The Parameterised Complexity of Computing the Maximum Modularity of a Graph

Kitty Meeks[1]
School of Computing Science, University of Glasgow, Glasgow, UK
kitty.meeks@glasgow.ac.uk

Fiona Skerman[2]
Department of Mathematics, Uppsala University, Uppsala, Sweden
fiona.skerman@math.uu.se

Abstract

The *maximum modularity* of a graph is a parameter widely used to describe the level of clustering or community structure in a network. Determining the maximum modularity of a graph is known to be NP-complete in general, and in practice a range of heuristics are used to construct partitions of the vertex-set which give lower bounds on the maximum modularity but without any guarantee on how close these bounds are to the true maximum. In this paper we investigate the parameterised complexity of determining the maximum modularity with respect to various standard structural parameterisations of the input graph G. We show that the problem belongs to FPT when parameterised by the size of a minimum vertex cover for G, and is solvable in polynomial time whenever the treewidth or max leaf number of G is bounded by some fixed constant; we also obtain an FPT algorithm, parameterised by treewidth, to compute any constant-factor approximation to the maximum modularity. On the other hand we show that the problem is W[1]-hard (and hence unlikely to admit an FPT algorithm) when parameterised simultaneously by pathwidth and the size of a minimum feedback vertex set.

2012 ACM Subject Classification Mathematics of computing → Graph algorithms, Theory of computation → Parameterized complexity and exact algorithms

Keywords and phrases modularity, community detection, integer quadratic programming, vertex cover, pathwidth

Digital Object Identifier 10.4230/LIPIcs.IPEC.2018.9

Acknowledgements The authors are grateful to Jessica Enright for some helpful initial discussions about the topic.

1 Introduction

The increasing availability of large network datasets has led to great interest in techniques to discover network structure. An important and frequently observed structure in networks is the existence of groups of vertices with many connections between them, often referred to as ‘communities’.

Newman and Girvan introduced the modularity function in 2004 [22]. Modularity gives a measure of how well a graph can be divided into communities and is used in the most

[1] KM is supported by a Royal Society of Edinburgh Personal Research Fellowship, funded by the Scottish Government.

[2] FS is supported by grants from Swedish Research Council and the Ragnar Söderberg Foundation. Research partially conducted while at the Heilbronn Institute for Mathematical Research, Bristol.

popular algorithms to cluster large networks. For example, the Louvain method, an iterative clustering technique, uses the modularity function to choose which parts from the previous step to fuse into larger parts at each step [2, 15]. The widespread use of modularity and empirical success in finding communities makes modularity an important function to study from an algorithmic point of view.

In this paper we are concerned with the computational complexity of computing the maximum modularity of a given input graph, and specifically in the following decision problem.

Modularity
Input: A graph G and a constant $q \in [0, 1]$.
Question: Is the maximum modularity of G at least q?

This problem was shown to be NP-complete in general by Brandes et. al. [3], using a construction that relies on the fact that all vertices of a sufficiently large clique must be assigned to the same part of an optimal partition. They also showed that a variation of the problem in which we wish to find the optimal partition into exactly two sets is hard; their proof for this relied again on the use of large cliques, but DasGupta and Desai [5] later showed that this 2-clustering problem remains NP-complete on d-regular graphs for any fixed $d \geq 9$. It has also been shown that it is NP-hard to approximate the maximum modularity within any constant factor [7], although there is a polynomial-time constant-factor approximation algorithm for certain families of scale-free networks [9]. The hardness of computing constant-factor multiplicative approximations in general has motivated research into approximation algorithms with an additive error [7, 16]: the best known result is an approximation algorithm with additive error roughly 0.42084 [16].

In this paper we initiate the study of the parameterised complexity of Modularity, considering its complexity with respect to several standard structural parameterisations. On the positive side, we show that the problem is in FPT when parameterised by the cardinality of a minimum vertex cover for the input graph G, and that it belongs to XP when parameterised by either the treewidth or max leaf number of G. The XP algorithm parameterised by treewidth can easily be adapted to give an FPT algorithm, parameterised by treewidth, to compute any constant-factor approximation maximum modularity. On the other hand, we demonstrate that Modularity, parameterised by treewidth, is unlikely to belong to FPT: we prove that the problem is W[1]-hard even when parameterised simultaneously by the pathwidth of G and the size of a minimum feedback vertex set for G. For background on parameterised complexity, and the complexity classes discussed here, we refer the reader to [4, 10].

These results follow the same pattern as those obtained for the problem Equitable Connected Partition [11], and indeed our hardness result involves a reduction from a specialisation of this problem. There are clear similarities between the two problems: in a partition that maximises the modularity, every part will induce a connected subgraph and, in certain circumstances, we achieve the maximum modularity with a partition into parts that are as equal as possible. However, the crucial difference between the two problems is that the input to Equitable Connected Partition includes the required number of parts, whereas Modularity requires us to maximise over all possible partition sizes; in fact, if we restrict to partitions with a specified parts, it is no longer necessarily true that a partition maximising the modularity must induce connected subgraphs. This difference makes reductions between the two problems non-trivial.

1.1 The modularity function

The definition of modularity was first introduced by Newman and Girvan in [22]. Many or indeed most popular algorithms used to search for clusterings on large datasets are based on finding partitions with high modularity [17, 14], and the heuristics within them sometimes also use local modularity optimisation, for example in the Louvain method [2]. See [23, 13] for surveys on community detection including modularity based methods.

Knowledge on the maximum modularity for classes of graphs helps to understand the behaviour of the modularity function. There is a growing literature on this which began with cycles and complete graphs in [3]. Bagrow [1] and Montgolfier et al. [6] showed some classes of trees have high maximum modularity which was extended in [19] to all trees with maximum degree $o(n)$, and furthermore to all graphs where the product of treewidth and maximum degree grows more slowly than the number of edges. Many random graph models also have high modularity, see [20, 21] for a treatment of Erdős-Renyi random graphs, [19] for random regular graphs and also [24] which includes the preferential attachment model.

