*, 2008], making it unappealing to solve the ILP directly for most practical applications.

A previous work [Newman, 2006] proposed a method for the 2-partition problem by solving a modified optimization problem with the objective function given in (2).[1] The formulation involves a modularity matrix $\mathbf{Q}$ where each element Q_{ij} is just the term $A_{ij} - \deg(v_i)\deg(v_j)/2m$ in (2). Also, the variable d_{ij} is rewritten as $(y_i y_j + 1)/2$ where $y_i = 1$ if $v_i \in C_1$ and $y_i = -1$ if $v_i \in C_2$. Moreover, we note that $\sum_{ij} A_{ij} = \sum_i \deg(v_i) = 2m$ and hence $\sum_{ij} Q_{ij} = 0$. Thus the modularity $q(\mathcal{P}_2)$ for a 2-partition $\mathcal{P}_2$ can be simplified to

$$\begin{aligned} q(\mathcal{P}_2) &= \frac{1}{4m}\sum_{ij} Q_{ij}(y_i y_j + 1) \\ &= \frac{1}{4m}\sum_{ij} Q_{ij} y_i y_j \\ &= \frac{1}{4m}\mathbf{y}^T\mathbf{Q}\mathbf{y} \end{aligned} \tag{3}$$

in a quadratic form of the vector $\mathbf{y} = (y_1, \ldots, y_n)^T \in \{-1, +1\}^n$. Maximizing $q(\mathcal{P}_2)$ is thus an IQP:

$$\begin{aligned} \max_{\mathbf{y}} \quad & \frac{1}{4m}\mathbf{y}^T\mathbf{Q}\mathbf{y} \\ \text{subject to} \quad & y_i \in \{-1, +1\},\ \forall i. \end{aligned} \tag{4}$$

The NP-completeness of this problem [Brandes *et al.*, 2008] has triggered some efforts to devise approximation methods.

An eigenvector-based method was proposed in [Newman, 2006]. The method is similar to spectral partitioning except that the matrix used is the modularity matrix instead of the Laplacian matrix. After solving the eigendecomposition problem, the vertices are assigned to clusters based on the signs of the elements in the eigenvector corresponding to the largest eigenvalue.

Another method based on a vector programming relaxation of the IQP was proposed in [Agarwal and Kempe, 2008]. It replaces each variable y_i by an n-dimensional vector $\mathbf{y}_i$ with constraints $\mathbf{y}_i^T\mathbf{y}_j = 1$ and solves the relaxed problem by semi-definite programming (SDP). After that, the result is rounded to give two partitions by randomly generating a hyperplane to separate the vectors $\mathbf{y}_i$ on the hypersphere. The best partition obtained out of 5,000 randomly generated hyperplanes is then reported as the solution.

[1] The final partition is obtained by tackling multiple 2-partition problems repeatedly.

3 Our Proposed Method

In this section, we first propose an extension of the IQP for the 2-partition problem in (4) to the general k-partition problem. We then propose an iterative rounding scheme by solving a convex relaxation of the IQP. Some issues on tackling the k-partition problem are discussed, including our recommendation of the convex relaxation scheme to use.

3.1 k-partition Integer Quadratic Program

To facilitate extension to the k-partition problem, we first give a different formulation of the IQP for the 2-partition problem. Our point of departure is to rewrite the variable d_{ij} in (2) as $\sum_{p=1}^{2}\left(\frac{z_{pi}+1}{2}\right)\left(\frac{z_{pj}+1}{2}\right) = \frac{1}{4}\left(\sum_{p=1}^{2} z_{pi}z_{pj} + 2\right)$ where $z_{pi} = +1$ if $v_i \in C_p$ and $z_{pi} = -1$ otherwise. Since $\sum_{ij} Q_{ij} = 0$, the modularity $q(\mathcal{P}_2)$ can be expressed as a sum of two quadratic terms:

$$q(\mathcal{P}_2) = \frac{1}{8m}(\mathbf{z}_1^T\mathbf{Q}\mathbf{z}_1 + \mathbf{z}_2^T\mathbf{Q}\mathbf{z}_2), \tag{5}$$

where $\mathbf{z}_p = (z_{p1}, \ldots, z_{pn})^T$. Note that (5) is equivalent to (3) for the 2-partition problem, with the only difference being that there is redundancy in this new formulation because $z_{1i} = -z_{2i}$.

