Clustering with Qualitative Information

Moses Charikar*
Princeton University

Venkatesan Guruswami†
University of Washington

Anthony Wirth‡
Princeton University

Abstract

We consider the problem of clustering a collection of elements based on pairwise judgments of similarity and dissimilarity. Bansal, Blum and Chawla [1] cast the problem thus: given a graph G whose edges are labeled "$+$" (similar) or "$-$" (dissimilar), partition the vertices into clusters so that the number of pairs correctly (resp. incorrectly) classified with respect to the input labeling is maximized (resp. minimized). Complete graphs, where the classifier labels every edge, and general graphs, where some edges are not labeled, are both worth studying. We answer several questions left open in [1] and provide a sound overview of clustering with qualitative information.

We give a factor 4 approximation for minimization on complete graphs, and a factor $O(\log n)$ approximation for general graphs. For the maximization version, a PTAS for complete graphs is shown in [1]; we give a factor 0.7664 approximation for general graphs, noting that a PTAS is unlikely by proving APX-hardness. We also prove the APX-hardness of minimization on complete graphs.

1. Introduction

The problem of grouping a corpus of data into clusters that contain similar items arises in numerous contexts and disciplines. Deservedly, it has been studied extensively in the algorithms and combinatorial optimization literature. Much of this literature works with the following abstraction of the problem: the input is represented as a table of *distances* between pairs of items where the distance between x and y represents *how different* x and y are. The goal is to find a clustering of the data that optimizes some function of the distances between items within or across clusters under some global constraint, such as knowledge of the total number of clusters. Quintessential examples include k-center, k-median, and k-sum clustering.

This clustering paper departs from the above distance paradigm. All we have at our disposal is *qualitative information* from a classifier that labels pairs of elements as similar or dissimilar. We are not provided with any quantitative information on *how different* pairs of elements are. Our aim is to produce a partitioning into clusters that puts similar objects in the same cluster and dissimilar objects in different clusters, to the extent possible. If there exists a clustering that is *correct* for every edge, then the problem is trivially solved by identifying as clusters connected components in the graph of similar pairs (see below). When the classifier has made mistakes, interesting and non-trivial questions arise; for instance, finding a clustering that differs from the classifier's verdicts on the fewest possible pairs. Bansal, Blum, and Chawla initiated the study of these kinds of clustering problems [1].

The obvious graph-theoretic formulation of the problem is the following: given a graph $G = (V, E)$ with each edge labeled either "$+$" (similar) or "$-$" (dissimilar), find a partitioning of the vertices into clusters that agrees as much as possible with the edge labels. The maximization version, denoted MaxAgree in this paper, seeks to maximize the number of agreements: the number of $+$ edges inside clusters plus the number of $-$ edges across clusters. The minimization version, denoted MinDisAgree, aims to minimize the number of disagreements: the number of $-$ edges within clusters plus the number of $+$ edges between clusters. An intriguing feature of this clustering problem is that, unlike most clustering formulations, we do not need to specify the number of clusters k as a parameter. We only have a single objective, and whether the optimal solution uses few or many clusters is automatically dictated by the edge labels.

If the classifier can be used to label every pair of elements as $+$ or $-$, then G will be a complete graph. So that we can capture situations where the classifier might be unable to tell if certain pairs of elements are similar or dissimilar, we do not insist on complete graphs. One upshot of the clustering will be to deduce the missing labels

*Email: moses@cs.princeton.edu. Supported by NSF ITR grant CCR-0205594, DOE Early Career Principal Investigator award DE-FG02-02ER25540, NSF CAREER award CCR-0237113 and an Alfred P. Sloan Fellowship

†Email: venkat@cs.washington.edu.

‡Email: awirth@cs.princeton.edu. Supported by a Gordon Wu Fellowship and NSF ITR grant CCR-0205594.

from the existing ones. Also, in some cases the classifier might provide confidence information on each of its labels. This can be captured by weights on the edges and one can then consider natural weighted versions of MAXAGREE and MINDISAGREE.

Previous Work Bansal *et al.* mainly focused on the case when G is the complete graph and presented results for both the maximization and minimization versions [1]. They established that the problems are NP-hard and therefore studied approximation algorithms with provable performance guarantees. They gave a polynomial time approximation scheme (henceforth, PTAS) for MAXAGREE on complete graphs. For the minimization version, they gave an approximation algorithm with performance ratio a constant. The constant is a rather large one, so it should be viewed as a qualitative result, demonstrating that a constant factor approximation can be achieved.

Recently, both Demaine and Immorlica [3], and Emanuel and Fiat [4], independently from each other and from this paper, announced results on clustering with qualitative information. Both papers focus on MINDISAGREE in general graphs. The authors of [3] present a factor $O(\log n)$ algorithm for general graphs, based on *region growing*, and demonstrate an approximation-preserving reduction from (weighted) minimum multicut. They also provide an $O(r^3)$ approximation algorithm for $K_{r,r}$-minor-free graphs. In [4], both reductions to and from minimum multicut are presented; in particular the authors show a reduction from unweighted multicut to unweighted MINDISAGREE.

Our Results In this paper, we present several results on clustering problems of this nature, and answer several questions left open by the work of Bansal *et al.* [1]. As a consequence, our results provide a better overview of the approximability of the various variants of clustering with qualitative information.

COMPLETE GRAPHS Our main algorithmic result here is a factor 4 approximation algorithm for MINDISAGREE on complete graphs. This significantly improves on the performance ratio achieved in [1]. The algorithm in [1] was combinatorial; in contrast our algorithm is based on a natural linear programming relaxation, and rounding the fractional solution (a semi-metric on the vertices) using the *region-growing* approach. The completeness of the graph allows us to to achieve a constant approximation using region-growing, instead of the usual logarithmic factor. The integrality gap of our LP formulation is 2 and we also show that beating factor 3 would require significant departure from our rounding strategy. To complement our algorithmic result, we also prove that MINDISAGREE on complete graphs is APX-hard (i.e., is NP-hard to approximate within some constant factor greater than 1) via a somewhat intricate reduction. The reduction used in [1] to prove NP-hardness does not yield APX-hardness. In contrast, the maximization version does admit a PTAS [1].