Given a set A of vertices, let $e(A)$ denote the number of edges within A, and let $\mathrm{vol}(A)$ (sometimes called the volume of A) denote the sum of the degree d_v (in the whole graph G) over the vertices v in A. For a graph G with $m \geq 1$ edges and a vertex partition $\mathcal{A}$ of G, set the modularity of $\mathcal{A}$ on G to be

$$q_{\mathcal{A}}(G) = \frac{1}{2m} \sum_{\mathsf{A} \in \mathcal{A}} \sum_{u,v \in A} \left(\mathbf{1}_{uv \in E} - \frac{d_u d_v}{2m} \right) = \frac{1}{m} \sum_{\mathsf{A} \in \mathcal{A}} e(\mathsf{A}) - \frac{1}{4m^2} \sum_{\mathsf{A} \in \mathcal{A}} \mathrm{vol}(\mathsf{A})^2;$$

the maximum modularity of G is $q^*(G) = \max_{\mathcal{A}}(G)$, where the maximum is over all partitions $\mathcal{A}$ of the vertices of G. Graphs with no edges are defined conventially to have modularity 0.

The modularity function is designed to score partitions highly when most edges fall within the parts and penalise partitions with very few or very big parts. These two objectives are encoded as the *edge contribution* or *coverage* $q^E_{\mathcal{A}}(G) = \frac{1}{m} \sum_{\mathsf{A} \in \mathcal{A}} e(\mathsf{A})$, and *degree tax* $q^D_{\mathcal{A}}(G) = \frac{1}{4m^2} \sum_{\mathsf{A} \in \mathcal{A}} \mathrm{vol}(\mathsf{A})^2$, in the modularity of a vertex partition $\mathcal{A}$ of G.

Note that for any graph with $m \geq 1$ edges $0 \leq q^*(G) < 1$. To see the lower bound, notice that the trivial partition which places all vertices in the same part has modularity zero. For example, complete graphs and stars have modularity 0 as noted in [3]. A graph consisting of c disjoint cliques of the same size has modularity $1 - 1/c$ with the optimal partition taking each clique to be a part.

As modularity is at most 1 it is sometimes useful to consider the *modularity defect* $\tilde{q}_{\mathcal{A}}(G) = 1 - q_{\mathcal{A}}(G)$. Denote by $\partial(A)$ the number of edges between A and the rest of the graph. Then

$$\tilde{q}_{\mathcal{A}}(G) = \frac{1}{2m} \sum_{A \in \mathcal{A}} \left(\partial(A) + \frac{\mathrm{vol}(A)^2}{2m} \right)$$

and we may equivalently minimise the modularity defect as maximise the modularity. In particular

$$\tilde{q}(G) = \min_{A \in \mathcal{A}} \tilde{q}_{\mathcal{A}}(G) = 1 - q^*(G).$$

We will make use of several facts about the maximum modularity of a graph.

▶ **Fact 1.1** (Lemma 3.3 of [3]). *If $\mathcal{A}$ is a partition of $V(G)$ such that $q_{\mathcal{A}}(G) = q^*(G)$ then no part A consists of a single vertex of degree 1.*

▶ **Fact 1.2** (Lemma 3.4 of [3]). *Suppose that G is a graph that contains no isolated vertices. If $\mathcal{A}$ is a partition of $V(G)$ such that $q_{\mathcal{A}}(G) = q^*(G)$ then, for every $A \in \mathcal{A}$, $G[A]$ is a connected subgraph of G.*

▶ **Fact 1.3** (Corollary 1 of [3]). *Let $G = (V, E)$ and suppose that $V_0 \subseteq V$ is a set of isolated vertices. Then $q(G) = q(G \setminus V_0)$. Moreover, if partitions $\mathcal{A}$ and $\mathcal{A}'$ agree on all vertices of $V \setminus V_0$, then $q_{\mathcal{A}}(G) = q_{\mathcal{A}'}(G)$.*

▶ **Fact 1.4** (Lemma 1 of [8],Lemma 2.1 of [5]). *For any integer $c > 0$ and any graph G,*

$$\max_{|\mathcal{A}| \leq c} q_{\mathcal{A}}(G) > q^*(G)\Big(1 - \frac{1}{c}\Big).$$

1.2 Notation and definitions

Given a graph $G = (V, E)$, and a set $U \subseteq V$ of vertices, we write $G[U]$ for the subgraph of G induced by U and $G \setminus U$ for $G[V \setminus U]$. Given two disjoint subsets of vertices $A, B \subseteq V$, we write $e(A, B)$ for the number of edges with one endpoint in A and the other in B. We shall often want to denote the number of edges between a set of vertices and the remainder of the graph so set $\partial(A) = e(A, \bar{A})$. If $\mathcal{P}$ is a partition of a set X, and $Y \subset X$, we write $\mathcal{P}[Y]$ for the restriction of $\mathcal{P}$ to Y.

A *vertex cover* of a graph $G = (V, E)$ is a set $U \subseteq V$ such that every edge has at least one endpoint in U; equivalently, $G \setminus U$ is an independent set (i.e. contains no edges). The *vertex cover number* of G is the smallest cardinality of any vertex cover of G. A *feedback vertex set* for G is a set $U \subseteq V$ such that $G \setminus U$ contains no cycles. Notice that the vertex cover number of G gives an upper bound on the size of the smallest feedback vertex set for G, written $\mathrm{fvs}(G)$. The *max leaf number* of G is the maximum number of leaves (degree one vertices) in any spanning tree of G.

A *tree decomposition* of a graph G is a pair $(T, \mathcal{D})$ where T is a tree and $\mathcal{D} = \{\mathcal{D}(t) : t \in V(T)\}$ is a collection of non-empty subsets of $V(G)$ (or *bags*), indexed by the nodes of T, satisfying:

1. $V(G) = \bigcup_{t \in V(T)} \mathcal{D}(t)$,
2. for every $e = uv \in E(G)$, there exists $t \in V(T)$ such that $u, v \in \mathcal{D}(t)$,
3. for every $v \in V(G)$, if $T(v)$ is defined to be the subgraph of T induced by nodes t with $v \in \mathcal{D}(t)$, then $T(v)$ is connected.

If T is in fact a path, we say that $(T, \mathcal{D})$ is a *path decomposition* of G. The *width* of the tree decomposition $(T, \mathcal{D})$ is defined to be $\max_{t \in V(T)} |\mathcal{D}(t)| - 1$, and the *treewidth* of G, written $\mathrm{tw}(G)$, is the minimum width over all tree decompositions of G. The *pathwidth* of G, $\mathrm{pw}(G)$, is the minimum width over all path decompositions of G.

