Maximizing the Correlation: Extending Grothendieck's Inequality to Large Domains

Dor Katzelnick
Department of Computer Science, Technion, Haifa, Israel
dkatzelnick@cs.technion.ac.il

Roy Schwartz
Department of Computer Science, Technion, Haifa, Israel
schwartz@cs.technion.ac.il

Abstract

Correlation Clustering is an elegant model where given a graph with edges labeled $+$ or $-$, the goal is to produce a clustering that agrees the most with the labels: $+$ edges should reside within clusters and $-$ edges should cross between clusters. In this work we study the MaxCorr objective, aiming to find a clustering that maximizes the difference between edges classified correctly and incorrectly. We focus on the case of bipartite graphs and present an improved approximation of 0.254, improving upon the known approximation of 0.219 given by Charikar and Wirth [FOCS'2004] and going beyond the 0.2296 barrier imposed by their technique. Our algorithm is inspired by Krivine's method for bounding Grothendieck's constant, and we extend this method to allow for more than two clusters in the output. Moreover, our algorithm leads to two additional results: (1) the first known approximation guarantees for MaxCorr where the output is constrained to have a bounded number of clusters; and (2) a natural extension of Grothendieck's inequality to large domains.

2012 ACM Subject Classification Theory of computation → Facility location and clustering; Mathematics of computing → Combinatorial optimization

Keywords and phrases Correlation Clustering, Grothendieck's Inequality, Approximation

Digital Object Identifier 10.4230/LIPIcs.APPROX/RANDOM.2020.49

Category APPROX

Funding *Roy Schwartz*: Research is supported by ISF grant 1336/16.

1 Introduction

Correlation Clustering is a classic model, where given objects and possibly inconsistent pairwise similarity (or dissimilarity) information regarding the objects, the goal is to cluster the objects in a way that agrees the most with the given information. Since its introduction by Bansal et al. [7] close to two decades ago, Correlation Clustering has found numerous practical applications in a wide range of settings: image segmentation [35], cross-lingual link detection [33], clustering gene expression patterns [5, 8], coreference resolution [14, 15, 28] and aggregating inconsistent clusterings [18], to name a few (refer to the survey [35] and the references therein for additional details). Moreover, from a theoretical perspective it can be shown that Correlation Clustering captures classic graph cuts problems, including Min $s-t$ Cut, Multiway Cut and Multicut.

Formally, in Correlation Clustering we are given an undirected graph $G = (V, E)$ equipped with non-negative edge weights $w : E \rightarrow \mathbb{R}_+$. Additionally, each edge is labeled either with a $+$ or a $-$, where $+$ indicates similarity and $-$ indicates dissimilarity. We consider E to be the disjoint union of E^+ (all edges labeled $+$) and E^- (all edges labeled $-$). Intuitively, two nodes connected by a $+$ edge ($-$ edge) are said to be similar (dissimilar) and the weight of the edge quantifies the strength of the similarity (dissimilarity) between the two nodes. A *clustering* of the graph is a partition $\mathcal{C} = \{S_1, \ldots, S_\ell\}$ of V, for some ℓ, where each S_i is called a cluster. Two nodes u and v are in *agreement* with respect to a clustering $\mathcal{C}$ if $(u, v) \in E^+$ (or $(u, v) \in E^-$) and u and v belong to the same cluster (different clusters), and are in *disagreement* otherwise.

The above leads to three natural objectives, which were introduced in the original paper of Bansal *et. al.* [7], measuring the compliance of a clustering $\mathcal{C}$ with the input: MaxAgree, MinDisagree and MaxCorr. Given a clustering $\mathcal{C}$, the first objective aims to maximize the total weight of edges that are in agreement, the second objective aims to minimize the total weight of edges that are in disagreement, and the third objective combines the previous two objectives and aims to maximize the difference between the total weight of edges in agreement and disagreement. Formally, given a clustering $\mathcal{C}$ the MaxCorr objective value of $\mathcal{C}$ is denoted by $\mathrm{Corr}(\mathcal{C})$ and is defined as follows: $\mathrm{Corr}(\mathcal{C}) \triangleq \mathrm{Agree}(\mathcal{C}) - \mathrm{DisAgree}(\mathcal{C})$, where:

$$\begin{aligned}
\mathrm{Agree}(\mathcal{C}) &= \sum_{(u,v)\in E^+:\mathcal{C}(u)=\mathcal{C}(v)} w_{u,v} + \sum_{(u,v)\in E^-:\mathcal{C}(u)\neq\mathcal{C}(v)} w_{u,v}\\
\mathrm{DisAgree}(\mathcal{C}) &= \sum_{(u,v)\in E^+:\mathcal{C}(u)\neq\mathcal{C}(v)} w_{u,v} + \sum_{(u,v)\in E^-:\mathcal{C}(u)=\mathcal{C}(v)} w_{u,v}.
\end{aligned}$$

In the above $\mathcal{C}(u)$ denotes the cluster in $\mathcal{C}$ that u belongs to. The goal is to find a clustering $\mathcal{C}$ that maximizes $\mathrm{Corr}(\mathcal{C})$.[1] A generalization of MaxCorr, denoted by Max-k-Corr, is where we are also given as input a parameter k that upper bounds the possible number of clusters in $\mathcal{C}$.

All the above objectives have been extensively studied for almost two decades for important special cases, such as bipartite graphs [1, 5, 6, 12], complete unweighted graphs [7, 10] and weights satisfying specific constraints [2, 12], as well as general graphs [10, 11, 17, 32]. The more general problem where the number of clusters is bounded has also been studied, e.g., [7, 16, 20, 24, 6]. From a practical perspective, the study of clustering of bipartite graphs was motivated by numerous applications, such as gene expression and biological data analysis [13, 27], and data mining applications [36].

The only known algorithm for MaxCorr is given by Charikar and Wirth [11] who elegantly reduced the problem to maximizing a quadratic form. We denote the latter by Max Quad: given a matrix $B \in \mathbb{R}^{n\times n}$ find $\mathbf{x} \in \{\pm 1\}^n$ maximizing $\mathbf{x}^T B\mathbf{x}$.[2] The algorithm of [11] works as follows (for simplicity of presentation we assume that $V = \{1, \ldots, n\}$): First, set $B_{i,j}$ to be $w_{i,j}$ if $(i, j) \in E^+$, $-w_{i,j}$ if $(i, j) \in E^-$ (and 0 otherwise) and approximately solve $\max_{\mathbf{x}\in\{\pm1\}^n}\{\mathbf{x}^T B\mathbf{x}\}$. Second, consider the clustering $\mathcal{C}_1 = \{S_1, S_2\}$ defined by $\mathbf{x}$: $S_1 = \{i : x_i = 1\}$ and $S_2 = \{i : x_i = -1\}$. Third, consider the clustering $\mathcal{C}_2 = \{S_1, \ldots, S_n\}$ of V into singletons, where $S_i = \{i\}$. Finally, return the best from $\mathcal{C}_1$ and $\mathcal{C}_2$, i.e., $\max\{\mathrm{Corr}(\mathcal{C}_1), \mathrm{Corr}(\mathcal{C}_2)\}$. [11] proved that if there exists an α approximation algorithm for Max Quad, the above reduction provides an approximation of $\alpha/(2 + \alpha)$ for MaxCorr. Since Max Quad is known to have a logarithmic approximation [11, 31], i.e., $\alpha = \Omega(1/\log n)$, this results in an approximation of $\Omega(1/\log n)$ for MaxCorr on general graphs.

We note that using the above exact same reduction, one can implicitly deduce an improved approximation algorithm for MaxCorr for bipartite graphs by simply substituting the approximation algorithm for Max Quad with one for maximizing a bipartite quadratic form.

