A Correlation Clustering Framework for Community Detection

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

Graph clustering, or community detection, is the task of identifying groups of closely related objects in a large network. In this paper we introduce a new community-detection framework called LambdaCC that is based on a specially weighted version of correlation clustering. A key component in our methodology is a clustering resolution parameter, λ, which implicitly controls the size and structure of clusters formed by our framework. We show that, by increasing this parameter, our objective effectively interpolates between two different strategies in graph clustering: finding a sparse cut and forming dense subgraphs. Our methodology unifies and generalizes a number of other important clustering quality functions including modularity, sparsest cut, and cluster deletion, and places them all within the context of an optimization problem that has been well studied from the perspective of approximation algorithms. Our approach is particularly relevant in the regime of finding dense clusters, as it leads to a 2-approximation for the cluster deletion problem. We use our approach to cluster several graphs, including large collaboration networks and social networks.

CCS CONCEPTS

• **Mathematics of computing → Graph algorithms**; **Approximation algorithms**; *Mathematical optimization*;

KEYWORDS

graph clustering; community detection; network analysis; correlation clustering; cluster deletion; sparsest cut

ACM Reference Format:
Nate Veldt, David F. Gleich, and Anthony Wirth. 2018. A Correlation Clustering Framework for Community Detection. In *WWW 2018: The 2018 Web Conference, April 23–27, 2018, Lyon, France.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3178876.3186110

1 INTRODUCTION

Identifying groups of related entities in a network is a ubiquitous task across scientific disciplines. This task is often called graph clustering, or community detection, and can be used to find similar proteins in a protein-interaction network, group related organisms in a food web, identify communities in a social network, and classify web documents, among numerous other applications.

Defining the right notion of a "good" community in a graph is an important precursor to developing successful algorithms for graph clustering. In general, a good clustering is one in which nodes inside clusters are more densely connected to each other than to the rest of the graph. However, no consensus exists as to the best way to determine the quality of network clusterings, and recent results show there cannot be such a consensus for the multiple possible reasons people may cluster data [32]. Common objective functions studied by theoretical computer scientists include normalized cut, sparsest cut, conductance, and edge expansion, all of which measure some version of the cut-to-size ratio for a single cluster in a graph. Other standards of clustering quality put a greater emphasis on the internal density of clusters, such as the cluster deletion objective, which seeks to partition a graph into completely connected sets of nodes (cliques) by removing the fewest edges possible.

Arguably the most widely used multi-cluster objective for community detection is modularity, introduced by Newman and Girvan [31]. Modularity measures the difference between the true number of edges inside the clusters of a given partitioning ("inner edges") minus the *expected* number of inner edges, where expectation is calculated with respect to a specific random graph model.

There are a limited number of results which have begun to unify distinct clustering measures by introducing objective functions that are closely related to modularity and depend on a tunable clustering resolution parameter [15, 33]. Reichardt and Bornholdt developed an approach based on finding the minimum-energy state of an infinite-range Potts spin glass. The resulting Hamiltonian function they study is viewed as a clustering objective with a resolution parameter, γ, which can be used as a heuristic for detecting overlapping and hierarchical community structure in a network. When $\gamma = 1$, the authors prove an equivalence between minimizing the Hamiltonian and finding the maximum modularity partitioning of a network [33]. Later, Delvenne et al. introduced a measure called the *stability* of a clustering, which generalizes modularity and also is related to the normalized cut objective and Fiedler's spectral clustering method for certain values of an input parameter [15].

The inherent difficulty in obtaining clusterings that are provably close to the optimal solution puts these objective functions at a disadvantage. Although both the *stability* and the Hamiltonian-Potts objectives provide useful interpretations for community detection, there are no approximation guarantees for either: all current algorithms are heuristics. Furthermore, it is known that maximizing

ACM ISBN 978-1-4503-5639-8/18/04.
https://doi.org/10.1145/3178876.3186110

modularity itself is not only NP-hard, but is also NP-hard to approximate to within any constant factor [19].

Our Contributions. In this paper, we introduce a new clustering framework based on a specially weighted version of correlation clustering [4]. Our partitioning objective for signed networks lies "between" the family of ±1 complete instances and the most general correlation clustering instances. Our framework comes with several novel theoretical properties and leads to many connections between clustering objectives that were previously not seen to be related. In summary, we provide:

- A novel framework LambdaCC for community detection that is related to modularity and the Hamiltonian, but is more amenable to approximation results.
- A proof that our framework interpolates between the sparsest cut objective and the cluster deletion problem, as we increase a single resolution parameter, λ.
- Several successful algorithms for optimizing our new objective function in both theory and practice, including a 2-approximation for cluster deletion, which improves upon the previous best approximation factor of 3.
- A demonstration of our methods in a number of clustering applications, including social network analysis and mining cliques in collaboration networks.

2 BACKGROUND AND RELATED WORK

Let G be an undirected and unweighted graph on n nodes V, with m edges E. For all $v \in V$, let d_v be node v's degree. Given $S \subseteq V$, let $\bar{S} = V\backslash S$ be the complement of S and $\text{vol}(S) = \sum_{v \in S} d_v$ be its volume. For every two disjoint sets of vertices $S, T \subseteq V$, $\text{cut}(S, T)$ indicates the number of edges between S and T. If $T = \bar{S}$, we write $\text{cut}(S) = \text{cut}(S, \bar{S})$. Let E_S denote the interior edge set of S. The edge density of a cluster is $\text{density}(S) = |E_S|/\binom{|S|}{2}$, the ratio between the number of edges to the number of pairs of nodes in S. By convention, the density of a single node is 1. We now present background and related work that is foundational to our results, including definitions for several common clustering objectives.

2.1 Correlation Clustering

An instance of correlation clustering is given by a signed graph where every pair of nodes i and j possesses two non-negative weights, w_{ij}^+ and w_{ij}^-, to indicate how similar and how dissimilar i and j are, respectively. Typically only one of these weights is nonzero for each pair i, j. The objective can be expressed as an integer linear program (ILP):

$$\begin{array}{ll} \text{minimize} & \sum_{i<j} w_{ij}^+ x_{ij} + w_{ij}^-(1 - x_{ij}) \\ \text{subject to} & x_{ij} \leq x_{ik} + x_{jk} \text{ for all } i, j, k \\ & x_{ij} \in \{0, 1\} \text{ for all } i, j. \end{array} \tag{1}$$

In the above formulation, x_{ij} represents "distance": $x_{ij} = 0$ indicates that nodes i and j are clustered together, while $x_{ij} = 1$ indicates they are separated. Including triangle inequality constraints ensures the output of the above ILP defines a valid clustering of the nodes. This objective counts the total *weight* of disagreements between the signed weights in the graph and a given clustering of its nodes. The disagreement (or "mistake") weight of a pair i, j is w_{ij}^- if the nodes are clustered together, but w_{ij}^+ if they are separated. We can equivalently define the agreement weight to be w_{ij}^+ if i, j are clustered together, but w_{ij}^- if they are separated. The optimal clusterings for maximizing agreements and minimizing disagreements are identical, but it is harder to approximate the latter.

Correlation clustering was introduced by Bansal et al., who proved the problem is NP-complete [4]. They gave a polynomial-time approximation scheme for the maximization version and a constant-factor approximation for minimizing disagreements in ±1-weighted graphs. Subsequently, Charikar et al. gave a factor 4-approximation for minimizing disagreements and proved APX-hardness of this variant. They also described an $O(\log n)$ approximation for minimization in general weighted graphs [12], proved independently by two different groups, who showed that minimizing disagreements is equivalent to minimum multicut [16, 20].

