Additive Approximation Algorithms for Modularity Maximization*

Yasushi Kawase[1], Tomomi Matsui[2], and Atsushi Miyauchi[3]

1 **Tokyo Institute of Technology, Tokyo, Japan**
kawase.y.ab@m.titech.ac.jp
2 **Tokyo Institute of Technology, Tokyo, Japan**
matsui.t.af@m.titech.ac.jp
3 **Tokyo Institute of Technology, Tokyo, Japan**
miyauchi.a.aa@m.titech.ac.jp

Abstract

The modularity is a quality function in community detection, which was introduced by Newman and Girvan (2004). Community detection in graphs is now often conducted through modularity maximization: given an undirected graph $G = (V, E)$, we are asked to find a partition $\mathcal{C}$ of V that maximizes the modularity. Although numerous algorithms have been developed to date, most of them have no theoretical approximation guarantee. Recently, to overcome this issue, the design of modularity maximization algorithms with provable approximation guarantees has attracted significant attention in the computer science community.

In this study, we further investigate the approximability of modularity maximization. More specifically, we propose a polynomial-time $\left(\cos\left(\frac{3-\sqrt{5}}{4}\pi\right) - \frac{1+\sqrt{5}}{8}\right)$-additive approximation algorithm for the modularity maximization problem. Note here that $\cos\left(\frac{3-\sqrt{5}}{4}\pi\right) - \frac{1+\sqrt{5}}{8} < 0.42084$ holds. This improves the current best additive approximation error of 0.4672, which was recently provided by Dinh, Li, and Thai (2015). Interestingly, our analysis also demonstrates that the proposed algorithm obtains a nearly-optimal solution for any instance with a high modularity value. Moreover, we propose a polynomial-time 0.16598-additive approximation algorithm for the maximum modularity cut problem. It should be noted that this is the first non-trivial approximability result for the problem. Finally, we demonstrate that our approximation algorithm can be extended to some related problems.

1998 ACM Subject Classification G.2.2 [Graph Theory] Network Problems

Keywords and phrases networks, community detection, modularity maximization, approximation algorithms

Digital Object Identifier 10.4230/LIPIcs.ISAAC.2016.43

1 Introduction

Identifying community structure is a fundamental primitive in graph mining [11]. Roughly speaking, a *community* (also referred to as a *cluster* or *module*) in a graph is a subset of vertices densely connected with each other, but sparsely connected with the vertices outside the subset. Community detection in graphs is a powerful way to discover components that have some special roles or possess important functions. For example, consider the

* The first author is supported by a Grant-in-Aid for Young Scientists (B) (No. 16K16005). The third author is supported by a Grant-in-Aid for JSPS Fellows (No. 26-11908).

graph representing the World Wide Web, where vertices correspond to web pages and edges represent hyperlinks between pages. Communities in this graph are likely to be the sets of web pages dealing with the same or similar topics, or sometimes link spam [14].

To date, numerous community detection algorithms have been developed, most of which are designed to maximize a *quality function*. Quality functions in community detection return some value that represents the *community-degree* for a given partition of the set of vertices. The best known and widely used quality function is the *modularity*, which was introduced by Newman and Girvan [24]. Let $G = (V, E)$ be an undirected graph consisting of $n = |V|$ vertices and $m = |E|$ edges. The modularity, a quality function for a partition $\mathcal{C} = \{C_1, \ldots, C_k\}$ of V (i.e., $\bigcup_{i=1}^{k} C_i = V$ and $C_i \cap C_j = \emptyset$ for $i \neq j$), can be written as

$$Q(\mathcal{C}) = \sum_{C \in \mathcal{C}} \left(\frac{m_C}{m} - \left(\frac{D_C}{2m} \right)^2 \right),$$

where m_C represents the number of edges whose endpoints are both in C, and D_C represents the sum of degrees of the vertices in C. The modularity represents the sum, over all communities, of the fraction of the number of edges within communities minus the expected fraction of such edges assuming that they are placed at random with the same degree distribution.

Although the modularity is known to have some drawbacks (e.g., the *resolution limit* [12]), community detection is now often conducted through modularity maximization: given an undirected graph $G = (V, E)$, we are asked to find a partition $\mathcal{C}$ of V that maximizes the modularity. Note that the modularity maximization problem has no restriction on the number of communities in the output partition; thus, the algorithms are allowed to specify the best number of communities by themselves. Brandes et al. [5] proved that modularity maximization is NP-hard. A wide variety of applications (and this hardness result) have promoted the development of modularity maximization heuristics. In fact, there are numerous algorithms based on various techniques such as greedy procedure [4, 24], simulated annealing [17], spectral optimization [23], and mathematical programming [1, 20, 6]. Although some of them are known to perform well in practice, they have no theoretical approximation guarantee at all.

Recently, to overcome this issue, the design of modularity maximization algorithms with provable approximation guarantees has attracted significant attention in the computer science community. DasGupta and Desai [8] designed a polynomial-time ϵ-additive approximation algorithm[1] for dense graphs (i.e., graphs with $m = \Omega(n^2)$) using an algorithmic version of the regularity lemma [13], where $\epsilon > 0$ is an arbitrary constant. Moreover, Dinh, Li, and Thai [9] very recently developed a polynomial-time 0.4672-additive approximation algorithm. This is the first polynomial-time additive approximation algorithm with a non-trivial approximation guarantee (that is applicable to any instance).[2] Note that, to our knowledge, this is the current best additive approximation error. Their algorithm is based on the semidefinite programming (SDP) relaxation and the hyperplane separation technique.

---

[1] A feasible solution is *α-additive approximate* if its objective value is at least the optimal value minus α. An algorithm is called an *α-additive approximation algorithm* if it returns an α-additive approximate solution for any instance. For an α-additive approximation algorithm, α is referred to as an *additive approximation error* of the algorithm.