[1] Equivalently one can formulate MaxCorr without the edge labels by negating the weight of $-$ edges and then maximizing the total weight of edges inside clusters minus the total weight of edges that cross between clusters.

[2] It is assumed that the diagonal of B is all zeros, i.e., $B_{i,i} = 0$ for every $i = 1, \ldots, n$.

We denote the latter by Max BiQuad: given $A \in \mathbb{R}^{n \times m}$ find $\mathbf{x} \in \{\pm 1\}^n$ and $\mathbf{y} \in \{\pm 1\}^m$ maximizing $\mathbf{x}^T A \mathbf{y}$.[3] Specifically, assuming G is a bipartite graph containing n vertices on one side and m vertices on the other, instead of the previously defined B consider a matrix $A \in \mathbb{R}^{n \times m}$ as follows: $A_{i,j}$ is set to $w_{i,j}$ if $(i,j) \in E^+$, $-w_{i,j}$ if $(i,j) \in E^-$ and 0 otherwise. Given A, approximately solve the instance of Max BiQuad defined by A, i.e., $\max_{\mathbf{x} \in \{\pm 1\}^n, \mathbf{y} \in \{\pm 1\}^m} \{\mathbf{x}^T A y\}$. The rest of [11]'s reduction remains unchanged. One can easily verify that the overall approximation guarantee of the reduction remains as before. Max BiQuad is known to have an approximation of ≈ 0.5611 [4, 26] that follows from Grothendieck's inequality, resulting in an improved approximation of $0.5611/(2 + 0.5611) = 0.219$ for MaxCorr when G is a bipartite graph.[4]

To the best of our knowledge, no other approximation algorithm for MaxCorr besides Charikar and Wirth [11] is known, both for general and bipartite graphs. It is important to note that when restricting attention to bipartite graphs, the algorithm of [11] imposes an intrinsic barrier of 0.2296 that follows from lower bounds on Grothendieck's constant K_G (see Section 1.2 for more details). Thus, it seems there is no much room to improve the current 0.219 approximation using [11]'s approach. Moreover, no approximation is known for Max-k-Corr, even when considering bipartite graphs. The reason for the latter is that the algorithm of [11] might output n singleton clusters, thus violating the bound k on the number of clusters. The only exception is Max-2-Corr, which coincides with Max Quad and Max BiQuad for general and bipartite graphs, respectively.

Grothendieck's Inequality with Large Domains

Our work closely relates to Grothendieck's inequality. This classic inequality, first presented in [21], states that there is a universal constant K_G such that for every matrix $A \in \mathbb{R}^{n \times m}$ (recall that S^d denotes the Euclidean unit sphere in $\mathbb{R}^{d+1}$):

$$\max_{\{\mathbf{u}_i\}_{i=1}^n \cup \{\mathbf{v}_j\}_{j=1}^m \subseteq S^{n+m-1}} \left\{ \sum_{i=1}^n \sum_{j=1}^m A_{i,j} \langle \mathbf{u}_i, \mathbf{v}_j \rangle \right\} \leq K_G \cdot \max_{\mathbf{x} \in \{\pm 1\}^n, \mathbf{y} \in \{\pm 1\}^m} \left\{ \sum_{i=1}^n \sum_{j=1}^m A_{i,j} x_i y_j \right\}.$$

The problem of bounding K_G, both upper and lower bounds, has been studied for more than half a century [9, 21, 26, 29, 30]. Specifically, upper bounding K_G is typically achieved by providing an approximation algorithm for Max BiQuad that rounds a fractional solution to the natural semi-definite relaxation: each x_i and y_j is assigned a unit vector $\mathbf{u}_i$ and $\mathbf{v}_j$ respectively and $x_i y_j$ is replaced with $\langle \mathbf{u}_i, \mathbf{v}_j \rangle$. Intuitively, K_G^{-1} can be viewed as the integrality gap of this relaxation.

One can easily note that the existence of a $1/\beta_k$ approximation for Max-k-Corr for bipartite graphs, that is based on rounding the natural semi-definite relaxation (see Section 2), implies a generalization of Grothendieck's inequality to larger domains. This generalization states that there exists an α_k, where $\alpha_k \leq \beta_k$, such that for every matrix $A \in \mathbb{R}^{n \times m}$,

$$\max_{\{\mathbf{u}_i\}_{i=1}^n \cup \{\mathbf{v}_j\}_{j=1}^m \in \mathcal{P}_k} \left\{ \sum_{i=1}^n \sum_{j=1}^m A_{i,j} \frac{(2(k-1) \langle \mathbf{u}_i, \mathbf{v}_j \rangle - (k-2))}{k} \right\} \leq$$
$$\alpha_k \cdot \max_{\mathbf{x} \in [k]^n, \mathbf{y} \in [k]^m} \left\{ \sum_{i=1}^n \sum_{j=1}^m A_{i,j} s(x_i, y_j) \right\}.$$

[3] Both problems, Max BiQuad and Max Quad, are known to be NP-hard [4, 11].
[4] Braverman et al. [9] showed an improvement over 0.5611 by a very small absolute constant, thus one can in fact obtain an approximation for MaxCorr in bipartite graphs that is slightly better than 0.219.

In the above $\mathcal{P}_k$ is defined as follows: $\mathcal{P}_k \triangleq \{\mathcal{U} \subseteq S^{n+m-1} : \langle \mathbf{u}, \mathbf{v} \rangle \geq -1/(k-1)\ \forall \mathbf{u}, \mathbf{v} \in \mathcal{U}\}$, s is the signed indicator function, i.e., $s(x_i, y_j) = 1$ if $x_i = y_j$ and $s(x_i, y_j) = -1$ if $x_i \neq y_j$, and $[k] = \{0, \ldots, k-1\}$. It is important to note that when $k = 2$ the above inequality reduces to the classic Grothendieck's inequality since $\mathcal{P}_2$ only enforces that all vectors $\{\mathbf{u}_i\}_{i=1}^n$ and $\{\mathbf{v}_j\}_{j=1}^m$ are unit vectors. Thus, Grothendieck's original inequality corresponds to the case where the size of the domain is 2. Hence, for larger values of k the above inequality can be viewed as an extension of Grothendieck's original inequality to larger domains. To the best of our knowledge, such an extension was not considered in the past. A special case of particular interest of the above extended inequality, in addition to the case $k = 2$, is when $k \to \infty$. In this case the above extended inequality reduces to:

$$\max_{\{\mathbf{u}_i\}_{i=1}^n \cup \{\mathbf{v}_j\}_{j=1}^m \in \mathcal{P}_\infty} \left\{ \sum_{i=1}^n \sum_{j=1}^m A_{i,j} \left(2 \langle \mathbf{u}_i, \mathbf{v}_j \rangle - 1\right) \right\} \leq$$
$$\alpha_\infty \cdot \max_{\mathbf{x} \in [n+m]^n, \mathbf{y} \in [n+m]^m} \left\{ \sum_{i=1}^n \sum_{j=1}^m A_{i,j} s(x_i, y_j) \right\},$$

where $\mathcal{P}_\infty$ only enforces that all vectors are unit and in the positive orthant. This case is interesting since one can easily observe that if MaxCorr admits an approximation of $1/\beta$ for bipartite graphs, that is based on rounding the natural semi-definite relaxation (see Section 2), then $\alpha_\infty \leq \beta$.

