ELSEVIER

Available online at www.sciencedirect.com

SCIENCE DIRECT®

Journal of Computer and System Sciences 71 (2005) 360–383

JOURNAL OF COMPUTER AND SYSTEM SCIENCES

www.elsevier.com/locate/jcss

Clustering with qualitative information☆

Moses Charikar[a,*,1], Venkatesan Guruswami[b,2], Anthony Wirth[a,3]

[a] *Department of Computer Science, Princeton University, 35 Olden St., Princeton, NJ 08544-2087, USA*
[b] *University of Washington, USA,*

Received 2 April 2004; received in revised form 3 October 2004

Available online 19 December 2004

Abstract

We consider the problem of clustering a collection of elements based on pairwise judgments of similarity and dissimilarity. Bansal et al. (in: Proceedings of 43rd FOCS, 2002, pp. 238–247) 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). It is worthwhile studying both complete graphs, in which every edge is labeled, and general graphs, in which some input edges might not have labels. We answer several questions left open by Bansal et al. (2002) and provide a sound overview of clustering with qualitative information.

Specifically, we demonstrate 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 was shown by Bansal et al. (2002), 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.

Keywords: Clustering; Approximation algorithm; LP rounding; Minimum multicut

☆ A preliminary version of this paper appeared in the *Proceedings of the* 44*th Annual IEEE Symposium on Foundations of Computer Science* (*FOCS*), October 2003.

* Corresponding author. Fax: +1 609 258 1771.

E-mail addresses: moses@cs.princeton.edu (M. Charikar), venkat@cs.washington.edu (V. Guruswami), awirth@cs.princeton.edu (A. Wirth).

[1] 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.

[2] Supported in part by NSF CAREER award CCF-0343672.

[3] Supported by a Gordon Wu Fellowship and NSF ITR Grant CCR-0205594.

0022-0000/$ - see front matter © 2004 Published by Elsevier Inc.
doi:10.1016/j.jcss.2004.10.012

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 the k-center, k-median, and k-sum clustering problems.

This clustering paper departs from the above distance paradigm. All we have at our disposal is *qualitative information* from a judge: a labeling of each pair of elements as either similar or dissimilar. We are not provided with any quantitative distance information about the pairs. 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 maximum extent possible. If there exists a clustering that is *correct* for every edge, then the problem is trivially solved by identifying as clusters the connected components in the graph of similar pairs (see below). When the judge has made mistakes, interesting and non-trivial questions arise: primarily, finding a clustering that differs from the judge's verdicts on the fewest possible pairs. Bansal et al. pointed out that correlation clustering corresponds to agnostic learning [16], when viewed as a machine learning problem. The edge labels are the examples and we are only allowed to use partitionings as hypotheses for the target function.

An 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 by 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 by 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 have only a single objective; whether the optimal solution uses few or many clusters is automatically dictated by the edge labels.

If every pair of elements is labeled either + or −, then G will be a complete graph. So that we can capture situations where the judge might be unable to tell if certain pairs of elements are similar or dissimilar, we do not insist on the input being a complete graph. One upshot of the clustering will be to deduce the missing labels from the existing ones. Also, in some instances the judge might provide confidence information for each of the labels. This is captured by assigning *weights* to the edges; one can then consider natural weighted versions of MaxAgree and MinDisAgree.

1.1. Previous and related work

The above problem on complete graphs seems to have been first considered by Ben-Dor et al. [3] motivated by some computational biology questions. Later, Shamir et al. [20] studied the computational complexity of the problem and showed that MaxAgree (and hence also MinDisAgree) is NP-hard for complete graphs. Shamir et al. used the term *Cluster editing* to refer to this problem; recent algorithms

for fixed parameter versions are presented by Gramm et al. [12]. Independently, Chen et al. [4] examined a very similar problem in the context of phylogeny trees, essentially showing that MinDisAgree is NP-hard.

As mentioned earlier, Bansal et al. [1] considered this problem independently. They initiated the study of approximate solutions to MinDisAgree and MaxAgree, focusing mainly on the case when G is complete. Bansal et al. gave a polynomial time approximation scheme (PTAS) for MaxAgree on complete graphs. For the minimization version MinDisAgree, they gave an approximation algorithm with constant performance ratio. 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. In the full version of their work [2], Bansal et al. provide a simple algorithm that is at most a factor three worse than the best partitioning into *two* clusters. They posed several open questions including those of demonstrating hardness of approximation results for complete graphs and understanding the problem on general graphs. These questions motivated a number of groups, such as ours, to work on this problem simultaneously.

Both Demaine and Immorlica [6], and Emanuel and Fiat [7], independently from each other and from this paper, announced results on clustering with qualitative information. These two papers focus on MinDisAgree in general graphs. Demaine and Immorlica [6] 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 MinDisAgree in $K_{r,r}$-minor-free graphs. In [7], both reductions to and from minimum multicut are presented; in particular the authors show a reduction from unweighted multicut to unweighted MinDisAgree. For MaxAgree on general graphs, Swamy [21], again independently from this paper, presented a factor 0.7666 approximation algorithm (very slightly better than the factor we present here).

1.2. Our results

In this paper, we 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 of the combinatorial algorithm in [1]. Our algorithm is based on a natural linear programming relaxation; it rounds the fractional solution (a semi-metric on the vertices) using the *region growing* approach. The completeness of the graph allows us to achieve a constant approximation using region growing, instead of the usual logarithmic factor [10]. The integrality gap of our LP formulation is 2 and we also show that beating factor 3 would require significant departure from our strategy. To complement our algorithmic result, we also prove that MinDisAgree on complete graphs is APX-hard (that is, 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 MaxAgree does admit a PTAS on complete graphs [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 (GVY) region-growing

algorithm for minimum multicut [10]. 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. A complementary hardness result follows for MinDisAgree. On the algorithmic side, the naive $\frac{1}{2}$-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 there exists a clustering that correctly classifies most of the edges, 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 would have to focus on instances where an almost perfect clustering exists for both the *yes* and *no* cases of the gap reduction.

1.3. Organization

We present algorithms for general graphs (for both the minimization and maximization variants) 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 [10].

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

A partitioning into clusters can be represented with a set of binary variables, one for each pair of vertices. If i and j are in the same cluster then x_{ij} is 0, if they are in different clusters then x_{ij} is 1. 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,

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

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 (1) summarizes the situation: $+(ij)$ indicates that the edge between i and j has a positive label, while $-(ij)$ indicates a negative label.