The problem has also been studied for the case where edges carry both positive and negative weights, satisfying probability constraints: for all pairs i, j, $w_{ij}^+ + w_{ij}^- = 1$. Ailon et al. gave a 2.5-approximation for this version of the problem based on an LP-relaxation, and additionally developed a very fast algorithm, called Pivot, that in expectation gives a 3-approximation [1]. Currently the best-known approximation factor for correlation clustering on ±1 instances is slightly smaller than 2.06, obtained by a careful rounding of the canonical LP relaxation [13].

2.2 Modularity and the Hamiltonian

One very popular measure of clustering quality is modularity, introduced in its most basic form by Newman and Girvan [31]. We more closely follow the presentation of modularity given by Newman [30]. The modularity Q of an underlying clustering is:

$$Q(x) = \frac{1}{2m} \sum_{i \neq j} \left(A_{ij} - P_{ij}\right)(1 - x_{ij}), \tag{2}$$

where $A_{ij} = 1$ if nodes i and j are adjacent, and zero otherwise, and x_{ij} is again the binary variable indicating "distance" between i and j in the corresponding clustering. The value P_{ij} represents the probability of an edge existing between i and j in a specific random graph model. The intent of this measure is to reward clusterings in which the actual number of edges inside a cluster is greater than the *expected* number of edges in the cluster, as determined by the choice for P_{ij}. Although there are many options, it is standard in the literature to set $P_{ij} = d_i d_j/(2m)$, since this preserves both the degree distribution and the expected number of edges between the original graph and null model.

By slightly editing the modularity function, we obtain the Hamiltonian objective of Reichardt and Bornholdt [33]:

$$\mathcal{H}(x) = -\sum_{i \neq j} \left(A_{ij} - \gamma P_{ij}\right)(1 - x_{ij}). \tag{3}$$

The primary difference between this and modularity is the inclusion of a clustering resolution parameter γ. If we fix $\gamma = 1$, minimizing (3) is equivalent to maximizing modularity. When varied, this parameter controls how much a clustering is penalized for putting two non-adjacent nodes together or separating adjacent nodes.

The Hamiltonian objective is in turn closely related to the *stability* of a clustering as defined by Delvenne et al., another generalization of modularity [15]. Roughly speaking, the stability of a partition measures the likelihood that a random walker, beginning at a node and following outgoing edges uniformly at random, will end up in the cluster it started in after a random walk of length t. This t serves as a resolution parameter, since the walker will tend to "wander" farther when t is increased, leading to the formation of larger clusters when the stability is maximized. Delvenne et al. showed that objective (3) is equivalent to a linearized version of the stability measure for a specific range of time steps t [15].

2.3 Sparsest Cut and Normalized Cut

One measure of cluster quality in an unsigned network G is the sparsest cut score, defined for a set $S \subseteq V$ to be $\phi(S) = \text{cut}(S)/|S| + \text{cut}(S)/|\bar{S}| = n \cdot \text{cut}(S)/(|S||\bar{S}|)$. Smaller values for $\phi(S)$ are desirable, since they indicate that S, in spite of its size, is only loosely connected to the rest of the graph. This measure differs by at most a factor of two from the related edge expansion measure: $\text{cut}(S)/(\min\{|S|, |\bar{S}|\})$. If we replace $|S|$ with $\text{vol}(S)$ in these two objectives, we obtain the normalized cut and the conductance measure respectively. In our work we focus on a multiplicative scaling of the sparsest cut objective that we call the *scaled sparsest cut*: $\psi(S) = \phi(S)/n = \text{cut}(S)/(|S||\bar{S}|)$, which is identical to sparsest cut in terms of multiplicative approximations. The best known approximation for finding the minimum sparsest cut of a graph is an $O(\sqrt{\log n})$-approximation algorithm due to Arora et al. [2].

2.4 Cluster Deletion

Cluster deletion is the problem of finding a minimum number of edges in G to be deleted in order to convert G into a disjoint set of cliques. This problem was first studied by Ben-Dor et al. [6], later formalized in the work of Natanzon et al. [29], who proved it is NP-hard, and Shamir et al. [36], who showed it is APX-hard. The latter studied the problem in conjunction with other related *edge-modification* problems, including cluster completion and cluster editing. Numerous fixed parameter tractability results are known for cluster deletion [8, 14, 23, 24], as well many results regarding special graphs for which the problem can be solved in polynomial time [10, 11, 17, 21]. Dessmark et al. proved that recursively finding maximum cliques will return a clustering with a cluster deletion score within a factor 2 of optimal [17], though in general this procedure is NP-hard.

3 THEORETICAL RESULTS

Our novel clustering framework takes an unsigned graph $G = (V, E)$ and converts it into a signed graph $G' = (V, E^+, E^-)$ on the same set of nodes, V, for a fixed clustering resolution parameter $\lambda \in (0, 1)$. Partitioning G' with respect to the correlation clustering objective will then induce a clustering on G. To construct the signed graph, we first introduce a node weight w_v for each $v \in V$. If $(i, j) \in E$, we place a positive edge between nodes i and j in G', with weight $(1-\lambda w_i w_j)$. For $(i, j) \notin E$, we place a negative edge between i and j in G', with weight $\lambda w_i w_j$. We consider two different choices for node weights w_v: setting $w_v = 1$ for all v (*standard*) or choosing $w_v = d_v$ (*degree-weighted*). In Figure 1 we illustrate the process of converting G into the LambdaCC signed graph, G'. The goal of LambdaCC is to find the clustering that minimizes disagreements in G', or equivalently minimizes the following objective function

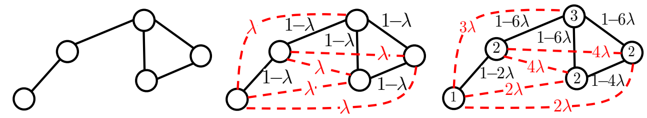


Figure 1: We convert a toy graph (left) into a signed graph for standard (middle) and degree-weighted (right) LambdaCC. Dashed red lines indicate negative edges.

expressed in terms of edges and non-edges in G:

$$\lambda CC(x) = \sum_{(i,j) \in E} (1 - \lambda w_i w_j) x_{ij} + \sum_{(i,j) \notin E} \lambda w_i w_j (1 - x_{ij}) \quad (4)$$

where $x = (x_{ij})$ represents the binary distances for the clustering.

3.1 Connection to Modularity

Despite a significant difference in approach and interpretation, the clustering that minimizes disagreements is the same clustering that minimizes the Hamiltonian objective (3), for a certain choice of parameters. To see this, we introduce node adjacency values A_{ij} in objective (4) and perform a few steps of algebra:

$$\begin{aligned} \lambda CC(x) &= \sum_{(i,j) \in E} (A_{ij} - \lambda w_i w_j) x_{ij} - \sum_{(i,j) \notin E} (A_{ij} - \lambda w_i w_j)(1 - x_{ij}) \\ &= \sum_{(i,j) \in E} (1 - \lambda w_i w_j) - \sum_{i<j} (A_{ij} - \lambda w_i w_j)(1 - x_{ij}) \,. \end{aligned}$$

Choosing $P_{ij} = w_i w_j/(2m)$ and $\gamma = 2m\lambda$, we see that:

$$\lambda CC(x) = \textstyle\sum_{(i,j) \in E} (1 - \lambda w_i w_j) + \mathcal{H}(x)/2 \,, \quad (5)$$

where the first term is just a constant. This theorem follows:

Theorem 3.1. *Minimizing disagreements for the LambdaCC objective is equivalent to minimizing $\mathcal{H}(x)$.*

The choice $P_{ij} = w_i w_j/(2m)$ is reminiscent of the graph null model most commonly used for modularity and the Hamiltonian. This best highlights the similarity between these objectives and degree-weighted LambdaCC.