CSP Over Large Domains

We note that MaxCorr and Max-k-Corr can be equivalently cast as a constraint satisfaction problem over a large domain with negative payoffs. Specifically, when considering bipartite graphs, we are given variables $x_1, \ldots, x_n, y_1, \ldots, y_m$, and weighted equality/inequality constraints containing exactly two variables per constraint (one from each type), i.e., $x_i = y_j$ or $x_i \neq y_j$. Given an assignment of values from the domain $[k]$ to each of the variables, the value of a constraint is its weight if it is satisfied and the negation of its weight if it is unsatisfied. The goal is to find an assignment maximizing the sum of values of the constraints. The above equivalence obviously applies to general graphs as well. CSP over large (non-binary) domains was studied, for example, by Guruswami and Raghavendra [22] (see the references therein for additional related work). Moreover, the classic problem of Max k-Cut, studied by Frieze and Jerum [19], is also a CSP (or equivalently a clustering problem) over a domain of size k.

1.1 Our Results

In this work we focus on MaxCorr and Max-k-Corr in bipartite graphs. We obtain the following main result for MaxCorr.

▶ **Theorem 1.** *There exists a polynomial-time* 0.254*-approximation algorithm for the problem of MaxCorr on bipartite graphs.*

There are three important things to note regarding Theorem 1. First, it improves upon the previously known approximation of 0.219 of Charikar and Wirth [11]. Second, it goes beyond the 0.2296 barrier which is intrinsic to the algorithm of [11] and follows from lower bounds on Grothendieck's constant K_G (see Section 1.2 for more details). Third, our algorithm that proves Theorem 1 produces at most three clusters, whereas the optimal solution might have any number of clusters.

We show how our main algorithm can be adapted to handle any bound k on the number of clusters. This results in an approximation factor that for $k = 2$ equals Krivine's [4, 26] guarantee of 0.5611, gracefully degrades as k increases, and reaches the guarantee of 0.254 of Theorem 1 for unbounded k (see Table 1 for approximation guarantees for some values of k). To the best of our knowledge, no previous approximation for MAX-k-CORR for bipartite graphs is known. Similarly to Theorem 1, the number of clusters produced by our algorithm for MAX-k-CORR in bipartite graphs does not exceed four, regardless of how many clusters are in the optimal solution. Moreover, allowing our algorithms to produce more clusters than four is not beneficial.

Table 1 Approximation for MAX-k-CORR.

k	2	3	4	5	6	10
approximation	0.561	0.397	0.348	0.32	0.309	0.285

The above approximation guarantees of MAXCORR and MAX-k-CORR for bipartite graphs lead to the following results regarding α_∞ and α_k for the extension of Grothendieck's inequality to large domains.

▶ **Theorem 2.** *α_k is a universal constant for every $k \geq 2$. Moreover, α_∞ is also a universal constant.*

The specific values of the bounds on α_∞ and α_k can be derived directly from our approximation factors and integrality gaps.

1.2 Our Techniques

The existing approach for approximating MAXCORR in bipartite graphs, i.e., the reduction of Charikar and Wirth [11], poses two difficulties. The first main difficulty is that this approach most likely cannot provide a much better approximation than its promised 0.219 guarantee. Recall that its approximation equals $\alpha/(2+\alpha)$, where α is the known approximation for MAX BIQUAD. It can be shown that the dependence on α in the approximation is tight, i.e., [11]'s algorithm cannot provide an approximation better than $\alpha/(2+\alpha)$. Thus, improving this algorithm requires improving α, a task that most likely will improve Grothendieck's constant K_G as well. Moreover, the known lower bound of $\beta = 1.6769$ on K_G [29] yields that [11]'s approach cannot provide an approximation better than $(1/\beta)/(2+1/\beta) \approx 0.2296$ for MAXCORR on bipartite graphs. The latter is true since the only known method for approximating MAX BIQUAD uses the standard semi-definite relaxation whose integrality gap is exactly K_G^{-1}. We note that our guarantee of 0.254 in Theorem 1 goes beyond this 0.2296 barrier that is inherent to the approach of [11]. The second main difficulty is that the approach of Charikar and Wirth [11] results in an algorithm that might produce up to n clusters. Therefore, it cannot be applied to MAX-k-CORR as it violates the constraint on the number of clusters k in the output.

To eschew the above difficulties, we adopt an approach that is inspired by Krivine's result for bounding K_G [26]. This approach is based on transforming two collections of disjoint vectors in such a way that allows random hyperplane rounding to be successful even when negative values are present in the objective function. We first show that a straightforward adaptation of this method yields an approximation of 0.198 for MAXCORR on bipartite graphs. To obtain our main result for MAXCORR we show that it suffices to extend Krivine's method to only three clusters and present suitable transformations to this end. Focusing on MAX-k-CORR we require more subtlety, as we build upon and adapt our two algorithms for

MaxCorr which are based on Krivine's method (the straightforward adaptation as well as the three cluster extension). We note that allowing our algorithms to produce more than four clusters is not beneficial.

1.3 Related Work

Many variants of Correlation Clustering were presented and studied throughout the years, both from theoretical and practical perspectives. We present here only some of the more relevant previous work. For general graphs, [7, 32] presented a 0.766-approximation for MaxAgree, [10, 17] presented an $O(\log n)$ approximation for MinDisagree, [11] presented an $\Omega(1/\log n)$ approximation for MaxCorr (while [3] presented a more refined guarantee of $\Omega(1/\log(\vartheta(\bar{G}))))$.

For complete unweighted graphs, Bansal *et al.* [7] presented a PTAS for MaxAgree, a constant factor approximation for MinDisagree, and a simple 3-approximation for MinDisagree restricted to two clusters. The approximation factor for MinDisagree was continually improved by Charikar et al. [10] to 4, Ailon et al. [2] to 2.5 and by Chawla et al. [12] to $2.06 - \varepsilon$. Moreover, Giotis et al. [20] presented a PTAS for MaxAgree and MinDisagree when the number of clusters is restricted to a constant k.

Focusing on bipartite graphs, Asteris et al. [6] presented a PTAS for MaxAgree on complete bipartite graphs, even when the number of clusters is restricted. Amit [5] presented a 11-approximation algorithm for MinDisagree on complete bipartite graphs. The latter was subsequently improved by [1] to a factor of 4 and by [12] to a factor of 3.

In the context of Grothendieck's Inequality, numerous studies were conducted. For bounding K_G, Grothendieck [21] showed that $\frac{\pi}{2} \leq K_G \leq \sinh(\frac{\pi}{2})$. Later on, Reeds [29] presented a lower bound of 1.6769. The upper bound was also improved by Rietz [30] to 2.261 and by Krivine [26] to 1.78821 who conjectured that this is the correct number. Many years later, Braverman et al. [9] showed that Krivine's bound is not the correct answer for K_G, by slightly improving the upper bound. These days, the value of K_G is still unknown. Many extensions of Grothendieck's Inequality were presented and studied, e.g., [3, 25]. In addition, Nesterov [34] presented a $(2/\pi)$-approximation for Max Quad for the special case the matrix A is positive semi-definite.

2 Semi-Definite Relaxations for MaxCorr and Max-k-Corr

In this section we present natural semi-definite programming relaxations for both MaxCorr and Max-k-Corr. These relaxation apply to both general and bipartite graphs. Focusing first on MaxCorr, we consider the following natural semi-definite programming formulation, denoted by SDP, which assigns to each vertex u a unit vector y_u.

$$\begin{aligned} \max \quad & \sum_{(u,v)\in E^+} w_{u,v}\cdot(2y_u\cdot y_v - 1) + \sum_{(u,v)\in E^-} w_{u,v}\cdot(1-2y_u\cdot y_v) && (1)\\ \text{s.t.} \quad & y_v\cdot y_v = 1 && \forall v\in V\\ & y_u\cdot y_v \geq 0 && \forall u,v\in V \end{aligned}$$

In the following lemma, we prove that (1) is a relaxation[5] for the MaxCorr problem.

▶ **Lemma 3.** *Let* $\mathrm{OPT}_{\mathrm{SDP}}$ *be the optimum value of* (1)*, and* OPT *be the value of the optimal clustering. Then it holds that* $\mathrm{OPT}_{\mathrm{SDP}} \geq \mathrm{OPT}$.