2 Positive results

In this section we identify a number of structural restrictions on the input graph that allow us to compute the maximum modularity of a graph, or a good approximation to this quantity, efficiently.

2.1 Parameterisation by vertex cover number

In this section we demonstrate that MODULARITY is in FPT when parameterised by the vertex cover number of the input graph.

▶ **Theorem 2.1.** Modularity, *parameterised by cardinality of a minimum vertex cover for the input graph G, is in* FPT.

To prove this result, we make use of recent work of Lokshtanov [18] which gives an FPT algorithm for the following problem.

Integer Quadratic Programming
Input: An $n \times n$ integer matrix Q, an $m \times n$ integer matrix A, and an m-dimensional vector $\mathbf{b}$.
Parameter: $n + \alpha$, where α is the maximum absolute value of any entry in A or Q.
Problem: Find a vector $\mathbf{x} \in \mathbb{Z}^n$ which minimises $\mathbf{x}^T Q\mathbf{x}$, subject to $A\mathbf{x} \leq \mathbf{b}$.

Before proving Theorem 2.1, we introduce some notation. Suppose that the graph $G = (V, E)$ has $|E| = m$, and that $U = \{u_1, \dots, u_k\}$ is a vertex cover for G. Let $\mathcal{P} = \{P_1, \dots, P_\ell\}$ be a partition of U, and set $W = V \setminus U$ (so W is an independent set).

We can partition the vertices of W into 2^k sets based on their *type*: the type $\tau_U(w) \in \{0,1\}^k$ of a vertex $w \in W$ describes which of the vertices in U are neighbours of w. Formally $\tau_U(w)_j = 1$ if $u_j w \in E(G)$ and $\tau_U(w)_j = 0$ otherwise. For each $\sigma \in \{0,1\}^k$, we set S_σ to be the set of all vertices in W with type exactly σ, that is, $S_\sigma = \{w \in W : \tau(w) = \sigma\}$.

Now let $\mathcal{A} = \{A_1, \dots, A_r\}$ be a partition of V. We write $x^{\mathcal{A}}_{\sigma,i}$ for the number of vertices of type σ which are assigned to A_i, that is, $x^{\mathcal{A}}_{\sigma,i} = |S_\sigma \cap A_i|$. Finally, we introduce 0-1 vectors to encode the sets $P_i \in \mathcal{P}$: for $1 \leq i \leq \ell$, we let $\pi^i \in \{0,1\}^k$ be given by $\pi^i_j = 1$ if $u_j \in P_i$, and $\pi^i_j = 0$ otherwise.

We now argue that, if the partition $\mathcal{A}$ extends $\mathcal{P}$, we can compute the modularity of $\mathcal{A}$ using only the values $x^{\mathcal{A}}_{\sigma,i}$, together with information about $\mathcal{P}$. This shows that the modularity depends only on the number of vertices of each type assigned to a given partition, and not the assignment of individual vertices. The proof is omitted due to space constraints.

▶ **Lemma 2.2.** *Let $U = \{u_1, \dots, u_k\}$ be a vertex cover for $G = (V, E)$, where $|E| = m$, and let $\mathcal{P}$ be a partition of U. If $\mathcal{A}$ is any partition of V which extends $\mathcal{P}$ and has the property that every $A \in \mathcal{A}$ has non-empty intersection with U, then*

$$q^E_{\mathcal{A}}(G) = \frac{1}{m}\sum_{i=1}^{\ell} e(P_i) + \frac{1}{m}\sum_{(\sigma,i)} x^{\mathcal{A}}_{\sigma,i}(\sigma \cdot \pi^i),$$

and

$$4m^2 q^D_{\mathcal{A}'} = 4\sum_i e(P_i)^2 + 4\sum_{(\sigma,i)} x^{\mathcal{A}}_{\sigma,i} e(P_i)(\sigma \cdot (\mathbf{1} + \pi^i)) + \sum_{(\sigma,i)(\sigma',j)} x^{\mathcal{A}}_{\sigma,i} x^{\mathcal{A}}_{\sigma',j} (\sigma \cdot (\mathbf{1} + \pi^i))(\sigma' \cdot (\mathbf{1} + \pi^j)).$$

Proof of Theorem 2.1. We will assume that the input to our instance of Modularity is a graph $G = (V, E)$, where $|E| = m$. We may assume without loss of generality that we are also given as input a vertex cover $U = \{u_1, \dots, u_k\}$ for G (as if not we can easily compute one in the allowed time). We may further assume that G does not contain any isolated vertices, as we can delete any such vertices (in polynomial time) without changing the value of the maximum modularity (by Fact 1.3).

Note that the total number of possible partitions of U into non-empty parts is equal to the k^{th} *Bell number*, B_k (and hence is certainly less than k^k). It therefore suffices to describe an fpt-algorithm which determines, given some partition $\mathcal{P}$ of U,

$$q^{\mathcal{P}}(G) = \max\{q_{\mathcal{A}}(G) : \mathcal{A}[U] = \mathcal{P}\}.$$

The maximum modularity of G can then be calculated by taking

$$\max\{q^{\mathcal{P}}(G) : \mathcal{P} \text{ is a partition of } U\}.$$

From now on, we consider a fixed partition $\mathcal{P} = \{P_1, \dots, P_\ell\}$ of U, and describe how to compute $q^{\mathcal{P}}(G)$.

It follows from Facts 1.1 and 1.2, together with the fact that W is an independent set that, if $\mathcal{A} = \{A_1, \dots, A_j\}$ is a partition of V which achieves the maximum modularity, then every part A_i has non-empty intersection with U. We will call a partition with this properties a U-partition of G. It then suffices to maximise the modularity over all U-partitions in order to determine the value of $q^{\mathcal{P}}(G)$.