3.2 Standard LambdaCC

While degree-weighted LambdaCC is more closely related to modularity and the Hamiltonian, standard LambdaCC (setting $w_v = 1$ for every $v \in V$) leads to strong connections between the sparsest cut objective and cluster deletion. This version corresponds to solving a correlation clustering problem where all positive edges have equal weight, $(1 - \lambda)$, while all negative edges have equal weight, λ. The objective function for minimizing disagreements is

$$\min \sum_{(i,j) \in E^+} (1 - \lambda) x_{ij} + \sum_{(i,j) \in E^-} \lambda (1 - x_{ij}) \,, \quad (6)$$

where we include the same constraints as in ILP (1). This is a strict generalization of the unit-weight correlation clustering problem [4] ($\lambda = 1/2$) indicating the problem in general is NP-hard (though it admits several approximation algorithms). If λ is 0 or 1, the problem is trivial to solve: put all nodes in one cluster or put each node in a singleton cluster, respectively. By selecting values for λ other than 0, 1/2, or 1, we uncover subtler connections between identifying sparse cuts and finding dense subgraphs in the network.

3.3 Connection to Sparsest Cut

Given G and λ, the weight of positive-edge mistakes in the standard LambdaCC objective made by a two-clustering $C = \{S, \bar{S}\}$ equals the weight of edges crossing the cut: $(1-\lambda)\,\mathrm{cut}(S)$. To compute the weight of negative-edge mistakes, we take the weight of all negative edges in the entire network, $\lambda\left(\binom{n}{2} - |E|\right)$, and then subtract the weight of negative edges between S and $\bar{S}$: $\lambda\left(|S||\bar{S}| - \mathrm{cut}(S)\right)$. Adding together all terms, we find that the LambdaCC objective for this clustering is

$$\mathrm{cut}(S, \bar{S}) - \lambda|S||\bar{S}| + \lambda\binom{n}{2} - \lambda|E|\,. \tag{7}$$

Note that if we minimize (7) over all 2-clusterings, we solve the decision version of the minimum scaled sparsest cut problem: a few steps of algebra confirm that there is some set $S \subseteq V$ with $\psi(S) = \mathrm{cut}(S)/(|S||\bar{S}|) < \lambda$ if and only if (7) is less than $\lambda\binom{n}{2} - \lambda|E|$.

In a similar way we can show that objective (6) is equivalent to

$$\min\ \frac{1}{2}\sum_{i=1}^{k}\mathrm{cut}(S_i) - \frac{\lambda}{2}\sum_{i=1}^{k}|S_i||\bar{S_i}| + \lambda\binom{n}{2} - \lambda|E|\,, \tag{8}$$

where we minimize over all clusterings of G (note that the number of clusters k is determined automatically by optimizing the objective). In this case, optimally solving objective (8) will tell us whether we can find a clustering $C = \{S_1, S_2, \dots, S_k\}$ such that $\sum_{i=1}^{k}\mathrm{cut}(S_i, \bar{S_i})/\left(\sum_{j=1}^{k}|S_j||\bar{S_j}|\right) < \lambda$. Hence LambdaCC can be viewed as a multi-cluster generalization of the decision version of minimum sparsest cut. We now prove an even deeper connection between sparsest cut and LambdaCC. Using degree-weighted LambdaCC yields an analogous result for normalized cut.

Theorem 3.2. *Let λ^* be the minimum scaled sparsest cut value for graph G.*

(a) For all $\lambda > \lambda^*$, optimal solution (8) partitions G into two or more clusters, each of which has scaled sparsest cut $\leq \lambda$. There exists some $\lambda' > \lambda^*$ such that the optimal clustering for LambdaCC is the minimum sparsest cut partition.

(b) For $\lambda \leq \lambda^*$, it is optimal to place all nodes into a single cluster.

Proof. **Statement (a)** Let S^* be some subset of V that induces a sparsest cut, i.e., $\psi(S^*) = \mathrm{cut}(S^*)/(|S^*||\bar{S}^*|) = \lambda^*$. The LambdaCC objective corresponding to $C = \{S^*, \bar{S}^*\}$ is

$$\mathrm{cut}(S^*) - \lambda|S^*||\bar{S}^*| + \lambda\binom{n}{2} - \lambda|E|\,. \tag{9}$$

When minimizing objective (8), we can always obtain a score of $\lambda\binom{n}{2} - \lambda|E|$ by placing all nodes into a single cluster. Note however that the the score of clustering $\{S^*, \bar{S}^*\}$ in expression (9) is strictly less than $\lambda\binom{n}{2} - \lambda|E|$ for all $\lambda > \lambda^*$. Even if $\{S^*, \bar{S}^*\}$ is not optimal, this means that when $\lambda > \lambda^*$, we can do strictly better than placing all nodes into one cluster. In this case let C^* be the optimal LambdaCC clustering and consider two of its clusters: S_i and S_j. The weight of disagreements between S_i and S_j is equal to the number of positive edges between them times the weight of a positive edge: $(1-\lambda)\,\mathrm{cut}(S_i, S_j)$. Should we form a new clustering by merging S_i and S_j, these positive disagreements will disappear; in turn, we would introduce $\lambda|S_i||S_j| - \lambda\,\mathrm{cut}(S_i, S_j)$ new mistakes, being negative edges between the clusters. Because we assumed C^* is optimal, we know that we cannot decrease the objective by merging two of the clusters, implying that

$$\begin{aligned}(1-\lambda)\,\mathrm{cut}(S_i, S_j) - \left(\lambda|S_i||S_j| - \lambda\,\mathrm{cut}(S_i, S_j)\right)\\ = \mathrm{cut}(S_i, S_j) - \lambda|S_i||S_j| \leq 0\,.\end{aligned}$$

Given this, we fix an arbitrary cluster S_i and perform a sum over all other clusters to see that

$$\sum_{j\neq i}\mathrm{cut}(S_i, S_j) - \sum_{j\neq i}\lambda|S_i||S_j| \leq 0$$

$$\implies \mathrm{cut}(S_i, \bar{S_i}) - \lambda|S_i||\bar{S_i}| \leq 0 \implies \mathrm{cut}(S_i, \bar{S_i})/\left(|S_i||\bar{S_i}|\right) \leq \lambda\,,$$

proving the desired upper bound on scaled sparsest cut.

Since G is a finite graph, there are a finite number of scaled sparsest cut scores that can be induced by a subset of V. Let $\tilde{\lambda}$ be the second-smallest scaled sparsest cut score achieved, so $\tilde{\lambda} > \lambda^*$. If we set $\lambda' = (\lambda^* + \tilde{\lambda})/2$, then the optimal LambdaCC clustering produces at least two clusters, since $\lambda' > \lambda^*$, and each cluster has scaled sparsest cut at most $\lambda' < \tilde{\lambda}$. By our selection of $\tilde{\lambda}$, all clusters returned must have scaled sparsest cut exactly equal to λ^*, which is only possible if the clustering returned has two clusters. Hence this clustering is a minimum sparsest cut partition of the network.

Statement (b) If $\lambda < \lambda^*$, forming a single cluster must be optimal, otherwise we could invoke Statement (a) to assert the existence of some nontrivial cluster with scaled sparsest cut less than or equal to $\lambda < \lambda^*$, contradicting the minimality of λ^*. If $\lambda = \lambda^*$, forming a single cluster or using the clustering $C = \{S^*, \bar{S}^*\}$ yield the same objective score, which is again optimal for the same reason. □

3.4 Connection to Cluster Deletion

For large λ, our problem becomes more similar to cluster deletion. We can reduce any cluster deletion problem to correlation clustering by taking the input graph G and introducing a negative edge of weight "∞" between every pair of non-adjacent nodes. This guarantees that optimally solving correlation clustering will yield clusters that all correspond to cliques in G. Furthermore, the weight of disagreements will be the number of edges in G that are cut, i.e., the cluster deletion score. We can obtain a generalization of cluster deletion by instead choosing the weight of each negative edge to be $\alpha < \infty$. The corresponding objective is

$$\sum_{(i,j)\in E^+} x_{ij} + \sum_{(i,j)\in E^-}\alpha(1 - x_{ij})\,. \tag{10}$$

If we substitute $\alpha = \lambda/(1-\lambda)$ we see this differs from objective (6) only by a multiplicative constant, and is therefore equivalent in terms of approximation. When $\alpha > 1$, putting dissimilar nodes together will be more expensive than cutting positive edges, so we would expect that the clustering which optimizes the LambdaCC objective will separate G into dense clusters that are "nearly" cliques. We formalize this with a simple theorem and corollary.

