Learning Resolution Parameters for Graph Clustering

Nate Veldt
Mathematics Department,
Purdue University
West Lafayette, Indiana
lveldt@purdue.edu

David F. Gleich
Computer Science Department,
Purdue University
West Lafayette, Indiana
dgleich@purdue.edu

Anthony Wirth
School of Computing and
Information Systems,
The University of Melbourne
Parkville, VIC, Australia
awirth@unimelb.edu.au

ABSTRACT

Finding clusters of well-connected nodes in a graph is an extensively studied problem in graph-based data analysis. Because of its many applications, a large number of distinct graph clustering objective functions and algorithms have already been proposed and analyzed. To aid practitioners in determining the best clustering approach to use in different applications, we present new techniques for automatically learning how to set clustering resolution parameters. These parameters control the size and structure of communities that are formed by optimizing a generalized objective function. We begin by formalizing the notion of a parameter fitness function, which measures how well a fixed input clustering approximately solves a generalized clustering objective for a specific resolution parameter value. Under reasonable assumptions, which suit two key graph clustering applications, such a parameter fitness function can be efficiently minimized using a bisection-like method, yielding a resolution parameter that fits well with the example clustering. We view our framework as a type of *single-shot* hyperparameter tuning, as we are able to learn a good resolution parameter with just a single example. Our general approach can be applied to learn resolution parameters for both local and global graph clustering objectives. We demonstrate its utility in several experiments on real-world data where it is helpful to learn resolution parameters from a given example clustering.

KEYWORDS

Graph clustering; community detection; resolution parameters

1 INTRODUCTION

Partitioning a collection of items into groups of similar items – that is, clustering – is a fundamental computational task. So commonly applied, there is a large and still-growing suite of objective functions, algorithms, and techniques for identifying *good* clusters. One powerful mathematical model for clustering is the graph, comprising nodes and (undirected) edges. For a broad overview of graph

clustering, refer to any one of a number of surveys [11, 13, 28, 31]. Nearly all clustering approaches favor clusters with high internal edge density and a low external edge density. A related, but not identical, notion is that a good cluster is a set of nodes with a small cut (i.e., few edges leaving the set), and a nontrivial size (e.g., a large number of nodes, or many internal edges).

Although most clustering approaches follow these general principles, there are many different ways to formalize such goals mathematically. In practice, this array of objective functions yields a large variety of different output clusterings. Indeed, many existing (theoretical) approaches to graph clustering assume that the user knows *a priori* which objective function is appropriate for their context or job. The main design task, leading to a practical solution, is then to develop good algorithms that exactly, or approximately, optimize the objective. However, we propose that it is more natural to assume that the user starts with some a priori knowledge about the desired *structure* of clusters in a given application domain. More specifically, they can provide at least *one* example of what a *good* clustering should look like. The revised goal is to find an objective function whose optimization yields the desired type of output.

Our approach. In this article, we show how to bootstrap from this one quality clustering to learn the appropriate objective function, chosen from a parameterized family of objective functions. The efficiency of our technique relies on the clustering objective function being linear, with linear constraints, tuned by a single parameter β. Given this type of objective, we formalize the notion of a parameter fitness function, which relies on a fixed example clustering C_x of a network, and takes a parameter β as input. The fitness function computes the ratio between the objective score of C_x, when β is chosen as the input parameter, and a lower bound on the optimal clustering objective for that β. Specifically, we use a concave piecewise linear lower bound over a wide family of what we refer to as *relaxed clusterings*. Minimizing the fitness function produces a parameter (and a corresponding objective function) that C_x exactly or at least *nearly* optimizes. Thus the aim, which can be realized via a guided binary search, is to identify the parameter setting in which C_x most stands out. In the remainder of this introduction, we flesh out the context for our technique.

Parameters. There are multiple families of objective functions whose members are specified by a tunable resolution parameter. This parameter controls the size and structure of clusters that are formed by optimizing the objective. Key examples of such *generalized clustering objective functions* include the Hamiltonian objective studied by Reichardt and Bornholdt [29], the stability objective of Delvenne et al. [10], and a multi-resolution variant of the map equation [32].

In this manuscript we focus on a related clustering framework that we developed in previous work [37], based on correlation clustering [5]. This framework is named LambdaCC, after its resolution parameter λ, which implicitly controls both the internal edge density as well as the cut sparsity of nodes formed by minimizing the objective. Furthermore, LambdaCC generalizes several well-studied objectives such as modularity clustering, sparsest cut, normalized cut, and cluster deletion. All of these objectives can be viewed as special cases of the objective for appropriate settings of λ.

Global and local. The above objectives are specifically designed for global clustering, in which the goal is to find a multi-cluster partitioning of an input graph. Local clustering objectives relying on resolution parameters also exist; these focus on finding a single cluster in a localized region of a graph. Flow-based methods such as FlowImprove [4], LocalImprove [26], and SimpleLocal [35] fit into this category. These methods repeatedly solve minimum s-t cut problems for different values of a parameter α, in order to minimize a ratio-style objective related to a cluster quality measure called *conductance*. This α can be viewed as a resolution parameter that balances a trade-off between forming a cluster with a small cut, and forming a cluster with a large overlap with a seed set in the graph.

Given the unifying power and versatility of generalized clustering objective functions, the challenge of finding the right clustering technique for a specific application can often be reduced to finding an appropriate resolution parameter. However, very little work has addressed how to set these parameters in practice, in particular to capture the specific clustering structure present in a certain application domain. In the past, solving generalized objective functions for a range of resolution parameters has been used to detect hierarchical clusterings in a network [29], or as a way to identify *stable* clusterings, which consistently optimize the objective over a range of parameter values [10, 17, 32]. While both are important applications of resolution-based clustering, the ability to detect a specified type of clustering structure is important regardless of a clustering's stability or the hierarchical structure of a network. Finally, while tuning hyperparameters is a standard procedure in the broader machine learning literature, most existing approaches are not specifically designed for tuning graph clustering resolution parameters. Furthermore hyperparameter tuning techniques typically rely on performing cross validation over a large number of training examples. We are concerned with learning good resolution parameters from a *single* example clustering that represents a meaningful partitioning in a certain application domain.

Our Contributions. In this paper we develop an approach for learning how to set resolution parameters for both local and global graph clustering problems. Our results for global graph clustering rely heavily on the LambdaCC framework we developed in past work [37]. We begin by formally defining a parameter fitness function for a given clustering. We then prove that under reasonable assumptions on the input clustering and clustering objective function used, we can find the minimizer of such a fitness function to within arbitrary precision using a simple bisection-like method. Our approach can be viewed as a type of *single-shot* hyperparameter tuning, as we are able to learn an appropriate setting of a resolution parameter when given a single example clustering. We display the utility of our approach in several local and global graph clustering

experiments. Our approach allows us to obtain improved community detection results on synthetic and real-world networks. We also show how our method can be used to measure the correlation between metadata attributes and community structure in social networks.

2 GRAPH CLUSTERING BACKGROUND

This section reviews the global LambdaCC [37] clustering objective and a local clustering objective that is based on regionally biased minimum cut computations [4, 26]. While there do exist many other objectives for local and global clustering, we focus on these two as they both rely crucially on resolution parameters.

Basic Notation. In this paper we consider unweighted and undirected graphs $G = (V, E)$, though many of the ideas can be extended to weighted graphs. Global graph clustering separates G into disjoint sets of nodes so that every node belongs to exactly one cluster. For local clustering, one is additionally given a set of reference or seed nodes $R \subset V$ and the objective is to find a good cluster that shares a nontrivial overlap with R. The degree of a node $i \in V$ is the number of edges incident to it; we denote this by d_i. The volume of a set $S \subseteq V$ is given by $\mathbf{vol}(S) = \sum_{i \in S} d_i$ and $\mathbf{cut}(S)$ measures how many edges cross from S to its complement set $\bar{S} = V \setminus S$. Further notation will be presented as needed in the paper.

2.1 Global Clustering with LambdaCC

The LambdaCC objective is a special case of correlation clustering (CC) [5], a framework for partitioning signed graphs. In correlation clustering, each pair of nodes (u, v) in a signed graph is associated with either a positive edge or a negative edge, as well as a nonnegative edge weight e_{uv} indicating the strength of the relationship between u and v. Given this input, the goal is to produce a clustering which minimizes the weight of *disagreements* or *mistakes*, which occur when positive edges are placed between clusters or negative edges are placed inside clusters.