GENERAL GRAPHS Bansal *et al.* did not give any algorithms for general graphs, but noted that MINDISAGREE is APX-hard. They provided evidence that MAXAGREE is unlikely to admit a PTAS (unlike the complete graph case) by showing that a PTAS would imply a much better algorithm for coloring 3-colorable graphs than is currently known. We give a factor $O(\log n)$ approximation algorithm for MINDISAGREE—this follows from a straightforward modification of the Garg, Vazirani, Yannakakis (henceforth GVY) algorithm for minimum multicut [7]. We also note that MINDISAGREE is at least as hard to approximate as multicut, so a constant factor approximation algorithm would be a major breakthrough.

We prove that MAXAGREE is APX-hard and thereby provide a concrete hardness result—in contrast to the above *evidence* of hardness based on a relation to graph coloring. On the algorithmic side, the naive $1/2$-approximation algorithm, namely choosing the better of placing all elements in a single cluster and placing each of them in a separate cluster, was the best known for MAXAGREE. We give a factor 0.766 approximation algorithm based on rounding a semidefinite programming relaxation. Moreover, if a clustering that correctly classifies most—say a fraction $(1-\varepsilon)$—of the edges exists, then our algorithm will also find one with a similar property (we defer the quantitative statement to the relevant technical section). Our interest in the latter result is due in part to the fact that it brings out some of the difficulty that must be overcome if one tries to prove a super-constant factor inapproximability result for MINDISAGREE. Such a result must focus on instances where an almost perfect clustering exists for both the *yes* and *no* cases of the gap reduction.

Organization We present algorithms for general graphs in Section 2. We then turn to complete graphs and describe our factor 4 approximation algorithm for MINDISAGREE in Section 3. Finally, we present the inapproximability results that complement our algorithms in Section 4.

2. Algorithms for general graphs

In this section, we consider the problems MINDISAGREE and MAXAGREE on general weighted graphs.

2.1. MINDISAGREE

We describe a natural LP relaxation for MINDISAGREE. This is very similar to the LP used in the GVY minimum multicut algorithm [7].

A partitioning into clusters can be represented by a set of binary variables, one for each pair of vertices. If i and j are in the same cluster x_{ij} is 1; if they are in different clusters

$$\begin{array}{lll} \text{minimize} & \sum_{+(ij)} w_{ij} \cdot x_{ij} + \sum_{-(ij)} w_{ij} \cdot (1 - x_{ij}) & \\ \text{such that} & x_{ik} \leq x_{ij} + x_{jk} & \text{for all } i, j, k \\ & x_{ij} \in \{0, 1\} & \text{for all } i, j \end{array}$$

Figure 1. Integer linear program for MINDISAGREE **in general weighted graphs**

x_{ij} is 0. Since each cluster is an equivalence class, we know that if $x_{ij} = 0$ and $x_{jk} = 0$ then $x_{ik} = 0$. We can express this fact using the triangle inequality,

$$x_{ik} \leq x_{ij} + x_{jk}.$$

The objective is to minimize the number of mistakes: the number of positive edges for which x_{ij} is one and the number of negative edges for which x_{ij} is zero. The integer program in Figure 1 summarizes the situation: $+(ij)$ indicates that the edge between i and j has a positive label, similarly for $-(ij)$. The LP relaxation is obtained by replacing the integer requirements in Figure 1 with $0 \leq x_{ij} \leq 1$ for all i, j.

Let the value of the optimal LP solution be denoted by $\mathsf{OPT_{LP}}$. A fairly straightforward application of the GVY region growing procedure yields a solution of cost at most $O(\log n)\mathsf{OPT_{LP}}$. We briefly describe the algorithm and outline the analysis.

We will refer to x_{ij} as the *distance* between i and j, which is consistent with the fact that x_{ij} is a semi-metric in the range $[0, 1]$. Intuitively, points that are *close* should be placed in the same cluster and points that are *far* should be placed in different clusters. Let $B_x(i, r)$ denote the set of points within distance r from i. For a set of vertices S, let $\delta(S)$ be the set of edges from S to $\overline{S}$.

Algorithm 1

1. $\mathcal{C} \leftarrow \emptyset$. /* Collection of clusters */
2. While there exist i, j in the graph such that $x_{ij} > 2/3$:
   (a) Let $S = B_x(i, r)$ for some $r < 1/3$.
   /* See proof for value of r */
   (b) $\mathcal{C} \leftarrow \mathcal{C} \cup \{S\}$.
   (c) Remove S and $\delta(S)$ from the current graph.
3. Return $\mathcal{C}$.

Theorem 1 *Algorithm 1 achieves a $O(\log n)$ approximation for* MINDISAGREE *on general graphs.*

Proof: The GVY region growing procedure suggests the choice of radius r in step 2(a) of the algorithm. Set $V_x^+(i, r)$ to be

$$\frac{\mathsf{OPT_{LP}}}{n} + \sum_{+(uv) \in B_x(i,r)} w_{ij} x_{ij} + \sum_{+(uv) \in \delta(B_x(i,r))} w_{ij} (r - x_{iu}).$$

This is the contribution to the LP solution from positive edges that have at least one endpoint in $B_x(i, r)$, plus an additional amount $\mathsf{OPT_{LP}}/n$. Let $W_x^+(i, r)$ denote the sum of weights of positive edges in $\delta(B_x(i, r))$. We pick $r < 1/3$ such that the ratio $W_x^+(i, r)/V_x^+(i, r)$ is minimized. The analysis technique in [7] can be used to show that there exists a radius $r < 1/3$ such that $W_x^+(i, r) \leq (3 \log n) V_x^+(i, r)$. This implies that the total weight of positive edges with end points in different clusters is at most $O(\log n)\mathsf{OPT_{LP}}$.

Now we account for the negative edges. For any two points i, j in the same cluster, $x_{ij} \leq 2/3$. Hence any negative edge ij that ends up inside a cluster in our solution contributes $w_{ij} \cdot (1 - x_{ij})$ to the LP, which is at least $w_{ij}/3$. On the other hand, we pay w_{ij} for this edge. This implies that the total weight of negative edges with end points in the same cluster is at most $O(\log n)\mathsf{OPT_{LP}}$. ■

The $O(\log n)$ approximation ratio we obtain from our LP is the best possible. Our LP formulation has integrality gap $\Omega(\log n)$, as shown by examples similar to the expander gap examples for minimum multicut.

We expect that a procedure such as this one, for learning distances from similarity judgment information, to have further applications in situations where no natural distance function exists.