We note in passing that solid lines indicate *positive* edges, whereas dashed lines indicate *negative* edges in the diagrams. The confidence that the judge places on the (dis)similarity label between $+i$ and j is represented by the weight w_{ij}. The LP relaxation is obtained by replacing the integer constraints in (1) with $0 \leqslant x_{ij} \leqslant 1$ for all i, j.

Let the value of the optimal LP solution be denoted by $\mathrm{OPT}_{\mathrm{LP}}$. A fairly straightforward application of the GVY region growing procedure yields a solution of cost at most $O(\log n)\mathrm{OPT}_{\mathrm{LP}}$. We briefly describe this algorithm, AlgGeneral, and outline its 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 whose distance from i is less than or equal to r. For a set of vertices S, let $\delta(S)$ be the set of edges between S and $\overline{S}$.

AlgGeneral

1. $\mathcal{C} \leftarrow \emptyset$. /* Collection of clusters */
2. While there exist i, j in the graph such that $x_{ij} > \frac{2}{3}$:
   (a) Let $S = B_x(i, r)$ for some $r < \frac{1}{3}$. /* 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. AlgGeneral *achieves an* $O(\log n)$ *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

$$\frac{\mathrm{OPT}_{\mathrm{LP}}}{n} + \sum_{+(uv)\in B_x(i,r)} w_{uv}x_{uv} + \sum_{+(uv)\in\delta(B_x(i,r))} w_{uv}(r - x_{iu}).$$

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 $\mathrm{OPT}_{\mathrm{LP}}/n$. Let $W_x^+(i, r)$ denote the sum of weights of positive edges in $\delta(B_x(i, r))$. We choose $r < \frac{1}{3}$ so that the ratio of $W_x^+(i, r)$ to $V_x^+(i, r)$ is minimized. The analysis technique in [10] can be used to show that there exists a radius $r < \frac{1}{3}$ such that $W_x^+(i, r) \leqslant (3 \log n)V_x^+(i, r)$. This and the triangle inequality imply that the total weight of positive edges with end points in different clusters is in $O(\log n)\mathrm{OPT}_{\mathrm{LP}}$.

Now we account for the negative edges. 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 $w_{ij}/3$, since $x_{ij} \leqslant \frac{2}{3}$. 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 $O(\log n)\mathrm{OPT}_{\mathrm{LP}}$. □

The $O(\log n)$ approximation ratio we obtain from our LP is asymptotically 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 [10].

We expect that a procedure such as this one, which *learns* distances from similarity judgment information, will have further applications in situations where no natural distance function exists.

2.2. MaxAgree

Since Bansal et al. [1] presented a PTAS for complete graphs, we need only look at general graphs for MaxAgree. Obtaining a $\frac{1}{2}$ 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.

2.2.1. 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* $\frac{2}{3} + \varepsilon$ *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 optimal LP solution assigns $x_{ij} = \frac{1}{2}$ for $+(ij)$ and $x_{ij} = 1$ for $-(ij)$, and so $\mathrm{OPT}_{\mathrm{LP}}$ is $n(n-1) + n^2/2$. On the other hand, the value of OPT for this instance is n^2: any instance with equal numbers of elements from A and B in each cluster suffices—we leave the proof to the reader. Hence the integrality gap is $2n/(3n-2)$, which approaches $\frac{2}{3}$ as n increases. □

2.2.2. Rounding a semidefinite program

We next consider a semidefinite program (SDP) for MaxAgree, as SDPs can be solved to arbitrary precision in polynomial time. To motivate the SDP, we associate a distinct basis vector with each cluster in a solution; for every vertex i in that cluster we set the unit vector v_i to be that basis vector. The agreement of the clustering solution can now be expressed in terms of the dot products $v_i \cdot v_j$. If vertices i and j are in the same cluster, then $v_i \cdot v_j = 1$, if not, $v_i \cdot v_j = 0$. With this vector solution in mind, we consider the SDP relaxation (2) for MaxAgree.

$$\begin{aligned} \text{maximize} \quad & \sum_{+(ij)} w_{ij}(v_i \cdot v_j) + \sum_{-(ij)} w_{ij}(1 - v_i \cdot v_j) \\ \text{subject to} \quad & v_i \cdot v_i = 1 \quad \text{for all } i, \\ & v_i \cdot v_j \geqslant 0 \quad \text{for all } i, j. \end{aligned} \tag{2}$$

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 $\mathrm{Best}(H_2, H_3)$.

Theorem 3. *Best* (H_2, H_3) *returns a solution in which the expected number of agreements is at least* $0.7664\,\mathrm{OPT}_{\mathrm{SDP}}$.

Proof. In order to analyze 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 Comb(H_2, H_3). Clearly, the approximation ratio of Comb(H_2, H_3) is a lower bound on the approximation ratio of 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 (non-negative) edge weights are common to both the integral formulation and its SDP relaxation 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 expected contribution to the number of agreements in 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 expected contribution to the number of agreements in 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

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

For $\alpha \leqslant 0.1316$, the minimum of the two expressions is $\frac{3}{4} + \alpha/8$. In fact the minimum value of the second expression is $\frac{3}{4} + \alpha/8$ 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 the SDP (2) is at most 0.828. Our example has three vertices, 1, 2, 3, in which edges (1, 2) and (2, 3) are positive, but (1, 3) is negative. The optimal 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 at most $2/(1+\sqrt{2}) \approx 0.828$.

Our SDP formulation does not, however, respect the triangle inequalities on the values $x_{ij} = 1 - v_i \cdot v_j$. Even with such constraints added, the example below shows that significant improvements to the approximation ratio may not be possible. Consider an instance on five vertices 0, 1, 2, 3, 4. Edges from 0 are positive, but all others are negative. With $v_0 = (0.5, 0.5, 0.5, 0.5)$, and v_i equal to the ith basis vector e_i, $\mathrm{OPT}_{\mathrm{SDP}} = 8$. However, $\mathsf{OPT} = 7$, with clusters {0, 1}, {2}, {3}, {4}, showing that we can rule out an SDP-based algorithm with approximation factor greater than least 7/8 that observes the triangle inequalities.

An alternative approach is to use the rounding scheme used by Frieze and Jerrum [9] 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. Swamy [21] recently carried out an analysis of such a rounding procedure and reported a factor 0.7666 approximation algorithm for MaxAgree.

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 expected agreement in $(1-O(\sqrt{\varepsilon}\log(1/\varepsilon)))W$. This strong result suggests there would be difficulty in proving super-constant inapproximability for MinDisAgree.