Theorem 3.3. *If C minimizes the LambdaCC objective for the unsigned network $G = (V, E)$, then the edge density of every cluster in C is at least λ.*

Proof. Take a cluster $S \in C$ and consider what would happen if we broke apart S so that each of its nodes were instead placed into its own singleton cluster. This means we are now making mistakes at every positive edge previously in S, which increases the weight

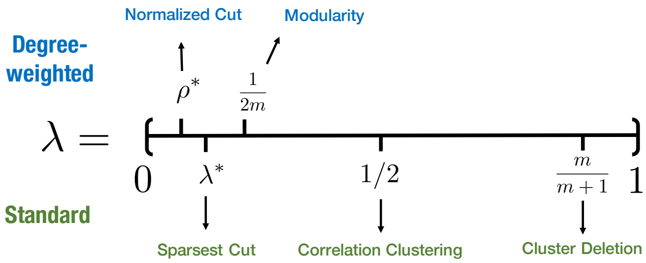


Figure 2: LambdaCC is equivalent to several other objectives for specific values of $\lambda \in (0, 1)$. Values λ^* and ρ^* are not known a priori, but can be obtained by solving LambdaCC for increasingly smaller values of λ.

Table 1: The best approximation factors known for standard LambdaCC, for $\lambda \in (0, 1)$, both for minimizing disagreements and maximizing agreements. We contribute two constant-factor approximations for minimizing disagreements when $\lambda > 1/2$.

	$\lambda \in (0, 1/2)$	$\lambda = 1/2$	$\lambda \in (1/2, 1)$
Max-Ag.	0.7666 [37]	PTAS [4]	0.7666 [37]
Min-Dis.	$O(\log n)$ [12, 16, 20]	2.06 [13]	$3\left(2 : \lambda > \frac{m}{m+1}\right)$

of disagreements by $(1 - \lambda)|E_S|$. On the other hand, there are no longer negative mistakes between nodes in S, so the LambdaCC objective would simultaneously decrease by $\lambda\left(\binom{|S|}{2} - |E_S|\right)$. The total change in the objective made by pulverizing S is

$$(1 - \lambda)|E_S| - \lambda\left(\binom{|S|}{2} - |E_S|\right) = |E_S| - \lambda\binom{|S|}{2},$$

which must be nonnegative, since C is optimal, so $|E_S| - \lambda\binom{|S|}{2} \geq 0$, which implies density$(S) = |E_S|/\binom{|S|}{2} \geq \lambda$. □

Corollary 3.4. *Let G have m edges. For every $\lambda > m/(m + 1)$, optimizing LambdaCC is equivalent to optimizing cluster deletion.*

Proof. All output clusters must have density at least $m/(m + 1)$, which is only possible if the density is actually 1, since m is the total number of edges in the graph. Therefore all clusters are cliques and the LambdaCC and cluster deletion objectives differ only by a multiplicative constant $(1 - \lambda)$. □

3.5 Equivalences and Approximations

We summarize the equivalence relationships between LambdaCC and other objectives in Figure 2. Accompanying this, Table 1 outlines the best-known approximation results both for maximizing agreements and minimizing disagreements for the standard LambdaCC signed graph. For degree-weighted LambdaCC, the best-known approximation factors for all λ are $O(\log n)$ [12, 16, 20] for minimizing disagreements, and 0.7666 for maximizing agreements [37]. Thus, LambdaCC is more amenable to approximation than modularity (and relatives) because of additive constants.

Algorithm 1 threeLP

Input: Signed graph $G' = (V, E^+, E^-)$, $\lambda \in (0, 1)$
Output: Clustering C of G'
Solve the LP relaxation of ILP (6), obtaining *distances* (x_{ij})
Let $\tilde{G} = (V, F^+, F^-)$, where $F^+ = \{(i, j) : x_{ij} < 1/3\}$ and $F^- = \{(i, j) : x_{ij} \geq 1/3\}$
Apply Pivot on $\tilde{G}$.

4 ALGORITHMS

We present several new algorithms, tailored specifically to our LambdaCC framework; some come with approximation guarantees, some are designed for efficiency.[1] Our first two methods rely on a key theorem of van Zuylen and Williamson, which can be used to prove approximation results for different cases of correlation clustering using pivoting algorithms, which operate by selecting a pivot node k, clustering it with all its positive neighbors, and recursing on the rest of the graph. If pivots are chosen uniformly at random this corresponds to the Pivot algorithm of Ailon et al. [1]. We state the theorem here for completeness, with minor changes to match our notation and presentation:

Theorem 4.1. *(Theorem 3.1 in [38]) Let $G = (V, W^+, W^-)$ be a signed, weighted graph where each pair of nodes (i, j) has positive and negative weights $w^+_{ij} \in W^+$ and $w^-_{ij} \in W^-$. Given a set of LP costs $\{c_{ij} : i \in V, j \in V, i \neq j\}$, and an unweighted graph $\tilde{G} = (V, F^+, F^-)$ satisfying the following assumptions:*

(i) $w^-_{ij} \leq \alpha c_{ij}$ for all $(i, j) \in F^+$ and $w^+_{ij} \leq \alpha c_{ij}$ for all $(i, j) \in F^-$,

(ii) $w^+_{ij} + w^+_{jk} + w^-_{ik} \leq \alpha\left(c_{ij} + c_{jk} + c_{ik}\right)$ for every bad triangle in $\tilde{G}$: $(i, j), (j, k) \in F^+$, $(i, k) \in F^-$,

then applying Pivot on $\tilde{G}$ will return a solution that costs $\alpha \sum_{i<j} c_{ij}$ in expectation.

A full proof is given by van Zuylen and Williamson [38], who also include a strategy for deterministically choosing pivots to achieve the same approximation.

4.1 3-Approximation for LambdaCC

In order to apply Theorem 4.1 we must compute the LP relaxation (6) for LambdaCC (where c_{ij} is the cost for edge (i, j)) to obtain distances x_{ij}, which we then round into an unweighted graph $\tilde{G}$. If we can construct $\tilde{G}$ to satisfy the assumptions of the theorem, all that remains is to apply Pivot to yield the desired approximation results. Pseudocode for our first method is displayed in Algorithm 1, which we call threeLP since it satisfies the following approximation guarantee:

Theorem 4.2. *Algorithm threeLP satisfies Theorem 4.1 with $\alpha = 3$ for standard LambdaCC when $\lambda > 1/2$.*

[1]For the initial submission of our work we presented a 5-approximation for LambdaCC when $\lambda > 1/2$ and a 4-approximation for cluster deletion by altering the approach of Charikar et al. [12]. Here we show improved approximations that we developed based on a helpful suggestion from an anonymous reviewer. We include the original approximation results in the full version of our paper [39].