2.2. MAXAGREE

Obtaining a $1/2$ approximation for MAXAGREE is trivial, as observed by Bansal *et al.* [1] for the complete graph. If the total weight of positive edges is greater than the total weight of negative edges, place all vertices in one cluster. Otherwise, put each of them in an individual cluster.

A linear program with poor integrality gap

Consider an LP relaxation for MAXAGREE similar to the LP used for MINDISAGREE. The constraints are exactly the same, but the objective is

$$\text{maximize} \quad \sum_{+(ij)} w_{ij} \cdot (1 - x_{ij}) + \sum_{-(ij)} w_{ij} \cdot x_{ij}$$

Theorem 2 *The integrality gap of the LP relaxation for* MAXAGREE *is no better than $2/3 + \varepsilon$ for any $\varepsilon > 0$.*

Proof: Our gap instance consists of two sets A and B of n vertices each. The graph is in fact complete, with every edge having a positive or negative label. The edges between A and B are positive; those with end points within the same set are negative. Thus there are n^2 positive edges and $n(n-1)$ negative edges. The LP solution assigns $x_{ij} = 1/2$ for positive edges ij and $x_{ij} = 1$ for negative edges ij, and so $\mathsf{OPT_{LP}}$ is $n(n-1) + n^2/2$. On the other hand, the value

$$\begin{array}{lll} \max & \sum_{+(ij)} w_{ij}(v_i \cdot v_j) + \sum_{-(ij)} w_{ij}(1 - v_i \cdot v_j) & \\ \text{s.t.} & v_i \cdot v_i = 1 & \text{for all } i \\ & v_i \cdot v_j \geq 0 & \text{for all } i, j \end{array}$$

Figure 2. SDP for maximizing agreements

of OPT for this instance is n^2—we leave the proof to the reader. Hence the integrality gap is $2n/(3n-2)$, which approaches $2/3$ as n increases. ■

Rounding a semidefinite program

We next consider a semidefinite program (SDP) for MAX-AGREE. To motivate the SDP, consider any clustering solution. Choose a collection of orthogonal unit vectors, one for each cluster in the solution. Every vertex i in the cluster is assigned the unit vector v_i corresponding to the cluster it is in. If vertices i and j are in the same cluster, then $v_i \cdot v_j = 1$, if not, $v_i \cdot v_j = 0$. The agreement of the clustering solution can now be expressed in terms of the dot products $v_i \cdot v_j$. With this vector solution in mind, we consider the SDP relaxation for MAXAGREE in Figure 2.

Consider the following general approach for rounding this SDP: Pick t random hyperplanes, dividing the set of vertices into 2^t clusters. We refer to this scheme as H_t. Our rounding scheme takes the better of the two solutions returned by H_2 and H_3, denoted by $\text{Best}(H_2, H_3)$.

Theorem 3 *The expected agreement of the solution returned by Best*(H_2, H_3) *is at least* $0.7664\ \mathsf{OPT}_{\mathsf{SDP}}$.

Proof: In order to analyze $\text{Best}(H_2, H_3)$, we consider a slightly different scheme: pick H_2 with probability $1-\alpha$ and pick H_3 with probability α, denoted by $\text{Comb}(H_2, H_3)$. Clearly the approximation ratio of $\text{Comb}(H_2, H_3)$ is a lower bound on the approximation ratio of $\text{Best}(H_2, H_3)$.

We perform an edge by edge analysis: For each edge ij, we measure the expected contribution to the solution produced relative to its SDP contribution. The edge weights are common to both OPT and the approximation and so can be ignored. Consider an edge ij such that the angle between v_i and v_j is $\theta \in [0, \pi/2]$. The probability that v_i and v_j are *not* separated by H_t is $(1-\theta/\pi)^t$.

If ij is a positive edge, the contribution to the SDP solution is $v_i \cdot v_j = \cos(\theta)$. On the other hand, the contribution to the agreement of $\text{Comb}(H_2, H_3)$ is

$$(1-\alpha)(1-\theta/\pi)^2 + \alpha(1-\theta/\pi)^3.$$

If ij is a negative edge, the contribution to the SDP solution is $1 - v_i \cdot v_j = 1 - \cos(\theta)$. On the other hand, the contribution to the agreement of $\text{Comb}(H_2, H_3)$ is

$$1 - (1-\alpha)(1-\theta/\pi)^2 - \alpha(1-\theta/\pi)^3.$$

Thus the approximation ratio can be bounded by

$$\min_{\theta \in [0, \pi/2]} \left\{ \frac{(1-\alpha)(1-\theta/\pi)^2 + \alpha(1-\theta/\pi)^3}{\cos(\theta)}, \frac{1 - (1-\alpha)(1-\theta/\pi)^2 - \alpha(1-\theta/\pi)^3}{1 - \cos(\theta)} \right\}$$

For $\alpha \leq 0.1316$, the minimum of the two expressions is $3/4 + \alpha/8$. In fact the minimum value of the second expression is $3/4 + \alpha/8$ for all $\alpha \in [0, 1]$ and is achieved when $\theta = \pi/2$. The upper bound on α is obtained by minimizing the first expression. Setting $\alpha = 0.1316$ yields a 0.7664 approximation. ■

The following simple example shows that the best approximation factor we can hope to achieve using this SDP is at most $2(\sqrt{2} - 1) \approx 0.8284$. Consequently, significant improvements to our approximation ratio may not be possible.

Consider an instance on three vertices $1, 2, 3$. Edges $(1, 2)$ and $(2, 3)$ are positive, but $(1, 3)$ is negative. The SDP solution consists of the vectors $v_1 = (1, 0), v_2 = (1/\sqrt{2}, 1/\sqrt{2}), v_3 = (0, 1)$, with objective value $1 + 2/\sqrt{2} = 1 + \sqrt{2}$. On the other hand, $\mathsf{OPT} = 2$, so the integrality gap is $2/(1 + \sqrt{2}) \approx 0.82843$.

An alternative approach might be be to use the rounding scheme used by Frieze and Jerrum [6] for MAX-k-CUT. The basic idea is to pick k random unit vectors (*spokes*) and assign each vector to the closest *spoke*. The analysis of such a scheme is quite involved and the gap example above suggests that pursuing this direction is unlikely to yield significant improvements.