[2] A 1-additive approximation algorithm is trivial because $Q(\{V\}) = 0$ and $Q(\mathcal{C}) < 1$ for any partition $\mathcal{C}$.

1.1 Our Contribution

In this study, we further investigate the approximability of modularity maximization. Our contribution can be summarized as follows:

1. We propose a polynomial-time $\left(\cos\left(\frac{3-\sqrt{5}}{4}\pi\right) - \frac{1+\sqrt{5}}{8}\right)$-additive approximation algorithm for the modularity maximization problem. Note here that $\cos\left(\frac{3-\sqrt{5}}{4}\pi\right) - \frac{1+\sqrt{5}}{8} < 0.42084$ holds; thus, this improves the current best additive approximation error of 0.4672, which was recently provided by Dinh, Li, and Thai [9]. Interestingly, our analysis also demonstrates that the proposed algorithm obtains a nearly-optimal solution for any instance with a high modularity value.
2. We propose a polynomial-time 0.16598-additive approximation algorithm for the maximum modularity cut problem. It should be noted that this is the first non-trivial approximability result for the problem.
3. We demonstrate that our additive approximation algorithm for the modularity maximization problem can be extended to some related problems.

First result

Let us describe our first result in details. Our additive approximation algorithm is also based on the SDP relaxation and the hyperplane separation technique. However, our analysis is essentially different from the one by Dinh, Li, and Thai [9], and is much more effective for many practical instances.

The algorithm by Dinh, Li, and Thai [9] reduces the SDP relaxation for the modularity maximization problem to the one for MaxAgree problem arising in correlation clustering (e.g., see [2] or [7]) by adding an appropriate constant to the objective function. Then, the algorithm adopts the SDP-based 0.7664-approximation algorithm[3] for MaxAgree problem [7]. Specifically, their algorithm generates 2 and 3 random hyperplanes to obtain feasible solutions, and then returns the better one. Their analysis of the additive approximation guarantee depends heavily on the above reduction; the additive approximation error of 0.4672 is just derived from $2(1-\kappa)$, where κ represents the approximation ratio of the SDP-based algorithm for MaxAgree problem (i.e., $\kappa = 0.7664$). The analysis of the SDP-based algorithm for MaxAgree problem [7] aims at multiplicative approximation rather than additive one. As a result, the analysis by Dinh, Li, and Thai [9] has caused a gap in terms of additive approximation. In fact, as shown in our analysis, their algorithm already has the approximation error of $\cos\left(\frac{3-\sqrt{5}}{4}\pi\right) - \frac{1+\sqrt{5}}{8}$ (< 0.42084).

In contrast, our algorithm and analysis do not depend on such a reduction. In fact, our algorithm just solves the SDP relaxation for the modularity maximization problem without any transformation. Moreover, our algorithm employs a hyperplane separation procedure that extends the one used in their algorithm. Specifically, our algorithm chooses an appropriate number of hyperplanes using the information of the optimal solution to the SDP relaxation so that the lower bound on the expected modularity value is maximized. It should be emphasized that our analysis directly evaluates an additive approximation error of the proposed algorithm, unlike the analysis by Dinh, Li, and Thai [9]. As a result, our analysis improves their additive approximation error, and demonstrates that the proposed

[3] A feasible solution is *α-approximate* if its objective value is at least α times the optimal value. An algorithm is called an *α-approximation algorithm* if it returns an α-approximate solution for any instance. For an α-approximation algorithm, α is referred to as an *approximation ratio* of the algorithm.

algorithm has a much better lower bound on the expected modularity value for many practical instances. In particular, for any instance with optimal value close to 1 (a trivial upper bound), our algorithm obtains a nearly-optimal solution. Note here that, as reported in previous work [4, 6, 25], there are many large real-world networks that have a partition with a very high modularity value. At the end of our analysis, we summarize a lower bound on the expected modularity value with respect to the optimal value of a given instance.

Second result

Here we describe our second result in details. The modularity maximization problem has no restriction on the number of clusters in the output partition. On the other hand, there also exist a number of problem variants with such a restriction. The maximum modularity cut problem is a typical one, where given an undirected graph $G = (V, E)$, we are asked to find a partition $\mathcal{C}$ of V consisting of at most two components (i.e., a bipartition $\mathcal{C}$ of V) that maximizes the modularity. This problem appears in many contexts in community detection. For example, a few hierarchical divisive heuristics for the modularity maximization problem repeatedly solve this problem either exactly [6] or heuristically [1], to obtain a partition $\mathcal{C}$ of V. Brandes et al. [5] proved that the maximum modularity cut problem is NP-hard (even on dense graphs). More recently, DasGupta and Desai [8] showed that the problem is NP-hard even on d-regular graphs with any fixed $d \geq 9$. However, to our knowledge, there exists no approximability result for the problem.

Our additive approximation algorithm adopts the SDP relaxation and the hyperplane separation technique, which is identical to the subroutine of the hierarchical divisive heuristic proposed by Agarwal and Kempe [1]. Specifically, our algorithm first solves the SDP relaxation for the maximum modularity cut problem (rather than the modularity maximization problem), and then generates a random hyperplane to obtain a feasible solution for the problem. Although the computational experiments by Agarwal and Kempe [1] demonstrate that their hierarchical divisive heuristic maximizes the modularity quite well in practice, the approximation guarantee of the subroutine in terms of the maximum modularity cut was not analyzed. Our analysis shows that the proposed algorithm is a 0.16598-additive approximation algorithm for the maximum modularity cut problem. At the end of our analysis, we again present a lower bound on the expected modularity value with respect to the optimal value of a given instance. This reveals that for any instance with optimal value close to $1/2$ (a trivial upper bound in the case of bipartition), our algorithm obtains a nearly-optimal solution.