The LambdaCC framework takes an unsigned graph $G = (V, E)$, a resolution parameter $\lambda \in (0, 1)$, and node weights w_u for each $u \in V$. It converts this input into a signed graph over which the correlation clustering objective can then be minimized. The signed graph $\tilde{G} = (V, E^+, E^-)$ is constructed as follows: for every $(u, v) \in E$, if $1 - \lambda w_u w_v \geq 0$, form a positive edge (u, v) in $\tilde{G}$, otherwise form a negative edge (u, v). In either case, the weight of this edge is $e_{uv} = |1 - \lambda w_u w_v|$. For every non-edge in the original graph $((u, v) \notin E)$, form a negative edge $(u, v) \in E^-$ in $\tilde{G}$ with weight $e_{uv} = \lambda w_u w_v$. The LambdaCC objective function then corresponds to the correlation clustering objective applied to $\tilde{G}$:

$$\min \sum_{(u,v) \in E^+} e_{uv}(1 - \delta_{uv}) + \sum_{(i,j) \in E^-} e_{uv} \delta_{uv}, \quad (1)$$

where δ_{uv} is a zero-one indicator function which encodes whether a clustering has placed nodes u, v together ($\delta_{uv} = 1$), or apart ($\delta_{uv} = 0$). There are two main choices for node weights: $w_u = 1$ for all $u \in V$ is the *standard* LambdaCC objective. For this simple case, we note that a node pair (u, v) which defines an edge in G will always correspond to a positive edge in G. In some applications it is useful to consider a *degree-weighted* version in which $w_u = d_u$. In this case, if $\lambda \leq 1/(d_{max}^2)$ then we can still guarantee that $E = E^+$.

However, for larger values of λ it may be possible that an edge in G gets mapped to a negative edge in $\tilde{G}$.

As a generalization of standard unweighted correlation clustering, LambdaCC is NP-hard, though many approximation algorithms and heuristics for correlation clustering have been developed in practice [2, 5, 8, 9]. In our previous work [37], we showed that a 3-approximation for standard LambdaCC can be obtained for any $\lambda \geq 1/2$ by rounding the following LP relaxation of objective (1):

$$
\begin{aligned}
\text{minimize} \quad & \sum_{(u,v)\in E}(1-\lambda)x_{uv} + \sum_{(u,v)\notin E}\lambda(1-x_{uv}) \\
\text{subject to} \quad & x_{uv} \leq x_{uw} + x_{vw} && \text{for all } u, v, w \\
& 0 \leq x_{uv} \leq 1 && \text{for all } u < v.
\end{aligned} \tag{2}
$$

Furthermore, even when a priori approximations are not guaranteed, solving the LP relaxation can be a very useful way to obtain empirical lower bounds for the objective in polynomial time. In follow up work [15], we provided improved approximations for $\lambda < 1/2$ based on rounding the LP, but noted an $\Omega(\log n)$ integrality gap for some small value of λ.

Equivalence Results. LambdaCC generalizes and unifies a large number of other clustering approaches. When $\lambda = 1/(2|E|)$, the degree-weighted version is equivalent to the popular maximum modularity clustering objective [23, 24]. Standard LambdaCC interpolates between the sparsest cut objective for a graph-dependent small value of λ, and the cluster deletion problem when $\lambda > |E|/(1+|E|)$. Given its relationship to modularity, LambdaCC is known to also be related to the stochastic block model [25] and a multi-cluster normalized cut objective [42].

2.2 Local Clustering Objectives

We next consider a class of clustering objectives that share some similarities with (1), but are designed for finding a single local cluster in a specific region of a large graph. With the input graph $G = (V, E)$, we additionally identify a set of seed or *reference* nodes R around which we wish to form a good community. One common measure for the "goodness" of a cluster S is the conductance objective:

$$
\phi(S) = \mathbf{cut}(S)/(\min\{\mathbf{vol}(S), \mathbf{vol}(\bar{S})\}), \tag{3}
$$

which is small when S is connected very well internally but shares few edges with $\bar{S}$. A number of graph clustering algorithms have been designed to minimize local variants of (3). These substitute the denominator of (3) with a measure of the overlap between an output cluster S and the reference set R. One such objective is the following local conductance measure:

$$
\phi_R(S) = \frac{\mathbf{cut}(S)}{\mathbf{vol}(R \cap S) - \varepsilon\mathbf{vol}(\bar{R} \cap S)}, \tag{4}
$$

which is minimized over all sets S such that the denominator of $\phi_R(S)$ is positive. This objective includes a *locality* parameter ε that controls how much overlap there should be between the seed set and output cluster. For a general overview of this clustering paradigm and its relationship to spectral and random-walk based techniques, we refer the reader to the work of Fountoulakis et al. [14]. Specific algorithms which minimize variants of (4) include FlowImprove [4], which always uses parameter $\varepsilon = \mathbf{vol}(R)/\mathbf{vol}(\bar{R})$, and LocalImprove [26] and SimpleLocal [35], both of which choose larger values of ε in order to keep computations more local. In the extreme case where we consider $\varepsilon = \infty$, the problem reduces to

finding the minimum conductance *subset* of a reference set R, which can be accomplished by the Minimum Quotient Improvement (MQI) algorithm of Lang and Rao [20].

Objective (4) can be efficiently minimized by repeatedly solving a minimum s-t cut problem on an auxiliary graph constructed from G, which introduces a sink node s attached to nodes in R, and a source node t attached to nodes in $\bar{R} = V \setminus R$. Edges are weighted with respect to the locality parameter ε and another parameter α. In order to detect whether there exists some set S with $\phi_R(S) \leq \alpha$, one can solve a local clustering objective corresponding to the minimum s-t cut objective on the auxiliary graph. We refer to this simply as the *local flow clustering objective*:

$$
\min \ f_\alpha(S) = \mathbf{cut}(S) + \alpha\mathbf{vol}(R \cap \bar{S}) + \alpha\varepsilon\mathbf{vol}(\bar{R} \cap S). \tag{5}
$$

If the set S minimizing f_α satisfies $f_\alpha(S) < \alpha\mathbf{vol}(R)$, then rearranging terms one can show that $\phi_R(S) < \alpha$. Thus, by performing binary search over α or repeatedly solving (5) for smaller and smaller α, one can minimize the local conductance measure (4).

Previous research has largely treated α as a temporary parameter used in one step of a larger algorithm seeking to minimize (4). Algorithms which minimize (4) do so by finding the smallest α such that the minimum of (5) is $\alpha\mathbf{vol}(R)$. We depart from this approach by instead treating α as a tunable resolution parameter for balancing two conflicting goals: finding clusters with a small cut, and finding clusters that have a large overlap with the seed set R. In the case where ε is treated as infinitely large and we are simply looking for subsets of a seed set R satisfying $\mathbf{vol}(R) \leq \mathbf{vol}(\bar{R})$, then in effect we are trying to solve the optimization problem:

$$
\min \mathbf{cut}(S) - \alpha\mathbf{vol}(S) + \alpha\mathbf{vol}(R) \quad \text{such that } S \subseteq R. \tag{6}
$$

This goal is related to, but ultimately should be contrasted with, the goal of minimizing the ratio $\mathbf{cut}(S)/\mathbf{vol}(S)$. The objectives are similar in that they both tend to prefer sets with small cut and large volume. We argue that treating α as a tunable parameter is in fact more versatile than simply minimizing the ratio score. In multiple applications it may be useful to find clusters with small cut and large volume, but different applications may put a different weight on each aspect of the objective. We observe that ε also plays an important role in the size and structure of the output community when it is less than ∞. For simplicity, in this paper we can treat this as a fixed constant, and in our experimental section we simply focus on objective (6).

2.3 Parametric Linear Programs

Before moving on we provide key background on parametric linear programming which will be important in our theoretical results. A standard linear program is a problem of the form

$$
\min_{\mathbf{x}} \ \mathbf{c}^T\mathbf{x} \quad \text{such that } \mathbf{A}\mathbf{x} \leq \mathbf{b} \tag{7}
$$

where $\mathbf{c}, \mathbf{b}$ are vectors and $\mathbf{A}$ is a constraint matrix. A *parametric* linear program is a related problem of the form

$$
\min_{\mathbf{x}} \ \mathbf{c}^T\mathbf{x} + \beta(\Delta\mathbf{c})^T\mathbf{x} \quad \text{such that } \mathbf{A}\mathbf{x} \leq \mathbf{b} \tag{8}
$$

where $\Delta\mathbf{c}$ is another vector of the same length as $\mathbf{c}$ and β is a parameter controlling the difference between (7) and (8). We state a well-known result about the solutions of (8) for different β. This result is not new; it follows directly from Proposition 2.3b from [1].

THEOREM 1. *Let $L(\beta)$ be the minimum of (8) for a fixed β. If we are given bounds a and b such that $L(\beta) \in \mathbb{R}$ for all $\beta \in [a, b]$, then L is a piecewise linear and concave function in β over this interval.*

Parametric LPs in Graph Clustering Applications. In our work it is significant to note that the linear programming relaxation of LambdaCC is a parametric linear program in λ. Furthermore, the local flow clustering objective can be cast as a parametric linear program in α, since this objective corresponds simply to a special case of the minimum *s-t* cut problem, which can be cast as an LP.

2.4 Related Work

Our work builds on previous results that introduced generalized objective functions with resolution parameters, including the Hamiltonian objective [29], clustering stability [10], a multiscale variant of the map equation [32], and the LambdaCC framework [37]. Recently Jeub et al. [17] introduced a technique for sampling values of a resolution parameter and applying hierarchical consensus clustering techniques. Our work on learning clustering resolution parameters differs from theirs in that we do not aim to provide hierarchical clusterings of a network. Instead we assume that there is a known fixed clustering, for which we wish to learn a single specific resolution parameter.

There exist many techniques for localized community detection based on seed set expansion. Among numerous others, these include spectral and random-walk based methods [3, 33], flow-based methods [4, 20, 26, 35], and other approaches which perform diffusions from a set of seed nodes and round embeddings via a sweep cut procedure [18, 39]. We build on these by interpreting hyperparameters associated with such methods as resolution parameters which can be learned to produce clusters of a certain type.