Now, by Lemma 2.2, we know that we can express the modularity of a U-partition $\mathcal{A}$ as

$$\begin{aligned} q_{\mathcal{A}}(G) = \frac{1}{m}\sum_{i=1}^{\ell} e(P_i) + \frac{1}{m}\sum_{(\sigma,i)} x^{\mathcal{A}}_{\sigma,i}(\sigma\cdot\pi^i) - \frac{1}{m^2}\sum_i e(P_i)^2 \\ - \frac{1}{m^2}\sum_{(\sigma,i)} x^{\mathcal{A}}_{\sigma,i} e(P_i)(\sigma\cdot(\mathbf{1}+\pi^i)) \\ - \frac{1}{4m^2}\sum_{(\sigma,i)(\sigma',j)} x^{\mathcal{A}}_{\sigma,i} x^{\mathcal{A}}_{\sigma',j}(\sigma\cdot(\mathbf{1}+\pi^i))(\sigma'\cdot(\mathbf{1}+\pi^j)). \end{aligned} \quad (1)$$

As we have fixed the partition $\mathcal{P}$, all values $e(P_i)$ can be regarded as fixed constants. In order to determine the maximum modularity we can obtain with a U-partition, we therefore need to find the values of $x^{\mathcal{A}}_{\sigma,i}$ which maximise this expression.

We can rewrite (1) as the sum of a constant term, two linear functions θ and ϕ of the $x^{\mathcal{A}}_{\sigma,i}$ and a quadratic function ψ of the $x^{\mathcal{A}}_{\sigma,i}$ (up to scaling by constants):

$$\begin{aligned} q_{\mathcal{A}}(G) = & \underbrace{\frac{1}{m}\sum_{i=1}^{\ell} e(P_i) - \frac{1}{m^2}\sum_i e(P_i)^2}_{\text{constant}} \\ & + \frac{1}{m}\underbrace{\sum_{(\sigma,i)} x^{\mathcal{A}}_{\sigma,i}(\sigma\cdot\pi^i)}_{\theta(\mathcal{A})} - \frac{1}{m^2}\underbrace{\sum_{(\sigma,i)} x^{\mathcal{A}}_{\sigma,i}\, e(P_i)(\sigma\cdot(\mathbf{1}+\pi^i))}_{\phi(\mathcal{A})} \\ & - \frac{1}{4m^2}\underbrace{\sum_{(i,\sigma)(j,\sigma'} x^{\mathcal{A}}_{\sigma,i} x^{\mathcal{A}}_{\sigma',j}(\sigma\cdot(\mathbf{1}+\pi^i))(\sigma'\cdot(\mathbf{1}+\pi^j))}_{\psi(\mathcal{A})}. \end{aligned}$$

To find the maximum value of $q_{\mathcal{A}}(G)$ over all U-partitions it therefore suffices to determine, for all possible values of $\theta(\mathcal{A})$ and $\phi(\mathcal{A})$, the minimum possible value of $\psi(\mathcal{A})$. Before describing how to do this, we observe that the number of combinations of possible values for $\theta(\mathcal{A})$ and $\phi(\mathcal{A})$ and is not too large. Note that $0 \le \sum_{\sigma,i} x^{\mathcal{A}}_{\sigma,i}(\sigma\cdot\pi^i) < nk$, and $0 \le \sum_{\sigma,i} x^{\mathcal{A}}_{\sigma,i} e(P_i)(\sigma\cdot(\mathbf{1}+\pi^i)) < n\binom{k}{2}2k < nk^3$, so the number of possible pairs $(\theta(\mathcal{A}), \phi(\mathcal{A}))$

is at most n^2k^4. Thus, if we know the minimum possible value of $\psi(\mathcal{A})$ corresponding to each possible pair $(\theta(\mathcal{A}), \phi(\mathcal{A}))$, we can compute the maximum modularity achieved by any U-partition $\mathcal{A}$ such that $(\theta(\mathcal{A}), \phi(\mathcal{A})) = (y, z)$, and maximising over the polynomial number of possible pairs (y, z) will give $q^{\mathcal{P}}(G)$.

Now, given a possible pair of values (y, z) for $(\theta(\mathcal{A}), \phi(\mathcal{A}))$, we describe how to compute

$$\min\{\psi(\mathcal{A}) : \mathcal{A} \text{ is a } U\text{-partition with } \theta(\mathcal{A}) = y \text{ and } \phi(\mathcal{A}) = z\}.$$

Our strategy is to express this minimisation problem as an instance of Integer Quadratic Programming and then apply the FPT algorithm of [18].

In this instance, we have $n = \ell 2^k \leq k2^k$, and our vector of variables $\mathbf{x} = (x_1, \dots, x_n)^T$ is given by

$$x_i = x^{\mathcal{A}}_{(\sigma_{i \mod 2^k}), \lceil i/2^k \rceil},$$

where $\sigma_1, \dots, \sigma_{2^k}$ is a fixed enumeration of all vectors in $\{0,1\}^k$. The matrix Q expresses the value of $\psi(\mathcal{A})$ in terms of $\mathbf{x}$: if we set $Q = \{q_{i,j}\}$ where

$$q_{i,j} = \left(\sigma_{(i \mod 2^k)} \cdot \left(\mathbf{1} + \pi^{\lceil i/2^k \rceil}\right)\right) \left(\sigma_{(j \mod 2^k)} \cdot \left(\mathbf{1} + \pi^{\lceil j/2^k \rceil}\right)\right),$$

then it is easy to see that $\psi(\mathcal{A}) = \mathbf{x}^T Q \mathbf{x}$. Note also that the maximum absolute value of any entry in Q is at most $4k^2$.

We now use the linear constraints to express the conditions that

1. $\theta(\mathcal{A}) = y$,
2. $\phi(\mathcal{A}) = z$, and
3. the values $x_{i,\sigma}$ correspond to a valid U-partition $\mathcal{A}$.

The first of these conditions can be expressed as a single linear constraint:

$$\sum_{(\sigma,i)} x^{\mathcal{A}}_{\sigma,i}(\sigma \cdot \pi^i) = y,$$

or equivalently $\mathbf{a}_1\mathbf{x} = y$ where $\mathbf{a}_1$ is the $1 \times n$ row vector with i^{th} entry equal to

$$\sigma_{(i \mod 2^k)} \cdot \pi^{\lceil i/2^k \rceil}.$$

We can similarly express the second condition as a single linear constraint:

$$\sum_{(\sigma,i)} x^{\mathcal{A}}_{\sigma,i}\, e(P_i)(\sigma \cdot (\mathbf{1} + \pi^i)) = z,$$

or equivalently $\mathbf{a}_2\mathbf{x} = z$, where $\mathbf{a}_2$ is the $1 \times n$ row vector with i^{th} entry equal to

$$e\left(P_{\lceil i/2^k \rceil}\right)\left(\sigma_{(i \mod 2^k)} \cdot \left(\mathbf{1} + \pi^{\lceil i/2^k \rceil}\right)\right).$$

Note that every entry in the vectors $\mathbf{a}_1$ and $\mathbf{a}_2$ has absolute value no more than $2k^3$. For the third condition, note that the values $x_{i,\sigma}$ correspond to a valid U-partition if and only if every $x_{i,\sigma}$ is non-negative, and for each σ we have $\sum_{i=1}^{\ell} x^{\mathcal{A}}_{i,\sigma} = |S_\sigma|$.