[5] The integrality gap of this relaxation is discussed in Appendix C.

Proof. Given an optimal solution $\mathcal{C}_{\text{OPT}}$, we can assign each y_v to be a standard unit vector in $\mathbb{R}^n$, while a pair of vectors y_u, y_v will be the same unit vector if and only if u and v are in the same cluster in $\mathcal{C}_{\text{OPT}}$. This way, all the constrains are satisfied: for each $v \in V$ we have that $y_v \cdot y_v = 1$, and for each $u, v \in V$ it holds that $y_u \cdot y_v \in \{0, 1\}$. Moreover, the value of this solution for the SDP formulation is the same as $\text{Corr}(\mathcal{C}_{\text{OPT}})$: for a plus edge (u, v), if u and v are in the same cluster in $\mathcal{C}_{\text{OPT}}$, then $y_u \cdot y_v = 1$ and so it contributes $w_{u,v}$. If they are in different cluster, then $y_u \cdot y_v = 0$ and the edge contributes $-w_{u,v}$ to the objective function. Similar argument can be placed for the minus edges. Therefore, it holds that $\text{OPT}_{\text{SDP}} \geq \text{OPT}$. ◀

Recall that in MAX-k-CORR the output can contain at most k clusters. Therefore, focusing on MAX-k-CORR we consider the following natural semi-definite formulation, inspired by the work of Karger *et. al.* on 3-COLORING [23], which we denote by SDP_k:

$$\begin{aligned} \max \quad & \sum_{(u,v)\in E^+} w_{u,v}\left(\frac{2(k-1)}{k} y_u \cdot y_v - \frac{k-2}{k}\right) + \sum_{(u,v)\in E^-} w_{u,v}\left(\frac{k-2}{k} - \frac{2(k-1)}{k} y_u \cdot y_v\right) \\ \text{s.t.} \quad & y_v \cdot y_v = 1 \qquad \forall v \in V \\ & y_u \cdot y_v \geq -1/(k-1) \qquad \forall u, v \in V \end{aligned} \tag{2}$$

One may observe, that if we are limited to two clusters only, i.e., $k = 2$, then SDP_2 is exactly the standard semi-definite relaxation for both MAX QUAD and MAX BIQUAD (as in Grothendieck's inequality). Moreover, when $k \to \infty$, namely k is large enough such that MAX-k-CORR becomes MAXCORR, SDP_∞ becomes the relaxation for MAXCORR. Similarly to MAXCORR, we prove the following lemma which shows that (2) is indeed a relaxation for the problem of MAX-k-CORR.

▶ **Lemma 4.** *Let $\text{OPT}_{\text{SDP}_\text{k}}$ be the optimum value of (2), and OPT_k be the value of the optimal clustering with at most k clusters. Then it holds that $\text{OPT}_{\text{SDP}_\text{k}} \geq \text{OPT}_k$.*

Proof. Given an optimal solution with at most k clusters, $\mathcal{C}_{\text{OPT}}$, we construct the following fractional solution. Let $v_1, \ldots, v_k$ be the vectors that form the $k-1$-regular simplex centered at the origin. We know that $v_1 + \cdots + v_k = 0$, and therefore

$$0 = \|v_1 + \cdots + v_k\|^2 = \sum_{i=1}^{k} v_i \cdot v_i + \sum_{i \neq j} v_i \cdot v_j = k + \sum_{i \neq j} v_i \cdot v_j.$$

Since the vertices of the regular simplex are in equal distance from each other, it must be the case that $v_i \cdot v_j = -1/(k-1)$ for all $i \neq j$. One can show that this is the most efficient solution, in terms of maximizing the minimum distance between k points on the Euclidean unit sphere.

Let us assign the vectors $\{y_v\}_{v \in V}$ to the vectors $v_1, \ldots, v_k$ where $y_u = y_v$ iff u and v are in the same cluster in $\mathcal{C}_{\text{OPT}}$. One can see that if $y_u \cdot y_v = 1$, it holds that

$$\frac{2(k-1)}{k} y_u \cdot y_v - \frac{k-2}{k} = \frac{2k-2-k+2}{k} = 1,$$

and if $y_u \cdot y_v = -1/(k-1)$, then it holds that

$$\frac{2(k-1)}{k} y_u \cdot y_v - \frac{k-2}{k} = \frac{-2-k+2}{k} = -1.$$

Hence, the value of the objective of (2) with this fractional solution equals the value of the integral solution. Thus, $\text{OPT}_{\text{SDP}_\text{k}} \geq \text{OPT}_k$. ◀

3 Approximation Algorithms for MaxCorr

In this section we present our approximation algorithms for MaxCorr on bipartite graphs, thus proving Theorem 1. First, for simplicity of presentation, we show that a straightforward adaptation of Krivine's method to SDP yields an approximation of 0.198. Second, we present an improved algorithm that achieves the guarantee of Theorem 1 by extending Krivine's method to more than two clusters.

3.1 Case Study: A Simple Two Clusters Algorithm

In this section we present a straightforward adaptation of Krivine's method to round SDP that produces at most two clusters. Denote by $V = V_1 \cup V_2$ the two sides of the graph and by n the total number of vertices in V. Let $A \in \mathbb{R}^{n\times n}$ be the matrix of the solution of (1). That is, $A_{i,j} = y_i \cdot y_j$ for all $i,j \in V$. Given a solution A, we can construct the corresponding vectors $\{y_v\}_{v\in V}$ in polynomial time. Additionally, we note that (1) can be solved, in polynomial time, and the value of the solution will be far from the optimum up to some additive error which is arbitrarily small.

Algorithm 1 Two Clusters Algorithm.

```
  Input: $G = (V_1, V_2, E)$ and $w : E \to \mathbb{R}$
  Output: clustering $\mathcal{C}$
1 Solve (1) for $G$ and $w$ to obtain a positive semi-definite matrix $A \in \mathbb{R}^{n\times n}$.
2 Define $\tilde{A} \in \mathbb{R}^{n\times n}$ as follows: $\tilde{A}_{i,j} \leftarrow \begin{cases} f(A_{i,j}) & \text{if } i \text{ and } j \text{ in different sides of } V \\ g(A_{i,j}) & \text{otherwise} \end{cases}$
3 Find vectors $\{\tilde{y}_i\}_{i\in V}$ s.t.: $\tilde{y}_i \cdot \tilde{y}_j = \tilde{A}_{i,j}\ \forall i,j \in V$ (via Cholesky decomposition of $\tilde{A}$).
4 Choose $z$ uniformly at random from $S^{n-1}$.
5 Set $C_1 \leftarrow \{i \in V : \tilde{y}_i \cdot z \geq 0\}$ and $C_2 \leftarrow \{i \in V : \tilde{y}_i \cdot z < 0\}$.
6 Return $\mathcal{C} = \{C_1, C_2\}$.
```

Next, we show how to choose the transformation functions f and g, and prove that the algorithm achieves the desired approximation factor. Specifically, we prove that it suffices to choose:

$$f(x) \triangleq \sin(c\pi x)\cos(\frac{c\pi}{2}) - \cos(c\pi x)\sin(\frac{c\pi}{2})$$
$$g(x) \triangleq \sinh(c\pi x)\cos(\frac{c\pi}{2}) + \cosh(c\pi x)\sin(\frac{c\pi}{2}),$$

where c is the solution to the equation $\sinh(c\pi)\cos(\frac{c\pi}{2}) + \cosh(c\pi)\sin(\frac{c\pi}{2}) = 1$. The reader is referred to Appendix A for the complete analysis.