Third result

Finally, we describe our third result. We first extend our additive approximation algorithm for the modularity maximization problem to the well-known graph partitioning problem called the clique partitioning problem [16]. Then, we apply the result for the following three special cases of the clique partitioning problem: the weighted modularity maximization problem [22], the directed modularity maximization problem [19], and Barber's bipartite modularity maximization problem [3], all of which are NP-hard (see [8] and [21]).

1.2 Related Work

Multiplicative approximation algorithms

There also exist multiplicative approximation algorithms for modularity maximization. DasGupta and Desai [8] designed an $\Omega(1/\log d)$-approximation algorithm for the modularity

maximization problem on d-regular graphs with $d \leq \frac{n}{2\log n}$. They extended the approximation algorithm to the weighted modularity maximization problem. On the other hand, Dinh and Thai [10] developed algorithms for scale-free graphs with a prescribed degree sequence. In their graphs, the number of vertices with degree d is fixed to some value proportional to $d^{-\gamma}$, where $-\gamma$ is called the power-law exponent.

Inapproximability results

There are some inapproximability results for the modularity maximization problem. Das-Gupta and Desai [8] showed that it is NP-hard to obtain a $(1-\epsilon)$-approximate solution for some constant $\epsilon > 0$ (even for complements of 3-regular graphs). More recently, Dinh, Li, and Thai [9] proved a much stronger statement, that is, there exists no polynomial-time $(1-\epsilon)$-approximation algorithm for *any* $\epsilon > 0$, unless P = NP. It should be noted that these results are on multiplicative approximation rather than additive one. In fact, there exist no inapproximability results in terms of additive approximation for modularity maximization.

1.3 Preliminaries

Here we introduce definitions and notation used in this paper. Let $G = (V, E)$ be an undirected graph consisting of $n = |V|$ vertices and $m = |E|$ edges. Let $P = V \times V$. By simple calculation, as mentioned in Brandes et al. [5], the modularity can be rewritten as

$$Q(\mathcal{C}) = \frac{1}{2m} \sum_{(i,j)\in P} \left(A_{ij} - \frac{d_i d_j}{2m} \right) \delta(\mathcal{C}(i), \mathcal{C}(j)),$$

where A_{ij} is the (i,j) component of the adjacency matrix A of G, d_i is the degree of $i \in V$, $\mathcal{C}(i)$ is the (unique) community to which $i \in V$ belongs, and δ is the Kronecker symbol equal to 1 if two arguments are identical and 0 otherwise. This form is useful to write mathematical programming formulations for modularity maximization. For convenience, we define

$$q_{ij} = \frac{A_{ij}}{2m} - \frac{d_i d_j}{4m^2} \quad \text{for each } (i,j) \in P.$$

We can divide the set P into the following two disjoint subsets: $P_{\geq 0} = \{(i,j) \in P \mid q_{ij} \geq 0\}$ and $P_{<0} = \{(i,j) \in P \mid q_{ij} < 0\}$. Clearly, we have $\sum_{(i,j)\in P_{\geq 0}} q_{ij} + \sum_{(i,j)\in P_{<0}} q_{ij} = \sum_{(i,j)\in P} q_{ij} = 0$, and thus $\sum_{(i,j)\in P_{\geq 0}} q_{ij} = \sum_{(i,j)\in P_{<0}} -q_{ij}$. We denote this value by q, i.e., $q = \sum_{(i,j)\in P_{\geq 0}} q_{ij}$. Note that for any instance, we have $q < 1$.

1.4 Paper Organization

In Section 2, we revisit the SDP relaxation for the modularity maximization problem, and then describe an outline of our algorithm. In Section 3, the approximation guarantee of the proposed algorithm is carefully analyzed. In Section 4, we present our additive approximation algorithm for the maximum modularity cut problem. We mention the extension of our additive approximation algorithm to some related problems in Section 5. Due to space limitations, some proofs are omitted, which can be found in the full version [18].

2 Algorithm

The modularity maximization problem can be formulated as follows:

$$\text{max.} \sum_{(i,j)\in P} q_{ij} (\boldsymbol{y}_i \cdot \boldsymbol{y}_j) \quad \text{s.t.} \quad \boldsymbol{y}_i \in \{\boldsymbol{e}_1, \ldots, \boldsymbol{e}_n\} \ (\forall i \in V),$$

Algorithm 1 Hyperplane(k)

Input: Graph $G = (V, E)$
Output: Partition $\mathcal{C}$ of V
1: Obtain an optimal solution $X^* = (x^*_{ij})$ to SDP
2: Generate k random hyperplanes and obtain a partition $\mathcal{C}_k = \{C_1, \ldots, C_{2^k}\}$ of V
3: **return** $\mathcal{C}_k$

where $\boldsymbol{e}_k$ $(1 \leq k \leq n)$ represents the vector that has 1 in the kth coordinate and 0 elsewhere. We denote by OPT the optimal value of this original problem. Note that for any instance, we have $\mathsf{OPT} \in [0, 1)$. We introduce the following semidefinite relaxation problem:

$$\mathsf{SDP}: \ \max. \sum_{(i,j)\in P} q_{ij}x_{ij} \quad \text{s.t.} \ \ x_{ii} = 1 \ (\forall i \in V), \quad x_{ij} \geq 0 \ (\forall i, j \in V), \quad X = (x_{ij}) \in \mathcal{S}^n_+,$$

where $\mathcal{S}^n_+$ represents the cone of $n \times n$ symmetric positive semidefinite matrices. It is easy to see that every feasible solution $X = (x_{ij})$ of SDP satisfies $x_{ij} \leq 1$ for any $(i, j) \in P$. Although the algorithm by Dinh, Li, and Thai [9] reduces SDP to the one for MaxAgree problem by adding an appropriate constant to the objective function, our algorithm just solves SDP without any transformation. Let $X^* = (x^*_{ij})$ be an optimal solution to SDP, which can be computed (with an arbitrarily small error) in time polynomial in n and m. Using the optimal solution X^*, we define the following two values:

$$z^*_+ = \frac{1}{q} \sum_{(i,j)\in P_{\geq 0}} q_{ij}x^*_{ij} \quad \text{and} \quad z^*_- = \frac{1}{q} \sum_{(i,j)\in P_{<0}} q_{ij}x^*_{ij},$$

both of which are useful in the analysis of the approximation guarantee of our algorithm. Clearly, we have $0 \leq z^*_+ \leq 1$ and $-1 \leq z^*_- \leq 0$.