Extension to the general case is now straightforward. Let $d_{ij} = \sum_{p=1}^{k}\left(\frac{z_{pi}+1}{2}\right)\left(\frac{z_{pj}+1}{2}\right) = \frac{1}{4}\left(\sum_{p=1}^{k} z_{pi}z_{pj} + 4 - k\right)$ and $\mathbf{Z} = (\mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_k) \in \{-1, +1\}^{n\times k}$. The k-partition optimization problem can be formulated as follows:

$$\begin{aligned} \max_{\mathbf{Z}} \quad & \frac{1}{8m}\mathrm{Tr}(\mathbf{Z}^T\mathbf{Q}\mathbf{Z}) \\ \text{subject to} \quad & z_{pi} \in \{-1, +1\},\ \forall p, \forall i \\ & \sum_{p=1}^{k} z_{pi} = -(k-2),\ \forall i. \end{aligned} \tag{6}$$

Unlike the ILP formulation based on (2) which requires the reflexivity, symmetry and transitivity constraints to be enforced explicitly on the variables d_{ij}, the IQP formulation in (6) automatically satisfies these properties. Moreover, we note that the number of optimization variables in the IQP formulation (6) is kn which is much less than $O(n^2)$ in the ILP if $k \ll n$. The value of k corresponds to the maximum number of clusters allowed. If a graph attains its maximum modularity with $\hat{k}$ clusters and we set $k \geq \hat{k}$, the optimal solution will still involve only $\hat{k}$ clusters. In practice, the number of clusters needed for attaining the maximum modularity is often found to be a small fraction of n due to the so-called resolution limit [Fortunato and Barthélemy, 2007]. Thus the assumption that $k \ll n$ is a very reasonable one that is satisfied in many real-world applications.

3.2 Convex Relaxation

Despite the favorable properties of the formulation in (6), its non-convexity makes it hard to solve directly. We propose here a simple yet effective convex relaxation scheme.

Auxiliary Function

We note that the form $\text{Tr}(\mathbf{Z}^T\mathbf{Q}\mathbf{Z})$ represents a real quadratic surface centered at the origin if $\mathbf{Z}$ is a real-valued matrix. An optimization method for optimizing continuous functions can then be used instead of handling the combinatorial nature of the original discrete IQP. Once the continuous optimization problem has been solved, we can round the continuous values of the solution to discrete values to approximate the discrete optimization variables in (6).

However, the continuous version of the optimization problem in (6) is still not convex because $\mathbf{Q}$ is an indefinite matrix. We observe that the rounding result is only affected by the relative ordering of the continuous values instead of the actual values. Let us consider an auxiliary function defined as $\text{Tr}(\mathbf{Z}^T\mathbf{Q}\mathbf{Z}) - \lambda\|\mathbf{Z}\|^2 = \text{Tr}(\mathbf{Z}^T\mathbf{Q}\mathbf{Z} - \lambda\mathbf{Z}^T\mathbf{Z})$. Maximizing this auxiliary function will give a solution that approximates the ordering of the continuous values that could be obtained based on the original objective function. If we set λ to the largest (positive) eigenvalue of $\mathbf{Q}$, the matrix $\mathbf{S} = \mathbf{Q} - \lambda\mathbf{I}$ will be negative semi-definite and hence $\text{Tr}(\mathbf{Z}^T\mathbf{S}\mathbf{Z})$ is concave downwards. Thus optimizing this auxiliary function is a convex optimization problem that can be used to approximate the ordering of the variables. We also note that for the discrete IQP in (6), optimizing the auxiliary function is exactly equivalent to optimizing the original function because $\|\mathbf{Z}\|^2 = kn$ for $z_{pi} \in \{-1, +1\}$.

Iterative Vertex Assignment

Based on the observations above, we propose an iterative algorithm for vertex assignment that assigns vertices to clusters in stages.[2]

Let $\mathcal{U} \subseteq V$ denote the set of vertices that have already been assigned to clusters. The algorithm ends when $\mathcal{U} = V$. In each iteration, with some vertices already assigned, it solves a convex QP to choose some vertices from $\widetilde{\mathcal{U}} = V \backslash \mathcal{U}$ to assign to clusters and update $\mathcal{U}$ and $\widetilde{\mathcal{U}}$ accordingly. The convex QP can be formulated as follows:

$$\begin{aligned} \max_{\mathbf{Z}} \quad & \frac{1}{8m}\text{Tr}(\mathbf{Z}^T\mathbf{S}\mathbf{Z}) \\ \text{subject to} \quad & z_{pi} \in [-1, +1],\ \forall p,\ \forall v_i \in \widetilde{\mathcal{U}} \\ & \sum_{p=1}^{k} z_{pi} = -(k-2),\ \forall v_i \in \widetilde{\mathcal{U}} \\ & z_{pi} \text{ set to assigned values } \pm 1,\ \forall p,\ \forall v_i \in \mathcal{U}. \end{aligned} \tag{7}$$

We can see that it searches within a hypercube on the auxiliary function.