Proof. We begin by stating the correspondence between the notation of Theorem 4.1 and the edge weights and LP costs for LambdaCC. The graph $G' = (V, E^+, E^-)$ is made up of positive edges of weight $(1-\lambda)$ and negative edges with weight λ, so

$$c_{ij} = (1-\lambda)x_{ij} \text{ and } (w^+_{ij}, w^-_{ij}) = (1-\lambda, 0) \text{ if } (i,j) \in E^+$$
$$c_{ij} = \lambda(1-x_{ij}) \text{ and } (w^+_{ij}, w^-_{ij}) = (0, \lambda) \text{ if } (i,j) \in E^-.$$

By construction, if $(i,j) \in F^-$ then $x_{ij} < 1/3$, otherwise $(i,j) \in F^+$ and we know $x_{ij} \geq 1/3$. The first two inequalities we need to check for Theorem 4.1 are

$$w^-_{ij} \leq \alpha c_{ij} \text{ for all } (i,j) \in F^+ \quad (11)$$
$$w^+_{ij} \leq \alpha c_{ij} \text{ for all } (i,j) \in F^- \quad (12)$$

where $\alpha = 3$. If $(i,j) \in F^+ \cap E^+$, then $w^-_{ij} = 0$ and inequality (11) is trivial since the left hand side is zero. Similarly, inequality (12) is trivial if $(i,j) \in F^- \cap E^-$. Assume then that $(i,j) \in F^+ \cap E^-$. Then $w^-_{ij} = \lambda$ and $c_{ij} = \lambda(1-x_{ij})$, and we know $x_{ij} < 1/3 \implies (1-x_{ij}) > 2/3$. Therefore:

$$w^-_{ij} = \lambda < 3\lambda\,(2/3) < 3\lambda(1-x_{ij}) = \alpha c_{ij}.$$

On the other hand, if $(i,j) \in F^- \cap E^+$, then $w^+_{ij} = (1-\lambda)$, $c_{ij} = (1-\lambda)x_{ij}$, and $x_{ij} \geq 1/3$, so we see:

$$w^+_{ij} = (1-\lambda) = 3(1-\lambda)\,(1/3) \leq 3(1-\lambda)x_{ij} = \alpha c_{ij}.$$

This concludes the proof for inequalities (11) and (12). Next we consider a triplet of nodes $\{i,j,k\}$ where $(i,j) \in F^+$, $(j,k) \in F^+$ but $(i,k) \in F^-$. This is called a *bad triangle* since we will have to violate at least one of these edges when clustering $\tilde{G}$. We must prove that:

$$w^+_{ij} + w^+_{jk} + w^-_{ik} \leq 3\left(c_{ij} + c_{jk} + c_{ik}\right), \quad (13)$$

which is somewhat tedious to show. The variables in (13) are highly dependent on the types of edges shared among nodes $\{i,j,k\}$ in the original signed graph G'; there are two possibilities for each edge for a total of eight cases. We give a proof here for the first case.

Case 1: Assume $(i,j) \in E^+$, $(j,k) \in E^+$, and $(i,k) \in E^-$. For this case, we note that $(c_{ij}, c_{jk}, c_{ik}) = ((1-\lambda)x_{ij}, (1-\lambda)x_{jk}, \lambda(1-x_{ik}))$ and $(w^+_{ij}, w^+_{jk}, w^-_{ik}) = (1-\lambda, 1-\lambda, \lambda)$. By our construction of F^+ and F^- we know that $x_{ij} < 1/3$, $x_{jk} < 1/3$, and by the triangle inequality we have $x_{ik} \leq x_{ij} + x_{jk} < 2/3$. Combining these facts:

$$\begin{aligned} 3(c_{ij} + c_{jk} + c_{ik}) &= 3\left((1-\lambda)(x_{ij} + x_{jk}) + \lambda(1-x_{ik})\right) \\ &\geq 3\left((1-\lambda)x_{ik} + \lambda(1-x_{ik})\right) = 3\left((1-2\lambda)x_{ik} + \lambda\right) \\ &> 3\left((1-2\lambda)2/3 + \lambda\right) = 2 - \lambda = w^+_{ij} + w^+_{jk} + w^-_{ik}. \end{aligned}$$

We rely above on the fact that $(1-2\lambda) < 0$, which restricts our proof to cases where $\lambda > 1/2$. Due to space constraints we defer the proof of the other seven cases to the full version of the paper [39]. □

4.2 2-Approximation for Cluster Deletion

In order to obtain an approximation algorithm for cluster deletion we alter threeLP in two ways:

- Begin by solving the LP relaxation of cluster deletion:

$$\begin{array}{ll} \text{minimize} & \sum_{(i,j)\in E^+} x_{ij} \\ \text{subject to} & x_{ij} \leq x_{ik} + x_{jk} \text{ for all } i,j,k \\ & x_{ij} \in [0,1] \text{ for all } (i,j) \in E^+ \\ & x_{ij} = 1 \text{ for all } (i,j) \in E^- \end{array} \quad (14)$$

- Define $F^+ = \{(i,j) : x_{ij} < 1/2\}$, $F^- = \{(i,j) : x_{ij} \geq 1/2\}$.

As before, we then run Pivot on $\tilde{G} = (V, F^+, F^-)$. We name the resulting procedure twoCD, and prove the following result:

Theorem 4.3. *Algorithm twoCD returns a 2-approximation for cluster deletion.*

Proof. First observe that no negative edge mistakes are made by performing Pivot on $\tilde{G}$: if k is the pivot and i,j are two positive neighbors of k in $\tilde{G}$, then $x_{ik} < 1/2$, $x_{jk} < 1/2$, and $x_{ij} \leq x_{ik} + x_{jk} < 1$. Since all distances are less than one, all nodes must share positive edges in G', because $x_{ij} = 1$ for any $(i,j) \in E^-$. The rest of the proof follows from showing that the newly constructed graph $\tilde{G}$ satisfies the assumptions of Theorem 4.1 when $\alpha = 2$. For the sake of space we defer details to the full version of the paper [39]. □

This result is particularly interesting given that no constant-factor approximation for cluster deletion has been explicitly presented in previous literature.[2] Until now, the best approximations for correlation clustering were stronger than any result known for cluster deletion; our result indicates that the latter problem is perhaps the easier of the two to approximate.

4.3 Scalable Heuristic Algorithms

As a counterpart to the previous approximation-driven approaches, we provide fast algorithms for LambdaCC based on greedy local heuristics. The first of these is GrowCluster, which iteratively selects an unclustered node uniformly at random and forms a cluster around it by greedily aggregating adjacent nodes, until there is no more improvement to the LambdaCC objective.

A variant of this, called GrowClique, is specifically designed for cluster deletion. It monotonically improves the LambdaCC objective, but differs in that at each iteration it uniformly at random selects k unclustered nodes, and greedily grows cliques around each of these seeds. The resulting cliques may overlap: at each iteration we select only the largest of such cliques.

Finally, since the LambdaCC and Hamiltonian objectives are equivalent, we can use previously developed algorithms and software for modularity-like objectives with a resolution parameter. In particular we employ adaptations of the *Louvain* method, an algorithm developed by Blondel et al. [7]. It iteratively visits each node in the graph and moves it to an adjacent cluster, if such a move gives a locally maximum increase in the modularity score. This continues until no move increases modularity, at which point the clusters are aggregated into super-nodes and the entire process is repeated on the aggregated network. By adapting the original Louvain method to make greedy local moves based on the LambdaCC objective, rather than modularity, we obtain a scalable algorithm

[2]The results of van Zuylen and Williamson for constrained correlation clustering can be used to obtain a 3-approximation for cluster deletion (Theorem 4.2 in [38]), though cluster deletion is not mentioned explicitly in their work.

that is known to provide good approximations for a related objective, and additionally adapts well to changes in our parameter λ. We refer to this as Lambda-Louvain. Both standard and degree-weighted versions of the algorithm can be achieved by employing existing generalized Louvain algorithms (e.g., the GenLouvain algorithm of Jeub et al. http://netwiki.amath.unc.edu/GenLouvain/). Our heuristic algorithms satisfy the following guarantee:

Theorem 4.4. *For every λ, both the algorithms GrowCluster and Lambda-Louvain either place all nodes in one cluster or they produce clusters that have scaled sparsest cut bounded above by λ.*

An analogous result for normalized cut holds when the algorithms greedily optimize degree-weighted LambdaCC. We provide a detailed proof in the full version of the paper [39].