We apply the hyperplane separation technique to obtain a feasible solution of the modularity maximization problem. Specifically, we consider the following general procedure: generate k random hyperplanes to separate the vectors corresponding to the optimal solution X^*, and then obtain a partition $\mathcal{C}_k = \{C_1, \ldots, C_{2^k}\}$ of V. For reference, the procedure is described in Algorithm 1. Note here that at this time, we have not yet mentioned how to determine the number k of hyperplanes we generate. As revealed in our analysis, we can choose an appropriate number of hyperplanes using the value of z^*_+ so that the lower bound on the expected modularity value of the output of Hyperplane(k) is maximized.

3 Analysis

In this section, we first analyze an additive approximation error of Hyperplane(k) for each positive integer $k \in \mathbb{Z}_{>0}$. Then, we provide an appropriate number $k^* \in \mathbb{Z}_{>0}$ of hyperplanes, which completes the design of our algorithm. Finally, we present a lower bound on the expected modularity value of the output of Hyperplane(k^*) with respect to the value of OPT.

When k random hyperplanes are generated independently, the probability that two vertices $i, j \in V$ are in the same cluster is given by $\left(1 - \arccos(x^*_{ij})/\pi\right)^k$, as mentioned in previous works (e.g., see [7] or [15]). For simplicity, we define the function $f_k(x) = (1 - \arccos(x)/\pi)^k$ for $x \in [0, 1]$. Here we present the lower convex envelope of each of $f_k(x)$ and $-f_k(x)$.

▶ **Lemma 1.** *For any positive integer k, the lower convex envelope of $f_k(x)$ is given by $f_k(x)$ itself, and the lower convex envelope of $-f_k(x)$ is given by the linear function $h_k(x) = -1/2^k + (1/2^k - 1)x$ for $x \in [0, 1]$.*

The following lemma lower bounds the expected modularity value of the output of Hyperplane(k).

▶ **Lemma 2.** *Let $\mathcal{C}_k$ be the output of* Hyperplane(k)*. For any positive integer k, it holds that*

$$\mathrm{E}[Q(\mathcal{C}_k)] \geq q\left(f_k(z_+^*) + h_k(-z_-^*)\right).$$

Proof. Recall that $\mathcal{C}_k(i)$ for each $i \in V$ denotes the (unique) cluster in $\mathcal{C}_k$ that includes the vertex i. Note here that $\delta(\mathcal{C}_k(i), \mathcal{C}_k(j))$ for each $(i,j) \in P$ is a random variable, which takes 1 with probability $f_k(x_{ij}^*)$ and 0 with probability $1 - f_k(x_{ij}^*)$. The expectation $\mathrm{E}[Q(\mathcal{C}_k)]$ can be transformed as follows:

$$\begin{aligned}\mathrm{E}[Q(\mathcal{C}_k)] &= \mathrm{E}\left[\sum_{(i,j)\in P} q_{ij}\delta(\mathcal{C}_k(i), \mathcal{C}_k(j))\right] \\ &= \sum_{(i,j)\in P} q_{ij} f_k(x_{ij}^*) = \sum_{(i,j)\in P_{\geq 0}} q_{ij} f_k(x_{ij}^*) + \sum_{(i,j)\in P_{<0}} -q_{ij}\cdot(-f_k(x_{ij}^*)).\end{aligned}$$

Using Lemma 1, we have

$$\begin{aligned}\mathrm{E}[Q(\mathcal{C}_k)] &\geq \sum_{(i,j)\in P_{\geq 0}} q_{ij} f_k(x_{ij}^*) + \sum_{(i,j)\in P_{<0}} -q_{ij} h_k(x_{ij}^*) \\ &= q\left(\sum_{(i,j)\in P_{\geq 0}} \left(\frac{q_{ij}}{q}\right) f_k(x_{ij}^*) + \sum_{(i,j)\in P_{<0}} \left(\frac{-q_{ij}}{q}\right) h_k(x_{ij}^*)\right) \\ &\geq q\left(f_k\left(\sum_{(i,j)\in P_{\geq 0}} \left(\frac{q_{ij}}{q}\right) x_{ij}^*\right) + h_k\left(\sum_{(i,j)\in P_{<0}} \left(\frac{-q_{ij}}{q}\right) x_{ij}^*\right)\right) \\ &= q\left(f_k(z_+^*) + h_k(-z_-^*)\right),\end{aligned}$$

where the last inequality follows from Jensen's inequality. ◀

The following lemma provides an additive approximation error of Hyperplane(k) by evaluating the above lower bound on $\mathrm{E}[Q(\mathcal{C}_k)]$ using the value of OPT.