3 THEORETICAL RESULTS

The major theoretical contribution of our work is a new framework for learning clustering resolution parameters based on minimizing a parameter fitness function for a given example clustering. We present results for a generic clustering objective and fitness function, and later show how to apply our results to LambdaCC and local flow clustering.

3.1 Problem Formulation

Let C denote a set of valid clusterings for a graph $G = (V, E)$. We consider a generic clustering objective function $f_\beta : C \to \mathbb{R}_{\geq 0}$ that depends on a resolution parameter β. The function takes as input a clustering $C \in C$, and outputs a nonnegative clustering quality score for C. We assume that smaller values of f_β are better. We intentionally allow f_β to be very general in order to develop broadly applicable theory. For intuition, one can think of f_β as being the LambdaCC function (1) with $\beta = \lambda$. Alternatively, one can picture f_β to be the local flow objective (5) with $\beta = \alpha$ and with C representing the set of bipartitions, i.e. for any $C \in C$, $C = \{S, \bar{S}\}$ for some set $S \subset V$.

Given some objective function f_β, a standard clustering paradigm is to assume that an appropriate value of β has already been chosen, and then the goal is to produce some clustering C that exactly or approximately minimizes f_β. In our work, we address an inverse question: given an example clustering C_x, how do we determine a parameter β such that C_x approximately minimizes f_β? Ideally we would like to solve the following problem:

Goal 1: Find $\beta > 0$ such that $f_\beta(C_x) \leq f_\beta(C)$ for all $C \in C$. (9)

In practice, however, C_x may not exactly minimize a generic clustering objective for any choice of resolution parameter. Thus we relax this to a more general and useful goal:

Goal 2: Find the minimum $\Delta \geq 1$ such that for some $\beta > 0$
$$f_\beta(C_x) \leq \Delta f_\beta(C) \text{ for all } C \in C. \tag{10}$$

This second goal is motivated by the study of approximation algorithms for clustering. In effect this asks: if we are given a certain clustering C_x, is C_x a good approximation to f_β for any choice of β? Note that this generalizes (9): if β can be chosen to satisfy Goal 1, then the same β will satisfy Goal 2 with $\Delta = 1$. Furthermore, it has the added advantage that, if solved, Goal 2 will produce a value Δ which communicates how well clusterings like C_x can be detected using variants of the objective function f_β. If Δ is near 1, it means that f_β is able to produce similar clusterings for a correct choice of β, whereas if Δ is very large this indicates that C_x will be difficult to find even for an optimal β, and thus a different approach will be necessary for detecting clusterings of this type.

Clustering Relaxations. While Goal 2 is a more reasonable target than Goal 1, it may still be a very challenging problem to solve when objective f_β is hard to optimize, e.g., if it is NP-hard. We thus consider one final relaxation that is slightly weaker than (10), but will be more feasible to work with. Let $\hat{C}$ denote a superset of C which includes not only clusterings for G, but also some notion of a relaxed clustering, and let $g_\beta : \hat{C} \to \mathbb{R}_{\geq 0}$ be an objective that assigns a score for every $C \in \hat{C}$. Furthermore, assume g_β represents a lower bound function for f_β: $g_\beta(C) \leq f_\beta(C)$ for all β and all $C \in C$. Our consideration of g_β is motivated by the fact that many NP-hard clustering objectives permit convex relaxations, which can be optimized in polynomial time over a larger set of relaxed clusterings that contain all valid clusterings of G as a subset. For example, the LambdaCC objective is NP-hard to optimize for every $\lambda \in (0, 1)$, but the linear programming relaxation for every λ can be solved in polynomial time, and is defined over relaxed clusterings in which pairs of nodes are assigned distances between 0 and 1. These relaxations can be rounded to produce good approximations to the original NP-hard objective [8, 9]. Since g_β is indeed easier to optimize than f_β, the following goal will be easier to approach but still provide strong guarantees for learning a good value of β:

Goal 3: Find the minimum $\Delta \geq 1$ such that for some $\beta > 0$
$$f_\beta(C_x) \leq \Delta g_\beta(C) \text{ for all } C \in \hat{C}. \tag{11}$$

If we can solve (11), this still guarantees that C_x is a Δ-approximation to f_β for an appropriately chosen β. For problems where f_β is very challenging to optimize, but g_β is not, this will be a much more feasible approach. In the next section we will focus on developing theory for addressing Goal 3, though we note that in applying this theory we can still choose $g_\beta = f_\beta$ and therefore instead address the stronger Goal 2 whenever this is feasible. We will take this approach when applying our theory to the local flow objective.

3.2 Parameter Fitness Function

We now present a parameter fitness function whose minimization is equivalent to solving (11). Functions f_β and g_β take a clustering or relaxed clustering as input and output an objective score. However, we wish to view β as an input parameter and we treat an example clustering C_x as a fixed input. Thus for convenience we introduce new related functions:

$$F(\beta) = f_\beta(C_x) \qquad (12)$$

$$G(\beta) = \min_{C \in \hat{C}} g_\beta(C) \qquad (13)$$

The ratio of these two functions defines the *parameter fitness function* that we seek to minimize:

$$\mathcal{P}(\beta) = \frac{F(\beta)}{G(\beta)}. \qquad (14)$$

Observe that this function is always greater than or equal to 1 since $G(\beta) \leq F(\beta)$ for any β. The minimizer of $\mathcal{P}$ is a resolution parameter β that minimizes the ratio between the clustering score of a fixed C_x and a lower bound on f_β. Thus, by minimizing (14) we achieve Goal 3 in (11) with $\Delta = \min_\beta \mathcal{P}(\beta)$.

In Section 2.3, we noted that the local flow clustering objective can be characterized as a parametric linear program, as can the LP relaxation of LAMBDACC. Furthermore, for a fixed clustering, both objective functions can be viewed as a linear function in terms of their resolution parameter. Motivated by these facts, we present a theorem which characterizes the behavior of the parameter fitness function $\mathcal{P}$ under certain reasonable conditions on the functions F and G. In the subsequent section we will use this result to show that $\mathcal{P}$ can be minimized to within arbitrary precision using an efficient bisection-like method.

THEOREM 2. *Assume $F(\beta) = a + b\beta$ for nonzero real numbers a and b. Let G be concave and piecewise linear in β, and assume $F(\beta) \geq G(\beta) \geq 0$ for all $\beta \in [\ell, r]$ where ℓ and r are nonnegative lower and upper (i.e. left and right) bounds for β. Then $\mathcal{P}$ satisfies the following two properties:*

(a) If $\beta^- < \beta < \beta^+$, then $\mathcal{P}(\beta)$ cannot be strictly greater than both $\mathcal{P}(\beta^-)$ and $\mathcal{P}(\beta^+)$.

*(b) If $\mathcal{P}(\beta^-) = \mathcal{P}(\beta^+)$, then $\mathcal{P}$ achieves its minimum in $[\beta^-, \beta^+]$.*

PROOF. Note that for some $\gamma \in (0, 1)$, $\beta = (1 - \gamma)\beta^+ + \gamma\beta^-$. By concavity of G and linearity of F, we know

$$\mathcal{P}(\beta) = \frac{F((1-\gamma)\beta^+ + \gamma\beta^-)}{G((1-\gamma)\beta^+ + \gamma\beta^-)} \leq \frac{(1-\gamma)F(\beta^+) + \gamma F(\beta^-)}{(1-\gamma)G(\beta^+) + \gamma G(\beta^-)}$$

$$\leq \max\left\{ \frac{(1-\gamma)F(\beta^+)}{(1-\gamma)G(\beta^+)}, \frac{\gamma F(\beta^-)}{\gamma G(\beta^-)} \right\} = \max\left\{ \mathcal{P}(\beta^+), \mathcal{P}(\beta^-) \right\},$$

which proves the first property. Now assume that $\mathcal{P}(\beta^-) = \mathcal{P}(\beta^+)$. Using property 1, we know as β increases from its lower to upper limit, $\mathcal{P}$ cannot increase and then decrease. Thus, either $\mathcal{P}$ attains its minimum on $[\beta^-, \beta^+]$, else $\mathcal{P}$ is a constant for all $\beta \in [\beta^-, \beta^+]$. If the latter is true, then for some $\beta \in [\beta^-, \beta^+]$ and some sufficiently small $\epsilon > 0$, G must be linear in the range $(\beta - \epsilon, \beta + \epsilon)$, since we know that G is piecewise linear. Therefore, $G(\beta) = c + d\beta$ and

$$\mathcal{P}(\beta) = (a + b\beta)/(c + d\beta) = constant \qquad (15)$$

for $\beta \in (\beta - \epsilon, \beta + \epsilon)$ and for some $c, d \in \mathbb{R}$. This ratio of linear functions can only be a constant if $a = c = 0$, or $b = d = 0$, or if
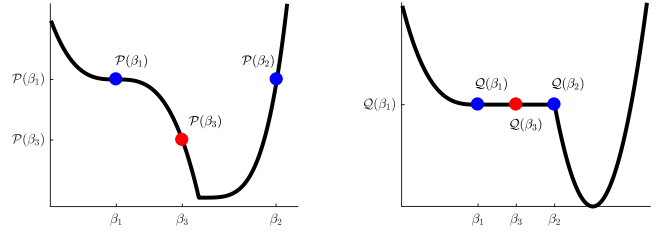
Figure 1: Function $\mathcal{P}$ satisfies both properties (a) and (b) in Theorem 2. If $\mathcal{P}(\beta_1) = \mathcal{P}(\beta_2)$, querying $\mathcal{P}$ at any point $\beta_3 \in [\beta_1, \beta_2]$ gets us closer to a minimizer. Function Q only satisfies property (a). If $Q(\beta_1) = Q(\beta_2)$, we can get stuck making queries inside a flat region of Q not near a minimizer.