It is convenient at this point 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 v as follows:

$$\rho = \frac{\sum_{+(ij)} w_{ij}(1 - v_i \cdot v_j)}{P}, \quad v = \frac{\sum_{-(ij)} w_{ij}(v_i \cdot v_j)}{N}.$$

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

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

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

We prove that the rounding scheme H_t with $t = \log(1/\varepsilon)$ satisfies the following two lemmas and then conclude with the main result of this section.

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

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

Since $\cos^{-1}(1-\rho)$ is in $O(\sqrt{\rho})$, the contribution of the positive edges is at least

$$P(1 - O(\sqrt{\rho}))^t \geqslant P(1 - tO(\sqrt{\rho})) \geqslant P - O(\sqrt{\varepsilon}\log(1/\varepsilon))W,$$

by Lemma 1. □

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

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 $Nv = \sum_{-(ij)} w_{ij}\varepsilon_{ij}$, the expected contribution of the negative edges is at least $N(1 - v - \varepsilon)$, for $t = \log(1/\varepsilon)$. □

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

$$\begin{array}{ll} \text{minimize} & \displaystyle\sum_{+(ij)} x_{ij} + \sum_{-(ij)} (1 - x_{ij}) \\ \text{subject to} & x_{ik} \leqslant x_{ij} + x_{jk} \quad \text{for all } i, j, k, \\ & 0 \leqslant x_{ij} \leqslant 1 \qquad\quad \text{for all } i, j. \end{array} \tag{3}$$

Proof. Lemmas 2 and 3 show that the expected number of agreements resulting from the $H_{\log(1/\varepsilon)}$ rounding scheme is at least

$$(P + N) - O(\sqrt{\varepsilon}\log(1/\varepsilon))W - (\varepsilon + v)N.$$

We note that $(\varepsilon + v)N \leqslant 2\varepsilon W$ and that ε is in $O(\sqrt{\varepsilon}\log(1/\varepsilon))$ as $\varepsilon \to 0$. Therefore the expected number of agreements is at least $W(1 - O(\sqrt{\varepsilon}\log(1/\varepsilon)))$.

3. MinDisAgree on complete graphs

We now study the clustering problem on complete graphs. As already mentioned, Bansal et al. [1] present a PTAS for MaxAgree on complete graphs, hence we focus on MinDisAgree. 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.

3.1. The four approximation

Our approach bears some similarity to the algorithm for MinDisAgree in general graphs, AlgGeneral, that we presented in Section 2.1. Once the linear relaxation (3) of the program for the is solved, in polynomial time, we are ready for our factor four approximation algorithm.

ALGCOMPLETE

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 whose distance from u is no greater than $\frac{1}{2}$, except u itself: $B_x(u, \frac{1}{2}) - \{u\}$.
4. If the average distance of the vertices in T from u is not less than $\frac{1}{4}$, 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).

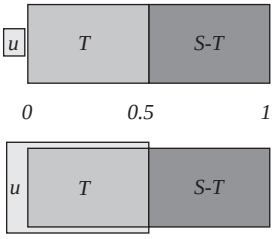

Fig. 1. Illustration of the two main choices in ALGCOMPLETE: numerical annotations are the *distances* from u.

We refer to x_{ij} not only as the *distance* between i and j, but also as the *length* of edge ij. The procedure we present, ALGCOMPLETE, illustrated also in Fig. 1, clearly describes a partitioning. We analyze its performance by comparing the number of mistakes incurred to the LP costs of appropriate edges.

Let us reflect on the natural intuition behind the algorithm. Intuitively, the LP solution x_{ui} gives a handle on how different u and i are: the smaller the value of x_{ui} the more incentive there is to place u and i in the same cluster. Therefore, it makes sense to cluster the points close to u (in a ball $B_x(u, r)$) in one cluster, say C, together with u. If both i and j are close to u, but are connected by a negative edge, we will cluster them together and make a mistake, but the LP cost of that edge $1 - x_{ij}$ will also be high since $x_{ij} \leqslant x_{iu} + x_{ju}$ must also be small. This basic strategy works well with negative edges. However, there is a problem if most of the vertices in C are near its *periphery*, that is, at distance close to r from u. In such a case, the LP might have very low cost x_{ij} for some $+(ij)$ crossing the cut, compared to the unit cost that the algorithm incurs on the same edge. A natural measure of whether this phenomenon could occur is the average distance from u of points in C. If this is large, then there could be many points on the periphery, and the above difficulty could occur, so we simply place u in its own cluster. It turns out, from the analysis that follows, that the best criterion for choosing between the ball cluster and a singleton cluster, is whether the average distance is greater or less than $\frac{1}{4}$.

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 $i < j$,

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

Observation 1. *The LP cost of a 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* $\max\{0, 1 - x_{ui} - x_{uj}\}$.

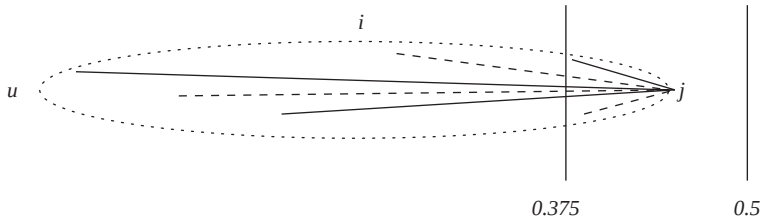

Fig. 2. Charging mistakes and LP costs to the further (fixed) vertex j.

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 of AlgComplete can be charged to the LP costs of the edges associated with the new cluster C. Let us now consider one iteration at a time, starting with the case when a singleton cluster is formed.

3.1.1. Singleton cluster

The edges associated with a singleton cluster are simply all the edges incident to u: the positive ones are the mistakes. We know from our choice in step 4 that

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

For $i \in T$, $1 - x_{ui} \geqslant x_{ui}$, so 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, which is at most $|T|$, is thus at most four times the LP cost of edges from u to T.

The remaining edges associated with this cluster are between u and $S - T$. Each positive mistake incident on u has distance, and thus LP cost, greater than $\frac{1}{2}$; so the number of mistakes is at most twice the LP cost of these edges.

3.1.2. Cluster with T

We now turn to the case in which $C = \{u\} \cup T$. 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 $\frac{3}{8}$ of u, then the LP cost of negative edge ij is at least $\frac{1}{4}$, by Observation 1. This accounts for the mistake within factor 4.

Each remaining negative edge mistake ij will be charged to vertex j, the vertex that is further from u (see Fig. 2).