Suppose we start with all vertices unassigned, i.e., $\mathcal{U} = \emptyset$. It will give a trivial solution with all $z_{pi} = -(k-2)/k$, since $\{z_{pi}\}_{p=1}^k$ are equal and $\|\mathbf{Z}\|^2$ is minimum. To break the symmetry, we randomly assign a vertex to the first cluster via an equality constraint and hence $\mathcal{U} \neq \emptyset$.

Our preliminary investigation shows that assigning all vertices to clusters after solving the QP once does not give very good results. Instead, we use an iterative procedure. After solving the QP in each iteration to obtain the solution $\hat{\mathbf{Z}}$, we only choose a fixed number C of the vertices with largest values and assign vertex v_i to cluster C_q if $q = \text{argmax}_p\, \hat{z}_{pi}$. This can be seen as a rounding scheme that converts the values $\hat{z}_{pi}$ from $[-1, +1]$ to $\{-1, +1\}$ for the corresponding z_{pi}. The rationale behind this scheme is that those vertices with high $\hat{z}_{pi}$ values have higher confidence to be assigned to the corresponding clusters. After choosing the vertices, we update $\mathcal{U}$ and $\widetilde{\mathcal{U}}$ and proceed to the next iteration as long as $\mathcal{U} \neq V$.

The algorithm is summarized in Algorithm 1 below. The parameter $\rho \in (0, 1)$ affects the rate of rounding. After each iteration, C ($= \max\{\lfloor \frac{n}{\ln n} \ln \frac{1}{\rho} \rfloor, 1\}$) vertices with the largest values are rounded, so the total number of iterations needed is around $\ln n (\ln \frac{1}{\rho})^{-1} = O(\ln n)$. In general, using a larger value of ρ takes more iterations to complete the whole procedure but gives better quality in terms of the modularity obtained. Thus choosing an appropriate value for ρ requires a tradeoff between efficiency and quality.

Algorithm 1 Iterative Algorithm for Vertex Assignment

```
C := max{⌊n/(ln n) ln(1/ρ)⌋, 1}
i := rand(n)
z_{1i} := 1; z_{pi} := −1 for p ≥ 2
while not all vertices have been assigned do
  perform QP to obtain solution Ẑ
  R := C vertices in Ũ with largest values of max_p ẑ_{pi}
  for each v_i ∈ R do
    q := argmax_p ẑ_{pi}
    z_{qi} := 1; z_{pi} := −1 for p ≠ q
  end for
end while
```

Issues on k-partition

The algorithm described above works well for 2-partition. When $k = 2$, the second constraint of (7) imposes the equality $z_{1i} = -z_{2i}$ and hence the signs of the variables can reflect how the corresponding vertices should be assigned to clusters. However, it does not work very well for the general k-partition problem especially when k is large. For example, in the beginning when only the first cluster C_1 has been assigned a randomly chosen vertex v_i, $\{z_{pi}\}_{p=2}^k$ are equal due to symmetry. If another vertex v_j is highly unlikely to belong to cluster C_1 with $z_{1j} = -1$, we will see that $z_{2i} = \cdots = z_{ki} = -(k-3)/(k-1)$ due again to symmetry. For large k, the value $-(k-3)/(k-1)$ is still very small (close to -1) making it hard to decide to which cluster v_j should be assigned.

We propose a revised form of (7) to alleviate this problem. We note that removing the bounded region constraint on $\mathbf{Z}$ does not alter the property of approximating the relative ordering. On the other hand, by allowing $z_{pi} \in (-\infty, +\infty)$, a vertex v_j can take a very negative z_{pj} value if it is highly unlikely to belong to cluster C_p. This makes it possible for some z_{pj} value to get significantly large to justify for assigning it to a currently unassigned cluster if it does not fit into any of the

[2]The reasons for not assigning all vertices at once will be made clear below.

already assigned clusters. The relaxed optimization problem is as follows:

$$\begin{aligned} \max_{\mathbf{Z}} \quad & \frac{1}{8m}\mathrm{Tr}(\mathbf{Z}^T\mathbf{S}\mathbf{Z}) \\ \text{subject to} \quad & z_{pi} \in (-\infty, \infty),\ \forall p,\ \forall v_i \in \widetilde{\mathcal{U}} \\ & \sum_{p=1}^{k} z_{pi} = -(k-2),\ \forall v_i \in \widetilde{\mathcal{U}} \\ & z_{pi} \text{ set to assigned values } \pm 1,\ \forall p,\ \forall v_i \in \mathcal{U}. \end{aligned} \tag{8}$$