$a = c$ and $b = d$. Since we assumed a and b were nonzero, the last case must hold, and thus $\mathcal{P}(\beta) = 1$ for every $\beta \in [\beta^-, \beta^+]$, so the minimizer is obtained in this case, since $\mathcal{P}(\beta) \geq 1$ for all β. $\square$

In the next section we present a method for finding the minimizer of a function satisfying properties (a) and (b) in Theorem 2 to within arbitrary precision. Before doing so, we highlight the importance of ensuring that *both* properties hold. In Figure 1 we plot two toy functions, $\mathcal{P}$ and Q. Although both satisfy property (a), only $\mathcal{P}$ additionally satisfies (b). Assume we do not have explicit representations of either function, but we can query them at specific points to help find their minimizers. Consider Figure 1. If we query $\mathcal{P}$ at points β_1 and β_2 to find that $\mathcal{P}(\beta_1) = \mathcal{P}(\beta_2)$, then choosing any third point $\beta_3 \in (\beta_1, \beta_2)$ will get us closer to the minimizer. However, if $Q(\beta_1) = Q(\beta_2)$ for some β_1, β_2, we cannot be sure these points are not part of a flat region of Q somewhere far from the minimizer. It thus becomes unclear how to choose a third point β_3 at which to query Q. If we choose some $\beta_3 \in (\beta_1, \beta_2)$ and find that $Q(\beta_3) = Q(\beta_2) = Q(\beta_1)$, the minimizer may be within $[\beta_1, \beta_2]$, within $[\beta_2, \beta_3]$, or in a completely different region. Thus it is important for the denominator of a parameter fitness function to be piecewise linear in addition to being concave, since this piecewise linear assumption guarantees property (b) will hold.

3.3 Minimizing $\mathcal{P}$

We now outline an approach for finding a minimizer of $\mathcal{P}$ to within arbitrary precision when Theorem 2 holds. Our approach is closely related to the standard bisection method for finding zeros of a continuous function f. Recall that standard bisection starts with a and b such that $sign(f(a)) \neq sign(f(b))$, and then computes $f(c)$ where $c = (a + b)/2$. Checking the sign of $f(c)$ allows one to determine whether the zero of f is located within the interval $[a, c]$ or $[b, c]$. Thus each new query of the function f halves the interval in which a zero must be located.

Assume $\mathcal{P}$ satisfies properties (a) and (b) in Theorem 2 over an interval $[\ell, r]$. To satisfy Goal 3, given in (11) in Section 3.1, it suffices to find any minimizer of $\mathcal{P}$, which we do by repeatedly halving the interval in which the minimizers of $\mathcal{P}$ must lie. Our approach differs from standard bisection in that we are trying to find a *minimizer* instead of the zero of some function. The key algorithmic difference is that querying $\mathcal{P}$ at a single point between two bounds will not always be sufficient to cut the search space in
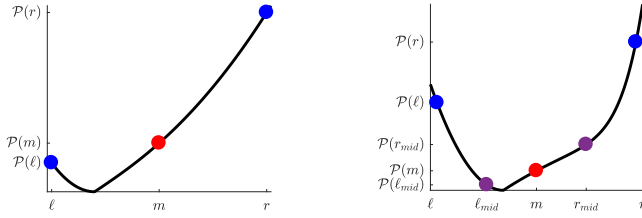
Figure 2: We evaluate $\mathcal{P}$ at left and right bounds (blue points), and at a midpoint m (red point). Left: If $\mathcal{P}(\ell) < \mathcal{P}(m) < \mathcal{P}(r)$, then we know the minimizer of $\mathcal{P}$ is in $[\ell, m]$, and we recursively call the *one-branch* phase (Algorithm 1) with new bounds ℓ and m. Right: If $\mathcal{P}(m) < \mathcal{P}(\ell) < \mathcal{P}(r)$, we don't know if the minimizer is in the left branch $[\ell, m]$ or right branch $[m, r]$. Evaluating $\mathcal{P}$ at the midpoint of each branch (purple points), we rule out branch $[m, r]$ and recursively call Algorithm 2 with new endpoints ℓ and m and midpoint ℓ_{mid}.

half. Consider Figure 2. Our method starts in a *one-branch* phase in which we know a minimizer lies between ℓ and r. If we compute $m = (\ell + r)/2$ and find that $\mathcal{P}(m)$ is between $\mathcal{P}(\ell)$ and $\mathcal{P}(r)$, this does in fact automatically cut our search space in half, as this implies that $\mathcal{P}$ is monotonic on either $[\ell, m]$ or $[m, r]$. However, if $\mathcal{P}(m) < \min\{\mathcal{P}(\ell), \mathcal{P}(r)\}$, then it is possible for the minimizer to reside within either the left branch $[\ell, m]$ or the right branch $[m, r]$. In this case, the method enters a *two-branch* phase in which it takes the midpoint of each branch ($\ell_{mid} = (\ell+m)/2$ and $r_{mid} = (m+r)/2$) and evaluates $\mathcal{P}(\ell_{mid})$ and $\mathcal{P}(r_{mid})$. If $\mathcal{P}$ returns the same value for two of the inputs (e.g., $\mathcal{P}(\ell) = \mathcal{P}(m)$), then by property (b) we have found a new interval containing the minimizer(s) of $\mathcal{P}$ that is at most half the length of $[\ell, r]$. Otherwise, we can use property (a) to deduce that the minimizer will be located within $[\ell, m]$, $[m, r]$, or $[\ell_{mid}, r_{mid}]$, and we recurse on the two-branch phase.

Algorithms 1 and 2 handle the one- and two-branch phases of the method respectively. The guarantees of our method are summarized in Theorem 3. We omit the full proof, since it follows directly from considering different simple cases and applying properties of $\mathcal{P}$ to halve the search space as outline above.

Theorem 3. *Consider a fixed clustering C_x and a corresponding parameter fitness function $\mathcal{P}_{C_x}$ satisfying the assumptions of Theorem 2. Running Algorithm 1 with input ℓ, r and a tolerance ϵ will produce a resolution parameter $\tilde{\beta}$ that is within ϵ of the minimizer of $\mathcal{P}_X$ over the interval $[\ell, r]$, in at most $\log_2((r - \ell)/\epsilon)$ recursive calls.*

4 APPLICATION TO SPECIFIC OBJECTIVES

Theorem 2 and our approach for minimizing $\mathcal{P}$ can be immediately applied to learn resolution parameters for the LambdaCC global clustering objective and the local flow clustering objective.

4.1 Local Clustering

For local clustering we consider the objective function f_α given in (5) and note that the set of valid clusterings C is the set of bipartitions. The example clustering we are given at the outset of the problem is $C_x = \{X, \bar{X}\}$ where $X \subset V$ is some nontrivial set of nodes representing a "good" cluster for a given application. We assume we

---

Algorithm 1 CheckOneBranch(ℓ, r, ϵ)

    Base case:
    if $r - \ell < \epsilon$ **then**
        return ℓ
    Recursive call:
5: Midpoint: $m = (\ell + r)/2$
    switch ℓ, m, r **do**
        case $\mathcal{P}(\ell) = \mathcal{P}(m) = \mathcal{P}(r)$
            return m
        case $\mathcal{P}(\ell) \leq \mathcal{P}(m) < \mathcal{P}(r)$
10:        **return** CheckOneBranch(ℓ, m, ϵ)
        case $\mathcal{P}(\ell) > \mathcal{P}(m) \geq \mathcal{P}(r)$
            return CheckOneBranch(m, r, ϵ)
        case $\mathcal{P}(\ell) > \mathcal{P}(m) < \mathcal{P}(r)$
            return CheckTwoBranches(ℓ, m, r, ϵ)

Algorithm 2 CheckTwoBranches(ℓ, m, r, ϵ)

    Base case:
    if $r - \ell < \epsilon$ **then**
        return m
    Recursive call:
5: Left midpoint: $\ell_{mid} = (\ell + m)/2$
    Right midpoint: $r_{mid} = (m + r)/2$
    switch ℓ_{mid}, m, r_{mid} **do**
        case $\mathcal{P}(\ell_{mid}) = \mathcal{P}(m) = \mathcal{P}(r_{mid})$
            return m
10:        **case** $\mathcal{P}(\ell_{mid}) = \mathcal{P}(m) \neq \mathcal{P}(r_{mid})$
            return CheckOneBranch(ℓ_{mid}, m, ϵ)
        case $\mathcal{P}(\ell_{mid}) \neq \mathcal{P}(m) = \mathcal{P}(r_{mid})$
            return CheckOneBranch(m, r_{mid}, ϵ)
        case $\mathcal{P}(\ell_{mid}) < \mathcal{P}(m) < \mathcal{P}(r_{mid})$
15:        **return** CheckTwoBranches($\ell, \ell_{mid}, m, \epsilon$)
        case $\mathcal{P}(\ell_{mid}) > \mathcal{P}(m) > \mathcal{P}(r_{mid})$
            return CheckTwoBranches(m, r_{mid}, r, ϵ)
        case $\mathcal{P}(\ell_{mid}) > \mathcal{P}(m) < \mathcal{P}(r_{mid})$
            return CheckTwoBranches($\ell_{mid}, m, r_{mid}, \epsilon$)