2.3. Almost satisfiable instances

Consider an instance for which the optimal SDP solution is $(1-\varepsilon)W$, where W is the total weight of all the edges. We show that in this case, it is possible to obtain a clustering with agreement $(1 - O(\sqrt{\varepsilon}\log(1/\varepsilon)))W$. Recall that the strong performance of our algorithm suggests the difficulty in proving super-constant inapproximability for MINDISAGREE.

Theorem 4 *The expected agreement as a result of rounding with* $H_{\log(1/\varepsilon)}$ *is* $W(1 - O(\sqrt{\varepsilon}\log(1/\varepsilon)))$.

Proof: It is convenient to define various parameters. Let P denote the total weight of the positive edges and N the total weight of the negative edges. We define ρ and ν as follows:

$$\begin{aligned} \rho &= \frac{\sum_{+(ij)} w_{ij}(1 - v_i \cdot v_j)}{P} \\ \nu &= \frac{\sum_{-(ij)} w_{ij}(v_i \cdot v_j)}{N}. \end{aligned}$$

Since $\mathsf{OPT}_{\mathsf{SDP}} = (1-\varepsilon)W$, we observe that $\varepsilon \cdot W = \rho \cdot P + \nu \cdot N$.

Lemma 1 $P\sqrt{\rho} \le W\sqrt{\varepsilon}$.

Proof: It is trivially true if $\rho \le \varepsilon$. Otherwise, by definition $P\rho \le W\varepsilon$, so $P\sqrt{\rho} \le W\varepsilon/\sqrt{\rho} < W\sqrt{\varepsilon}$. ■

We prove that the rounding scheme H_t with $t = \log(1/\varepsilon)$ satisfies the following two lemmas and thus are done. ■

Lemma 2 *The expected contribution from the positive edges is at least* $P - O(\sqrt{\varepsilon}\log(1/\varepsilon))W$.

Proof: Define ε_{ij} to be $1 - v_i \cdot v_j$, so the expected weight of positive edges that are *not* cut in the solution is

$$\sum_{+(ij)} w_{ij} \left[1 - \cos^{-1}(1-\varepsilon_{ij})/\pi)\right]^t .$$

The function $(1-\cos^{-1}(x)/\pi)^t$ is convex, so by applying Jensen's inequality, we obtain the lower bound

$$P\left[1 - \cos^{-1}(1-\rho)/\pi\right]^t .$$

As $\cos^{-1}(1-\rho)$ is $O(\sqrt{\rho})$, the contribution of the positive edges, is at least $P(1 - O(\sqrt{\rho}))^t$, which by Lemma 1 is greater than or equal to $P - O(\sqrt{\varepsilon}\log(1/\varepsilon))W$. ■

Lemma 3 *The expected contribution from the negative edges is at least* $N(1-\varepsilon-\nu)$.

Proof: Now redefine ε_{ij} to be $v_i \cdot v_j$. The expected weight of negative edges that *are* cut in the solution is

$$\sum_{-(ij)} w_{ij} \left(1 - \left[1 - \cos^{-1}(\varepsilon_{ij})/\pi\right]^t\right) .$$

Again, convexity tells us that

$$\left[1 - \cos^{-1}(\varepsilon_{ij})/\pi)\right]^t$$

is no greater than

$$\varepsilon_{ij}\left(1 - \cos^{-1}(1)/\pi\right)^t + (1-\varepsilon_{ij})\left(1 - \cos^{-1}(0)/\pi\right)^t .$$

This is bounded above by $\varepsilon_{ij} + 1/2^t$. Since $N\nu = \sum_{-(ij)} w_{ij}\varepsilon_{ij}$, the expected contribution of the negative edges is at least $N(1-\nu-\varepsilon)$, for $t = \log(1/\varepsilon)$. ■

3. MinDisAgree in the complete graph

We present a factor four algorithm for minimizing disagreements in the complete graph. In contrast to Bansal *et al.* [1], who devised a combinatorial algorithm with factor 17433, our algorithm uses a linear programming formulation of the problem.

$$\begin{array}{lll} \text{minimize} & \sum_{+(ij)} x_{ij} + \sum_{-(ij)} (1 - x_{ij}) & \\ \text{such that} & x_{ik} \le x_{ij} + x_{jk} & \text{for all } i,j,k \\ & 0 \le x_{ij} \le 1 & \text{for all } i,j \end{array}$$

Figure 3. Relaxed linear program for minimizing disagreements

3.1. The four approximation

Our approach bears some similarity to the algorithm for MinDisAgree in general graphs that we presented in Section 2.1. Figure 3 shows the linear relaxation of the program for the complete graph. Having solved this linear program, in polynomial time, we are ready for our factor four approximation algorithm.

We refer to x_{ij} not only as the *distance* between i and j, but also as the *length* of edge ij. Algorithm 2, presented below, clearly describes a partitioning. We analyze its performance by comparing the number of mistakes incurred with the LP costs of appropriate edges.

Algorithm 2

1. Let $S = V$ and repeat the following steps until S is empty.
2. Select a vertex u arbitrarily from S.
3. Let T be the set of vertices at distance no greater than $1/2$ from u, except u itself: $B_x(u, 1/2) - \{u\}$.
4. If the average distance of the vertices in T from u is not less than $1/4$, then make $C = \{u\}$ a singleton cluster and jump to step 6.
5. If the average distance is less than $1/4$, then make $C = \{u\} \cup T$ a cluster.
6. Let $S = S - C$ and jump to step 2 (the start of the loop).

At each iteration of the loop, we relabel the vertices (other than u) so that $i < j$ if $x_{ui} < x_{uj}$, breaking ties arbitrarily. The triangle inequality tells us that for $i < j$,

$$x_{uj} \le x_{ui} + x_{ij} \quad \text{and} \quad x_{ij} \le x_{ui} + x_{uj}.$$

Observation 1 *The LP cost of the positive edge* ij*,* x_{ij}*, is at least* $x_{uj} - x_{ui}$*. The LP cost of a negative edge* ij*,* $1 - x_{ij}$*, is at least* $\max\{0, 1 - x_{ui} - x_{uj}\}$.

Associated with the new cluster, C, are the edges within C and the edges between C and $S - C$. We show that the mistakes in each iteration can be charged to the LP costs of the edges associated with C. Let us now consider one iteration at a time, starting with the case when a singleton cluster is formed.