3.2 Producing More Clusters – Better Approximation

Algorithm 1 always clusters the graph into at most two clusters while ignoring the fact that the optimal solution might contain a larger number of clusters. Building upon Algorithm 1, we present an algorithm that clusters the graph into at most three clusters. Though this seems like only a minor change, it enables us to improve the approximation factor to 0.254, well beyond the 0.2296 barrier of the approach of Charikar and Wirth [11].

We note that Algorithm 2 differs from Algorithm 1 since it uniformly at random picks one of the two initial clusters and splits it again using another independent random hyperplane. This subtle change incurs a significant change in the choice of f and g, as will be evident

Algorithm 2 Three Clusters Algorithm.

```
   Input: $G = (V_1, V_2, E)$ and $w : E \to \mathbb{R}$
   Output: clustering $\mathcal{C}$
1  Solve (1) for $G$ and $w$ to obtain a positive semi-definite matrix $A \in \mathbb{R}^{n\times n}$.
2  Define $\tilde{A} \in \mathbb{R}^{n\times n}$ as follows: $\tilde{A}_{i,j} \leftarrow \begin{cases} f(A_{i,j}) & \text{if } i \text{ and } j \text{ in different sides of } V \\ g(A_{i,j}) & \text{otherwise} \end{cases}$
3  Find vectors $\{\tilde{y}_i\}_{i\in V}$ s.t.: $\tilde{y}_i \cdot \tilde{y}_j = \tilde{A}_{i,j}\ \forall i,j \in V$ (via Cholesky decomposition of $\tilde{A}$).
4  Choose $z$ and $z'$ independently and uniformly from $S^{n-1}$.
5  Set $C' \leftarrow \{i \in V : \tilde{y}_i \cdot z \geq 0\}$ and $C'' \leftarrow \{i \in V : \tilde{y}_i \cdot z < 0\}$.
6  Let $C_1$ be a uniform random cluster from $C'$ and $C''$, and $C$ the remaining cluster.
7  Set $C_2 \leftarrow \{i \in C : \tilde{y}_i \cdot z' \geq 0\}$ and $C_3 \leftarrow \{i \in C : \tilde{y}_i \cdot z' < 0\}$.
8  Return $\mathcal{C} = \{C_1, C_2, C_3\}$.
```

from the analysis. Furthermore, this subtle change also makes the analysis more challenging, compared to Krivine's original method. The reason is that the separation probability of two vectors is quadractically dependent on the angle between them, as opposed to linear dependency in Krivine's original method. In order to analyze this algorithm, we first calculate the probability that two vertices will be in the same cluster.

▶ **Lemma 5.** *Let $u, v \in V$ whose corresponding vectors are $\tilde{y}_u, \tilde{y}_v$ respectively. Then,*

$$\Pr[u, v \textit{ are in the same cluster}] = \frac{1}{2}\left(1 - \frac{\arccos(\tilde{y}_u \cdot \tilde{y}_v)}{\pi}\right) + \frac{1}{2}\left(1 - \frac{\arccos(\tilde{y}_u \cdot \tilde{y}_v)}{\pi}\right)^2.$$

Proof. If u and v are in different clusters after splitting V into C' and C'', then u and v will remain apart. Otherwise, with probability $1/2$ we will choose to split their cluster, and then the chance of separating u and v is the same as before. That is,

$$\Pr[u, v \text{ are in the same cluster}] = \left(1 - \frac{\arccos(\tilde{y}_u \cdot \tilde{y}_v)}{\pi}\right) \cdot \left(\frac{1}{2} + \frac{1}{2}\left(1 - \frac{\arccos(\tilde{y}_u \cdot \tilde{y}_v)}{\pi}\right)\right).$$

This concludes the proof. ◀

Similarly to Lemma 11 we have the following lemma. In what follows, $X_{u,v}$ is the random variable denoting the contribution of the edge (u, v) to the value of the output of Algorithm 2.

▶ **Lemma 6.** *For every edge $(u, v) \in E$:*

$$\mathbb{E}[X_{u,v}] = \begin{cases} w_{u,v} \cdot \left(1 - 3 \cdot \frac{\arccos(\tilde{y}_u \cdot \tilde{y}_v)}{\pi} + \frac{\arccos(\tilde{y}_u \cdot \tilde{y}_v)^2}{\pi^2}\right) & (u, v) \in E^+ \\ w_{u,v} \cdot \left(3 \cdot \frac{\arccos(\tilde{y}_u \cdot \tilde{y}_v)}{\pi} - \frac{\arccos(\tilde{y}_u \cdot \tilde{y}_v)^2}{\pi^2} - 1\right) & (u, v) \in E^-. \end{cases}$$

Proof. For every edge $(u, v) \in E^+$:

$$\begin{aligned} \mathbb{E}[X_{u,v}] &= w_{u,v} \cdot \Pr[u, v \text{ are in the same cluster}] - w_{u,v} \cdot (1 - \Pr[u, v \text{ are in the same cluster}]) \\ &= w_{u,v} \cdot \left(2\left(\frac{1}{2}\left(1 - \frac{\arccos(\tilde{y}_u \cdot \tilde{y}_v)}{\pi}\right) + \frac{1}{2}\left(1 - \frac{\arccos(\tilde{y}_u \cdot \tilde{y}_v)}{\pi}\right)^2\right) - 1\right) \\ &= w_{u,v} \cdot \left(1 - 3 \cdot \frac{\arccos(\tilde{y}_u \cdot \tilde{y}_v)}{\pi} + \frac{\arccos(\tilde{y}_u \cdot \tilde{y}_v)^2}{\pi^2}\right). \end{aligned}$$

For a minus edge, we can simply multiply the above by -1. ◀

The following lemma lies at the heart of the argument as it provides the existence of suitable transformations f and g.

▶ **Lemma 7.** *There exist functions $f, g : \mathbb{R} \to \mathbb{R}$ and a constant $c \in (0,1)$ such that the matrix $\tilde{A}$ obtained in Algorithm 2 can be decomposed into unit vectors $\{\tilde{y}_v\}_{v\in V}$ in $\mathbb{R}^n$, and these vectors satisfy that for all $(u,v) \in E$:*

$$\begin{cases} \left(1 - 3 \cdot \frac{\arccos(\tilde{y}_u \cdot \tilde{y}_v)}{\pi} + \frac{\arccos(\tilde{y}_u \cdot \tilde{y}_v)^2}{\pi^2}\right) = c \cdot (2y_u \cdot y_v - 1) & \text{if } (u,v) \in E^+ \\ \left(3 \cdot \frac{\arccos(\tilde{y}_u \cdot \tilde{y}_v)}{\pi} - \frac{\arccos(\tilde{y}_u \cdot \tilde{y}_v)^2}{\pi^2} - 1\right) = c \cdot (1 - 2y_u \cdot y_v) & \text{if } (u,v) \in E^-. \end{cases}$$

Proof. We wish to find the appropriate functions f, g and the constant c, that will satisfy the above conditions. That is, we need to solve the equation:

$$\left(1 - 3 \cdot \frac{\arccos(\tilde{y}_u \cdot \tilde{y}_v)}{\pi} + \frac{\arccos(\tilde{y}_u \cdot \tilde{y}_v)^2}{\pi^2}\right) = c \cdot (2y_u \cdot y_v - 1)\,.$$

It has two solutions: $\arccos(\tilde{y}_u \cdot \tilde{y}_v) = \frac{1}{2}(3\pi \pm \pi\sqrt{5 - 4c + 8c(y_u \cdot y_v)})$. Therefore, we have two candidates for f: $\cos(\frac{1}{2}(3\pi - \pi\sqrt{5-4c+8cx}))$ and $\cos(\frac{1}{2}(3\pi + \pi\sqrt{5-4c+8cx}))$. We know that $\arccos(x)$ is bounded between 0 and π, and expect that f will satisfy $f(1) > f(0)$, so we dismiss the latter candidate solution and remain with:

$$f(x) = \cos(\frac{1}{2}(3\pi - \pi\sqrt{5 - 4c + 8cx})).$$

Moreover, since $y_u \cdot y_v \geq 0$ and $0 \leq c \leq 1$, it holds that $5 - 4c + 8c(y_u \cdot y_v) \geq 0$ for all u, v and so f is well defined for our case. In fact, we can see that f is well defined for all $x \geq {(4c-5)}/{(8c)}$. Hence, we consider the Taylor expansion of f at $x = 0$, defined by $p(x) = \sum_{k=0}^{\infty}(k!)^{-1}f^{(k)}(0)x^k$. Let f_k be the coefficient of x^k in $p(x)$. We note that the coefficients depend on the value of c. Thus, $p(x)$ has the following two properties, described in two technical lemmas that are given without a proof.