---

are also given a reference set R (with $\mathbf{vol}(R) \leq \mathbf{vol}(\bar{R})$) that defines a region of the graph in which we are searching for clusters. As noted previously, f_α can be viewed as a parametric linear program, and furthermore it will evaluate to a non-negative finite number for any $\alpha > 0$. Thus by Theorem 1, $G(\alpha) = \min_S f_\alpha(S)$ is concave and piecewise linear and we can apply Theorem 2. More explicitly, the local clustering parameter fitness function is

$$\mathcal{P}_X(\alpha) = \frac{\mathbf{cut}(X) + \alpha\mathbf{vol}(\bar{X} \cap R) + \alpha\varepsilon\mathbf{vol}(X \cap \bar{R})}{\min_S [\mathbf{cut}(S) + \alpha\mathbf{vol}(\bar{S} \cap R) + \alpha\varepsilon\mathbf{vol}(S \cap \bar{R})]}. \quad (16)$$

If we focus on finding clusters that are subsets of R, using objective (6), we have a simplified fitness function:

$$\mathcal{P}_X(\alpha) = \frac{\mathbf{cut}(X) - \alpha\mathbf{vol}(X) + \alpha\mathbf{vol}(R)}{\min_{S \subseteq R}[\mathbf{cut}(S) - \alpha\mathbf{vol}(S) + \alpha\mathbf{vol}(R)]}. \quad (17)$$

When we apply Algorithm 1 to minimize (16) or (17), we can query $\mathcal{P}_X$ in the time it takes to evaluate a linear function and the time it

takes to solve the *s-t* cut problem (5). This can be done extremely quickly using localized min-cut computations [20, 26, 35, 38].

Functions (16) and (17) should be minimized over $\alpha \in [\alpha^*, \mathbf{cut}(R)]$, where α^* is either the minimum of (4) if we are minimizing (16), or is the minimum conductance for a subset of R if we are minimizing (17). One can show that for any α outside this range, objectives (5) and (6) will be trivially minimized by $S = R$, so it is not meaningful to optimize these objectives for these α. In practice one can additionally set stricter upper and lower bounds if desired.

4.2 Global Clustering Approach

We separately consider the standard and degree-weighted versions of LAMBDACC when applying Theorem 2 to global graph clustering.

Standard LAMBDACC. For the standard objective, it is useful to consider the scaled version of LAMBDACC obtained by dividing (1) by $1 - \lambda$ and substituting for a new resolution parameter $\gamma = \lambda/(1 - \lambda)$. Then the objective is

$$\min \ \textstyle\sum_{(u,v)\in E}(1 - \delta_{uv}) + \sum_{(u,v)\notin E}\gamma\delta_{uv}. \tag{18}$$

The denominator of the parameter fitness function for this scaled LAMBDACC problem would be

$$G(\gamma) = \min_{\mathbf{x}\in\mathcal{X}} \ \textstyle\sum_{(u,v)\in E} x_{uv} + \sum_{(u,v)\notin E}\gamma(1 - x_{uv}) \tag{19}$$

where $\mathcal{X}$ represents the set of linear constraints for the linear program (2). Note that $G(\gamma)$ will be finite for every $\gamma \geq 0$, so Theorem 1 holds. Thus G is concave and piecewise linear as required by Theorem 2. Next, for a fixed clustering C_x, let P_x be the number of positive mistakes (pairs of nodes that are separated despite sharing an edge) and N_x be the number of negative mistakes (pairs of nodes that are clustered together but share no edge). Then objective (18) for this clustering is $P_x + \gamma N_x$, and we see that this fits the linear form given in Theorem 2 as long as the example clustering satisfies $P_x > 0$ and $N_x > 0$, which will be the case for nearly any nontrivial clustering one might consider. Finally, note that the parameter fitness function for (18) would be exactly the same as the parameter fitness function for the standard LAMBDACC objective, since scaling by $(1 - \lambda)$ makes no difference if we are going to minimize the ratio between the clustering objective and its LP relaxation. The parameter fitness function for standard LAMBDACC is therefore

$$\mathcal{P}_{C_x}(\lambda) = \frac{(1 - \lambda)P_x + \lambda N_x}{\min_{\mathbf{x}}\left[\sum_{uv\in E}(1 - \lambda)x_{uv} + \sum_{uv\notin E}\lambda(1 - x_{uv})\right]} \tag{20}$$

and it satisfies the assumptions of Theorem 2 as long as $P_x > 0$, $N_x > 0$, and we optimize over $\lambda \in (0, 1)$.

Degree-weighted LAMBDACC. Showing how Theorem 2 applies to degree-weighted LAMBDACC requires slightly more work, though the same basic principles hold. The LP-relaxation of the objective is still a parametric linear program, thus is still concave and piecewise linear in λ over the interval $(0, 1)$. The denominator of the parameter fitness function in this case would be:

$$\min \ \textstyle\sum_{(u,v)\in E^+} e_{uv}(1 - \delta_{uv}) + \sum_{(u,v)\in E^-} e_{uv}\delta_{uv}. \tag{21}$$

where e_{uv} is defined in the degree-weighted fashion (see Section 2.1). For a fixed example clustering C_x encoded by a function $\delta_x = (\delta_{uv})$, we can rearrange this into the form $a + \lambda b$ where $a = \sum_{(u,v)\in E}(1 - \delta_{uv})$ and $b = \sum_{(u,v)\notin E} d_u d_v \delta_{uv} - \sum_{(u,v)\in E} d_u d_v(1 -$
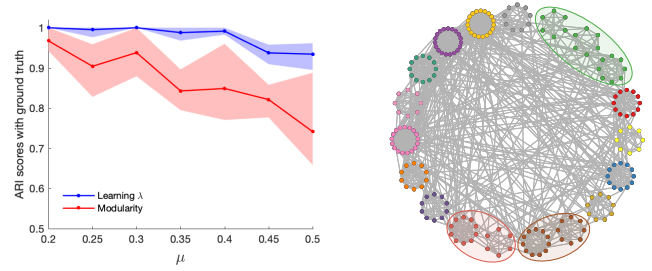
Figure 3: Left: ARI scores for detecting ground truth in LFR graphs. Solid lines indicate mean scores, and colored regions show the range of scores across 5 test graphs for each μ. Right: one of the 5 LFR test graphs for $\mu = 0.3$. Modularity ($\lambda = 1/(2|E|)$) makes mistakes by putting distinct ground truth clusters together (highlighted). For this example our approach perfectly detects the ground truth.

δ_{uv}). These values are simple to compute, and as long as they are both nonzero, the results of Theorem 2 apply. In some extreme cases it is possible that $a = 0$ or $b = 0$, but we expect this to be rare. Furthermore, our general approach may still work even when $a = 0$ or $b = 0$, Theorem 2 simply does not analyze this case. We leave it as future work to develop more refined sufficient and necessary conditions such that Algorithm 1 is guaranteed to minimize $\mathcal{P}$.

5 EXPERIMENTS

We consider several local and global clustering experiments in which significant benefit can be gained from learning resolution parameters rather than using previous off-the-shelf algorithms and objective functions. We implement Algorithms 1 and 2 in the Julia programming language for both local and global parameter fitness functions. Computing the LAMBDACC linear programming relaxation can be challenging due to the size of the constraint set. For our smaller graphs we apply Gurobi optimization software, and for larger problems we use recently developed memory-efficient projection methods [30, 36]. For the local-flow objective we use a fast Julia implementation we developed in recent work [38]. Our experiments were run on a machine with two Intel Xeon E5-2690 v4 processors. Code for our experiments and algorithms are available at https://github/nveldt/LearnResParams.

5.1 Learning Parameters for Synthetic Datasets

Although modularity is a widely-applied objective function for community detection, Fortunato and Barthélemy [12] demonstrated that it is unable to accurately detect communities below a certain size threshold in a graph. In our first experiment we demonstrate that learning resolution parameters for LAMBDACC allows us to overcome the *resolution limit* of modularity, and better detect community structure in synthetic networks. We generate a large number of synthetic LFR benchmark graphs [19], in a parameter regime that is chosen to be difficult for modularity. All graphs contain 200 nodes, average degree 10, max degree 20, and community sizes between 5 and 20 nodes. We test a range of mixing parameters μ, which controls the fraction of edges that connect nodes in different communities ($\mu = 0$ means all edges are inside the communities).