We can therefore express all three conditions in the form $A\mathbf{x} = \mathbf{b}$, where A is a $\left(4 + (\ell+1)2^k\right) \times n$ and $\mathbf{b}$ is a $\left(4 + (\ell+1)2^k\right)$-dimensional vector (notice that we use two inequalities to express each of the linear equality constraints).

Altogether, this means that a solution to this instance of Integer Quadratic Programming will determine the values of $x^{\mathcal{A}}_{i,\sigma}$ which minimize (out of all values corresponding to some U-partition $\mathcal{A}$) the value of $\psi(\mathcal{A})$, subject to the additional requirement that $\theta(\mathcal{A}) = y$ and $\phi(\mathcal{A}) = z$. Note that the number of variables n is at most $k2^k$ and the largest absolute value of any entry in A or Q is at most $2k^3$, so the parameter in the instance of Integer Quadratic Programming is bounded by a function of k. This completes the proof. ◀

We note the algorithm described can easily be modified to output an optimal partition.

2.2 Parameterisation by treewidth

In this section we demonstrate that Modularity, parameterised by the treewidth of the input graph G, belongs to XP and so is solvable in polynomial time on graph classes whose treewidth is bounded by some fixed constant. We further show that for any fixed $\varepsilon > 0$ there is an FPT-algorithm, parameterised by treewidth, which computes a factor $(1-\varepsilon)$-approximation; i.e. returning a value between $(1-\varepsilon)q^*$ and q^* where q^* is the maximum modularity of the graph.

▶ **Theorem 2.3.** Modularity *parameterised by the treewidth of the input graph G is in* XP.

This result makes use of standard dynamic programming techniques, and details are omitted due to space constraints. The key property of modularity we use is that each part in a partition that maximises modularity must induce a connected subgraph, so it suffices to keep track of feasible statistics for each part that intersects the bag of the tree decomposition under consideration, together with the contribution to modularity from parts that are entirely "below" the current bag.

To obtain our FPT approximation result, we use a very similar approach; the key is to restrict our attention to partitions with only a constant number of parts. For any constant $c \in \mathbb{N}$, we write $q^*_{\leq c}(G)$ for the maximum modularity for G achievable with a partition into at most c parts, that is

$$q^*_{\leq c}(G) = \max_{|\mathcal{A}| \leq c} q_{\mathcal{A}}(G).$$

We refer to the problem of deciding whether $q^*_{\leq c}(G) \geq q$ for a given input graph G and constant $q \in [0,1]$ as c-Modularity. We now argue that c-Modularity is in FPT parameterised by the treewidth of the input graph.

▶ **Lemma 2.4.** c-Modularity *is in* FPT *when parameterised by the treewidth of the input graph.*

The crucial difference from our XP algorithm above is the fact that, when we fix the number of parts in the partition, we can no longer assume that every part is connected. However, if the maximum number of parts c is a constant, we can keep track of the necessary statistics for every possible part, not just those that intersect the bag under consideration.

Recall (Fact 1.4) that $q^*(G) \geq q^*_{\leq c}(G)) > q^*(G)\Big(1 - \frac{1}{c}\Big)$; thus, for any constant $\epsilon > 0$, we obtain a multiplicative approximation with relative error at most ϵ by solving $\lceil \frac{1}{\epsilon} \rceil$-Modularity. This immediately gives the following result.

▶ **Corollary 2.5.** *Given any constant $\epsilon > 0$, there is an FPT-algorithm, parameterised by the treewidth of the input graph G, that that returns a partition $\mathcal{A}$ with $q_{\mathcal{A}}(G) > (1-\varepsilon)q^*(G)$.*

We conclude this section by noting that sparse graphs, in particular graphs G with low tree width, $\mathrm{tw}(G)$, and maximum degree, $\triangle(G)$, can have high maximum modularity. In particular Theorem 1.11 of [19] shows $q^*(G) \geq 1 - 2((\mathrm{tw}(G)+1)\triangle(G)/|E(G)|)^{1/2}$.

2.3 Parameterisation by max leaf number

In this section we demonstrate that MODULARITY can be solved in time linear in the number of connected subgraphs of the input graph G; as a consequence of this result, we deduce that the problem belongs to XP when parameterised by the max leaf number of G.

▶ **Theorem 2.6.** *Let G be a graph on n vertices with m edges and at most h connected subgraphs. Then* MODULARITY *can be solved in time $\mathcal{O}(h^2 n)$.*

This is achieved by means of a dynamic programming strategy, in which we compute the maximum value of a modularity-like function for each connected subgraph in turn, considering the subgraphs in nondecreasing order of their number of vertices. Full details of the proof are omitted due to space constraints.

It is known that, if the max leaf number of G is c, then G is a subdivision of some graph H on at most $4c$ vertices [12]; a graph on n vertices that is a subdivision of such a graph H has at most $2^{4c} n^{(4c)^2}$ connected subgraphs (once we have decided which branch vertices belong to a subgraph, it remains only to decide where to cut each path from one of the chosen branch vertices to one we have not chosen). Thus, if the max leaf number of G is bounded by a constant it follows that G has at most a polynomial number of connected subgraphs, and the following result is an immediate consequence of Theorem 2.6.

▶ **Corollary 2.7.** MODULARITY *is in* XP *when parameterised by the max leaf number of the input graph G.*

We conjecture that this result is not optimal, and that MODULARITY is in fact in FPT with respect to this parameterisation.

3 Hardness results

In this section we complement our positive result about the FPT approximability of the problem parameterised by treewidth by demonstrating that computing the exact value of the maximum modularity is hard even in a more restricted setting.