We have also considered another approximation scheme to further relax the problem by removing the equality constraints:

$$\begin{aligned} \max_{\mathbf{Z}} \quad & \frac{1}{8m}\mathrm{Tr}(\mathbf{Z}^T\mathbf{S}\mathbf{Z}) \\ \text{subject to} \quad & z_{pi} \in (-\infty, \infty),\ \forall p,\ \forall v_i \in \widetilde{\mathcal{U}} \\ & z_{pi} \text{ set to assigned values } \pm 1,\ \forall p,\ \forall v_i \in \mathcal{U}. \end{aligned} \tag{9}$$

This relaxation basically removes the direct interaction between clusters and considers the assignment of vertices to clusters independently. We can also use a simple rounding scheme to assign vertices to clusters by identifying the largest $\hat{z}_{pi}$ values as before.

Note that in (9) the assigned vertices in $\mathcal{U}$ set affine plane constraints ($z_{pi} = \pm 1$) to yield a related convex QP problem. Let $\mathbf{X}$ be equal to $\mathbf{Z}$ with the rows corresponding to the assigned vertices in $\mathcal{U}$ removed. By substituting the assigned values of z_{pi} into the objective function, the constrained QP in (9) is equivalent to the following unconstrained QP:

$$\max_{\mathbf{X}} \quad \frac{1}{8m}\mathrm{Tr}\left(\mathbf{X}^T\mathbf{S}'\mathbf{X} + 2\mathbf{F}^T\mathbf{X}\right), \tag{10}$$

where $\mathbf{S}'$ and $\mathbf{F}$ are appropriate matrices. Setting the first derivative of the objective function in (10) to zero, it is easy to see that this problem is equivalent to solving the linear system $\mathbf{S}'\mathbf{X} = -\mathbf{F}$ which has a closed-form solution.

4 Experimental Validation

In this section, we present experimental results of several realizations of Algorithm 1 and compare them empirically with some closely related methods in the community detection literature. For easy reference, we use M1 to refer to using (8) for the QP in Algorithm 1 and M2 to using (9). For each method, we allow two variants corresponding to running with restart (R) and without restart (N). For example, M2R refers to M2 with restart. In the algorithm, while choosing vertices for assignment (rounding) with the restart (R) option after solving a QP, further rounding will be terminated to restart a new QP if a previously unassigned cluster is assigned the first element. The rationale behind the restart option is that assigning to a new cluster may result in a large change in modularity obtained by the QP. For example, assigning two vertices to a newly assigned cluster together may not mean these two vertices are close to each other. We use MATLAB to implement our methods, with M1R/M1N implemented using the large-scale algorithm (allowing only equality constraints) of the `quadprog` QP solver and M2R/M2N implemented using the `pcg` preconditioned conjugate gradients method for linear systems.[3] More implementation details will be provided in Section 5. For all four variants, we need to set the parameter k for the k-partition problem as well as the rounding parameter ρ.

Our first set of experiments is for the 2-partition problem. The two representative and closely related methods for tackling this problem are the eigenvector-based method (Eig) of [Newman, 2006] and the vector programming method (VP) of [Agarwal and Kempe, 2008]. For the general k-partition problem, there does not exist any method in the literature based on modularity maximization that tackles the problem directly. Instead, the methods (such as Eig and VP) for 2-partition are applied repeatedly to tackle a 2-partition subproblem at a time, often requiring a postprocessing step at the end such as applying the Kernighan-Lin algorithm [Kernighan and Lin, 1970] in order to obtain reasonable results. Thus they are best regarded as hybrid methods. Therefore our second set of experiments for the k-partition problem will mainly be based on studying the effect of varying the k parameter on the four variants of our algorithm. Nevertheless, we will quote some results reported for other methods on the same datasets to put our work in perspective. Note, however, that the results reported for CNM and our method do not apply any postprocessing method while those for other methods do.

We use six network datasets commonly used by other researchers working on this topic, with network size (i.e., vertex set size n) varying from 34 to 27,519. These datasets are the karate club network of Zachary (karate) [Zachary, 1977], a social network of bottlenose dolphins (dolp) [Lusseau *et al.*, 2003], a collaboration network of jazz musicians (jazz) [Gleiser and Danon, 2003], a metabolic network for nematode (meta) [Jeong *et al.*, 2000], a network of email contacts at a university (email) [Guimerà *et al.*, 2003], and a coauthorship network of scientists working in condensed matter physics (phy) [Newman, 2001].