So *fix* j and assume x_{uj} lies in the range $(\frac{3}{8}, \frac{1}{2}]$. Observation 1 tells us that the total LP cost of all the 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 $i < j$, and let n_j stand for the number of such negative edges. The total cost

is then

$$p_j x_{uj} + n_j(1 - x_{uj}) - \sum_{i:i<j} x_{ui}. \tag{4}$$

Since we are including T in C, we know that the average value of x_{ui} is less than $\frac{1}{4}$ for $i \in T$. The summation above is over the set $\{i : i < j\}$, but since $x_{ui} \geqslant \frac{3}{8}$ for $i > j$, the average value of the summation terms in (4) is less than $\frac{1}{4}$. Hence the LP cost is greater than

$$p_j x_{uj} + n_j(1 - x_{uj}) - \frac{p_j + n_j}{4}. \tag{5}$$

The number of mistakes associated with j is merely n_j. The LP cost is bounded below by a linear function (5) that ranges from $p_j/8 + 3n_j/8$, when $x_{uj} = \frac{3}{8}$, to $p_j/4 + n_j/4$, when $x_{uj} = \frac{1}{2}$. Therefore the LP cost is at least $n_j/4$ and all the (negative) mistakes are accounted for within factor four. Since this property holds for every j in the range $(\frac{3}{8}, \frac{1}{2}]$, 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 $\frac{1}{2}$ boundary: $x_{ui} \leqslant \frac{1}{2}$, but $x_{uj} > \frac{1}{2}$. In particular, if $x_{uj} \geqslant \frac{3}{4}$, then $x_{uj} - x_{ui} \geqslant \frac{1}{4}$ and so each such positive edge pays for itself within factor four.

Again, we associate each remaining edge with the vertex that is further from u. So fix j and assume that x_{uj} is in the range $(\frac{1}{2}, \frac{3}{4})$. The LP cost of the edges 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 (5). This time, the linear function lower bound ranges between $p_j/4 + n_j/4$, when $x_{uj} = \frac{1}{2}$, and $p_j/2$, when $x_{uj} = \frac{3}{4}$. The number of (positive) mistakes is p_j so again we can pay for these within factor 4 of the LP cost. This argument holds for all j and thus for all positive edge mistakes.

3.1.3. Summary

Each choice of cluster leads to a ratio of at most four between the number of mistakes and the 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. AlgComplete *achieves a factor* 4 *approximation for* MinDisAgree *on complete graphs.*

As we remarked earlier, if we assume that all positive edges are correct, the problem is trivial as it reduces to finding connected components. Shamir et al. [20] studied the *cluster deletion* problem, in which all *negative* edges are deemed to be correct and must be cut, and showed it to be APX-hard. In this case, the problem analogous to MinDisAgree is to find a clustering with the fewest possible positive edges crossing cluster boundaries. Our algorithm for MinDisAgree also achieves a 4 approximation in this variant. The idea is to add the constraints $x_{ij} = 1$ in the linear program for each $-(ij)$, and then run

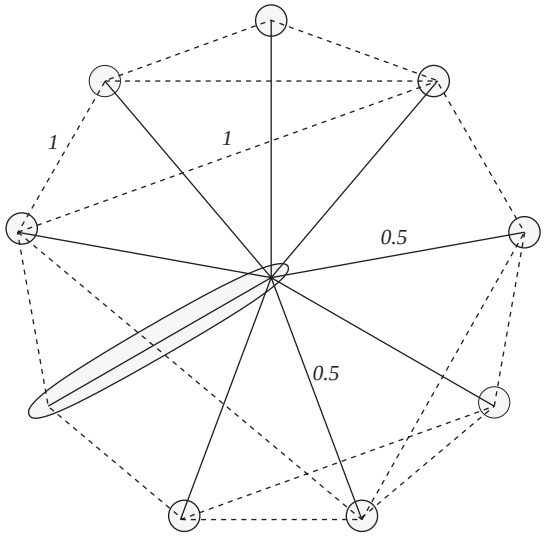

Fig. 3. MINDISAGREE instance with integrality gap almost 2, showing both the fractional optimum (with distances) and integral optimum (with clusters). Some edges have been omitted for clarity.

ALGCOMPLETE on the LP solution. We make the minor amendment, which does not affect the proof of Theorem 5 substantially, that T does not include the vertices whose distance from u is exactly $\frac{1}{2}$. Thus each cluster C has diameter less than 1 and the endpoints of a negative edge are never placed in the same cluster. The analysis for the number of mistakes on positive edges remains identical. With this variant, as with MINDISAGREE, it is an interesting question whether the factor 4 can be improved.

3.2. Approximation limitations

3.2.1. Integrality gap

Any approximation technique that is based on the linear program (3) is limited by its integrality gap. The following *star* example, in Fig. 3, 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 $\frac{1}{2}$ and the negative edges have length 1, so $\mathrm{OPT}_{\mathrm{LP}} = n/2$. 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.

3.2.2. 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 $B_x(u, \frac{1}{2})$. If we restrict ourselves to clusters of the form $B_x(u, r)$, or $\{u\}$, then we are confounded by the following star type example. Admittedly, this example is not an optimal fractional solution to the linear program, but it is a feasible solution and thus Observation 1, on which our technique is based, applies.

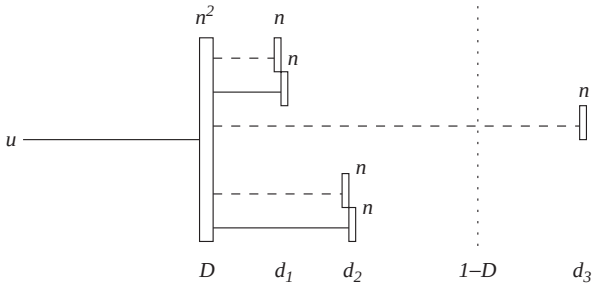

Fig. 4. Feasible solution example showing that with k thresholds our techniques cannot give an approximation ratio better than $3 + 1/k$. The instance is complete, but we have chosen not to show edges that have little impact on the calculations.

The positive and negative labels are identical to the previous star, but now every edge has fractional length $\frac{1}{3}$. If our cluster radius is less than $\frac{1}{3}$ then we have a singleton cluster $\{u\}$, in which case the gap ratio is 3. Alternatively, if the radius is at least $\frac{1}{3}$ 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.

3.2.3. Using fixed radii

Our factor four algorithm chose between a singleton cluster and a fixed cluster radius of $\frac{1}{2}$. A more general algorithm might 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—for cluster balls are specified *in advance*.

Theorem 6. *Given a set of thresholds, of which k are greater than* $\frac{1}{4}$*, then our analysis techniques, which rely only on the solution being feasible, cannot be used to show an approximation ratio better than* $3 + 1/k$.

Proof. Consider the analysis of the following feasible solution, shown in Fig. 4, to the MinDisAgree LP, which could occur in a single iteration of region growing.