▶ **Theorem 3.1.** MODULARITY*, parameterised simultaneously by the pathwidth and the size of a minimum feedback vertex set for the input graph, is* W[1]*-hard.*

Our proof of this result relies on the hardness of the following problem.

EQUITABLE CONNECTED PARTITION (ECP)
Input: A graph $G = (V, E)$ and $r \in \mathbb{N}$.
Question: Is there a partition of V into r classes $V_1, \dots, V_r$ such that $|V_i| - |V_j| \leq 1$ for all $1 \leq i < j \leq r$, and the induced subgraph $G[V_i]$ is connected for each $i \in 1, \dots, r$?

The parameterised complexity of ECP was investigated thoroughly in [11]. Among other results, the problem is shown to be W[1]-hard even when parameterised simultaneously by r, $\text{pw}(G)$ and $\text{fvs}(G)$. In proving this hardness result, the authors implicitly consider the following variation of ECP.

Anchored Equitable Connected Partition (AECP)
Input: A graph $H = (V_H, E_H)$, and a set of distinguished *anchor vertices* $a_1, \dots, a_r \in V$.
Question: Is there a partition of V_H into r classes $V_1, \dots, V_r$ such that $a_i \in V_i$ for all i, $||V_i| - |V_j|| \leq 1$ for all $1 \leq i < j \leq r$, and the induced subgraph $G[V_i]$ is connected for each $i \in 1, \dots, r$?

From the proof of [11, Theorem 1] we can extract the following statement about the hardness of AECP.

▶ **Lemma 3.2** ([11], implicit in proof of Theorem 1)**.** *AECP is* W[1]*-hard, parameterised simultaneously by* $\mathrm{pw}(H)$ *and* $\mathrm{fvs}(H)$*, even if the following conditions hold simultaneously:*

1. *H is connected;*
2. *the graph H' obtained from H by deleting all vertices of degree one is a subdivision of a 3-regular graph $\tilde{H}$;*
3. *the branch vertices of H' (i.e. vertices of $\tilde{H}$) are precisely the anchor vertices $a_1, \dots, a_r$;*
4. *$r \geq 4$ is even and divides $|V_H|$;*
5. *$H \setminus \{a_1, \dots, a_r\}$ is a disjoint union of isolated vertices and paths with pendant edges.*

In the proof of Theorem 3.1, it is useful to analyse the 'per unit modularity defect' $f_m(B)$ of vertex subsets B. Intuitively, the following lemma (proof omitted due to space constraints) says that, if you are restricted to parts B with $\delta(B) = 4$, the modularity maximising volume is $\mathrm{vol}(B) = 2\sqrt{2m}$. Moreover, while it would usually be better to take parts with $\delta(B) < 4$ these parts are actually worse (i.e. higher $f_m(B)$ value) if their volumes are too big or too small. The function $f_m(B)$ plays a similar role to the *n-cost* in Proposition 1 of [19].

▶ **Lemma 3.3.** *Assume that* $m \geq 1$, $\mathrm{vol}(B) \geq 1$ *and define*

$$f_m(B) = \frac{\partial(B)}{\mathrm{vol}(B)} + \frac{\mathrm{vol}(B)}{2m}.$$

Then the following properties hold:
0: *if* $\partial(B) = 0$ *and* $\mathrm{vol}(B) > 4\sqrt{2m}$ *then* $f_m(B) > 2\sqrt{2/m}$*.*
1: *if* $\partial(B) = 1$ *and* $\mathrm{vol}(B) > 3.7321\sqrt{2m}$ *or* $\mathrm{vol}(B) < 0.2679\sqrt{2m}$ *then* $f_m(B) > 2\sqrt{2/m}$*.*
2: *if* $\partial(B) = 2$ *and* $\mathrm{vol}(B) > 3.4143\sqrt{2m}$ *or* $\mathrm{vol}(B) < 0.5857\sqrt{2m}$ *then* $f_m(B) > 2\sqrt{2/m}$*.*
3: *if* $\partial(B) = 3$ *and* $\mathrm{vol}(B) > 3\sqrt{2m}$ *or* $\mathrm{vol}(B) < \sqrt{2m}$ *then* $f_m(B) > 2\sqrt{2/m}$*.*
4: *if* $\partial(B) = 4$ *and* $\mathrm{vol}(B) = 2\sqrt{2m}$ *then* $f_m(B) = 2\sqrt{2/m}$*.*
5: *if* $\partial(B) \geq 5$ *then* $f_m(B) > 2\sqrt{2/m}$*.*

Proof of Theorem 3.1. We give a reduction from AECP. Let $(H, \{a_1, \dots, a_r\})$ be the input to an instance of AECP; we will describe how to construct a graph G, where $\mathrm{pw}(G)$ and $\mathrm{fvs}(G)$ are both bounded by a function of r, together with an explicit $q_0 \in (0, 1)$ such that (G, q_0) is a yes-instance for Modularity if and only if $(H, \{a_1, \dots, a_r\})$ is a yes-instance for AECP.

We may assume without loss of generality that our instance of AECP satisfies all of the conditions of Lemma 3.2.

We define a new graph G, obtained from H by adding the following (see Figure 1):

- α new leaves adjacent to each anchor vertex $a_1, \dots, a_r$,
- β isolated edges disjoint from G, and
- an arbitrary perfect matching on the anchor vertices $a_1, \dots, a_r$,

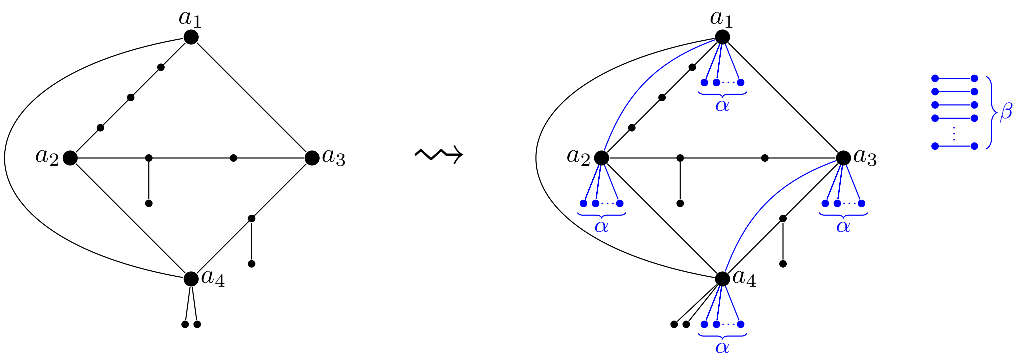

Figure 1 Possible input graph H with anchors a_1, a_2, a_3, a_4 and the graph G constructed from it by adding α new leaves adjacent to each anchor, β isolated edges and a perfect matching between the anchors.