For each μ from 0.2 to 0.5, in increments of 0.05, we generate six LFR networks, one for training and five for testing. On the training graph, we minimize the degree-weighted LambdaCC parameter fitness function to learn a resolution parameter λ_{best}. This takes between roughly half an hour (for $\mu = 0.2$) to just over three hours (for $\mu = 0.5$), solving the underlying LambdaCC LP with Gurobi software. We then cluster the five test LFR examples using a generalized version of the Louvain method [7], as implemented by Jeub et al. [16]. We separately run the method with two resolution parameters: $\lambda = 1/(2|E|)$, the standard setting for modularity, and $\lambda = \lambda_{best}$. Learning λ_{best} significantly improves adjusted Rand index (ARI) scores for detecting the ground truth (see Figure 3).

5.2 Local Community Detection

Next we demonstrate that a small amount of semi-supervised information about target communities in real-world networks can allow us to learn good resolution parameters, leading to more robust community identification. Additionally, minimizing the parameter fitness function provides a way to measure the extent to which *functional* communities in a network correspond to topological notions of community structure in networks.

Data. We consider four undirected networks, DBLP, Amazon, Orkut, LiveJournal, which are all available on the SNAP repository [22], and come with sets of nodes that can be identified as "functional communities" (see Yang and Leskovec [40]). For example, members of the social network Orkut may explicitly identify as being part of a user-formed group. Such user groups can be viewed simply as metadata about the network, though these still correspond to some notion of community organization that may be desirable to detect. Following an approach taken in previous work [38], we specifically consider the ten largest communities from the 5000 best functional communities as identified by Yang and Leskovec [40].

Experimental Setup and Results. We treat each functional community as an example cluster X. We build a superset of nodes R by growing X from a breadth first search until we have a superset of size $5|X|$, breaking ties arbitrarily. The size of R is chosen so that it comprises a localized region of a large graph, but is still significantly larger than the target cluster X hidden inside of it. We compare two approaches for detecting X within R. As a baseline approach we extract the best conductance subset of R. Then as our new approach we assume we are given $\mathbf{cut}(X)$ and $\mathbf{vol}(X)$ as additional semi-supervised information. This allows us to minimize the parameter fitness function (17), without knowing what X is. This outputs a resolution parameter α_X, and we then minimize $\mathbf{cut}(S) - \alpha_X \mathbf{vol}(S) + \alpha \mathbf{vol}(R)$ over $S \subseteq R$ to output a set S_X.

Table 1 reports conductance, set size, runtimes, and F1 scores for both approaches, averaged over the ten communities in each network. Learning resolution parameters leads to significantly better F1 scores on every dataset. Additionally, learning resolution parameters for local clustering can be done much more quickly than learning λ for LambdaCC.

New Insights. In addition to improving semi-supervised community detection, minimizing $\mathcal{P}_X$ allows us to measure how well a functional community matches the topological notion of a cluster.
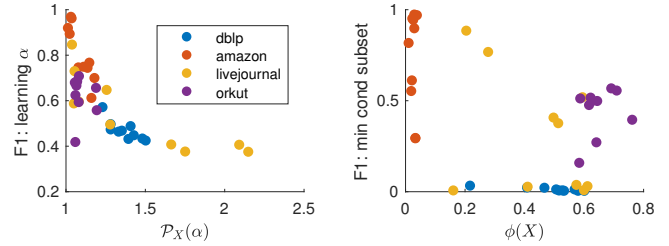
Figure 4: Left: F1 scores for detecting clusters X by learning α vs. the minimum of $\mathcal{P}_X$. Right: F1 scores obtained by finding $\min_{S \subseteq R} \phi(S)$ vs. $\phi(X)$. See the main text for discussion.

Figure 4 shows a scatter plot of F1 community recovery scores against the minimum of $\mathcal{P}_X$ for each experiment from Table 1. We note a downward sloping trend: small values of $\mathcal{P}_X$ near 1 tend to indicate that a cluster is highly "detectable," whereas a higher value of $\mathcal{P}_X$ gives some indication that the functional community may not in fact correspond to a good structural community. We also plot the F1 recovery scores for finding the minimum conductance subset of R against the conductance of functional communities. In this case we do not see any clear pattern, and we learn very little about the relationship between structural and functional communities.

5.3 Meta-Data and Global Clustering

Next we use our techniques to measure how strongly metadata attributes in a network are associated with actual community structure. In general, sets of nodes sharing metadata attributes should not be viewed as "ground truth" clusters [27], although they may still shed light on the underlying clustering structure of a network.

Email Network. We first consider the largest connected component of an email network [21, 41]. Each of the 986 nodes in the graph represents a faculty member at a European university, and edges represent email correspondence between members. We remove edge weights and directions, and consider an example clustering C_x formed by assigning faculty in the same academic department to the same cluster. We use our bisection method to approximately minimize the global parameter fitness function for the degree-weighted LambdaCC objective. We run our method until we find the best resolution parameter to within a tolerance of 10^{-8}, yielding a resolution parameter $\lambda_x = 6.5 \times 10^{-5}$ and a fitness score of $\mathcal{P}_{C_x}(\lambda_x) = 1.34$.

To assess how good or bad a score of 1.34 is for this particular application, we construct a new fake metadata attribute by performing a random permutation of the department labels, which gives a clustering C_{fake}. Approximately minimizing $\mathcal{P}_{C_{fake}}$ yields a resolution parameter $\lambda_{fake} = 3.25 \times 10^{-5}$ and a score $\mathcal{P}_{C_{fake}}(\lambda_{fake}) = 2.16$. The gap between the minima of $\mathcal{P}_{C_{fake}}$ and $\mathcal{P}_{C_x}$ indicates that although the true metadata partitioning does not perfectly map to clustering structure in the network, it nevertheless shares some meaningful correlation with the network's connectivity patterns. To further demonstrate this, we run the generalized Louvain algorithm [7, 16], using the resolution parameters λ_x and λ_{fake}. Running the clustering heuristic with λ_x outputs a clustering that has a normalized mutual information score (NMI) of 0.71 and an adjusted Rand index (ARI) score of 0.55 with C_x. Using λ_{fake}, we get NMI and ARI scores of only 0.05 and 0.003 respectively when comparing with C_{fake}.

Table 1: Output statistics on SNAP datasets for finding the minimum conductance (mc) subset, and learning a resolution parameter (lr). Runtime is given in seconds. We also provide average size ($|T|$) and conductance ($\phi(T)$) for target communities.

Graph	$|V|$	$|E|$	$|T|$	$\phi(T)$	F1		ϕ		run.		size	
					mc	lr	mc	lr	mc	lr	mc	lr
DBLP	317,080	1,049,866	3902	0.49	0.01	**0.47**	0.02	0.16	4.9	11.4	31	11680
Amazon	334,863	925,872	190	0.03	0.73	**0.80**	0.00	0.02	0.3	0.7	142	288
LiveJournal	3,997,962	34,681,189	988	0.45	0.30	**0.54**	0.06	0.10	13.7	31.3	1556	2940
Orkut	3,072,441	117,185,083	3877	0.65	0.44	**0.62**	0.42	0.46	129.8	272.6	2353	5727

Social Networks. We repeat the above experiment on the smallest social network in the Facebook 100 datasets [34], Caltech36. This network is a subset of Facebook with $n = 769$ nodes, defined by users at the California Institute of Technology at a certain point in September 2005. Every node in the network comes with anonymized metadata attributes reporting student/faculty status, gender, major, second major, residence, graduation year, and high school. We treat each metadata attribute as an example clustering C_x. Any node with a value of 0 for an attribute we treat as its own cluster, as this indicates the node has no label for the given attribute. We do not run Algorithm 1 for each individual C_x, since this would involve redundant computations of the LAMBDACC LP relaxations for many of the same values of λ. Instead, we evaluate the denominator of $\mathcal{P}$, which is the same for all example clusterings, at 20 equally spaced λ values between $1/(8|E|)$ and $2/(|E|)$. We set values of λ to be inversely proportional to the number of edges, since we expect the effect of a resolution parameter to depend on a network's size. We note for example that the resolution parameter corresponding to modularity is $\lambda = 1/(2|E|)$, which is also inversely proportional to $|E|$. Computing all of the LP bounds is the bottleneck in our computations, and takes just under 2.5 hours using a recently developed parallel solver for the correlation clustering relaxation [30].

Having evaluated the denominator of $\mathcal{P}$ at these values, we can quickly find the minimizer of $\mathcal{P}$ for each metadata attribute and a permuted fake metadata attribute to within an error of less than 10^{-5}. The smallest values of the parameter fitness function $\mathcal{P}$ for both real and permuted (fake) metadata attributes are given below:

	S/F	Gen	Maj.	Maj. 2	Res.	Yr	HS
$\min \mathcal{P}_{real}$	1.30	1.73	2.03	2.12	1.35	1.57	2.11
$\min \mathcal{P}_{fake}$	1.65	1.80	2.12	2.12	2.11	2.09	2.12

We note that the smallest values of $\mathcal{P}$, as well as the largest gap between $\mathcal{P}$ for true and fake metadata clusterings, are obtained for the student/faculty status, residence, and graduation year attributes. This indicates that these attributes share the strongest correlation with the community structure at this university, which is consistent with independent results on the Facebook 100 datasets [34, 37].

5.4 Local Clustering in Social Networks

In our final experiment we continue exploring the relationship between metadata and community structure in Facebook 100 datasets. We find that minimizing a *local* parameter fitness function $\mathcal{P}$ can be a much better way to measure the community structure of a set of nodes than simply considering the set's conductance.