▶ **Lemma 3.** *For any positive integer k, it holds that*

$$\mathrm{E}[Q(\mathcal{C}_k)] \geq \mathsf{OPT} - q\left(z_+^* - f_k(z_+^*) + \frac{1}{2^k}\right).$$

Proof. Clearly, (z_+^*, z_-^*) satisfies $q(z_+^* + z_-^*) \geq \mathsf{OPT}$ and $z_-^* \leq 0$. Thus, we obtain

$$\begin{aligned}q\left(f_k(z_+^*) + h_k(-z_-^*)\right) &\geq \left(\mathsf{OPT} - q(z_+^* + z_-^*)\right) + q\left(f_k(z_+^*) + h_k(-z_-^*)\right) \\ &= \left(\mathsf{OPT} - q(z_+^* + z_-^*)\right) + q\left(f_k(z_+^*) - 1/2^k + (1/2^k - 1)(-z_-^*)\right) \\ &= \mathsf{OPT} - q\left(z_+^* - f_k(z_+^*) + 1/2^k + (1/2^k)z_-^*\right) \\ &\geq \mathsf{OPT} - q\left(z_+^* - f_k(z_+^*) + 1/2^k\right).\end{aligned}$$

Combining this with Lemma 2, we have $\mathrm{E}[Q(\mathcal{C}_k)] \geq q\left(f_k(z_+^*) + h_k(-z_-^*)\right) \geq \mathsf{OPT} - q\left(z_+^* - f_k(z_+^*) + 1/2^k\right)$, as desired. ◀

For simplicity, we define the function $g_k(x) = x - f_k(x) + 1/2^k$ for $x \in [0,1]$. Then, the inequality of the above lemma can be rewritten as

$$\mathrm{E}[Q(\mathcal{C}_k)] \geq \mathsf{OPT} - q \cdot g_k(z_+^*).$$

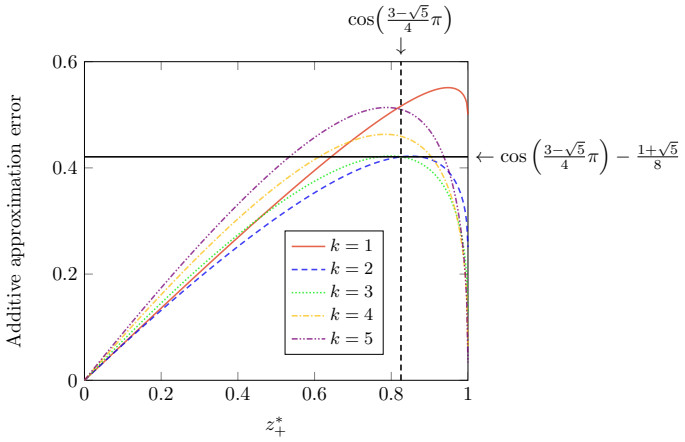

Figure 1 A brief illustration of the additive approximation error of Hyperplane(k) with respect to the value of z_+^*. For simplicity, we replace q by its upper bound 1. Specifically, the function $g_k(x) = x - f_k(x) + 1/2^k$ for $x \in [0,1]$ is plotted for $k = 1, 2, 3, 4$, and 5, as examples. The point $\left(\cos\left(\frac{3-\sqrt{5}}{4}\pi\right), \cos\left(\frac{3-\sqrt{5}}{4}\pi\right) - \frac{1+\sqrt{5}}{8}\right)$ is an intersection of the functions $g_2(x)$ and $g_3(x)$.

Figure 1 plots the above additive approximation error of Hyperplane(k) with respect to the value of z_+^*.

As can be seen, the appropriate number of hyperplanes (i.e., the number of hyperplanes that minimizes the additive approximation error) depends on the value of z_+^*. Intuitively, we wish to choose k^{**} that satisfies $k^{**} \in \arg\min_{k \in \mathbb{Z}_{>0}} g_k(z_+^*)$. However, it is not clear whether Hyperplane(k^{**}) runs in polynomial time. In fact, the number k^{**} becomes infinity if the value of z_+^* approaches 1. Therefore, alternatively, our algorithm chooses

$$k^* \in \operatorname*{arg\,min}_{k \in \{1,\ldots,\max\{3,\lceil \log_2 n \rceil\}\}} g_k(z_+^*).$$

First, we analyze the worst-case performance of Hyperplane(k^*). The following lemma says that (i) the worst-case performance of Hyperplane(k^*) is exactly the same as that of Hyperplane(k^{**}); and moreover (ii) to achieve the same worst-case performance as that of Hyperplane(k^{**}), it suffices to choose k from the set $\{2, 3\}$.

▶ **Lemma 4.** *Let S be one of the sets $\{2,3\}$, $\{1,\ldots,\max\{3,\lceil \log_2 n \rceil\}\}$, and $\mathbb{Z}_{>0}$. It holds that*

$$\max_{x \in [0,1]} \min_{k \in S} g_k(x) = \cos\left(\frac{3-\sqrt{5}}{4}\pi\right) - \frac{1+\sqrt{5}}{8}.$$

▶ Remark. Here we consider the algorithm that executes Hyperplane(2) and Hyperplane(3), and then returns the better solution. Note that this algorithm is essentially the same as that proposed by Dinh, Li, and Thai [9]. The above lemma implies that the algorithm by Dinh, Li, and Thai [9] already has the worst-case performance exactly the same as that of Hyperplane(k^*) (and Hyperplane(k^{**})). However, as shown below, Hyperplane(k^*) has a much better lower bound on the expected modularity value for many instances.

Finally, we present a lower bound on the expected modularity value of the output of Hyperplane(k^*) with respect to the value of OPT (rather than z_+^*). The following lemma is useful to show that the lower bound on the expected modularity value with respect to the value of OPT is not affected by the change from k^{**} to k^*.

▶ **Lemma 5.** *For any* $k' \in \arg\min_{k\in\mathbb{Z}_{>0}} g_k(\mathsf{OPT})$*, it holds that* $k' \leq \max\{3, \lceil \log_2 n \rceil\}$.

We are now ready to prove the main result of this paper.