▶ **Lemma 8** (The convergence radius of p). *The function $f(x)$ is equal to its Taylor series, $p(x)$, for all $x \in (\frac{4c-5}{8c}, -\frac{4c-5}{8c})$, and in particular, $p(x) = f(x)$ for all $x \in [-1,1]$ if $c < 5/12$.*

▶ **Lemma 9** (The signs of the coefficients of p). *For all $c \in (0.25, 0.4)$, it holds that $f_0 < 0$, $f_1 > 0$, $f_2 < 0$ and for all $k \geq 2$, $f_{2k} > 0$ and $f_{2k-1} < 0$.*

Thus, from now on we assume that c is in the range $(0.25, 0.4)$, and can consider the function $f(-x)$. It holds that for all $x \in [-1,1]$,

$$f(-x) = p(-x) = \sum_{k=0}^{\infty} f_{2k}x^{2k} - \sum_{k=0}^{\infty} f_{2k+1}x^{2k+1} = f_0 - f_1x + f_2x^2 + \sum_{k=3}^{\infty}|f_k|x^k.$$

Therefore, we conclude that $\sum_{k=0}^{\infty}|f_k|x^k = f(-x) - 2f_0 + 2f_1x - 2f_2x^2$. Hence, let us define $g(x) = f(-x) - 2f_0 + 2f_1x - 2f_2x^2$. We can prove that $\tilde{A}$ is positive semi-definite, and can be decomposed to the above vectors $\{\tilde{y}_v\}_{v\in V}$ in a similar way to Lemma 13. Next, we wish to find the value of c for which the new vectors $\{\tilde{y}_v\}_{v\in V}$ will be unit vectors. Plugging $x = 1$ in the equation $g(x) = 1$ and then solving for c in the required range, will yield the solution $c \approx 0.254013$. This completes the proof of Lemma 7. ◀

Proof of Theorem 1. According to Lemma 6 and Lemma 7, it follows that Algorithm 2 achieves an approximation of 0.254 for the problem of MaxCorr on bipartite graphs. ◀

As a result from Theorem 1, we obtain an upper bound on α_∞, proving the second part of Theorem 2. Additionally, we note that Algorithm 2 can be slightly improved. If we consider the probability that two vertices will be in the same cluster, we can notice that it is the same as if we split the hyper-sphere with one hyper-plane with probability 1/2, or with two hyper-planes with probability 1/2. Hence, we can find the value of p for which the algorithm that splits the hyper-sphere with one hyper-plane with probability p or with two hyper-planes with probability $1-p$, achieves the best approximation factor (with the above analysis). It turns out that the best p is very close to 1/2, i.e., if we set $p = 0.49$ the approximation factor will be $c \approx 0.2551$. As previously mentioned, allowing the algorithm to produce more than four clusters is not beneficial (refer to Appendix B for additional details).

4 Restricting the Number of Clusters – Approximating Max-k-Corr

In this section we focus on Max-k-Corr, where the solution is constrained to contain at most k clusters. We build upon Algorithms 1 and 2 to obtain our results.

4.1 Two Clusters – Adapting Algorithm 1

We start with adapting Algorithm 1 to Max-k-Corr. It is obvious that this algorithm produces a feasible solution, no matter which transformations f and g we employ, since $k \geq 2$ and it produces at most two clusters. However, since SDP$_k$ changes with k, the original transformations f and g, that were chosen when considering MaxCorr and SDP (equivalently SDP$_\infty$), do not yield a meaningful approximation anymore. Thus, all that remains is to choose these transformations, given k. This is done by requiring that the transformed vectors satisfy the following for every edge $(u,v) \in E$:

$$1 - 2\frac{\arccos(\tilde{y}_u \cdot \tilde{y}_v)}{\pi} = c\left(\frac{2(k-1)}{k} y_u \cdot y_v - \frac{k-2}{k}\right).$$

To satisfy the above we choose f as follows:

$$\begin{aligned} f(x) &= \cos\left(\frac{\pi}{2k}(k + c(k-2) - 2cx(k-1))\right) \\ &= \sin\left(c\pi x \frac{k-1}{k}\right)\cos\left(\frac{c\pi}{2}\frac{k-2}{k}\right) - \cos\left(c\pi x\frac{k-1}{k}\right)\sin\left(\frac{c\pi}{2}\frac{k-2}{k}\right). \end{aligned}$$

Hence, the function g will be:

$$g(x) = \sinh\left(c\pi x\frac{k-1}{k}\right)\cos\left(\frac{c\pi}{2}\frac{k-2}{k}\right) + \cosh\left(c\pi x\frac{k-1}{k}\right)\sin\left(\frac{c\pi}{2}\frac{k-2}{k}\right).$$

The analysis continues the same as in Algorithm 1, and we require for the transformed vectors to be unit vectors, i.e., that $g(1) = 1$. Solving this equation for each value of k, yields the desired approximation factor c. These appear in Table 2 for some values of k.

Table 2 Approximation factors obtained by adapting Algorithm 1 for Max-k-Corr.

k	2	3	4	5	10	50	1000
c	0.5611	0.3441	0.2901	0.2654	0.2269	0.2034	0.1985

We can see that when $k = 2$, we have exactly the constant obtained by Krivine. As k increases the approximation factor gracefully degrades, and when $k \to \infty$ it approaches 0.1985 (the approximation we obtained for MaxCorr with Algorithm 1). Additionally, we note that this result proves the first part of Theorem 2 as it provides upper bounds on α_k for $k \geq 2$.

4.2 More Than Two Clusters – Adapting Algorithm 2

The success of Algorithm 2 implies that allowing more than two clusters in the output might be beneficial. However, adapting this approach, and specifically Algorithm 2, to Max-k-Corr requires much care as the value of k will have a considerable influence on the resulting algorithm and its transformations.

Specifically, given k, we need to choose from the following clustering techniques: (1) two clusters produced by a random hyperplane (similarly to Algorithm 1); (2) three clusters produced by a random hyperplane and then further partioning of one of the clusters by an additional random hyperplane (similarly to Algorithm 2); and (3) four clusters produced by choosing two random hyperplanes. The exact convex combination of these techniques will uniquely dictate the transformations f and g that will ensure the success of the algorithm. We refer the reader to Appendix B for a more detailed discussion of the above, the approximation guarantees obtained for specific values of k, and additional details, e.g., why not use a larger number of hyperplanes to construct a clustering.

References

1 Nir Ailon, Noa Avigdor-Elgrabli, Edo Liberty, and Anke Van Zuylen. Improved approximation algorithms for bipartite correlation clustering. *SIAM Journal on Computing*, 41(5):1110–1121, 2012.

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

3 Noga Alon, Konstantin Makarychev, Yury Makarychev, and Assaf Naor. Quadratic forms on graphs. *Inventiones mathematicae*, 163(3):499–522, 2006.