Data. We perform experiments on all Facebook 100 networks, focusing on the student/faculty status, gender, residence, and graduation year metadata attributes. For the Caltech dataset in the last experiment, these attained the lowest scores for a *global* parameter fitness function, and furthermore these are the only attributes with a significant number of sets with nontrivial conductance. For the graduation year attribute, we focus on classes between 2006 to 2009, since these correspond to the four primary classes on each campus when the networks were crawled in September of 2005 [34].

Experimental Setup. We return to an approach similar to our first experiment. For each network and metadata attribute, we consider sets of nodes identified by the same metadata label, e.g., X may represent all students in the class of 2008 at the University of Chicago. We will refer to these simply as *metadata sets*. A label of zero indicates no attribute is known, so we ignore these sets. We also discard sets that are larger than half the graph, or smaller than 20 nodes. We restrict to considering metadata sets with conductance at most 0.7, since conductance scores too close to 1 indicate that a set has little to no meaningful connectivity pattern. For each remaining metadata set X, we grow a superset R around X using a breadth first search, and stop growing when R contains half the nodes in the graph or is three times the size of X. We then minimize $\mathcal{P}_X$ as given by (17) to learn a resolution parameter α_X. This allows us to find $S_X = \mathrm{argmin}_{S \subseteq R} \mathbf{cut}(S) - \alpha_X \mathbf{vol}(S)$, and we then compute the F1 score between S_X and X. Our goal here is not to develop a new method for community detection. Rather, computing the F1 score and the minimum of $\mathcal{P}_X$ provide ways to measure how well a metadata set conforms to a topological notion of community structure, and how detectable the set is from an algorithmic perspective.

Results. While computing conductance scores provides a good first order measure of a node's community structure, we find that minimizing $\mathcal{P}$ provides more refined information for the detectability of clusters. In Figure 5 we show scatter plots of F1 detection scores against both $\min \mathcal{P}_X$ as well as $\phi(X)$ for each metadata set X. We see that especially for the gender and residence metadata sets across all networks, there is a much clearer relationship between F1 scores and $\min \mathcal{P}_X$. Values of $\mathcal{P}_X$ very close to 1 map to F1 scores near 1, and as $\mathcal{P}_X$ increases we see a downward sloping trend in F1 scores. In the conductance plot we do not see the same trend.

Figures 5c and 5d show results for metadata sets associated with the 2006-2009 graduation years. For this attribute there appears to be a relationship between both conductance and the $\min \mathcal{P}$ scores. Furthermore, in both plots we see a separation of the points roughly into two clusters. A deeper exploration of these trends reveals that
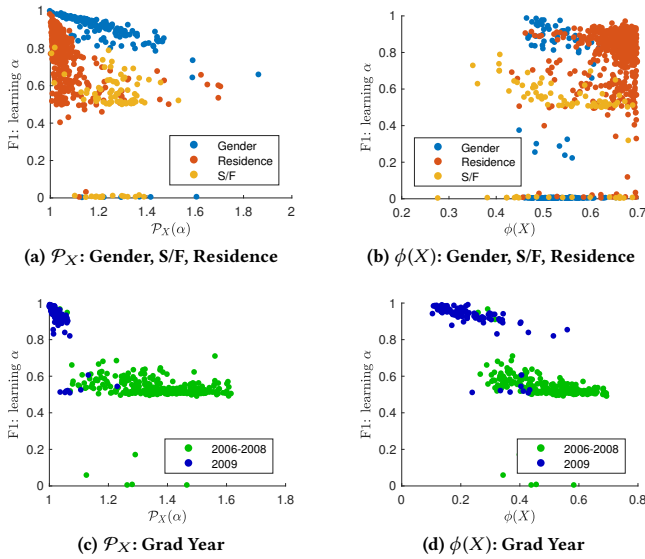
Figure 5: Minimizing $\mathcal{P}_X$ gives insight into the connectivity structure of different metadata sets in Facebook 100 datasets. Plotting F1 detection scores against the minimum of $\mathcal{P}_X$ shows especially clear trends for the gender, residence, and year attributes. Plots (c) and (d) highlight an anomaly in the connectivity patterns of the 2009 graduating year classes.
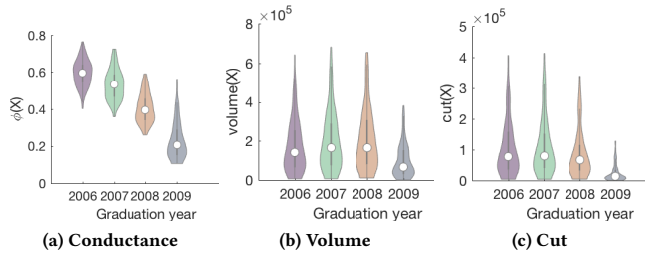
Figure 6: Small conductance and cut scores in Facebook 100 datasets indicate that freshman in 2005 were largely connecting on Facebook with people in their same class.

the 2009 graduation class accounts for the majority of one of these two clusters, and there appears to be an especially clear trend between F1 detection scores and both $\phi(X)$ and $\mathcal{P}_X$ for this class.

New Insights. Figure 6 shows violin plots for $\phi(X)$, **cut**(X), and **vol**(X) for metadata sets associated with graduation years from 2006 to 2009. Overall, conductance decreases as graduation year increases. We notice that class sizes for the 2009 graduation year are much smaller on average. When these datasets were generated, Facebook users needed a .edu email address to register an account. Thus, in September 2005, the graduation class of 2009 was made up primarily of new freshman who just started college, many of whom had not registered a Facebook account yet. Interestingly, we see a slight decrease in the median cut score from 2007 to 2008, and a significant decrease from 2008 to 2009 (Figure 6c). This suggests

that although there were fewer freshman on Facebook at the time, on average they had a greater tendency to establish connections on Facebook among peers in their same graduation year.

Figure 6 suggests that in the early years of Facebook, with each new year, students in the same graduating class tended to form tighter Facebook circles with members in their own class. To further explore this hypothesis, for each of the 100 Facebook datasets we consider each node from a graduating class between 2006 and 2009. In each network we compute the average *in-class connection ratio*, i.e., the number of Facebook friends each person has inside the same graduating class, divided by the total number of Facebook connections that the person has across the entire university. In 97 out of 100 datasets (all networks except Caltech36, Hamilton46, and Santa74), this ratio strictly increases as graduation year increases. For Hamilton46 and Santa74, the ratio is still significantly higher for the 2009 graduation class than any other class. If we average this ratio across all networks, as the graduation year increases from 2006 to 2009, the ratios strictly increase: 0.39 for 2006, 0.45 for 2007, 0.57 for 2008, and 0.75 for the class of 2009. Previous work on has already highlighted the influence of the graduation year attribute on the connectivity structure of Facebook 100 datasets on the whole [34]. Our exploration of these datasets has further allowed us to uncover interesting new connectivity patterns that exist between different individual graduation years.

6 DISCUSSION AND FUTURE WORK

We have introduced a new framework and theory for learning resolution parameters based on minimizing a fitness function associated with a single example clustering of interest. There are several open questions for improving our specific approach. Our bisection-like algorithm is designed to be general enough to minimize a large class of functions to within arbitrary precision. However, by making additional assumptions on either specific clustering objectives or the fixed example clustering, one may be able to develop improved algorithms for minimizing the parameter fitness function in practice. Another open question is to study which other graph clustering objectives can fit into out framework, beyond just the LAMBDACC global objective and the local flow clustering objective, and whether, for example, the approach can be applied to clustering in directed graphs.

Our work can be viewed as one approach to the more general goal of learning objective functions for graph clustering applications. This general goal could involve more techniques than simply learning resolution parameters. For example, in future work we wish to explore how to learn small *motif* subgraph patterns [6] in an example clustering that may be indicative of a desirable type of clustering structure in an application of interest.

ACKNOWLEDGMENTS

REFERENCES

[1] Ilan Adler and Renato D.C. Monteiro. 1992. A geometric view of parametric linear programming. *Algorithmica* 8 (1992), 161–176.

[2] Nir Ailon, Moses Charikar, and Alantha Newman. 2008. Aggregating inconsistent information: ranking and clustering. *Journal of the ACM (JACM)* 55, 5 (2008), 23.

[3] Reid Andersen, Fan Chung, and Kevin Lang. 2006. Local Graph Partitioning using PageRank Vectors. In *Proceedings of the 47th Annual IEEE Symposium on Foundations of Computer Science.* http://www.math.ucsd.edu/~fan/wp/localpartition.pdf

[4] Reid Andersen and Kevin Lang. 2008. An Algorithm for Improving Graph Partitions. In *Proceedings of the 19th annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2008).* 651–660.

[5] Nikhil Bansal, Avrim Blum, and Shuchi Chawla. 2004. Correlation Clustering. *Machine Learning* 56 (2004), 89–113.