where the values of α and β will be determined later. Notice that, even with these modifications, $G \setminus \{a_1, \ldots, a_r\}$ is still a disjoint union of isolated vertices and paths with pendant edges; hence $\mathrm{pw}(G) \leq r+1$ and $\mathrm{fvs}(G) \leq r$. We set $m = |E(G)|$ so $m = |E(H)|+\alpha r+\beta+r/2$.

We now define our instance of Modularity to be (G, q_0), where

$$q_0 = 1 - \frac{\beta}{m^2} - \frac{2\sqrt{2}(m-\beta)}{m^{3/2}}$$

We now argue that (G, q_0) is a yes-instance if and only if $(H, \{a_1, \ldots, a_r\})$ is a yes-instance for AECP. Recall that

$$q^*(G) = 1 - \min_{\mathcal{A}} \sum_{A \in \mathcal{A}} \left(\frac{\partial(A)}{2m} + \frac{\mathrm{vol}(A)^2}{4m^2} \right).$$

and that the partition $\mathcal{A}$ which achieves the minimum in the expression above is exactly the modularity maximal $\mathcal{A}$. In any modularity optimal partition, $\mathcal{A}$, each isolated edge will form its own part: this follows from Facts 1.1 and 1.2. Write V' for vertices of G without the vertices supporting the β isolated edges, and let the minimisation be over $\mathcal{A}'$ which are vertex partitions of V'. We then have

$$q^*(G) = 1 - \frac{\beta}{m^2} - \min_{\mathcal{A}'} \sum_{A \in \mathcal{A}'} \frac{\partial(A)}{2m} + \frac{\mathrm{vol}(A)^2}{4m^2}.$$

Rearranging, we see that

$$\begin{aligned} 1 - \frac{\beta}{m^2} - q^*(G') &= \frac{m-\beta}{m} \min_{\mathcal{A}'} \sum_{A \in \mathcal{A}'} \frac{\mathrm{vol}(A)}{2(m-\beta)} \left(\frac{\partial(A)}{\mathrm{vol}(A)} + \frac{\mathrm{vol}(A)}{2m} \right) \\ &= \frac{m-\beta}{m} \min_{\mathcal{A}'} \sum_{A \in \mathcal{A}'} \frac{\mathrm{vol}(A)}{2(m-\beta)} f_m(A) \qquad (2) \\ &\geq \frac{m-\beta}{m} \min_{A \subseteq V'} f_m(A). \qquad (3) \end{aligned}$$

The last inequality holds because $\sum_A \mathrm{vol}(A) = 2(m-\beta)$ and so (2) is a weighted sum of the $f_m(A)$ with total weight one. This, together with the fact that no A has zero volume, also implies that (2)$\geq$(3) with equality if and only if $f_m(A) = \min_{B \subset V'} f_m(B)$ for every $A \in \mathcal{A}'$.

Note that, since $\mathcal{A}'$ is the restriction of some modularity optimal partition $\mathcal{A}$ to a connected component of G, we may assume that, for all $A \in \mathcal{A}'$, $G[A]$ is connected. Moreover, if v is a pendant vertex adjacent to u then u and v are in the same part in $\mathcal{A}'$; we call a partition with this last property (or, abusing notation, a set that would not violate this condition in a partition) 'pendant-consistent'.

We make the following claim, whose proof is omitted due to space constraints; we write $s = |H|/r$ for the desired part size in our instance of AECP.

▶ **Claim 3.4.** *Suppose that $\alpha > 32|E(H)|^2$ and that we have $\sqrt{2m} = s + \alpha + 1$. Then:*

(a) *for any connected, pendant-consistent set $B \subseteq V'$ we have $f_m(B) \geq 2\sqrt{2/m}$, and if $f_m(B) = 2\sqrt{2/m}$ then B contains exactly one anchor and $\mathrm{vol}(B) = 2\sqrt{2m}$;*

(b) *if $(H, \{a_1, \dots, a_r\})$ is a yes-instance, then there is a vertex partition $\mathcal{A}'$ of V' so that $f_m(A) = 2\sqrt{2/m}$ for all $A \in \mathcal{A}'$;*

(c) *if there is a vertex partition $\mathcal{A}' = \{A_1, \dots, A_r\}$ of V' so that for all $A_i \in \mathcal{A}$, $f_m(A_i) = 2\sqrt{2/m}$, A is pendant-consistent and $G[A]$ is connected for all $A \in \mathcal{A}'$, then $(H, \{a_1, \dots, a_r\})$ is a yes-instance.*

Note that, by Claim 3.4(a) and line (3), we always have

$$q^*(G) \leq q_0 = 1 - \frac{\beta}{m^2} - \frac{2\sqrt{2}(m-\beta)}{m^{3/2}}.$$

Hence in particular (G, q_0) is a yes-instance if and only if there is a partition $\mathcal{A}'$ of V' such that $\forall A \in \mathcal{A}'$ $f_m(A) = 2\sqrt{2/m}$.

Claim 3.4(b) , together with line (2), implies that, if $(H, \{a_1, \dots, a_r\})$ is a yes-instance, then so is (G, q_0). Converesly, if (G, q_0) is a yes-instance, it follows from Claim 3.4(c), that $(H, \{a_1, \dots, a_r\})$ is a yes-instance.