5 EXPERIMENTS

We begin by comparing our new methods against existing correlation clustering algorithms on several small networks. This shows our algorithms for LambdaCC are superior to common alternatives. We then study how well-known graph partitioning algorithms implicitly optimize the LambdaCC objective for various λ. In subsequent experiments, we apply our methods to clique detection in collaboration and gene networks, and to social network analysis.

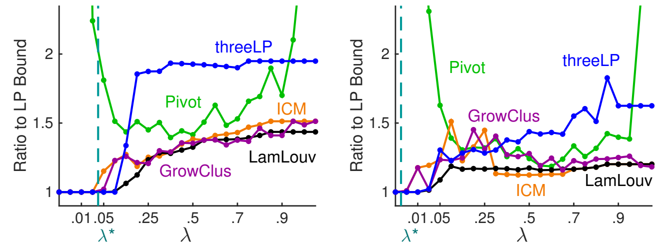
5.1 LambdaCC on Small Networks

In our first experiment, we show that Lambda-Louvain is the best general-purpose correlation clustering method for minimizing the LambdaCC objective. We test this on four small networks: Karate [40], Les Mis [26], Polbooks [27], and Football [22]. Figure 3 shows the performance of our algorithms, Pivot, and ICM, for a range of λ values. Pivot is the fast algorithm of Ailon et al. [1], which selects a uniform random node and clusters its neighbors with it. ICM is the energy-minimization heuristic algorithm of Bagon and Galun [3].

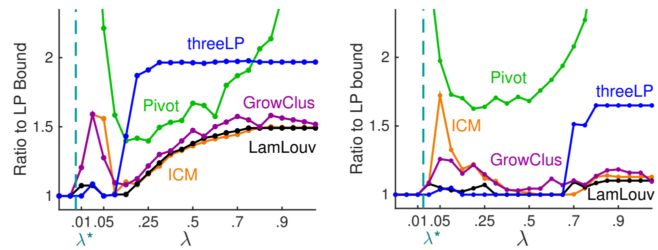
We find that threeLP gives much better than a 3-approximation in practice. Pivot is much faster, but performs poorly for λ close to 0 or 1. ICM is also much quicker than solving the LP relaxation, but is still limited in scalability, as it is intended for correlation clustering problems where most edge weights are 0 (not the case for LambdaCC). On the other hand, GrowCluster and Lambda-Louvain are scalable and give good approximations for all input networks and values of λ.

5.2 Standard Clustering Algorithms

Many existing clustering algorithms implicitly optimize different parameter regimes of the LambdaCC objective. We show this by running several clustering algorithms on a 1000-node synthetic graph generated from the BTER model [35]. We do the same on the largest component (4158 nodes) of the ca-GrQc collaboration network from the *arXiv* e-print website. We then compute LambdaCC objective scores for each algorithm for a range of λ values. We first cluster each graph using Graclus [18] (forming two clusters), Infomap [9], and Louvain [7]. To form dense clusters, we also partition the networks by recursively extracting the maximum clique (called RMC), and by recursively extracting the maximum quasi-clique (RMQC), i.e., the largest set of nodes with inner edge density



(a) Karate, n = 34, λ^* = .0267 **(b) Les Mis, n = 77, λ^* = .0045**



(c) Polbooks, n = 105, λ^* = .0069 **(d) Football, n = 115, λ^* = .0184**

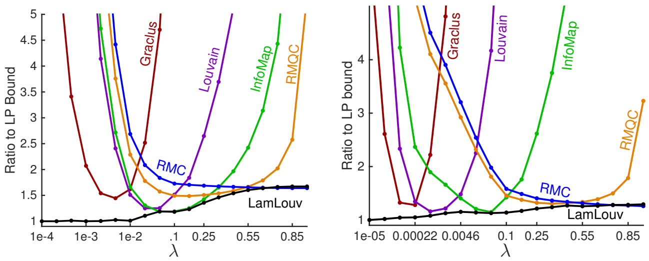
Figure 3: We optimize the standard LambdaCC objective with five correlation clustering algorithms on four small networks. The y-axis reports the ratio between each algorithm's score and the lower bound on the optimal objective determined by solving the LP relaxation. Lambda-Louvain (black) and GrowCluster (purple) perform well for all λ, in addition to being the most scalable algorithms. In each plot, a dashed vertical line indicates the optimal scaled sparsest cut value, λ^*, for that network.

bounded below by some $\rho < 1$ (here we use $\rho = 0.6$). The last two procedures must solve an NP-hard objective at each step, but for reasonably sized graphs there is available clique and quasi-clique detection software [28, 34].

After each algorithm has produced a single clustering of the unsigned network, we evaluate how the LambdaCC objective score of that clustering changes as we vary λ. This allows us to observe whether an algorithm is effective in approximating the LambdaCC objective for a certain range of λ values. For comparison we run Lambda-Louvain for each different value of λ. Figure 4 reports the ratio between the LambdaCC objective score of each clustering and the LP-relaxation lower bound. These plots illustrate that our framework and Lambda-Louvain effectively interpolate between several well-established strategies in graph partitioning, and can serve as a good proxy for any clustering task for which any one of these algorithms is known to be effective.

5.3 Cluster Deletion in Large Collaboration Networks

The connection between LambdaCC and cluster deletion provides a new approach for enumerating groups in large networks. Here we evaluate GrowClique for cluster deletion and use it to cluster two large collaboration networks, one formed from a snapshot of the author-paper DBLP dataset in 2007, and the other generated using



(a) BTER graph with 1000 nodes **(b) Largest component of ca-GrQc**

Figure 4: We illustrate the performance of well-known clustering algorithms in approximating the LambdaCC objective on (a) one synthetic and (b) and one real-world graph. The bowl-shaped curves indicate that each algorithm implicitly optimizes the LambdaCC objective in a different parameter regime. The y-axis reports the ratio between each clustering's objective score and the LP-relaxation lower bound. Lambda-Louvain effectively interpolates between all the clustering strategies seen here.

Table 2: Cluster deletion scores for GrowClique (GC), ProjectClique (PC) and RMC on two collaboration networks. GrowClique is unaware of the underlying player-project network, and does not solve an NP-hard objective at each iteration, yet returns very good results. Best score for each dataset is *emphasized.*

Graph	Nodes	Edges	GC	PC	RMC
Actors	341,185	10,643,420	*8,085,286*	8,086,715	8,087,241
DBLP	526,303	1,616,814	945,489	946,295	*944,087*

actor-movie information from the NotreDame actors dataset [5]. The original data in both cases is a bipartite network indicating which *players* (i.e., authors or actors) have parts in different *projects* (papers or movies respectively). We transform each bipartite network into a graph in which nodes are players and edges represent collaboration on a project.

At each iteration GrowClique grows 500 (possibly overlapping) cliques from random seeds and selects the largest to be included in the final output. We compare against RMC, an expensive method which provably returns a 2-approximation to the optimal cluster deletion objective [17]. We also design ProjectClique, a method that looks at the original bipartite network and recursively identifies the project associated with the largest number of players not yet assigned to a cluster. These players form a clique in the collaboration network, so ProjectClique clusters them together, then repeats the procedure on remaining nodes.

Table 2 shows that GrowClique outperforms ProjectClique in both cases, and slightly outperforms RMC on the actor network. Our method is therefore competitive against two algorithms that in some sense have an unfair advantage over it: ProjectClique employs knowledge not available to GrowClique regarding the original bipartite dataset, and RMC performs very well mainly because it solves an NP-hard problem at each step.