4.1 2-partition Experiments

We run implementations of M2R, Eig and VP on a Windows-based Genuine Intel(R) CPU T2500 @ 2.00GHz machine. Table 1 shows the experimental results reporting both clustering quality and efficiency.

Network	Size	M2R	Eig	VP
karate	34	0.3718[0.09]	0.3715[0.01]	0.3718[7.46]
dolp	62	0.4027[0.18]	0.3899[0.01]	0.4027[8.22]
jazz	198	0.3193[0.25]	0.3048[0.01]	0.3151[15.3]
meta	453	0.3188[0.27]	0.2648[0.01]	0.3175[65.0]
email	1,133	0.3469[0.51]	0.2850[0.02]	0.3675[607]
phy	27,519	0.4250[10.2]	0.0346[0.75]	-

Table 1: Modularity $q(\mathcal{P}_2)$ and running time (in seconds shown inside square brackets) of M2R, Eig and VP.

[3] Our MATLAB implementation can be downloaded from `http://www.cse.ust.hk/~dyyeung/code/cqp.rar`.

For M2R, we set $\rho = 0.9$ and perform 10 runs for each dataset. The modularity and running time reported are the average values over 10 runs. For 2-partition, since $z_{1i} = -z_{2i}$ for all $v_i \in \mathcal{U}$, the optimization variables in (9) automatically satisfy $z_{1j} = -z_{2j}$ for all $v_j \in \widetilde{\mathcal{U}}$. Therefore M1R and M2R are equivalent when $k = 2$. We choose M2R instead of M1R here because solving a linear system using `pcg` is 3 to 6 times faster.

The randomness of our algorithm comes from the initial random vertex selection for assigning to the first cluster. We notice that the result obtained is not very sensitive to the random initialization. For the karate network, for instance, we obtain the optimal modularity (0.3718) in 9 out of 10 runs and a suboptimal value (0.3715) only once when the first vertex chosen happens to lie between two clusters. For all the networks tested, the standard deviation of the modularity obtained over 10 runs is always below 0.0023.

From the results, we notice that M2R always outperforms Eig in terms of the modularity measure and achieves clustering quality comparable to VP but requires significantly less running time. Although Eig is simple and hence very efficient, it can only make use of the leading eigenvector with the largest eigenvalue which cannot capture well the network behavior especially for large networks. For example, for the phy network, using the leading eigenvector can only give a very low modularity of 0.0346 but it can reach 0.3599 if the eigenvector corresponding to the 18th largest eigenvalue is used instead. Although VP can often obtain a near-optimal quality, its need for $O(n^2)$ optimization variables makes it infeasible for moderately large networks, including the phy network. Our algorithm maintains a good tradeoff between quality and efficiency, with roughly linear time complexity with respect to the number of vertices.

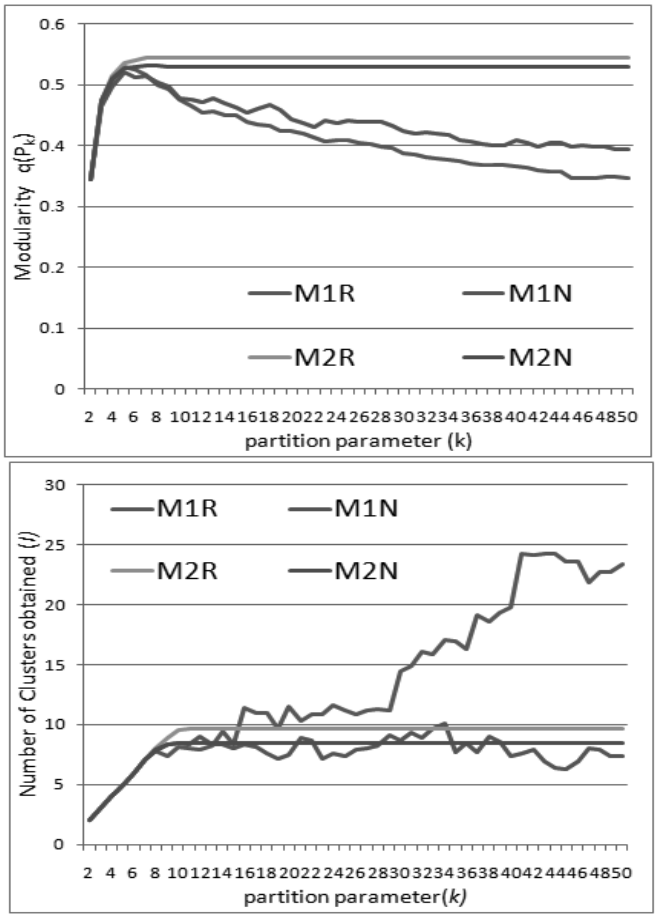

Figure 1: Modularity $q(\mathcal{P}_k)$ (upper) and number of clusters l obtained (lower) by varying k for the email network when $\rho = 0.81$.