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

For every threshold that lies in the range $(\frac{1}{4}, D)$, the number of mistakes is dominated by n^2 and the LP cost is dominated by Dn^2. Therefore the integrality gap is $1/D$, which tends to $\to 3 + 1/k$ as $\varepsilon \to 0$.

For every other threshold, the LP cost is dominated by the edges between the n^2 D-vertices and the vertices in the other sets. The LP cost of the edges to $d_i - \varepsilon$ and $d_i + \varepsilon$ could be as low as

$$n^3[(d_i + \varepsilon) - D] + n^3[1 - (d_i - \varepsilon) - D] = n^3[1 + 2\varepsilon - 2D] \\ \to n^3 \cdot \frac{k+1}{3k+1} \quad \text{as } \varepsilon \to 0.$$

The LP cost of the negative edges between the D-vertices and the $d_i - \varepsilon$-vertices, where $d_i > D$, could be zero. For each threshold between D and $1 - D$, of which there are $k' \leqslant k$, the number of mistakes is $(k' + 1)n^3$. Therefore the ratio of mistakes to LP cost could be as high as

$$\frac{k'+1}{k'} \cdot \frac{3k+1}{k+1},$$

which is $3 + 1/k$ when $k' = k$, and greater otherwise. The total LP cost associated with thresholds whose distance is greater than $1 - D$ may be no greater than before. Since the number of mistakes is *at least* $(k' + 1)n^3$, we cannot prove an approximation ratio any better than $3 + 1/k$. □

Note then that our factor four algorithm, which has one threshold greater than $\frac{1}{4}$, 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 on 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 all 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 $-(i_1, i_m)$,

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

$$\begin{aligned} \text{minimize} \quad & \sum_{+(ij)} x_{ij} + \sum_{-(ij)} (1 - x_{ij}) \\ \text{subject to} \quad & \sum_{j=1}^{m-1} x_{i_j, i_{j+1}} - x_{i_m, i_1} \geqslant 0 \quad \text{for all } C(i_1, \ldots, i_m), \\ & x_{ij} \leqslant 1 \quad \text{for all } -(ij), \\ & x_{ij} \geqslant 0 \quad \text{for all } i, j. \end{aligned} \tag{6}$$

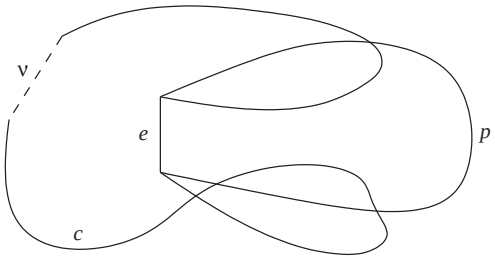

Fig. 5. Construction of a new NEPPC: Positive edge e is part of a tight NEPPC c, which has one negative edge v; edge e is also in a cycle with positive path p.

We call this type of polygon a *negative edge with positive path cycle* (NEPPC), and denote it by $C(i_1, \ldots, i_m)$. Elsewhere [7], NEPPCs have been called erroneous cycles.

We now show that the NEPPC constraints are a sufficiently large set that they imply all the triangle (inequality) constraints for *optimal* solutions to the linear program (6). The following simple observation, together with the consequent lemma, is the key.

Observation 2. *In an optimal solution to the linear program* (6)*, 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 an optimal solution to LP* (6)*, the polygon inequalities apply to every cycle of positive edges.*

Proof. Consider a positive path p that is incident to both endpoints of positive edge e, with $x_e > x_p$ in an optimal solution (abusing notation). Since the length of e cannot be zero, Observation 2 tells us that e lies in some tight NEPPC c. Assume for the moment that c does not share any vertices with p except for the endpoints of e. Now, consider the NEPPC c' that is formed by replacing e in c with p. Since c was tight, but p is shorter than e, c' must violate its NEPPC inequality.

It may be that p and c share some vertices other than the endpoints of e. If so, then form a NEPPC c' by building a positive path p' in the following way, where v refers to the negative edge in c (see also Fig. 5).

1. Start at one endpoint of v and walk along c until it intersects p.
2. Now start at the other endpoint of v and walk in the other direction along c until it intersects p.
3. Complete the path p' by walking along the subpath of p that joins the intersection points, but does not include e.

Note that the intersection points above are well-defined, as p must meet c at the very least at the endpoints of e. Clearly p' and v form an NEPPC c', but the length of p' is bounded by the sum of the lengths of $c - e - v$ and of p. Since c was tight,

$$x_v = x_{c-v} = x_{c-e-v} + x_e > x_{c-e-v} + x_p \geqslant x_{p'},$$

hence the NEPPC inequality for c' is breached. □

Corollary 1. *In every triangle of positive edges the triangle inequalities are satisfied in an optimal solution to* (6).

We are now able to prove our main result of this section.

Theorem 7. *The linear program with only NEPPC polygon constraints* (6) *is equivalent to the triangle inequality program* (3)*, in the sense that their sets of optimal solutions are the same.*

Proof. We first show that any optimal solution to (6) must satisfy the triangle inequalities.

Although the corollary above deals with all-positive triangles, there are still a number of different cases and configurations to consider. We therefore leave the details to the reader, but note the following general principles of the proof technique.

Consider some triangle in the graph that is not covered by the corollary above: it must have at least one negative edge. If a negative edge has length one, then some of the triangle inequalities are trivially satisfied. Otherwise, the negative edge is contained in a tight NEPPC. The combination of tight NEPPCs and positive triangle edges allows us to use either the NEPPC constraints or Lemma 4 to be sure that the triangle inequality constraints are observed.

$$\begin{aligned} \text{minimize} \quad & \sum_{+(ij)} x_{ij} + \sum_{-(ij)} x'_{ij} \\ \text{subject to} \quad & \sum_{j=1}^{m-1} x_{i_j,i_{j+1}} + x'_{i_m,i_1} \geqslant 1 \quad \text{for all } C(i_1,\ldots,i_m), \\ & x_{ij} \geqslant 0 \quad \text{for all } +(ij), \\ & x'_{ij} \geqslant 0 \quad \text{for all } -(ij). \end{aligned} \tag{7}$$

Finally, since the linear program (6) is a relaxation of the original (3), the two formulations must have the same set of optimal solutions. □

We note that 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 an LP with only positive coefficients (7), in which the $x'_{ij} \leqslant 1$ constraints are unnecessary. In any feasible solution to (7), the sum of the terms around any NEPPC is at least 1. If the variables x_{ij} and x'_{ij} are binary, then we have the following interpretation: 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 [8]. This feedback edge set interpretation might lead to an algorithm with approximation ratio better than four.