Singleton cluster

The edges associated with a singleton cluster are simply all the edges adjacent to u: the positive ones are the mistakes.

We know from our choice in step 4 that

$$\sum_{i \in T} x_{ui} \geq |T|/4.$$

For $i \in T$, $1 - x_{ui} \geq x_{ui}$, so Observation 1 shows that the LP cost of *all* edges from u to T is at least $|T|/4$. The number of positive edge mistakes from u to T is thus at most $|T|$; this is at most four times the LP cost of edges from u to T.

The remaining edges connect vertices outside T to u. Each positive mistake from u has distance, and thus LP cost, greater than $1/2$; the number of mistakes is thus at most twice the LP cost of these edges.

Cluster with T

There are two kinds of mistakes in this case: negative edges inside C and positive edges between C and $S - C$.

(i) Negative edge mistakes

If both i and j are within distance $3/8$ of u, then the LP cost of negative edge ij is at least $1/4$, by Observation 1. This accounts for the mistake within factor 4.

Each remaining negative edge mistake ij will be charged to vertex j, the one that is further from u. So fix j and assume x_{uj} lies in the range $(3/8, 1/2]$. The total LP cost of edges within C associated with j is at least

$$\sum_{i:i<j,+(ij)} (x_{uj} - x_{ui}) + \sum_{i:i<j,-(ij)} (1 - x_{ui} - x_{uj}).$$

We let $x_{vv} = 0$ for all v so that this summation is well-defined. Denote by p_j the number of positive edges ij for which i is less than this fixed j. Similarly n_j stands for the number of such negative edges. The sum is then

$$p_j x_{uj} + n_j(1 - x_{uj}) - \sum_{i:i<j} x_{ui}.$$

Since we are including T in C, we know that $\sum_{i \in T} x_{ui} < |T|/4$. The set $\{i : i < j\} - \{u\}$ is a subset of T, but importantly the only terms missing have distance from u at least $3/8$. Therefore the average distance of the vertices in this set from u is still less than $1/4$ and including u certainly preserves this property. Hence the LP cost is greater than

$$p_j x_{uj} + n_j(1 - x_{uj}) - \frac{p_j + n_j}{4}. \qquad (1)$$

The number of mistakes associated with j is merely n_j. The LP cost is bounded below by a linear function (1) that lies between $p_j/8 + 3n_j/8$, when $x_{uj} = 3/8$, and $p_j/4 + n_j/4$, when $x_{uj} = 1/2$. Hence the LP cost is at least $n_j/4$ and thus all the mistakes are accounted for within factor four. Since this property holds for every j in the range $(3/8, 1/2]$, we conclude that the total number of negative edge mistakes is accounted for by appropriate LP edge costs within factor four.

(ii) Positive edge mistakes

Consider positive edges ij that cross the distance $1/2$ boundary. That is $x_{ui} \leq 1/2$, but $x_{uj} > 1/2$ and $+(ij)$. In particular, if $x_{uj} \geq 3/4$, then $x_{uj} - x_{ui} \geq 1/4$ and so each such positive edge pays for itself within factor four.

Again, we associate each remaining mistake edge with the vertex that is further from u. So fix j and assume that x_{uj} is in the range $(1/2, 3/4)$. The LP cost associated with j is

$$p_j x_{uj} + n_j(1 - x_{uj}) - \sum_{i \in T \cup \{u\}} x_{ui},$$

which is strictly greater than (1). This time, the linear function lower bound ranges between $p_j/4 + n_j/4$, when $j = 1/2$, and $p_j/2$, when $j = 3/4$. The number of (positive) mistakes is p_j so again we can pay for these within factor 4 using the LP cost. This argument holds for all j and thus for all positive edge mistakes.

Summary

Each choice of cluster leads to a ratio of at most four between mistakes and linear programming cost of associated edges. Since in past iterations we never charged to edges within S, and in future iterations we charge only to edges within $S - C$, we have a factor four approximation algorithm.

Theorem 5 *Algorithm 2 achieves a* 4 *approximation for* MINDISAGREE *on complete graphs.*

3.2. Approximation limitations

Integrality gap

Any approximation technique that is based on the linear program in Figure 3 is limited by its integrality gap. The following *star* example shows this gap is at least two. Place n vertices around a single center vertex so that the center is joined to the others with positive edges, but the perimeter vertices have negative edges between them. In an optimum fractional solution the positive edges have length $1/2$, the negative edges have length 1, so $\mathsf{OPT}_{\mathsf{LP}} = n/2$. An optimal clustering places all the perimeter vertices in singleton clusters, except for one, which is in a cluster with the center, so $\mathsf{OPT} = n - 1$. The gap, $2(n - 1)/n$, has limit 2 as n increases.

Limitations of region growing

The approximation technique we used, based on GVY region growing, cannot achieve a factor better than three. Our algorithm cuts a cluster C out of the set S, where C is chosen according to the distance relation x. We allowed ourselves two options for C: the singleton set $\{u\}$ or $B_x(u, 1/2)$. If we restrict ourselves to clusters of the form $B_x(u, r)$, or $\{u\}$, then we are confounded by the following star example.

Admittedly, this example is not an optimal fractional solution to the linear program, but it is consistent with Observation 1, on which our technique is based. The positive and negative labels are identical to the previous star, but now every edge has fractional length $1/3$. If our cluster radius is less than $1/3$ then we have a singleton cluster $\{u\}$, in which case the gap ratio is 3. Alternatively, if the radius is at least $1/3$ then all the vertices are in one cluster and the number of mistakes is $n(n-1)/2$. Since the LP cost is $n(n-1)/6+n/3$, the gap is $3(n-1)/(n+1)$, which tends to 3 as n increases. Therefore, no radius based approximation algorithm can beat a factor of three.

Using fixed radii

Our factor four algorithm chose between a singleton and a fixed radius of $1/2$. A more general algorithm would select the cluster radius based on the values of the x distance relation. We saw that even if this option were available, we could not achieve an approximation factor better than three. We now show that in some sense our algorithm is the best possible if the radius candidates—call them thresholds—are specified *in advance*.

Theorem 6 *Given a set of thresholds, of which k are greater than $1/4$, then our analysis techniques, which rely on Observation 1, cannot show an approximation ratio better than $3 + 1/k$.*

Proof: This proof concerns the analysis of a single iteration of removing C from S, rather than a complete graph example.