4.2 k-partition Experiments

We now compare the four variants of our algorithm for the k-partition problem on the email network by varying the parameter k. For each variant, we perform 10 runs for each value of k and report the average result.

Figure 1 summarizes the results obtained by the four variants. We can see that M1 is generally inferior to M2. Although they have similar behavior when k is less than the optimal number of clusters $\hat{k}$, the performance of M1 deteriorates as k increases. For M2, the modularity increases initially and becomes steady when k exceeds around 10. Moreover, the number of clusters l obtained by M1 is not very stable and varies a lot as k increases, making it crucial to choose an appropriate k value for M1 to work effectively. If k is set too large, it is difficult to have an appropriate rounding scheme when the equality constraints of (8) vary. On the other hand, for M2, since the interaction between clusters is removed after eliminating the equality constraints, the l value obtained is independent of k when k exceeds around 10. This explains why M2 always reports the same result for larger k values. Previous work shows that $\hat{k}$ is about 8 to 11 [Noack and Rotta, 2009], implying that the l value found by our algorithm is a good approximation of $\hat{k}$.

Allowing for restart usually leads to a larger l value but requires some extra iterations. When ρ is large enough, the restart option makes very little difference. When ρ is not large, M2R usually outperforms M2N as shown in Figure 1 because the restart option avoids assigning too many vertices to a newly assigned cluster. However, the effect on M1 is not obvious and it depends on the dataset used. The benefit of the restart option is generally more significant when ρ is small or l is large, but the number of extra iterations needed increases with l.

In general, we recommend using M2R over other variants because M2 is more stable than M1 and allowing restart generally gives better performance.

4.3 Comparison with Other Methods

We now quote some results from the literature reported for other methods to give a comparison with our method. The four methods included in our comparison are an agglomerative algorithm (CNM) [Clauset *et al.*, 2004], an eigenvector-based divisive algorithm (Eig) [Newman, 2006], a vector programming algorithm (VP) [Agarwal and Kempe, 2008], and a mutli-level algorithm (MLA) [Noack and Rotta, 2009]. For our algorithm, we use the `pcg` implementation of M2R and run it on a Linux-based Genuine Intel(R) Core(TM)2 Duo

E8400 @ 3.00GHz machine. Both the average and maximum modularity and average running time over 10 runs are reported for our method. Table 2 summarizes the results for all six network datasets. Note that no results were reported for CNM and Eig on the dolp network and VP cannot be applied to large networks such as the phy network. All methods compared except CNM applied postprocessing algorithms for fune-tuning in order to obtain the results reported here.

Network	q_{mean}	q_{max}	time	CNM	Eig	VP	MLA
karate	.4103	.4174	.099	.381	.419	.420	.4197
dolp	.5219	.5268	.212	-	-	.526	.5276
jazz	.4400	.4414	.269	.439	.442	.445	.4447
meta	.4224	.4285	1.45	.402	.435	.450	.4461
email	.5348	.5465	2.66	.494	.572	.579	.5774
phy	.7830	.7832	6311	.668	.723	-	.8143

Table 2: Average and maximum modularity $q(\mathcal{P}_k)$ and average running time (in seconds) over 10 runs by M2R for $\rho = 0.9$ as well as modularity reported for other methods in the literature.

We can see that M2R outperforms CNM consistently for all datasets. For small networks, M2R generally obtains close to optimal modularity values. For larger networks, it can still obtain competitive results even without applying any postprocessing step.[4] For the karate network, we note that Eig without postprocessing can only obtain a modularity of 0.393 which is much lower than that obtained by M2R. Unlike the other methods which are hierarchical in taking an agglomerative or divisive approach to tackle the modularity maximization problem, a major difference of our method is that it tackles the k-partition problem directly. As a consequence, our method can tackle the k-partition problem directly even when k is set to be smaller than $\hat{k}$ while other methods cannot.