5.4 Clustering Yeast Genes

The study of cluster deletion and cluster editing (equivalent to ±1-correlation clustering, or to $\lambda = 1/2$) was originally motivated by applications to clustering genes using expression patterns [6, 36]. Standard LambdaCC is a natural framework for this, since it generalizes both objectives and interpolates between them as λ ranges from $1/2$ to $m/(m+1)$. We cluster genes of the *Saccharomyces cerevisiae* yeast organism using microarray expression data collected by Kemmeren et al. [25]. With the 200 expression values from the dataset, we compute correlation coefficients between all pairs of genes. We threshold these at 0.9 to obtain a small graph of 131 nodes corresponding to unique genes, which we cluster with twoCD. For this cluster deletion experiment, our algorithm returns the optimal solution: solving the LP-relaxation returns a solution that is in fact integral. We validate each clique of size at least three returned by twoCD against known gene-association data from the Saccharomyces Genome Database (SGD) and the String Consortium Database (see Table 3). With one exception, these cliques match groups of genes that are known to be strongly associated, according to at least one validation database. The exception is a cluster with four genes (YHR093W, YIL171W, YDR490C, and YOR225W), three of which, according to the SGD are not known to be associated with any Gene Ontology term. We conjecture that this may indicate a relationship between genes not previously known to be related.

Table 3: We list cliques of size ≥ 3 in the optimal clustering (found by twoCD) of a network of 131 yeast genes. We validate each cluster using the SGD GO slim mapper tool, which identifies any GO term (function, process, or component of the organism) for a given gene. We list one GO term shared by all genes in the cluster, if one exists. The Term % column reports the percentage of all genes in the organism associated with this term. A low percentage indicates a cluster of genes that share a GO term that is not widely shared among other genes. The final column shows the minimum String association score between every pair of genes in the cluster, a number between 0 and 1000 (higher is better). Any non-zero score is a strong indication of gene association, as the majority of String scores between genes of *S. cerevisiae* are zero. All clusters, except the third, either have a high minimum String score or are all associated with a specific GO term.

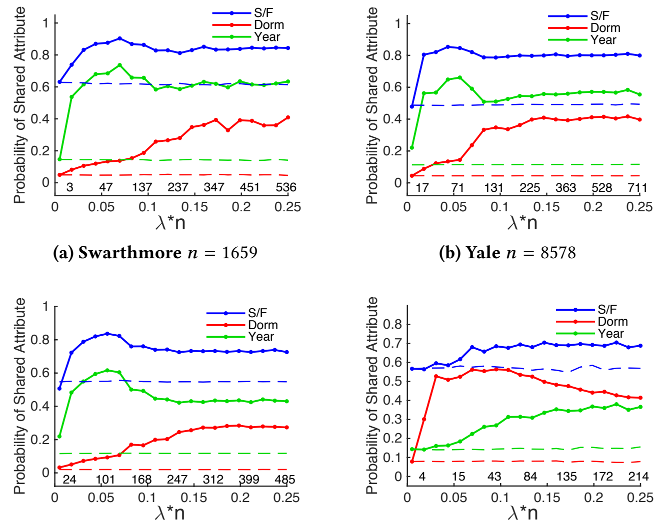
Clique #	Size	Shared GO term	Term %	String
1	6	nucleus	34.3	0
2	4	nucleus	34.3	202
3	4	N/A	-	0
4	4	vitamin metabolic process	0.7	980
5	3	cytoplasm	67.0	990
6	3	cytoplasm	67.0	998
7	3	N/A	-	962
8	3	cytoplasm	67.0	996
9	3	N/A	-	973
10	3	transposition	1.7	0

5.5 Social Network Analysis with LambdaCC

Clustering a social network using a range of resolution parameters can reveal valuable insights about how links are formed in the network. Here we examine several graphs from the Facebook100 dataset, each of which represents the induced subgraph of the Facebook network corresponding to a US university at some point in 2005. The networks come with anonymized meta-data, reporting attributes such as major and graduation year for each node. While meta-data attributes are not expected to correspond to ground-truth communities in the network [32], we do expect them to play a role in how friendship links and communities are formed. In this experiment we illustrate strong correlations between the link structure of the networks and the dorm, graduation year, and student/faculty status meta-data attributes. We also see how these correlations are revealed, to different degrees, depending on our choice of λ.

Given a Facebook subgraph with n nodes, we cluster it with degree-weighted Lambda-Louvain for a range of λ values between $0.005/n$ and $0.25/n$. In this clustering, we refer to two nodes in the same cluster as an *interior pair*. We measure how well a meta-data attribute M correlates with the clustering by calculating the proportion of interior pairs that share the same value for M. This value, denoted by $P(M)$, can also be interpreted as the probability of selecting an interior pair uniformly at random and finding that they agree on attribute M. To determine whether the probability is meaningful, we compare it against a null probability $P(\tilde{M})$: the probability that a random interior pair agree at a *fake* meta-data attribute $\tilde{M}$. We assign to each node a value for the fake attribute $\tilde{M}$ by performing a random permutation on the vector storing values for true attribute M. In this way, we can compare each true attribute M against a fake attribute $\tilde{M}$ that has the same exact proportion of nodes with each attribute value, but does not impart any true information regarding each node.

In Figure 5 we plot results for each of the three attributes $M \in \{dorm, year, s/f \text{ (student/faculty)}\}$ on four Facebook networks, as λ is varied. In all cases, we see significant differences between $P(M)$ and $P(\tilde{M})$. In general, $P(year)$ and $P(s/f)$ reach a peak at small values of λ when clusters are large, whereas $P(dorm)$ is highest when λ is large and clusters are small. This indicates that the first two attributes are more highly correlated with large sparse communities in the network, whereas sharing a dorm is more correlated with smaller, denser communities. Caltech, a small residential university, is an exception to these trends and exhibits a much stronger correlation with the dorm attribute, even for very small λ.



(a) Swarthmore $n = 1659$ **(b) Yale $n = 8578$** **(c) Cornell $n = 18660$** **(d) Caltech $n = 769$**

Figure 5: On four university Facebook graphs, we illustrate that the dorm (red), graduation year (green), and student/-faculty (S/F) status (blue) meta-data attributes all correlate highly with the clustering found by Lambda-Louvain for each λ. Above the x-axis we show the number of clusters formed, which increases with λ. The y-axis reports the probability that two nodes sharing a cluster also share an attribute value. Each attribute curve is compared against a null probability, shown as a dashed line of the same color. The large gaps between each attribute curve and its null probability indicate that the link structure of each network is highly correlated with these attributes. In general, probabilities for *year* and s/f status are highest for small λ, whereas *dorm* has a higher correlation with smaller, denser communities in the network. Caltech is an exception to the general trend; see the main text for discussion.

6 DISCUSSION

We have introduced a new clustering framework that unifies several other commonly-used objectives and offers many attractive theoretical properties. We prove that our objective function interpolates between the sparsest cut objective and the cluster deletion problem, as we vary a single input parameter, λ. We give a 3-approximation algorithm for our objective when $\lambda \geq 1/2$, and a related method which improves the best approximation factor for cluster deletion from 3 to 2. We also give scalable procedures for greedily improving our objective, which are successful in a wide variety of clustering applications. These methods are easily modified to add must-cluster and cannot-cluster constraints, which makes them amenable to many applications. In future work, we will continue exploring approximations when $\lambda < 1/2$.

ACKNOWLEDGEMENTS

This work was supported by several funding agencies: Nate Veldt and David Gleich are supported by NSF award IIS-154648, David Gleich is additionally supported by NSF awards CCF-1149756 and CCF-093937 as well as the DARPA Simplex program and the Sloan Foundation. Anthony Wirth is supported by the Australian Research Council. We thank Flavio Chierichetti for several helpful conversations and also thank the anonymous reviewers for several helpful suggestions for improving our work, in particular for mentioning connections to the work of van Zuylen and Williamson [38], which led to significantly improved approximation results.