4 Noga Alon and Assaf Naor. Approximating the cut-norm via grothendieck's inequality. In *Proceedings of the thirty-sixth annual ACM symposium on Theory of computing*, pages 72–80, 2004.

5 Noga Amit. *The bicluster graph editing problem.* PhD thesis, Tel Aviv University, 2004.

6 Megasthenis Asteris, Anastasios Kyrillidis, Dimitris Papailiopoulos, and Alexandros Dimakis. Bipartite correlation clustering: Maximizing agreements. In *Artificial Intelligence and Statistics*, pages 121–129, 2016.

7 Nikhil Bansal, Avrim Blum, and Shuchi Chawla. Correlation clustering. *Machine learning*, 56(1-3):89–113, 2004.

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

9 Mark Braverman, Konstantin Makarychev, Yury Makarychev, and Assaf Naor. The grothendieck constant is strictly smaller than krivine's bound. ieee 52nd annual symposium on foundations of computer science focs 2011, 453–462. *IEEE Computer Soc., Los Alamitos, CA*, 2011.

10 Moses Charikar, Venkatesan Guruswami, and Anthony Wirth. Clustering with qualitative information. *Journal of Computer and System Sciences*, 71(3):360–383, 2005.

11 Moses Charikar and Anthony Wirth. Maximizing quadratic programs: Extending grothendieck's inequality. In *45th Annual IEEE Symposium on Foundations of Computer Science*, pages 54–60. IEEE, 2004.

12 Shuchi Chawla, Konstantin Makarychev, Tselil Schramm, and Grigory Yaroslavtsev. Near optimal lp rounding algorithm for correlationclustering on complete and complete k-partite graphs. In *Proceedings of the forty-seventh annual ACM symposium on Theory of computing*, pages 219–228, 2015.

13 Yizong Cheng and George M Church. Biclustering of expression data. In *Ismb*, volume 8, pages 93–103, 2000.

14 William Cohen and Jacob Richman. Learning to match and cluster entity names. In *In ACM SIGIR-2001 Workshop on Mathematical/Formal Methods in Information Retrieval*, 2001.

15 William W. Cohen and Jacob Richman. Learning to match and cluster large high-dimensional data sets for data integration. In *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '02, page 475–480, New York, NY, USA, 2002. Association for Computing Machinery.

16 Tom Coleman, James Saunderson, and Anthony Wirth. A local-search 2-approximation for 2-correlation-clustering. In *European Symposium on Algorithms*, pages 308–319. Springer, 2008.

17 Erik D Demaine, Dotan Emanuel, Amos Fiat, and Nicole Immorlica. Correlation clustering in general weighted graphs. *Theoretical Computer Science*, 361(2-3):172–187, 2006.

18 Vladimir Filkov and Steven Skiena. Integrating microarray data by consensus clustering. *International Journal on Artificial Intelligence Tools*, 13(04):863–880, 2004.

19 Alan M. Frieze and Mark Jerrum. Improved approximation algorithms for MAX k-cut and MAX BISECTION. *Algorithmica*, 18(1):67–81, 1997. `doi:10.1007/BF02523688`.

20 Ioannis Giotis and Venkatesan Guruswami. Correlation clustering with a fixed number of clusters. *Theory of Computing*, 2(13):249–266, 2006. `doi:10.4086/toc.2006.v002a013`.

21 Alexandre Grothendieck. *Résumé de la théorie métrique des produits tensoriels topologiques*. Soc. de Matemática de São Paulo, 1956.

22 Venkatesan Guruswami and Prasad Raghavendra. Constraint satisfaction over a non-boolean domain: Approximation algorithms and unique-games hardness. In *Approximation, Randomization and Combinatorial Optimization. Algorithms and Techniques*, pages 77–90. Springer, 2008.

23 David R. Karger, Rajeev Motwani, and Madhu Sudan. Approximate graph coloring by semidefinite programming. *J. ACM*, 45(2):246–265, 1998. `doi:10.1145/274787.274791`.

24 Marek Karpinski and Warren Schudy. Linear time approximation schemes for the gale-berlekamp game and related minimization problems. In *Proceedings of the forty-first annual ACM symposium on Theory of computing*, pages 313–322, 2009.

25 Subhash Khot and Assaf Naor. Grothendieck-type inequalities in combinatorial optimization. *arXiv preprint*, 2011. `arXiv:1108.2464`.

26 Jean-Louis Krivine. Constantes de grothendieck et fonctions de type positif sur les spheres. *Séminaire Analyse fonctionnelle (dit" Maurey-Schwartz")*, pages 1–17, 1978.

27 Sara C Madeira and Arlindo L Oliveira. Biclustering algorithms for biological data analysis: a survey. *IEEE/ACM transactions on computational biology and bioinformatics*, 1(1):24–45, 2004.

28 Andrew McCallum and Ben Wellner. Toward conditional models of identity uncertainty with application to proper noun coreference. In *Proceedings of the IJCAI-2003 Workshop on Information Integration on the Web*, pages 79–86, August 2003.

29 JA Reeds. A new lower bound on the real grothendieck constant. *Manuscript (http://www. dtc. umn. edu/ reedsj/bound2. dvi)*, 88(96):107, 1991.

30 Ronald E Rietz. A proof of the grothendieck inequality. *Israel journal of mathematics*, 19(3):271–276, 1974.

31 Cornelis Roos, Tamás Terlaky, Arkadi Nemirovski, and Kees Roos. On maximization of quadratic form over intersection of ellipsoids with common center. *Mathematical Programming*, 86, September 1999.

32 Chaitanya Swamy. Correlation clustering: maximizing agreements via semidefinite programming. In *Proceedings of the fifteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 526–527. Society for Industrial and Applied Mathematics, 2004.

33 Jurgen Van Gael and Xiaojin Zhu. Correlation clustering for crosslingual link detection. In *IJCAI*, pages 1744–1749, 2007.

34 David P. Williamson and David B. Shmoys. *The Design of Approximation Algorithms*. Cambridge University Press, 2011. URL: `http://www.cambridge.org/de/knowledge/isbn/item5759340/?site_locale=de_DE`.

35 Anthony Wirth. Correlation clustering. *C. Sammut and G. Webb, editors, Encyclopedia of Machine Learning*, pages 227—-231, 2010.

36 Hongyuan Zha, Xiaofeng He, Chris Ding, Horst Simon, and Ming Gu. Bipartite graph partitioning and data clustering. In *Proceedings of the tenth international conference on Information and knowledge management*, pages 25–32, 2001.

A Analysis of Algorithm 1

Lemma 10 is widely known and is given without proof (see, e.g., [3]).