Our method also has competitive advantage in terms of running time. For networks with $n < 1000$, it is very fast and gives the results almost instantly. Empirically we found that the time complexity of our algorithm is about linear in the number of variables, i.e., $O(kn)$. The main reason for the dramatic increase in running time for the phy network is due to the large l value. The restart option for M2R adds around l extra iterations. For instance, at $\rho = 0.9$, the number of iterations reported by M2N is only 98, while M2R is 208 as $l = 128$. M2N only takes about 1800s to complete while M2R takes more than 6000s.

In summary, these experiments show that our method can scale well with the network size and can obtain results comparable to methods that are computationally much more demanding or require postprocessing.

5 Some Implementation Details and Complexity Analysis

In this section, we will provide some implementation details including specific techniques to speed up our algorithm.

[4]Although we could also perform postprocessing to further improve the results, our priority here is to push the limit of a principled approach based entirely on a well-formulated optimization problem.

The MATLAB functions needed for our optimization problem do not support direct optimization of the trace form $\text{Tr}(\mathbf{Z}^T\mathbf{S}\mathbf{Z})$ with respect to a matrix variable $\mathbf{Z}$. As such, we construct a matrix $\mathbf{M}$ in the form of a block diagonal matrix with $\mathbf{S}$ appearing k times in the diagonal blocks and use the vec operator to vectorize $\mathbf{Z}$ to a long column vector $\mathbf{z} \in \mathbb{R}^{nk}$. Thus we can rewrite $\text{Tr}(\mathbf{Z}^T\mathbf{S}\mathbf{Z})$ as $\mathbf{z}^T\mathbf{M}\mathbf{z}$ without involving the trace operator. However, directly storing $\mathbf{M}$ requires $O(kn^2)$ space. To overcome this problem, we note that $\mathbf{S}$ can be expressed as $\mathbf{S} = \mathbf{A} - \lambda\mathbf{I} - \frac{\mathbf{d}\mathbf{d}^T}{2m}$ where $\mathbf{d}$ is the degree vector (i.e., a vector of the degrees of the n vertices) and hence it is sufficient to store only $\mathbf{A}$ and $\mathbf{d}$ which require $O(m+n)$ space in total. Since $\mathbf{A}$ is typically sparse for the applications of interest to us, m is typically $O(n)$ rather than $O(n^2)$. Besides the great saving in terms of storage, this scheme also leads to significant speedup in matrix multiplication. While performing the matrix multiplication $\mathbf{S}\mathbf{z}_1$ directly requires $O(n^2)$ operations, performing it in the form $\mathbf{A}\mathbf{z}_1 - \lambda\mathbf{z}_1 - \frac{\mathbf{d}(\mathbf{d}^T\mathbf{z}_1)}{2m}$ instead requires only $O(m+n)$ operations. Therefore, the matrix multiplication $\mathbf{M}\mathbf{z} = ((\mathbf{S}\mathbf{z}_1)^T, \ldots, (\mathbf{S}\mathbf{z}_k)^T)^T$ requires only $O(k(m+n))$ operations and $O(m+kn)$ space.

Using the scheme above, we can use `quadprog` for M1 and `pcg` for M2 by passing function arguments for computing $\mathbf{M}\mathbf{z}$. Normally `quadprog` or `pcg` converges in $T(kn)$ steps as there are kn variables. In addition, our algorithm requires $O(\ln n)$ iterations. Therefore, the total time complexity is $O(T(kn)k(n+m)\ln n)$. This agrees with our empirical results showing that the running time is roughly linear to $O(kn)$ if we take $T(kn) = O(1)$ for these datasets because the number of steps needed varies in a small range (1 to 39)[5] and also $m = O(n)$, resulting in the overall time complexity of $O(kn\ln n)$. We note that the $\ln n$ factor is not easy to observe unless n is very large. With the restart option, around $l \leq k$ extra iterations are added and thus the overall time complexity increases to $O(kn(\ln n + k))$.

Moreover, we can speed up `quadprog` further by substituting the assigned values to give a sum of quadratic and linear terms similar to (10). Doing so reduces the number of variables from kn to $k|\widetilde{\mathcal{U}}|$. However, this requires deleting appropriate rows and columns in $\mathbf{S}$ (and hence $\mathbf{A}$) to obtain $\mathbf{S}'$. Deleting rows and columns in a sparse matrix can be slow if we use an ordinary row/column deletion method. To overcome this problem, we delete appropriate rows and columns from an identity matrix (which can be done efficiently) to construct a (sparse) matrix $\mathbf{B}$ so that $\mathbf{A}\mathbf{B}$ is equal to $\mathbf{A}$ with the appropriate columns deleted. Deletion of rows can be done in a similar way. This greatly reduces the time used to find $\mathbf{S}'$ and $\mathbf{F}$ in (10) and is very useful for both `quadprog` and `pcg`.