It remains only to show that we can choose suitable values of α and β. Set α to be the least integer such that

$$\alpha \geq 32|E(H)|^2, \quad (\alpha + s + 1)^2 > 2|E(H)| + 2\alpha r + r \quad \text{and} \quad \alpha = s + 1 \pmod{2}. \tag{4}$$

Recall that r is even. This, along with our parity constraint between α and s, implies that $(\alpha + s + 1)^2 - r$ is even. Thus we can choose β to be

$$\beta = \frac{1}{2}\left((\alpha + s + 1)^2 - r\right) - |E(H)| - \alpha r; \tag{5}$$

note β is positive because we set α so that $(\alpha + s + 1)^2 > 2|E(H)| + 2\alpha r + r$. Finally, observe that we do have $\sqrt{2m} = s + \alpha + 1$ because, by the chosen value of β,

$$m = |E(H)| + \alpha r + \beta + r/2 = (s + \alpha + 1)^2/2. \qquad ◀$$

4 Conclusions and Open Problems

We have shown that Modularity belongs to FPT when parameterised by the vertex cover number of the input graph, and that the problem is solvable in polynomial time on input graphs whose treewidth or max leaf number is bounded by some fixed constant; we also showed that there is an FPT algorithm, parameterised by treewidth, which computes any constant-factor approximation to the maximum modularity. In contrast with the positive approximation result, we demonstrated that the problem is unlikely to admit an exact FPT when the treewidth is taken to be the parameter, as it is W[1]-hard even when parameterised simultaneously by the treewidth and size of a minimum feedback vertex set for the input graph.

We conjecture that our XP algorithm parameterised by max leaf number is not optimal, and that MODULARITY in fact belongs to FPT with respect to this parameterisation. Another natural open question arising from our work is whether our approximation result can be extended to larger classes of graphs, for example those of bounded cliquewidth or bounded expansion.

References

1 J. P. Bagrow. Communities and bottlenecks: Trees and treelike networks have high modularity. *Physical Review E*, 85(6):066118, 2012.

2 V. D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre. Fast unfolding of communities in large networks. *Journal of Statistical Mechanics: Theory and Experiment*, 2008(10):P10008, 2008.

3 U. Brandes, D. Delling, M. Gaertler, R. Gorke, M. Hoefer, Z. Nikoloski, and D. Wagner. On Modularity Clustering. *IEEE Trans. on Knowl. and Data Eng.*, 20(2):172–188, February 2008. doi:10.1109/TKDE.2007.190689.

4 M. Cygan, F. V. Fomin, Ł. Kowalik, D. Lokshtanov, D. Marx, M. Pilipczuk, M. Pilipczuk, and S. Saurabh. *Parameterized Algorithms*. Springer, Cham, 2015.

5 B. DasGupta and D. Desai. On the complexity of Newman's community finding approach for biological and social networks. *Journal of Computer and System Sciences*, 79(1):50–67, 2013. doi:10.1016/j.jcss.2012.04.003.

6 F. De Montgolfier, M. Soto, and L. Viennot. Asymptotic modularity of some graph classes. In *Algorithms and Computation*, pages 435–444. Springer, 2011.

7 T. N. Dinh, X. Li, and M. T. Thai. Network Clustering via Maximizing Modularity: Approximation Algorithms and Theoretical Limits. In *2015 IEEE International Conference on Data Mining*, pages 101–110, November 2015. doi:10.1109/ICDM.2015.139.

8 T. N. Dinh and M. T. Thai. Finding community structure with performance guarantees in scale-free networks. In *Privacy, Security, Risk and Trust (PASSAT) and 2011 IEEE Third Inernational Conference on Social Computing (SocialCom), 2011 IEEE Third International Conference on*, pages 888–891. IEEE, 2011.

9 T. N. Dinh and M. T. Thai. Community Detection in Scale-Free Networks: Approximation Algorithms for Maximizing Modularity. *IEEE Journal on Selected Areas in Communications*, 31(6):997–1006, June 2013. doi:10.1109/JSAC.2013.130602.

10 R. G. Downey and M. R. Fellows. *Fundamentals of Parameterized Complexity*. Springer London, 2013.

11 R. Enciso, M. R. Fellows, J. Guo, I. Kanj, F. Rosamond, and O. Suchý. *What Makes Equitable Connected Partition Easy*, pages 122–133. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009. doi:10.1007/978-3-642-11269-0_10.

12 V. Estivill-Castro, M. Fellows, M. Langston, and F. Rosamond. FPT is P-time extremal structure I. In *Algorithms and Complexity in Durham 2005, Proceedings of the first ACiD Workshop, volume 4 of Texts in Algorithmics*, pages 1–41. King's College Publications, 2005.

13 S. Fortunato. Community detection in graphs. *Physics Reports*, 486(3):75–174, 2010.

14 S. Fortunato and D. Hric. Community detection in networks: A user guide. *Physics Reports*, 659:1–44, 2016.

15 I. S. Jutla, L. G. S. Jeub, and P. J. Mucha. A generalized Louvain method for community detection implemented in MATLAB. *URL http://netwiki.amath.unc.edu/GenLouvain*, 2011.

16 Y. Kawase, T. Matsui, and A. Miyauchi. Additive Approximation Algorithms for Modularity Maximization. In Seok-Hee Hong, editor, *27th International Symposium on Algorithms* *and Computation (ISAAC 2016)*, volume 64 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 43:1–43:13, Dagstuhl, Germany, 2016. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. doi:10.4230/LIPIcs.ISAAC.2016.43.

17 A. Lancichinetti and S. Fortunato. Limits of modularity maximization in community detection. *Physical Review E*, 84(6):066122, 2011.

18 D. Lokshtanov. Parameterized Integer Quadratic Programming: Variables and Coefficients. arXiv:1511.00310 [cs.DS], 2015.

19 C. McDiarmid and F. Skerman. Modularity of regular and treelike graphs. *Journal of Complex Networks*, 5, 2017.

20 C. McDiarmid and F. Skerman. Modularity of Erdős-Rényi Random Graphs. In *29th International Conference on Probabilistic, Combinatorial and Asymptotic Methods for the Analysis of Algorithms*, volume 1, 2018.

21 C. McDiarmid and F. Skerman. Modularity of Erdős-Rényi random graphs. arXiv:1808.02243 [math.CO], 2018.

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

23 M. Porter, J.-P. Onnela, and P. Mucha. Communities in networks. *Notices of the AMS*, 56(9):1082–1097, 2009.

24 L. O. Prokhorenkova, P. Prałat, and A. Raigorodskii. Modularity of models of complex networks. *Electronic Notes in Discrete Mathematics*, 61:947–953, 2017.