▶ **Theorem 6.** *Let* $\mathcal{C}_{k^*}$ *be the output of* $\mathsf{Hyperplane}(k^*)$*. It holds that*

$$\mathrm{E}[Q(\mathcal{C}_{k^*})] \geq \mathsf{OPT} - q\left(\cos\left(\frac{3-\sqrt{5}}{4}\pi\right) - \frac{1+\sqrt{5}}{8}\right).$$

In particular, if $\mathsf{OPT} \geq \cos\left(\frac{3-\sqrt{5}}{4}\pi\right)$ *holds, then* $\mathrm{E}[Q(\mathcal{C}_{k^*})] > \mathsf{OPT} - q\min_{k\in\mathbb{Z}_{>0}} g_k(\mathsf{OPT})$. *Note here that* $q < 1$ *and* $\cos\left(\frac{3-\sqrt{5}}{4}\pi\right) - \frac{1+\sqrt{5}}{8} < 0.42084$.

Proof. From Lemmas 3 and 4, it follows directly that

$$\mathrm{E}[Q(\mathcal{C}_{k^*})] \geq \mathsf{OPT} - q\left(\cos\left(\frac{3-\sqrt{5}}{4}\pi\right) - \frac{1+\sqrt{5}}{8}\right).$$

Here we prove the remaining part of the theorem. Assume that $\mathsf{OPT} \geq \cos\left(\frac{3-\sqrt{5}}{4}\pi\right)$ holds. By simple calculation, for any $k \in \mathbb{Z}_{>0}$, we have $g_k''(x) < 0$ for $x \in (0,1)$. This means that for any $k \in \mathbb{Z}_{>0}$, the function $g_k(x)$ is strictly concave, and moreover, so is the function $\min_{k\in\{1,\ldots,\max\{3,\lceil\log_2 n\rceil\}\}} g_k(x)$. From Lemma 4 and the fact that $\min_{k\in\{1,\ldots,\max\{3,\lceil\log_2 n\rceil\}\}} g_k\left(\cos\left(\frac{3-\sqrt{5}}{4}\pi\right)\right) = \cos\left(\frac{3-\sqrt{5}}{4}\pi\right) - \frac{1+\sqrt{5}}{8}$ holds, we see that the function $\min_{k\in\{1,\ldots,\max\{3,\lceil\log_2 n\rceil\}\}} g_k(x)$ attains its maximum at $x = \cos\left(\frac{3-\sqrt{5}}{4}\pi\right)$. Thus, the function $\min_{k\in\{1,\ldots,\max\{3,\lceil\log_2 n\rceil\}\}} g_k(x)$ is strictly monotonically decreasing over the interval $\left[\cos\left(\frac{3-\sqrt{5}}{4}\pi\right), 1\right]$. Therefore, we have

$$\begin{aligned}\mathrm{E}[Q(\mathcal{C}_{k^*})] &\geq \mathsf{OPT} - q\min_{k\in\{1,\ldots,\max\{3,\lceil\log_2 n\rceil\}\}} g_k(z_+^*)\\ &> \mathsf{OPT} - q\min_{k\in\{1,\ldots,\max\{3,\lceil\log_2 n\rceil\}\}} g_k(\mathsf{OPT}) = \mathsf{OPT} - q\min_{k\in\mathbb{Z}_{>0}} g_k(\mathsf{OPT}),\end{aligned}$$

where the second inequality follows from $z_+^* \geq \mathsf{OPT}/q > \mathsf{OPT}$ and the last equality follows from Lemma 5. ◀

Figure 2 depicts the above lower bound on $\mathrm{E}[Q(\mathcal{C}_{k^*})]$. As can be seen, if OPT is close to 1, then $\mathsf{Hyperplane}(k^*)$ obtains a nearly-optimal solution. For example, for any instance with $\mathsf{OPT} \geq 0.99990$, it holds that $\mathrm{E}[Q(\mathcal{C}_{k^*})] > 0.96109$, i.e., the additive approximation error is less than 0.03891. For such instances, the reduction-based analysis by Dinh, Li, and Thai [9] provides no guarantee better than the worst-case performance.

▶ Remark. The additive approximation error of $\mathsf{Hyperplane}(k^*)$ depends on the value of $q < 1$; the less the value of q, the better the additive approximation error. Thus, it is interesting to find some graphs that have a small value of q. For instance, for any regular graph G that satisfies $m = \frac{\alpha}{2}n^2$, it holds that $q = 1 - \alpha$, where α is an arbitrary constant in $(0,1)$. Here we prove the statement. Since G is regular, we have $d_i = 2m/n = \alpha n$ for any $i \in V$. Hence, for any $\{i,j\} \in E$, it holds that $q_{ij} = \frac{A_{ij}}{2m} - \frac{d_i d_j}{4m^2} = \frac{1}{\alpha n^2} - \frac{1}{n^2} > 0$. Therefore, we have

$$q = \sum_{(i,j)\in P_{\geq 0}} q_{ij} = 2\sum_{\{i,j\}\in E} q_{ij} = 2m\left(\frac{1}{\alpha n^2} - \frac{1}{n^2}\right) = 1 - \alpha.$$

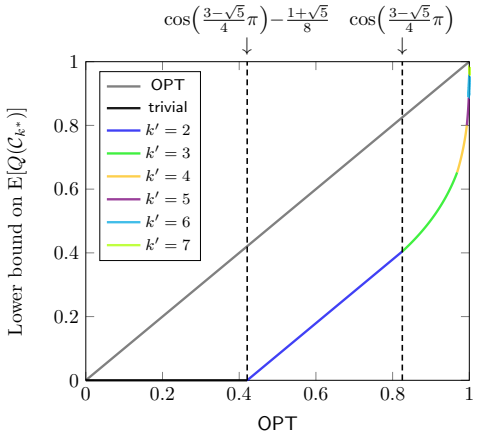

Figure 2 A brief illustration of the lower bound on the expected modularity value of the output of Hyperplane(k^*) with respect to the value of OPT. For simplicity, we replace q by its upper bound 1. Note that $k' \in \arg\min_{k\in\mathbb{Z}_{>0}} g_k(\mathsf{OPT})$.