Imagine that there are n^2 vertices at distance $D = k/(3k+1) - \varepsilon$ from u, and that for each threshold distance d_i in the range $(D, 1-D]$ there are n vertices at distance $d_i + \varepsilon$. The edges those between the D-vertices and each of the other groups are positive. There are also n vertices at distance $d_i - \varepsilon$ for each d_i greater than D; they have negative edges to the D-vertices. Finally, every edge between u and some other vertex is positive. We ignore all other edges as their costs are dominated by the edges adjacent to the D-vertices.

Simple calculations show that no matter what size clustering radius is used, the ratio between the number of edges cost and the total LP cost approaches $3+1/k$, as n increases. ■

$$
\begin{array}{lll}
\text{minimize} & \sum_{+(ij)} x_{ij} + \sum_{-(ij)} (1 - x_{ij}) & \\
\text{such that} & \sum_{j=1}^{m-1} x_{i_j, i_{j+1}} - x_{i_m, i_1} \geq 0 & \\
 & \qquad\qquad \text{for all } C(i_1, \ldots, i_m) & \\
 & x_{ij} \leq 1 \qquad\qquad \text{for all } -(i,j) & \\
 & x_{ij} \geq 0 \qquad\qquad\quad \text{for all } i, j &
\end{array}
$$

Figure 4. NEPPC inequality linear program for minimizing disagreements

Note then that our factor four algorithm, which has one threshold greater than $1/4$, is the best we could hope for with these techniques and just one threshold.

3.3. The connection to feedback edge sets

Using an alternative linear programming formulation, we demonstrate the link between MinDisAgree in complete graphs and a feedback edge set problem.

Polygon inequalities are generalizations of triangle inequalities: the length of one edge in a polygon is at most the sum of the lengths of the other edges in the polygon. A *full* set of polygon inequalities is equivalent to a full set of triangle inequalities. Our new formulation, however, contains only one type of polygon inequality: the length of a negative edge is at most the sum of the lengths of edges in a *positive path* connecting its endpoints. More precisely, for all $i_1, i_2, \ldots, i_m$ such that $+(i_1, i_2), \ldots, +(i_{m-1}, i_m)$, but $-(i_1, i_m)$,

$$\sum_{j=1}^{m-1} x_{i_j, i_{j+1}} - x_{i_1, i_m} \geq 0.$$

We call this type of polygon a *negative edge with positive path cycle* (NEPPC), and denote it by $C(i_1, \ldots, i_m)$. In [4] this is called an erroneous cycle.

Theorem 7 *The linear program with only NEPPC polygon constraints, in Figure 4, is equivalent to the triangle inequality program in Figure 3, in the sense that their sets of optimal solutions are the same.*

Proof: **(Sketch)** The following simple observation is the key to the proof.

Observation 2 *In an optimal solution, a positive edge either has length zero, or it is part of some tight NEPPC constraint. Likewise, an optimal negative edge either has length one or is part of some tight NEPPC constraint.*

Lemma 4 *In any cycle of positive edges the polygon inequalities apply.*

$$
\begin{array}{lll}
\text{minimize} & \sum_{+(ij)} x_{ij} + \sum_{-(ij)} x'_{ij} & \\
\text{such that} & \sum_{j=1}^{m-1} x_{i_j,i_{j+1}} + x_{i_m,i_1} \geq 1 & \\
& \qquad\qquad \text{for all } C(i_1,\ldots,i_m) & \\
& x_{ij} \geq 0 \qquad \text{for all } +(ij) & \\
& x'_{ij} \geq 0 \qquad \text{for all } -(ij) &
\end{array}
$$

Figure 5. Positive coefficient NEPPC program for minimizing disagreements

Proof: The principle is that if a positive path p is shorter than the positive edge e joining its endpoints, then the tight NEPPC that includes e, will break a polygon inequality when e is replaced with p. ■

Lemma 4 and the principle of its proof can be used to show that at optimality the NEPPC constraints ensure that all triangle constraints are satisfied. The linear program in Figure 4 is a relaxation of the original in Figure 3. Therefore the two formulations have the same set of optimal solutions. [Theorem 7] ■

One can also prove an integral equivalent to Theorem 7: any optimal $\{0,1\}$ solution to the NEPPC constraint LP is an optimal solution to the MINDISAGREE problem, in a complete graph.

If we replace each $(1-x_{ij})$ term with x'_{ij} for each negative edge, we obtain the LP in Figure 5. The constraints $x'_{ij} \leq 1$ are unnecessary as each term has a positive coefficient. Since the sum of the terms around any NEPPC is at least 1, restricting the variables to $\{0,1\}$ leads to an interesting situation. Around any cycle that contains *exactly* one negative edge we must *select* at least one edge. That is, we need a feedback edge set for the set of cycles with exactly one negative edge. If the cycles of interest were those with *at least* one negative edge, we would already have a factor two approximation algorithm [5]. Perhaps this feedback edge set interpretation might lead to an algorithm with approximation ratio better than four.

4. Hardness of approximation

4.1. MINDISAGREE in general graphs

We first show that minimum multicut reduces in an approximation preserving way to MINDISAGREE. Note that Bansal *et al.* [1] make a similar observation, though they use the all-pairs version of multicut, usually called multiway cut, for the reduction. Reducing from the more general multicut problem, as other groups have also done independently [3, 4], provides us with evidence of the difficulty of approximating MINDISAGREE within any constant factor. In contrast, multiway cut has approximation algorithms with performance ratio a very small constant, 1.3438 being the current best [2, 11].

Theorem 8 *Minimum multicut reduces in an approximation preserving way to* MINDISAGREE.

Proof: Given a graph G with k pairs (s_i, t_i) which have to be separated, form an instance of Correlation Clustering H. The positive edges are the edges of G with unit weight, and for each i, $1 \leq i \leq k$, we add a (negative) edge between s_i and t_i with weight $-W$ for some large positive integer W, say $W = n^2$. If (s_i, t_i) happens to be an edge in G, then it will be a (negative) edge of weight $-(W-1)$ in H. We can make the instance unweighted by replacing a negative edge of weight $-M$ by M parallel length two paths. Each path has a fresh intermediate vertex, with one edge of weight 1 and the other of weight -1. Clearly, the minimum cost clustering must have s_i and t_i in different clusters for every i. The cost of the solution is simply the number of positive edges that lie between clusters, which is the same as the cost of the multicut. ■

Since minimum multicut is known to be APX-hard [8], we conclude that MINDISAGREE is also APX-hard. Moreover, an improvement over the $O(\log n)$ approximation ratio, which we achieved in Section 2.1, would solve one of the major open problems in the area of approximation algorithms: Can minimum multicut be approximately solved within a factor better than $O(\log n)$?