As a final comment, we note that there is also some similarity to the notion of *balance* in *signed* graphs, as used in the social sciences [19]. Each person in some group is represented by a node in a graph; there is an edge between a pair of nodes if there is some strong relationship between the people, with the sign of the edge reflecting the nature of the relationship. A group, and therefore the graph, is called balanced if every cycle in the graph contains an even number of negative edges. There exist linear time algorithms to determine whether a signed graph is balanced. However, some graphs are neither completely balanced nor completely unbalanced and there is ongoing research to measure the degree of balance in them.

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 [6,7], 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 [5,15].

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

Proof. Given a graph G with k pairs (s_i, t_i), in which each s_i must be separated from each t_i, form an instance H of MinDisAgree. The edges of G become positive edges in H with unit weight. For each i, $1 \leqslant i \leqslant k$, we add a (negative) edge between s_i and t_i with weight $-W$ for some large positive integer W, say $W = n^2$. We can make the instance unweighted by replacing a negative edge of weight $-W$ by W 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 [11], we conclude that MinDisAgree is also APX-hard. Furthermore, an improvement over the $O(\log n)$ approximation ratio, which we matched 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 in $o(\log n)$?

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 minimum edge deletion graph bipartization (also known as Min-Uncut) reduces to minimum multicut in an approximation preserving way. The factor $O(\log n)$ approximation for Min-Uncut works by reducing it to a multicut instance on which the GVY algorithm is run [10]. It is implicit in Khot's work [17] that a certain conjecture about Unique games would result in Min-Uncut being NP-hard to approximate within any constant factor. Therefore, under the same conjecture, it is NP-hard to approximate minimum multicut, and also MinDisAgree, within any constant factor.

Emanuel and Fiat [7] also present an approximation preserving reduction in the reverse direction to Theorem 8, from MinDisAgree to minimum multicut. This shows that the approximability of MinDisAgree is identical to that of the fundamental minimum multicut problem.

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

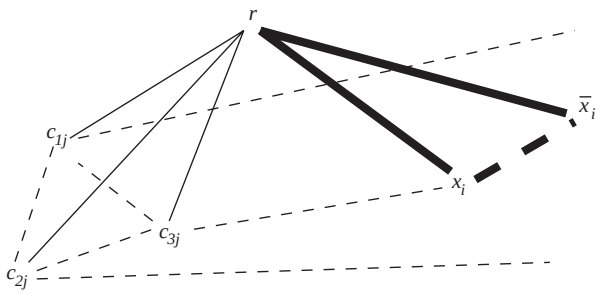

Fig. 6. Reduction from MAX 3SAT to MAXAGREE instance. The jth clause has three vertices c_{1j}, c_{2j}, c_{3j}. The ith variable has two vertices $x_i, \bar{x}_i$. Solid lines represent positive edges, dashed negative edges; thick lines represent edges of weight B_i.

$\varepsilon > 0$. However, the issue of 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 $\frac{79}{80} + \varepsilon$. Furthermore, it is* NP*-hard to approximate the unweighted version of* MAXAGREE *within a factor of $\frac{115}{116} + \varepsilon$.*

Proof. We reduce from MAX 3SAT, which is NP-hard to approximate within a factor of $\frac{7}{8} + \varepsilon$, even on satisfiable instances [14]. 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 $1 \leqslant i \leqslant n$, and *clause* vertices c_{1j}, c_{2j}, c_{3j} for each clause C_j, $1 \leqslant j \leqslant m$. The edges and their weights are defined as follows (see also Fig. 6):

- The root r is connected to each c_{pj}, $p = 1, 2, 3$, 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 pth 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 edges 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 appropriate c_{pj}s.

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, but false if it is in the root-cluster. 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 r-cluster, 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 r-cluster, but c_{2j} and c_{3j} in singleton clusters. Consequently, we have four agreements: the negative edges between the c_{pj}s and the positive edge (c_{1j}, r). The negative weight edges between c_{1j}, c_{2j}, and c_{3j} ensure that, regardless of how many of C_j's literals are true, we always achieve the same number of agreements whenever C_j is satisfied.

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

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

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 $\mathsf{OPT}_\phi \leqslant (\frac{7}{8} + \varepsilon)m$ is NP-hard [14].

In order to obtain a result for unweighted (± 1)-labeled 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 $\frac{115}{116} + \varepsilon$ 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 sum of the weights 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 $\frac{29}{28} - \varepsilon$.*

4.3. *MinDisAgree in complete graphs*

In addition to their constant factor approximation algorithm, Bansal et al. [1] 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 of the complexity of the problem on complete graphs, complementing our factor four approximation algorithm.

Theorem 11. *There exists some constant $c > 1$ for which it is* NP*-hard to approximate* MinDisAgree *on complete graphs within a factor of c.*

Proof. We give a reduction from the max 2-colorable subgraph problem on bounded degree 3-uniform hypergraphs. Here, the input is a 3-uniform hypergraph $H = (V, S)$ where each hyperedge in $S = \{e_1, e_2, \ldots, e_m\}$ consists of three elements of $V = \{v_1, \ldots, v_n\}$ with the added restriction that each

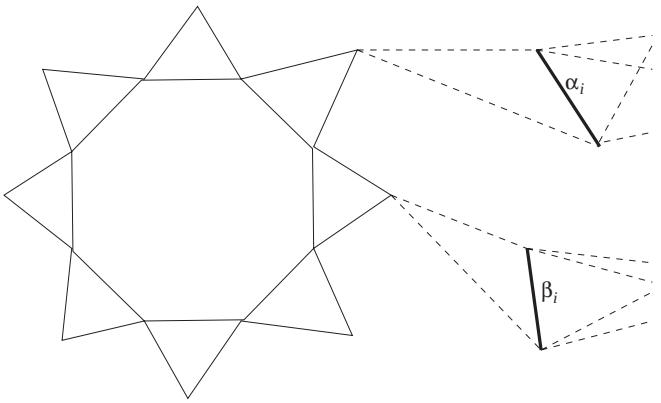

Fig. 7. Part of the graph G constructed from the hypergraph H, showing a flower, its petals, and an α, β edge pair.

element of V occurs in at most B hyperedges, for some absolute constant B (so that $m \leqslant Bn/3$). The goal is to find a 2-coloring of V that maximizes the number of hyperedges that are *split* by the coloring, that is, are bichromatic. It is known that for some absolute constants $\gamma > 0$ and B (integer), given such a 3-uniform hypergraph it is NP-hard to distinguish between the following two cases: (i) H is 2-colorable, i.e., there exists a 2-coloring of its vertices under which no hyperedge is monochromatic, and (ii) every 2-coloring of V leaves at least a fraction γ of hyperedges in S monochromatic. This follows for example from the reduction used to show the hardness of max 3-set splitting in [13]. The starting point for that reduction is a constraint satisfaction problem, called MAXSNE$_4$ in [13], that is shown to be hard to approximate in [14]. The hardness result from [14] also holds under a bounded occurrence restriction, and therefore the 3-uniform hypergraph constructed by the reduction in [13] can also be assumed to have degree bounded by an absolute constant B.