[6] Austin R. Benson, David F. Gleich, and Jure Leskovec. 2016. Higher-order organization of complex networks. *Science* 353, 6295 (2016), 163–166. https://doi.org/10.1126/science.aad9029 arXiv:http://science.sciencemag.org/content/353/6295/163.full.pdf

[7] Vincent D Blondel, Jean-Loup Guillaume, Renaud Lambiotte, and Etienne Lefebvre. 2008. Fast unfolding of communities in large networks. *Journal of Statistical Mechanics: Theory and Experiment* 2008, 10 (2008), P10008. http://stacks.iop.org/1742-5468/2008/i=10/a=P10008

[8] Moses Charikar, Venkatesan Guruswami, and Anthony Wirth. 2005. Clustering with qualitative information. *J. Comput. System Sci.* 71, 3 (2005), 360 – 383. Learning Theory 2003.

[9] Shuchi Chawla, Konstantin Makarychev, Tselil Schramm, and Grigory Yaroslavtsev. 2015. Near optimal LP rounding algorithm for correlation clustering on complete and complete k-partite graphs. In *Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing.* ACM, 219–228.

[10] J.-C. Delvenne, Sophia N Yaliraki, and Mauricio Barahona. 2010. Stability of graph communities across time scales. *Proceedings of the National Academy of Sciences* 107, 29 (2010), 12755–12760.

[11] Santo Fortunato. 2010. Community detection in graphs. *Physics Reports* 486, 3 (2010), 75 – 174. https://doi.org/10.1016/j.physrep.2009.11.002

[12] Santo Fortunato and Marc Barthélemy. 2007. Resolution limit in community detection. *Proceedings of the National Academy of Sciences* 104, 1 (2007), 36–41. https://doi.org/10.1073/pnas.0605965104 arXiv:https://www.pnas.org/content/104/1/36.full.pdf

[13] Santo Fortunato and Darko Hric. 2016. Community detection in networks: A user guide. *Physics Reports* 659 (2016), 1 – 44. https://doi.org/10.1016/j.physrep.2016.09.002 Community detection in networks: A user guide.

[14] K. Fountoulakis, D. F. Gleich, and M. W. Mahoney. 2017. An Optimization Approach to Locally-Biased Graph Algorithms. *Proc. IEEE* 105, 2 (Feb 2017), 256–272. https://doi.org/10.1109/JPROC.2016.2637349

[15] David F. Gleich, Nate Veldt, and Anthony Wirth. 2018. Correlation Clustering Generalized. In *29th International Symposium on Algorithms and Computation (ISAAC 2018)*, Vol. 123. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, Dagstuhl, Germany, 44:1–44:13. https://doi.org/10.4230/LIPIcs.ISAAC.2018.44

[16] Lucas G. S. Jeub, Marya Bazzi, Inderjit S. Jutla, and Peter J. Mucha. 2011-2017. A generalized Louvain method for community detection implemented in MATLAB. http://netwiki.amath.unc.edu/GenLouvain

[17] Lucas G. S. Jeub, Olaf Sporns, and Santo Fortunato. 2018. Multiresolution Consensus Clustering in Networks. *Scientific Reports* 8, 1 (2018), 3259. https://doi.org/10.1038/s41598-018-21352-7

[18] Kyle Kloster and David F. Gleich. 2014. Heat Kernel Based Community Detection. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD 2014).* 1386–1395. https://doi.org/10.1145/2623330.2623706

[19] Andrea Lancichinetti and Santo Fortunato. 2009. Benchmarks for testing community detection algorithms on directed and weighted graphs with overlapping communities. *Phys. Rev. E* 80 (Jul 2009), 016118. Issue 1. https://doi.org/10.1103/PhysRevE.80.016118

[20] Kevin Lang and Satish Rao. 2004. A Flow-Based Method for Improving the Expansion or Conductance of Graph Cuts. In *Integer Programming and Combinatorial Optimization*. Lecture Notes in Computer Science, Vol. 3064. Springer Berlin Heidelberg, 325–337. https://doi.org/10.1007/978-3-540-25960-2_25

[21] Jure Leskovec, Jon Kleinberg, and Christos Faloutsos. 2007. Graph Evolution: Densification and Shrinking Diameters. *ACM Trans. Knowl. Discov. Data* 1, 1, Article 2 (March 2007). https://doi.org/10.1145/1217299.1217301

[22] Jure Leskovec and Andrej Krevl. 2014. SNAP Datasets: Stanford Large Network Dataset Collection. http://snap.stanford.edu/data.

[23] Mark EJ Newman. 2006. Finding community structure in networks using the eigenvectors of matrices. *Physical review E* 74, 3 (2006), 036104.

[24] Mark EJ Newman and Michelle Girvan. 2004. Finding and evaluating community structure in networks. *Physical review E* 69, 026113 (2004).

[25] M. E. J. Newman. 2016. Equivalence between modularity optimization and maximum likelihood methods for community detection. *Phys. Rev. E* 94 (Nov 2016), 052315. Issue 5. https://doi.org/10.1103/PhysRevE.94.052315

[26] Lorenzo Orecchia and Zeyuan Allen Zhu. 2014. Flow-Based Algorithms for Local Graph Clustering. In *Proceedings of the 25th ACM-SIAM Symposium on Discrete Algorithms (SODA 2014).* 1267–1286. http://arxiv.org/abs/1307.2855

[27] Leto Peel, Daniel B. Larremore, and Aaron Clauset. 2017. The ground truth about metadata and community detection in networks. *Science Advances* 3, 5 (2017). https://doi.org/10.1126/sciadv.1602548 arXiv:http://advances.sciencemag.org/content/3/5/e1602548.full.pdf

[28] Mason A Porter, Jukka-Pekka Onnela, and Peter J Mucha. 2009. Communities in networks. *Notices of the AMS* 56, 9 (2009), 1082–1097.

[29] Jörg Reichardt and Stefan Bornholdt. 2006. Statistical mechanics of community detection. *Physical Review E* 74, 016110 (2006).

[30] Cameron Ruggles, Nate Veldt, and David Gleich. 2019. A Parallel Projection Method for Metric Constrained Optimization. *arXiv preprint arXiv:1901.10084* (2019).

[31] Satu Elisa Schaeffer. 2007. Graph clustering. *Computer Science Review* 1, 1 (2007), 27 – 64. https://doi.org/10.1016/j.cosrev.2007.05.001

[32] Michael T. Schaub, Renaud Lambiotte, and Mauricio Barahona. 2012. Encoding dynamics for multiscale community detection: Markov time sweeping for the map equation. *Phys. Rev. E* 86 (Aug 2012), 026112. Issue 2. https://doi.org/10.1103/PhysRevE.86.026112

[33] Daniel A. Spielman and Shang-Hua Teng. 2013. A Local Clustering Algorithm for Massive Graphs and Its Application to Nearly Linear Time Graph Partitioning. *SIAM J. Comput.* 42, 1 (2013), 1–26. https://doi.org/10.1137/080744888

[34] Amanda L. Traud, Peter J. Mucha, and Mason A. Porter. 2012. Social structure of Facebook networks. *Physica A: Statistical Mechanics and its Applications* 391, 16 (2012), 4165–4180. https://doi.org/10.1016/j.physa.2012.03.0

[35] Nate Veldt, David Gleich, and Michael Mahoney. 2016. A Simple and Strongly-Local Flow-Based Method for Cut Improvement. In *Proceedings of The 33rd International Conference on Machine Learning (ICML 2016)*, Vol. 48. PMLR, New York, New York, USA, 1938–1947. http://proceedings.mlr.press/v48/veldt16.html

[36] Nate Veldt, David Gleich, Anthony Wirth, and James Saunderson. 2018. A Projection Method for Metric-Constrained Optimization. *arXiv preprint arXiv:1806.01678* (2018).

[37] Nate Veldt, David F. Gleich, and Anthony Wirth. 2018. A Correlation Clustering Framework for Community Detection. In *Proceedings of the 2018 World Wide Web Conference (WWW 2018)*. International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva, Switzerland, 439–448. https://doi.org/10.1145/3178876.3186110

[38] Nate Veldt, Christine Klymko, and David F. Gleich. 2019. Flow-Based Local Graph Clustering with Better Seed Set Inclusion. In *Proceedings of the 2019 SIAM International Conference on Data Mining.*

[39] Di Wang, Kimon Fountoulakis, Monika Henzinger, Michael W. Mahoney, and Satish Rao. 2017. Capacity Releasing Diffusion for Speed and Locality. In *Proceedings of the 34th International Conference on Machine Learning (ICML 2017)*, Vol. 70. PMLR, International Convention Centre, Sydney, Australia, 3598–3607. http://proceedings.mlr.press/v70/wang17b.html

[40] Jaewon Yang and Jure Leskovec. 2015. Defining and evaluating network communities based on ground-truth. *Knowledge and Information Systems* 42, 1 (01 Jan 2015), 181–213. https://doi.org/10.1007/s10115-013-0693-z

[41] Hao Yin, Austin R. Benson, Jure Leskovec, and David F. Gleich. 2017. Local Higher-Order Graph Clustering. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD 2017)*. ACM, New York, NY, USA, 555–564. https://doi.org/10.1145/3097983.3098069

[42] Linbin Yu and Chris Ding. 2010. Network Community Discovery: Solving Modularity Clustering via Normalized Cut. In *Proceedings of the Eighth Workshop on Mining and Learning with Graphs (MLG 2010)*. ACM, New York, NY, USA, 34–36. https://doi.org/10.1145/1830252.1830257