We also note the following fact concerning the perceived difficulty of multicut which does not seem to have been explicitly pointed out in the literature. It is well known that Min-2CNF-Deletion reduces to minimum multicut in an approximation preserving way; in fact the factor $O(\log n)$ approximation for Min-2CNF-deletion works by reducing it to a multicut instance and then running the GVY algorithm on the multicut instance. Recently, Khot [12] proved the NP-hardness of approximating Min-2CNF-deletion within any constant factor based on a conjecture about certain Unique games. Therefore, under the same conjecture, it is NP-hard to approximate minimum multicut, and therefore also MINDISAGREE within any constant factor.

In the next section, we study the maximization version. As a corollary of our hardness result for MAXAGREE, we will also record an explicit constant factor hardness for MINDISAGREE (Theorem 10).

4.2. MAXAGREE in general graphs

Bansal *et al.* [1] provided evidence for the APX-hardness of MAXAGREE by showing that a PTAS for MAXAGREE would lead to a polynomial time algorithm for $O(n^{\varepsilon})$ coloring a 3-colorable graph for every $\varepsilon > 0$. However, a concrete NP-hardness result for approximating MAXAGREE remained open and is resolved here.

Theorem 9 *For every $\varepsilon > 0$, it is NP-hard to approximate the* weighted *version of* MAXAGREE *within a factor of* $79/80 + \varepsilon$. *Furthermore, it is NP-hard to approximate the* unweighted *version of* MAXAGREE *within a factor of* $115/116 + \varepsilon$.

Proof: We reduce from MAX 3SAT, which is NP-hard to approximate within a factor of $(7/8 + \varepsilon)$ even on satisfiable instances [10]. Let ϕ be an instance of MAX 3SAT with variables $x_1, x_2, \ldots, x_n$ and clauses $C_1, C_2, \ldots, C_m$. We also assume that for each i, x_i and $\bar{x}_i$ each appear in the same number of clauses. This is a minor restriction and the inapproximability result for MAX 3SAT stands.

Construct a graph G with integer edge weights from the instance ϕ as follows. The vertices of G are a *root* vertex r, *variable* vertices $x_i, \bar{x}_i$ for $1 \le i \le n$, and *clause* vertices c_{1j}, c_{2j}, c_{3j} for each clause C_j, $1 \le j \le m$. The edges and their weights are defined as follows:

- The root r is connected to each c_{pj}, $p \in \{1, 2, 3\}$, by a weight 1 edge, and is connected to x_i and $\bar{x}_i$ by a weight B_i edge, where B_i is the number of clauses in which x_i (and $\bar{x}_i$) appears.
- A weight $-B_i$ edge connects x_i and $\bar{x}_i$ for each $i = 1, 2, \ldots, n$.
- The vertices c_{1j}, c_{2j}, c_{3j} corresponding to each clause form a triangle with weight -1 edges.
- Finally, if the p'th variable in clause C_j is x_i, for $p = 1, 2, 3$ (assuming some fixed ordering of variables in each clause), then a weight -1 edge connects c_{pj} with x_i.

We now prove that the optimum value of G as an instance of MAXAGREE is $9m + \mathsf{OPT}_\phi$ where OPT_ϕ is the maximum number of clauses of ϕ that can be simultaneously satisfied.

To that end, we show that any clustering can be modified to a specific format, still maximizing the number of agreements. Since the only positive edges incident to x_i and $\bar{x}_i$ are the ones joining them to r, each of x_i and $\bar{x}_i$ can be assumed to be either a singleton cluster or part of the cluster containing r. If both x_i and $\bar{x}_i$ are in the cluster with r, then we can make one of them, say x_i, a singleton and the number of agreements will not decrease, since we will lose B_i for the edge (r, x_i), but will gain B_i for the edge $(x_i, \bar{x}_i)$. Similarly, if both x_i and $\bar{x}_i$ are singletons, we can place x_i in the cluster containing r — we will gain a value of B_i for the edge (r, x_i) and might lose at most a value of B_i for the edges connecting x_i to the c_{pj}'s for each clause C_j in which x_i occurs.

Once in this format, a clustering corresponds to a truth assignment to the variables of ϕ in a natural way: variable x_i is true if it is in a singleton cluster, and false otherwise. Now for each clause C_j, we can cluster the vertices c_{pj}, $p = 1, 2, 3$, in the following way without decreasing the number of agreements. If C_j is not satisfied by the above assignment, which means all its literals are in the cluster containing r, we place each c_{pj} in a singleton cluster for $p = 1, 2, 3$. If C_j is satisfied, say because its first literal is set true, then we place c_{1j} in the cluster containing r, and c_{2j} and c_{3j} in singleton clusters. This way, we correctly classify all the negative weight edges incident upon the c_{pj}'s, and we correctly classify one of the three edges (r, c_{pj}) for $p = 1, 2, 3$ for each satisfied clause C_j.

It is easily seen that the total weight of correctly clustered edges equals

$$\Big(\sum_{i=1}^{n} 2B_i\Big) + 6m + m^* = 9m + m^*,$$

where m^* is the number of clauses satisfied by the above assignment. Therefore the optimum value of this instance of MAXAGREE is $9m + \mathsf{OPT}_\phi$. The claimed result follows since distinguishing between the cases $\mathsf{OPT}_\phi = m$ and $\mathsf{OPT}_\phi \le (7/8 + \varepsilon)m$ is NP-hard [10].

In order to obtain a result for unweighted (± 1) graphs, we replace each positive (resp. negative) edge of weight B_i (resp. $-B_i$) by B_i length two paths whose edges have weights $1, 1$ (resp. $1, -1$), as in the proof of Theorem 8. Now, if a weight B_i (positive or negative) edge is correctly clustered, then all the $2B_i$ newly constructed edges agree with the labeling; otherwise we get only B_i agreements. Using this gadget, we conclude that there is a $115/116 + \varepsilon$ inapproximability factor for the unweighted version of MAXAGREE; we omit the straightforward calculations. ■

Since the number of disagreements in an optimum clustering is simply the total weight of edges minus the number of agreements, the above reduction also establishes the following.