The first step in the reduction is to construct a graph G from the hypergraph H. This step is analogous to the reduction from MAX 3SAT to 3-dimensional matching in Section 9.4 of Papadimitrìou [18] and is sketched in Fig. 7. Specifically, for each $v_i \in V$, we construct a *flower* structure F_i with $4s_i$ vertices U_i, where $s_i \leqslant B$ is the number of hyperedges in which v_i occurs. The set U_i consists of $2s_i$ vertices that form an induced cycle, together with $2s_i$ *petal* vertices each of which is adjacent to the two endpoints of one of the $2s_i$ cycle edges. Let O_i (resp., E_i) be the petal vertices with odd (resp., even) indices according to an arbitrary cyclic ordering of the vertices as $1, 2, \ldots, 2s_i$. One can then pick two distinct collections of s_i vertex-disjoint triangles in the graph F_i by picking either all the triangles containing the petal vertices in O_i or all those containing the petal vertices in E_i—these collections are accordingly called *odd* and *even* collections, respectively. The choice of one of these collections will capture which one of the two colors given to the vertex v_i—this is the crux of the approach guiding the reduction. Now, corresponding to each hyperedge $e_j = (v_{j_1}, v_{j_2}, v_{j_3})$, we create two-independent edges α_j, β_j in G. We add an edge from each endpoint of one of them, say α_j, to the vertex in O_{j_1} that corresponds to the occurrence of v_{j_1} in e_j. Recall that there are s_{j_1} vertices in O_{j_1} so a different one of them will be used for each connection corresponding to each of the s_{j_1} different hyperedges containing v_{j_1}. We make similar connections between the endpoints of α_j and appropriate vertices of O_{j_2} and O_{j_3}. The endpoints of the second edge β_j are similarly connected to appropriate vertices in the *even* petal sets E_{j_1}, E_{j_2}, and E_{j_3}.

Denote by N the total number of vertices in G: clearly $N = \sum_{i=1}^{n} 4s_i + 4m = 16m$. By construction, G is 4-regular and therefore the number of edges in G, denoted by M, is $2N$—the crucial point is that G is sparse and $M = O(N)$. Finally, we construct an instance of MinDisAgree on a complete graph on N vertices by labeling all edges in G as positive and the remaining edges as negative—let us denote by $\mathcal{I}$ the resulting ± 1-weighted copy of K_N. This completes our reduction, and clearly the transformation from the 3-uniform hypergraph H to $\mathcal{I}$ can be computed in polynomial time.

Consider any clustering, call it $\mathcal{C}$, of the vertices of $\mathcal{I}$, or equivalently of G. Let the *value* of a cluster be the number of edges of G within the cluster minus the number of non-edges of G within the cluster—that is, the correlation associated with edges inside the cluster. Define the value of the clustering $\mathcal{C}$, denoted $\mathsf{value}(\mathcal{C})$, to be the sum of the values of all the clusters in $\mathcal{C}$. It is easy to verify that the number of disagreements (or mistakes) in the clustering $\mathcal{C}$, denote it $\mathsf{DisAg}(\mathcal{C})$, satisfies $\mathsf{DisAg}(\mathcal{C}) = M - \mathsf{value}(\mathcal{C})$.

We now define the value $\mathsf{val}_{\mathcal{C}}(v)$ of a vertex v, with respect to the clustering $\mathcal{C}$, to be the value of the cluster containing v divided by the number of vertices in that cluster. This way the value of a cluster is equally divided among its constituent vertices. For example, if a vertex is in a singleton cluster, its value is 0, if it is in an *edge* cluster, its value is $\frac{1}{2}$, if it belongs to a triangle cluster, its value is 1, and so on. Note that $\mathsf{value}(\mathcal{C})$ equals the sum of the values (under $\mathcal{C}$) of all the vertices.

(i) H is 2-colorable. We first claim that if H is 2-colorable, then there is a clustering $\mathcal{C}^*$ of G in which every vertex has value 1, and therefore $\mathsf{value}(\mathcal{C}^*) = N$. In what follows, a *diamond* refers to the complete graph K_4 on four vertices minus one edge. Let $f : V \to \{\mathsf{Red}, \mathsf{Blue}\}$ be a 2-coloring under which every hyperedge of H is bichromatic. First, we pick the following clusters. For each flower structure F_i, we pick the s_i triangles of the *odd* collection (those containing the vertices in O_i) if $f(v_i) = \mathsf{Red}$, and those belonging to the *even* collection (the ones containing the vertices in E_i) if $f(v_i) = \mathsf{Blue}$. We know each hyperedge e_j is bichromatic, so assume for definiteness that two of its vertices v_{j_1}, v_{j_2} are colored Red and the third one v_{j_3} is colored Blue. Then, for this j, we pick two clusters, one a triangle containing the edge α_j together with its neighbor in O_{j_3}, and the other a diamond containing the edge β_j together with its neighbors in E_{j_1} and E_{j_2}.

It is easy to check that the clustering $\mathcal{C}^*$ defined above covers all the vertices of G. Since each vertex of G is in either a triangle or a diamond cluster, it has a value of 1 and $\mathsf{value}(\mathcal{C}^*) = N$, as claimed.

(ii) H has at least γ fraction of edges monochromatic. We now wish to argue that if every 2-coloring of H leaves γm hyperedges monochromatic, then *every* clustering $\mathcal{C}'$ of G must have value at most $(1-\delta)N$ for some $\delta > 0$. The following claim is crucial to understanding how good clusterings (those with large value) of G must appear.

Claim. *In any clustering of $\mathcal{C}$ of G, the value of every vertex is at most* 1, *and if* $\mathsf{val}_{\mathcal{C}}(v) = 1$, *then v must belong to a cluster which is either a triangle or a diamond. Moreover, the supremum* $(1-\rho)$ *of the non-triangle and non-diamond vertex values is strictly less than* 1.

The claim can be proved by straightforward inspection of the structure of the graph G since it is so sparsely connected—we omit the details. The claim asserts that $\rho > 0$; in fact one can show that $\rho = 0.2$, but all we require is that ρ is a strictly positive constant.