Algorithm 2 Modularity Cut

Input: Graph $G=(V,E)$
Output: Bipartition $\mathcal{C}$ of V
1: Obtain an optimal solution $X^*=(x^*_{ij})$ to $\mathsf{SDP}_{\mathsf{cut}}$
2: Generate a random hyperplane and obtain a bipartition $\mathcal{C}_{\mathrm{out}}=\{C_1,C_2\}$ of V
3: **return** $\mathcal{C}_{\mathrm{out}}$

4 Maximum Modularity Cut

The maximum modularity cut problem can be formulated as follows:

$$\text{max.} \quad \frac{1}{2}\sum_{(i,j)\in P} q_{ij}(y_iy_j+1) \quad \text{s.t.} \quad y_i\in\{-1,1\}\ (\forall i\in V).$$

We denote by $\mathsf{OPT}_{\mathsf{cut}}$ the optimal value of this original problem. Note that for any instance, it holds that $\mathsf{OPT}_{\mathsf{cut}}\in[0,1/2]$, as shown in DasGupta and Desai [8]. We introduce the following semidefinite relaxation problem:

$$\mathsf{SDP}_{\mathsf{cut}}: \ \text{max.} \quad \frac{1}{2}\sum_{(i,j)\in P} q_{ij}(x_{ij}+1) \quad \text{s.t.} \quad x_{ii}=1\ (\forall i\in V), \quad X=(x_{ij})\in\mathcal{S}^n_+,$$

where recall that $\mathcal{S}^n_+$ represents the cone of $n\times n$ symmetric positive semidefinite matrices. Let $X^*=(x^*_{ij})$ be an optimal solution to $\mathsf{SDP}_{\mathsf{cut}}$, which can be computed (with an arbitrarily small error) in time polynomial in n and m. Note here that x^*_{ij} may be negative for $(i,j)\in P$ with $i\neq j$, unlike SDP in the previous section.

We generate a random hyperplane to separate the vectors corresponding to the optimal solution X^*, and then obtain a bipartition $\mathcal{C}=\{C_1,C_2\}$ of V. For reference, the procedure is described in Algorithm 2. As mentioned above, this algorithm is identical to the subroutine of the hierarchical divisive heuristic for the modularity maximization problem, which was proposed by Agarwal and Kempe [1].

The main result of this section is the following theorem.

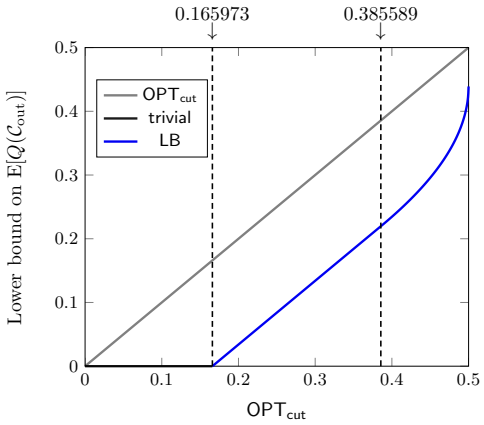

Figure 3 An illustration of the lower bound on the expected modularity value of the output of Algorithm 2.

▶ **Theorem 7.** *Let* $\alpha = \min_{-1<x<1} \frac{1-\arccos(x)/\pi}{(x+1)/2}$ $(\simeq 0.878567)$. *Let* $\mathcal{C}_{\mathrm{out}}$ *be the output of Algorithm 2. It holds that*

$$\mathrm{E}[Q(\mathcal{C}_{\mathrm{out}})] > \mathsf{OPT}_{\mathsf{cut}} - 0.16598.$$

In particular, if $\mathsf{OPT}_{\mathsf{cut}} \geq \frac{\sqrt{\pi^2-4}}{2\pi}$ $(\simeq 0.385589)$ *holds, then* $\mathrm{E}[Q(\mathcal{C}_{\mathrm{out}})] \geq \frac{\alpha}{2} - \frac{\arccos(2\cdot\mathsf{OPT}_{\mathsf{cut}})}{\pi}$.

Figure 3 depicts the above lower bound on $\mathrm{E}[Q(\mathcal{C}_{\mathrm{out}})]$. As can be seen, if $\mathsf{OPT}_{\mathsf{cut}}$ is close to 1/2, then Algorithm 2 obtains a nearly-optimal solution.

5 Extension

We first extend our additive approximation algorithm for the modularity maximization problem to the clique partitioning problem. In the clique partitioning problem, we are given a finite set V and a weight function $c : V \times V \to \mathbb{R}$. Let $P = V \times V$. The aim is to find a partition $\mathcal{C}$ of V that maximizes the sum of weights of the pairs within the same clusters, i.e.,

$$Q_{\mathsf{CPP}}(\mathcal{C}) = \sum_{(i,j)\in P} c_{ij}\delta(\mathcal{C}(i), \mathcal{C}(j)).$$

Although our definition is slightly different from the traditional one (see e.g., [16]), it remains essentially the same. Clearly, the modularity maximization problem can be reduced to the clique partitioning problem. In fact, it suffices to set $c_{ij} = q_{ij}$ for each $(i, j) \in P$.

We can extend $\mathsf{Hyperplane}(k)$ to the clique partitioning problem by replacing q_{ij} with c_{ij} in the description of the algorithm. Furthermore, we can also extend our analysis of the additive approximation error of $\mathsf{Hyperplane}(k)$. Note here that we should redefine

$$z^*_+ = \frac{1}{c_+} \sum_{(i,j)\in P_{\geq 0}} c_{ij}x^*_{ij} \quad \text{and} \quad z^*_- = \frac{1}{c_-} \sum_{(i,j)\in P_{<0}} c_{ij}x^*_{ij},$$

where $c_+ = \sum_{(i,j)\in P_{\geq 0}} c_{ij}$ and $c_- = \sum_{(i,j)\in P_{<0}} c_{ij}$. This is due to the fact that $c_+ = c_-$ does not necessarily hold. For the clique partitioning problem, we have the following key lemma, which is a generalization of Lemma 3.