Theorem 10 *For every $\varepsilon > 0$, it is NP-hard to approximate both the* weighted *and* unweighted *versions* MINDISAGREE *within a factor of* $29/28 - \varepsilon$.

4.3. MINDISAGREE in complete graphs

In addition to their constant factor approximation algorithm, the authors of [1] also proved the NP-completeness of MINDISAGREE on *complete* graphs. Their reduction does not yield any hardness of approximation result, but they do show that the maximization version admits a PTAS on complete graphs.

Theorem 11 nicely completes the picture with respect to the complexity of the problem on complete graphs, and also complements our factor four approximation algorithm.

Theorem 11 *For some constant $a > 1$, it is NP-hard to approximate* MINDISAGREE *on* complete graphs *within a factor of a.*

Proof: **(Sketch)** We show a reduction from "MAX 2-colorable subgraph" on bounded-degree 3-uniform hypergraphs. In this problem the input is a 3-uniform hypergraph $H = (V, S)$ in which each hyperedge in $S = \{e_1, e_2, \ldots, e_m\}$ consists of three elements of $V = \{v_1, v_2, \ldots, v_n\}$ and each vertex is in at most B hyperedges. The objective is to find a 2-coloring that maximizes the number of hyperedges that are bichromatic. It is NP-hard to distinguish whether a 2-coloring exists with no monochromatic hyperedges, or at least a fraction γ of S is monochromatic [9, 10].

The first step is to construct a graph G (whose elements we call *nodes*, rather than vertices) from hypergraph H. For each $v_i \in V$ we build a *flower* F_i with $4s_i$ nodes, where $s_i \leq B$ is the number of hyperedges that contain v_i. There is an (internal) cycle of $2s_i$ nodes, together with $2s_i$ *petal* nodes. Each petal has an edge to each of two consecutive cycle nodes (see Section 9.4 of [13] for a similar reduction). Label the petal nodes of F_i in order, with arbitrary starting point, and refer to the odd ones as O_i and the even as E_i. For each hyperedge e_j, create two edges α_j and β_j in G. For each vertex i in e_j, create edges from both endpoints of α_j to the kth petal in O_i, where hyperedge e_j is the kth occurrence ($k \leq s_i$) of vertex i. Likewise, connect the endpoints of β_j to the even petals.

The total number of nodes in G is $N = \sum_{i=1}^{n} 4s_i + 4m = 16m$. Since G is 4-regular, the number of edges M equals $2N$. We construct a complete graph K_N with positive labels on the edges of G and negative labels on the other edges. Given a clustering, let the *value* of a single cluster be the number of positive edges minus the number of negative edges. The total value of a clustering is merely the sum of the cluster values. The value of a node is the value of the cluster it is in, divided by the number of nodes in that cluster. For example, if a cluster is a singleton, its value is zero. The nodes in an *edge* cluster have value $1/2$; those in a triangle have value 1. A diamond cluster is K_4 with exactly one negative edge (that is, one edge is missing in G) and its nodes have value 1.

We can show that if H is 2-colorable—there is a coloring with no monochromatic hyperedges—then there exists a clustering of value N.

The complementary result is that if every 2-coloring of H has γm monochromatic hyperedges, then every clustering has value at most $(1 - \delta)N$, for some $\delta > 0$. To prove this, we show that the value of every node equals 1 if and only if it lies in a triangle or diamond cluster. Otherwise the value is at most $1 - \rho$ for $\rho > 0$. The argument continues by showing that there exists a coloring for which every monochromatic hyperedge has at least one vertex whose flower is not of full value—that is, some flower nodes have value less than 1. It can be shown that with γm monochromatic edges the total cluster value is at most $(1-\xi)N$, where $\xi = 3\rho\gamma/(16B^2) > 0$. Since the number of disagreements in a clustering is $M = 2N$ minus its value, the result is an N versus $(1 + \xi)N$ gap for MINDISAGREE in complete graphs. ■

References

[1] N. Bansal, A. Blum, and S. Chawla. Correlation clustering. In *Proc. of 43rd FOCS*, pages 238–47, 2002.

[2] G. Calinescu, H. Karloff, and Y. Rabani. An improved approximation algorithm for multiway cut. *JCSS*, 60:564–74, 2000.

[3] E. Demaine and N. Immorlica. Correlation clustering with partial information. In *Proc. of 6th APPROX*, 2003. To appear.

[4] D. Emanuel and A. Fiat. Correlation clustering—minimizing disagreements on arbitrary weighted graphs. In *Proc. of 11th ESA*, 2003. To appear.

[5] G. Even, J. Naor, B. Schieber, and L. Zosin. Approximating minimum subset feedback sets in undirected graphs with applications. *SIAM J Disc. Math.*, 25:255–67, 2000.

[6] A. Frieze and M. Jerrum. Improved approximation algorithms for MAX k-CUT and MAX BISECTION. In E. Balas and J. Clausen, editors, *Proc. of 4th IPCO*, volume 920 of *LNCS*, pages 1–13. Springer, 1995.

[7] N. Garg, V. Vazirani, and M. Yannakakis. Approximate max-flow min-(multi)cut theorems and their applications. *SIAM J Comp.*, 25:235–51, 1996.

[8] N. Garg, V. Vazirani, and M. Yannakakis. Primal-dual approximation algorithms for integral flow and multicut in trees. *Algorithmica*, 18:3–20, 1997.

[9] V. Guruswami. Inapproximability results for set splitting and satisfiability with no mixed clauses. In K. Jansen and S. Khuller, editors, *Proc. of 3rd APPROX*, volume 1913 of *LNCS*, pages 155–66. Springer, 2000.

[10] J. Håstad. Some optimal inapproximability results. *JACM*, 48:798–859, 2001.

[11] D. Karger, P. Klein, C. Stein, M. Thorup, and N. Young. Rounding algorithms for a geometric embedding of minimum multiway cut. In *Proc. of 31st STOC*, pages 668–78, 1999.

[12] S. Khot. On the power of unique 2-prover 1-round games. In *Proc. of 34th STOC*, pages 767–75, 2002.

[13] C. Papadimitriou. *Computational Complexity*. Addison Wesley Longman, 1994.