Now suppose there exists a clustering $\mathcal{C}'$ with $\mathsf{value}(\mathcal{C}') = (1-\delta)N$. A simple counting argument shows that we must have at least

$$n - \delta N/\rho = n - 16\delta m/\rho$$

values of i for which *every* vertex in the flower structure F_i has value equal to 1. Call the vertex $v_i \in V$ for each such i *good*. Also call an hyperedge of H good if *all* three of its vertices are good. Since there are at most $16\delta m/\rho$ bad vertices in V, there are at most $16\delta mB/\rho$ bad hyperedges.

Suppose we could prove that there is a 2-coloring of H under which every good hyperedge is bichromatic, then, since every 2-coloring of H leaves at least γm monochromatic hyperedges, we would have $16\delta B/\rho \geqslant \gamma$. As a consequence,

$$\mathsf{value}(\mathcal{C}') = (1-\delta)N \leqslant (1-\zeta)N,$$

where $\zeta = \rho\gamma/(16B)$, and there would be a gap of N versus $(1-\zeta)N$ for the value of the best clustering in the two cases. Recalling that

$$\mathsf{DisAg}(\mathcal{C}) = M - \mathsf{value}(\mathcal{C}) = 2N - \mathsf{value}(\mathcal{C}),$$

we would get a gap of N versus $(1+\zeta)N$ for the number of disagreements in the best clustering. Since $\zeta > 0$ this will prove the theorem.

Therefore it only remains to prove that there is a 2-coloring g of H under which every *good* hyperedge is bichromatic. Consider a good vertex v_i: we know all internal cycle vertices in the flower structure F_i have value 1. Since there is no diamond structure containing any of these vertices, the claim tells us they must all be covered by vertex-disjoint triangles. There are only two ways to achieve this: either the triangles containing the odd petals O_i are picked, or those containing the even petals E_i are picked. We set $g(v_i) = \mathsf{Red}$ in the former case and $g(v_i) = \mathsf{Blue}$ in the latter case (the colors given to the bad vertices are of no concern). We now prove that every good hyperedge is bichromatic under this coloring. Indeed, let e_j be a hyperedge on three good vertices $v_{j_1}, v_{j_2}, v_{j_3}$, and suppose all of them are colored Red under g. Let $w_1 \in E_{j_1}$ be the vertex that is adjacent to the endpoints of β_j. Since $\mathsf{val}_{\mathcal{C}'}(w_1) = 1$, w_1 must be clustered together with the edge β_j. The same holds for the analogous vertices w_2, w_3 from E_{j_2} and E_{j_3}, respectively. But now w_1 belongs to a cluster that contains at least five elements (namely the endpoints of β_j and w_1, w_2, w_3) and therefore w_1 cannot have value 1, a contradiction. We conclude that all good hyperedges are bichromatic under g and the proof is complete. □

Acknowledgments

The authors thank the reviewers for their insightful suggestions, which improved the clarity of this article.

References

[1] N. Bansal, A. Blum, S. Chawla, Correlation clustering, in: Proceedings of 43rd FOCS, 2002, pp. 238–247.
[2] N. Bansal, A. Blum, S. Chawla, Correlation clustering, Machine Learning 56 (2004) 89–113.
[3] A. Ben-Dor, R. Shamir, Z. Yakhini, Clustering gene expression patterns, J. Comput. Biol. 6 (1999) 281–297.
[4] Z. Chen, T. Jiang, G. Lin, Computing phylogenetic roots with bounded degrees and errors, SIAM J. Comput. 32 (4) (2003) 864–879.
[5] G. Calinescu, H. Karloff, Y. Rabani, An improved approximation algorithm for multiway cut, J. Comput. System Sci. 60 (2000) 564–574.
[6] E. Demaine, N. Immorlica, Correlation clustering with partial information, in: Proceedings of Sixth APPROX, 2003, pp. 1–13.

[7] D. Emanuel, A. Fiat, Correlation clustering—minimizing disagreements on arbitrary weighted graphs, in: Proceedings of 11th ESA, 2003, pp. 208–220.
[8] G. Even, J. Naor, B. Schieber, L. Zosin, Approximating minimum subset feedback sets in undirected graphs with applications, SIAM J. Discrete Math. 25 (2000) 255–267.
[9] A. Frieze, M. Jerrum, Improved approximation algorithms for and MAX *k*-CUT and MAX BISECTION, in: E. Balas, J. Clausen (Eds.), Proceedings of Fourth IPCO, Lecture Notes in Computer Science, vol. 920, Springer, Berlin, 1995, pp. 1–13.
[10] N. Garg, V. Vazirani, M. Yannakakis, Approximate max-flow min-(multi)cut theorems and their applications, SIAM J. Comput. 25 (1996) 235–251.
[11] N. Garg, V. Vazirani, M. Yannakakis, Primal-dual approximation algorithms for integral flow and multicut in trees, Algorithmica 18 (1997) 3–20.
[12] J. Gramm, J. Guo, F. Hüffner, R. Niedermeier, Graph-modeled data clustering: Fixed-parameter algorithms for clique generation, in: Proceedings of Fifth CIAC, 2003, pp. 108–119.
[13] V. Guruswami, Inapproximability results for set splitting and satisfiability problems with no mixed clauses, Algorithmica 38 (3) (2003) 451–469.
[14] J. Håstad, Some optimal inapproximability results, J. Assoc. Comput. Mach. 48 (2001) 798–859.
[15] D. Karger, P. Klein, C. Stein, M. Thorup, N. Young, Rounding algorithms for a geometric embedding of minimum multiway cut, in: Proceedings of 31st STOC, 1999, pp. 668–678.
[16] M. Kearns, R. Schapire, L. Sellie, Toward efficient agnostic learning, Machine Learning 17 (1994) 115–142.
[17] S. Khot, On the power of unique 2-prover 1-round games, in: Proceedings of 34th STOC, 2002, pp. 767–775.
[18] C. Papadimitrìou, Computational Complexity, Addsion-Wesley, Longman, New York, 1994.
[19] F. Roberts, Discrete mathematics, in: International Encyclopedia on Social and Behavioural Sciences, Elsevier, Amsterdam, 2001, pp. 3743–3746.
[20] R. Shamir, R. Sharan, D. Tsur, Cluster graph modification problems, in: Proceedings of 28th Workshop on Graph Theory (WG), 2002, pp. 379–390.
[21] C. Swamy, Correlation clustering: maximizing agreements via semidefinite programming, in: Proceedings of 15th SODA, 2004, pp. 519–520.