REFERENCES

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

[2] Sanjeev Arora, Satish Rao, and Umesh Vazirani. 2009. Expander flows, geometric embeddings and graph partitioning. *Journal of the ACM (JACM)* 56, 2 (2009).

[3] Shai Bagon and Meirav Galun. 2011. Large Scale Correlation Clustering Optimization. *arXiv* cs.CV (2011), 1112.2903.

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

[5] Albert-László Barabási and Réka Albert. 1999. Emergence of Scaling in Random Networks. *Science* 286, 5439 (1999), 509–512. https://doi.org/10.1126/science.286.5439.509

[6] Amir Ben-Dor, Ron Shamir, and Zohar Yakhini. 1999. Clustering gene expression patterns. *Journal of computational biology* 6, 3-4 (1999), 281–297.

[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] Sebastian Böcker and Peter Damaschke. 2011. Even faster parameterized cluster deletion and cluster editing. *Inform. Process. Lett.* 111, 14 (2011), 717 – 721. https://doi.org/10.1016/j.ipl.2011.05.003

[9] Ludvig Bohlin, Daniel Edler, Andrea Lancichinetti, and Martin Rosvall. 2014. Community detection and visualization of networks with the map equation framework. In *Measuring Scholarly Impact*. Springer, 3–34.

[10] Flavia Bonomo, Guillermo Duran, Amedeo Napoli, and Mario Valencia-Pabon. 2015. A one-to-one correspondence between potential solutions of the cluster deletion problem and the minimum sum coloring problem, and its application to P4-sparse graphs. *Inform. Process. Lett.* 115 (2015), 600–603.

[11] Flavia Bonomo, Guillermo Duran, and Mario Valencia-Pabon. 2015. Complexity of the cluster deletion problem on subclasses of chordal graphs. *Theoretical Computer Science* 600 (2015), 59–69.

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

[13] 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.

[14] Peter Damaschke. 2009. *Bounded-Degree Techniques Accelerate Some Parameterized Graph Algorithms*. Springer Berlin Heidelberg, Berlin, Heidelberg, 98–109. https://doi.org/10.1007/978-3-642-11269-0_8

[15] 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.

[16] Erik D. Demaine and Nicole Immorlica. 2003. Correlation Clustering with Partial Information. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques: 6th International Workshop on Approximation Algorithms for Combinatorial Optimization Problems, APPROX 2003 and 7th International Workshop on Randomization and Approximation Techniques in Computer Science, RANDOM 2003, Princeton, NJ, USA, August 24-26, 2003. Proceedings*, Sanjeev Arora, Klaus Jansen, José D. P. Rolim, and Amit Sahai (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 1–13.

[17] Anders Dessmark, Jesper Jansson, Andrezej Lingas, Eva-Marta Lundell, and Mia Person. 2007. On the Approximability of Maximum and Minimum Edge Clique Partition Problems. *International Journal of Foundations of Computer Science* 18, 02 (2007), 217–226.

[18] Inderjit S Dhillon, Yuqiang Guan, and Brian Kulis. 2007. Weighted graph cuts without eigenvectors a multilevel approach. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 29, 11 (2007).

[19] Thang N Dinh, Xiang Li, and My T Thai. 2015. Network clustering via maximizing modularity: Approximation algorithms and theoretical limits. In *Proceedings of the 2015 IEEE International Conference on Data Mining (ICDM)*. IEEE, 101–110.

[20] Dotan Emanuel and Amos Fiat. 2003. Correlation Clustering – Minimizing Disagreements on Arbitrary Weighted Graphs. In *Algorithms - ESA 2003: 11th Annual European Symposium, Budapest, Hungary, September 16-19, 2003. Proceedings*, Giuseppe Di Battista and Uri Zwick (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 208–220. https://doi.org/10.1007/978-3-540-39658-1_21

[21] Yong Gao, Donovan R Hare, and James Nastos. 2013. The cluster deletion problem for cographs. *Discrete Mathematics* 313, 23 (2013), 2763–2771.

[22] Michelle Girvan and Mark EJ Newman. 2002. Community structure in social and biological networks. *Proceedings of the National Academy of Sciences* 99, 12 (2002), 7821–7826.

[23] Jens Gramm, Jiong Guo, Falk Hüffner, and Rolf Niedermeier. 2003. Graph-Modeled Data Clustering: Fixed-Parameter Algorithms for Clique Generation. In *Algorithms and Complexity: 5th Italian Conference, CIAC 2003, Rome, Italy, May 28–30, 2003. Proceedings*, Rossella Petreschi, Giuseppe Persiano, and Riccardo Silvestri (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 108–119. https://doi.org/10.1007/3-540-44849-7_17

[24] Jens Gramm, Jiong Guo, Falk Hüffner, and Rolf Niedermeier. 2004. Automated Generation of Search Tree Algorithms for Hard Graph Modification Problems. *Algorithmica* 39, 4 (2004), 321–347. https://doi.org/10.1007/s00453-004-1090-5

[25] Patrick Kemmeren, Katrin Sameith, Loes AL van de Pasch, Joris J Benschop, Tineke L Lenstra, Thanasis Margaritis, Eoghan O'Duibhir, Eva Apweiler, Sake van Wageningen, Cheuk W Ko, et al. 2014. Large-scale genetic perturbations reveal regulatory networks and an abundance of gene-specific repressors. *Cell* 157, 3 (2014), 740–752.

[26] D. E. Knuth. 1993. *The Stanford GraphBase: A Platform for Combinatorial Computing*. Addison-Wesley, Reading, MA.

[27] V. Krebs. 2004. Books about US Politics. (2004). http://networkdata.ics.uci.edu/data.php?d=polbooks Hosted at UCI Data Repository.

[28] Guimei Liu and Limsoon Wong. 2008. Effective Pruning Techniques for Mining Quasi-Cliques. In *Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2008, Antwerp, Belgium, September 15-19, 2008, Proceedings, Part II*, Walter Daelemans, Bart Goethals, and Katharina Morik (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 33–49. "https://doi.org/10.1007/978-3-540-87481-2_3"

[29] Assaf Natanzon, Ron Shamir, and Roded Sharan. 1999. Complexity classification of some edge modification problems. In *International Workshop on Graph-Theoretic Concepts in Computer Science*. Springer, 65–77.

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

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

[32] 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

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

[34] Ryan A. Rossi, David F. Gleich, and Assefaw H. Gebremedhin. 2015. Parallel Maximum Clique Algorithms with Applications to Network Analysis. *SIAM Journal on Scientific Computing* 37, 5 (2015), C589–C616. https://doi.org/10.1137/14100018X

[35] C. Seshadhri, Tamara G. Kolda, and Ali Pinar. 2012. Community Structure and Scale-free Collections of Erdős-Rényi Graphs. *Physical Review E* 85, 5, Article 056109 (May 2012). https://doi.org/10.1103/PhysRevE.85.056109

[36] Ron Shamir, Roded Sharan, and Dekel Tsur. 2004. Cluster graph modification problems. *Discrete Applied Mathematics* 144 (2004), 173–182.

[37] Chaitanya Swamy. 2004. Correlation clustering: maximizing agreements via semidefinite programming. In *Proceedings of the Fifteenth Annual ACM-SIAM Symposium on Discrete Algorithms*. Society for Industrial and Applied Mathematics, 526–527.

[38] Anke van Zuylen and David P. Williamson. 2009. Deterministic Pivoting Algorithms for Constrained Ranking and Clustering Problems. *Mathematics of Operations Research* 34, 3 (2009), 594–620. https://doi.org/10.1287/moor.1090.0385 arXiv:https://doi.org/10.1287/moor.1090.0385

[39] Nate Veldt, David Gleich, and Tony Wirth. 2017. Unifying Sparsest Cut, Cluster Deletion, and Modularity Clustering Objectives with Correlation Clustering. *arXiv* cs.DS (2017). https://arxiv.org/abs/1712.05825

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