▶ **Lemma 8.** *Let $\mathcal{C}_k$ be the output of* Hyperplane(k)*. It holds that*

$$\mathrm{E}[Q_{\mathsf{CPP}}(\mathcal{C}_k)] \geq \mathsf{OPT} - \left(c_+ z_+^* - c_+ f_k(z_+^*) + c_- \frac{1}{2^k}\right).$$

Using this lemma, we can choose an appropriate number k^* of hyperplanes and analyze the additive approximation error of Hyperplane(k^*). Note that if $c_+ = c_-$ holds, then we obtain the additive approximation error of $c_+ \left(\cos\left(\frac{3-\sqrt{5}}{4}\pi\right) - \frac{1+\sqrt{5}}{8}\right)$.

Finally, we mention the results for the following three problems: the weighted modularity maximization problem [22], the directed modularity maximization problem [19], and Barber's bipartite modularity maximization problem [3]. These problems are all special cases of the clique partitioning problem, where $c_+ = c_- < 1$ holds. For the detailed description of the above problems, see the full version [18]. We have the following corollary.

▶ **Corollary 9.** *There exist polynomial-time $\left(\cos\left(\frac{3-\sqrt{5}}{4}\pi\right) - \frac{1+\sqrt{5}}{8}\right)$-additive approximation algorithms for the weighted modularity maximization problem, the directed modularity maximization problem, and Barber's bipartite modularity maximization problem.*

References

1 G. Agarwal and D. Kempe. Modularity-maximizing graph communities via mathematical programming. *Eur. Phys. J. B*, 66(3):409–418, 2008.

2 N. Bansal, A. Blum, and S. Chawla. Correlation clustering. *Mach. Learn.*, 56(1-3):89–113, 2004.

3 M. J. Barber. Modularity and community detection in bipartite networks. *Phys. Rev. E*, 76:066102, 2007.

4 V. D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre. Fast unfolding of communities in large networks. *J. Stat. Mech.: Theory Exp.*, 2008:P10008, 2008.

5 U. Brandes, D. Delling, M. Gaertler, R. Gorke, M. Hoefer, Z. Nikoloski, and D. Wagner. On modularity clustering. *IEEE Trans. Knowl. Data Eng.*, 20(2):172–188, 2008.

6 S. Cafieri, A. Costa, and P. Hansen. Reformulation of a model for hierarchical divisive graph modularity maximization. *Ann. Oper. Res.*, 222(1):213–226, 2014.

7 M. Charikar, V. Guruswami, and A. Wirth. Clustering with qualitative information. *J. Comput. Syst. Sci.*, 71(3):360–383, 2005.

8 B. DasGupta and D. Desai. On the complexity of Newman's community finding approach for biological and social networks. *J. Comput. Syst. Sci.*, 79(1):50–67, 2013.

9 T. N. Dinh, X. Li, and M. T. Thai. Network clustering via maximizing modularity: Approximation algorithms and theoretical limits. In *ICDM'15*, pages 101–110, 2015.

10 T. N. Dinh and M. T. Thai. Community detection in scale-free networks: Approximation algorithms for maximizing modularity. *IEEE J. Sel. Areas Commun.*, 31(6):997–1006, 2013.

11 S. Fortunato. Community detection in graphs. *Phys. Rep.*, 486(3):75–174, 2010.

12 S. Fortunato and M. Barthélemy. Resolution limit in community detection. *Proc. Natl. Acad. Sci. U.S.A.*, 104(1):36–41, 2007.

13 A. Frieze and R. Kannan. The regularity lemma and approximation schemes for dense problems. In *FOCS'96*, pages 12–20, 1996.

14 D. Gibson, R. Kumar, and A. Tomkins. Discovering large dense subgraphs in massive graphs. In *VLDB'05*, pages 721–732, 2005.

15 M. X. Goemans and D. P. Williamson. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *J. ACM*, 42(6):1115–1145, 1995.

16 M. Grötschel and Y. Wakabayashi. A cutting plane algorithm for a clustering problem. *Math. Program.*, 45(1-3):59–96, 1989.

17 R. Guimerà and L. A. N. Amaral. Functional cartography of complex metabolic networks. *Nature*, 433(7028):895–900, 2005.

18 Yasushi Kawase, Tomomi Matsui, and Atsushi Miyauchi. Additive approximation algorithms for modularity maximization. *arXiv:1601.03316*, 2016.

19 E. A. Leicht and M. E. J. Newman. Community structure in directed networks. *Phys. Rev. Lett.*, 100:118703, 2008.

20 A. Miyauchi and Y. Miyamoto. Computing an upper bound of modularity. *Eur. Phys. J. B*, 86:302, 2013.

21 A. Miyauchi and N. Sukegawa. Maximizing Barber's bipartite modularity is also hard. *Optim. Lett.*, 9(5):897–913, 2015.

22 M. E. J. Newman. Analysis of weighted networks. *Phys. Rev. E*, 70:056131, 2004.

23 M. E. J. Newman. Modularity and community structure in networks. *Proc. Natl. Acad. Sci. U.S.A.*, 103(23):8577–8582, 2006.

24 M. E. J. Newman and M. Girvan. Finding and evaluating community structure in networks. *Phys. Rev. E*, 69:026113, 2004.

25 L. Waltman and N. J. van Eck. A smart local moving algorithm for large-scale modularity-based community detection. *Eur. Phys. J. B*, 86(11):1–14, 2013.