We search for the appropriate value of k by doubling it each time to rerun M2. We stop this procedure when $l < k$

[5]As $|\widetilde{\mathcal{U}}|$ gets smaller, `pcg` converges in much fewer steps. The number of steps ranges from 14 to 39 in the first `pcg` operation for all six networks, but it gradually decreases and needs only one or two steps to converge in the last `pcg` operation. For example, in the phy network, the initial numbers of steps are: 39, 29, 23, 23, 19, 16, 16, 14, 14, ...

or the improvement in modularity is very small. The stable number of clusters, denoted l', can be found when k is at most equal to $2l'$. In addition, the number of steps for pcg to converge is independent of k since the exact solution of the linear system for each cluster can be found using the conjugate gradient method in at most n steps. Hence $T(kn)$ can be reduced to $O(n)$.[6] So the total time complexity for finding l' is $O(n(n+m)\ln n \sum_{i=1}^{\ln l'+1} 2^i) = O(l'n(n+m)\ln n)$. If we assume $l' \approx \hat{k}$, $\hat{k} = O(\ln n)$ and $m = O(n)$ for social networks, the overall time complexity is $O(n^2 \ln^2 n)$. As for space complexity, the overall complexity is $O(m + kn) = O(n \ln n)$.

6 Conclusion

We have presented a new formulation which allows us to perform modularity maximization for tackling the general k-partition problem directly. Our formulation requires only kn variables in an IQP as compared to $O(n^2)$ variables in an ILP formulation. Based on a relaxation of the formulation, we have presented an iterative algorithm for tackling the problem. Our experiments give very encouraging results. Not only is the proposed method scalable in terms of both time and space complexity, but it also gives very competitive results with respect to the modularity measure. Moreover, our method provides a way to determine the number of communities in the network as long as the maximum number allowed in the formulation is no less than the actual number. This capability makes our method applicable under more general settings. Our future research will also consider the use of quality measures other than modularity for solving the k-partition problem under an optimization framework.

Acknowledgment

This research has been supported by General Research Fund 621310 from the Research Grants Council of Hong Kong.

References

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

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

[Clauset *et al.*, 2004] A. Clauset, M. E. J. Newman, and C. Moore. Finding community structure in very large networks. *Physical Review E*, 70(6):066111, Dec 2004.

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

[6] In practice, $T(kn)$ is found to be much smaller than $O(n)$ probably because the constant of proportionality is small and hence noticeable change can only be observed when the network size n is huge.

[Girvan and Newman, 2002] M. Girvan and M. E. J. Newman. Community structure in social and biological networks. *Proceedings of the National Academy of Sciences of the United States of America*, 99(12):7821–7826, June 2002.

[Gleiser and Danon, 2003] P. Gleiser and L. Danon. Community Structure in Jazz. *Advances in Complex Systems*, 6(4):565–573, 2003.

[Guimerà *et al.*, 2003] R. Guimerà, L. Danon, A. Díaz-Guilera, F. Giralt, and A. Arenas. Self-similar community structure in a network of human interactions. *Physical Review E*, 68(6):065103, Dec 2003.

[Jeong *et al.*, 2000] H. Jeong, B. Tombor, R. Albert, Z. N. Oltvai, and A. L. Barabàsi. The large-scale organization of metabolic networks. *Nature*, 407(6804):651–654, Oct 2000.

[Kernighan and Lin, 1970] B. Kernighan and S. Lin. An efficient heuristic procedure for partitioning graphs. *Bell System Technical Journal*, pages 291–307, 1970.

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

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

[Newman *et al.*, 2006] M. Newman, A.L. Barabási, and D.J. Watts, editors. *The Structure and Dynamics of Networks*. Princeton University Press, Princeton, NJ, USA, 2006.

[Newman, 2001] M. E. J. Newman. The structure of scientific collaboration networks. *Proceedings of the National Academy of Sciences of the United States of America*, 98(2):404–409, 2001.

[Newman, 2006] M. E. J. Newman. Modularity and community structure in networks. *Proceedings of the National Academy of Sciences of the United States of America*, 103(23):8577–8582, 6 2006.

[Noack and Rotta, 2009] A. Noack and R. Rotta. Multi-level algorithms for modularity clustering. In Jan Vahrenhold, editor, *SEA*, volume 5526 of *Lecture Notes in Computer Science*, pages 257–268. Springer, 2009.

[Shi and Malik, 1997] J. Shi and J. Malik. Normalized cuts and image segmentation. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 731–737, San Juan, Puerto Rico, 17–19 June 1997.

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