▶ **Lemma 10.** *Let y_u, y_v be two unit vectors in $\mathbb{R}^n$ and z be a random unit vector chosen uniformly from S^{n-1}. Then,*

$$\Pr[\operatorname{sign}(y_u \cdot z) = \operatorname{sign}(y_v \cdot z)] = 1 - \frac{\arccos(y_u \cdot y_v)}{\pi}.$$

▶ **Lemma 11.** *Let $\tilde{y}_u, \tilde{y}_v$ be the vectors representing u and v respectively. Additionally, denote by $X_{u,v} = w_{u,v} \cdot \operatorname{sign}(\tilde{y}_u \cdot z)\operatorname{sign}(\tilde{y}_v \cdot z)$ the random variable that represents the contribution of the edge (u, v) to the value of the solution. Therefore,*

$$\mathbb{E}[X_{u,v}] = \begin{cases} w_{u,v}\left(1 - 2\frac{\arccos(\tilde{y}_u \cdot \tilde{y}_v)}{\pi}\right) & (u,v) \in E^+ \\ w_{u,v}\left(2\frac{\arccos(\tilde{y}_u \cdot \tilde{y}_v)}{\pi} - 1\right) & (u,v) \in E^-. \end{cases}$$

Proof. From Algorithm 1, we can see that that two vertices u, v will be in the same cluster if and only if $\operatorname{sign}(\tilde{y}_u \cdot z) = \operatorname{sign}(\tilde{y}_v \cdot z)$. Therefore, for every plus edge $(u, v) \in E^+$,

$$\begin{aligned}\mathbb{E}[X_{u,v}] &= w_{u,v} \Pr[\operatorname{sign}(\tilde{y}_u \cdot z) = \operatorname{sign}(\tilde{y}_v \cdot z)] + (-1)w_{u,v}(1 - \Pr[\operatorname{sign}(\tilde{y}_u \cdot z) = \operatorname{sign}(\tilde{y}_v \cdot z)]) \\ &= w_{u,v}(2\Pr[\operatorname{sign}(\tilde{y}_u \cdot z) = \operatorname{sign}(\tilde{y}_v \cdot z)] - 1).\end{aligned}$$

Additionally, for every minus edge $(u, v) \in E^-$,

$$\begin{aligned}\mathbb{E}[X_{u,v}] &= w_{u,v}(1 - \Pr[\operatorname{sign}(\tilde{y}_u \cdot z) = \operatorname{sign}(\tilde{y}_v \cdot z)]) + (-1)w_{u,v} \Pr[\operatorname{sign}(\tilde{y}_u \cdot z) = \operatorname{sign}(\tilde{y}_v \cdot z)] \\ &= w_{u,v}(1 - 2\Pr[\operatorname{sign}(\tilde{y}_u \cdot z) = \operatorname{sign}(\tilde{y}_v \cdot z)])\end{aligned}$$

Consequently, we can use Lemma 10 to complete this proof. ◀

The following corollary is immediate from linearity of expectation.

▶ **Corollary 12.** *Let X_{ALG} be the value of the output of Algorithm 1 . Then*

$$\mathbb{E}[X_{ALG}] = \sum_{(u,v)\in E^+} w_{u,v}\left(1 - 2\frac{\arccos(\tilde{y}_u \cdot \tilde{y}_v)}{\pi}\right) + \sum_{(u,v)\in E^-} w_{u,v}\left(2\frac{\arccos(\tilde{y}_u \cdot \tilde{y}_v)}{\pi} - 1\right).$$

Proof. Follows directly from Lemma 11 and the linearity of expectation. ◀

The following lemma lies at the heart of the argument as it proves the existence of suitable transformations f and g.

▶ **Lemma 13.** *There exist functions $f, g : \mathbb{R} \to \mathbb{R}$ and a constant $c \in (0,1)$ such that the matrix $\tilde{A}$ obtained in Algorithm 1 can be decomposed into unit vectors $\{\tilde{y}_v\}_{v\in V}$ in $\mathbb{R}^n$, and these vectors satisfy that for all $(u,v) \in E^+$,*

$$1 - 2\frac{\arccos(\tilde{y}_u \cdot \tilde{y}_v)}{\pi} = c(2y_u \cdot y_v - 1)$$

and for all $(u,v) \in E^-$,

$$2\frac{\arccos(\tilde{y}_u \cdot \tilde{y}_v)}{\pi} - 1 = c(1 - 2y_u \cdot y_v).$$

Proof. First, we want to choose a function f such that

$$1 - 2\frac{\arccos(f(y_u \cdot y_v))}{\pi} = c(2y_u \cdot y_v - 1)$$

for all $(u,v) \in E$. This will satisfy the condition on plus edges and minus edges as well. The constant c will be determined later. We choose the function f in the following way:

$$f(x) \triangleq \cos\left(\frac{\pi}{2}(1 - c(2x-1))\right) = \sin\left(c\pi x - \frac{c\pi}{2}\right) = \sin(c\pi x)\cos(\frac{c\pi}{2}) - \cos(c\pi x)\sin(\frac{c\pi}{2}),$$

where the last equality follows from the identity: $\sin(a-b) = \sin(a)\cos(b) - \cos(a)\sin(b)$ for all $a, b \in \mathbb{R}$. In addition, using the Taylor series expansion of $\sin(x)$ and $\cos(x)$, f can be written as follows:

$$f(x) = \cos(\frac{c\pi}{2})\sum_{k=0}^{\infty}\frac{(-1)^k(c\pi x)^{2k+1}}{(2k+1)!} - \sin(\frac{c\pi}{2})\sum_{k=0}^{\infty}\frac{(-1)^k(c\pi x)^{2k}}{(2k)!}.$$

Let f_k be the coefficient of x^k in the Taylor expansion of f. We define the following functions:

$$g(x) = \sum_{k=0}^{\infty}|f_k|x^k, \qquad a(x) = \sum_{k=0}^{\infty}\sqrt{|f_k|}x^k, \qquad b(x) = \sum_{k=0}^{\infty}\text{sign}(f_k)\sqrt{|f_k|}x^k.$$

Let a_k and b_k be the coefficients of x^k in the above series expansions of a and b. Recall that our goal is to show that there exist vectors $\{\tilde{y}_v\}_{v\in V}$ such that $\tilde{y}_u \cdot \tilde{y}_v = \tilde{A}_{u,v}$ for all $u, v \in V$. Given the set of vectors $\{y_v\}_{v\in V}$, we define the following vectors: if $u \in V_1$ then

$$y'_u = (a_0, a_1 y_u, a_2(y_u \otimes y_u), a_3(y_u \otimes y_u \otimes y_u), \dots)$$

and if $u \in V_2$ then

$$y'_u = (b_0, b_1 y_u, b_2(y_u \otimes y_u), b_3(y_u \otimes y_u \otimes y_u), \dots).$$

We note that the k-times tensor product $y_u \otimes y_u \otimes \cdots \otimes y_u$ are in fact n^k coordinates in the vector y'_u, as n is the dimension of the vector y_u. For every vector v and integer k, we denote by $v^{\otimes k}$ the k-times tensor product of v with itself. It is a known fact, that for every two vectors u, v and integer k, it holds that $u^{\otimes k} \cdot v^{\otimes k} = (u \cdot v)^k$.

Thus, we can see that if u, v are not both in V_1 or V_2, that is, not in the same side of the bipartite graph, then

$$y'_u \cdot y'_v = \sum_{k=0}^{\infty} a_k b_k (y_u^{\otimes k} \cdot y_v^{\otimes k}) = \sum_{k=0}^{\infty} a_k b_k (y_u \cdot y_v)^k = \sum_{k=0}^{\infty} f_k (y_u \cdot y_v)^k = f(y_u \cdot y_v).$$

Otherwise, if u and v are in the same side of the graph, without loss of generality, let it be V_1, then:

$$y'_u \cdot y'_v = \sum_{k=0}^{\infty} a_k^2 (y_u^{\otimes k} \cdot y_v^{\otimes k}) = \sum_{k=0}^{\infty} |f_k| (y_u \cdot y_v)^k = g(y_u \cdot y_v).$$

That is, it holds that $\tilde{A}_{i,j} = y'_i \cdot y'_j$ for all $i, j \in V$. Consequently, the matrix $\tilde{A}_{i,j} \in \mathbb{R}^{n \times n}$ is the symmetric Gramian matrix of the vectors $\{y'_v\}_{v \in V}$, and can be decomposed in polynomial time with Cholesky decomposition into $\tilde{A} = \tilde{Y}\tilde{Y}^T$. The rows of $\tilde{Y}$ will be the desired vectors $\{\tilde{y}_v\}_{v \in V}$.

To complete the proof we need to show that these vectors are unit vectors. That is, we want to show that $y'_u \cdot y'_u = 1$ for all $u \in V$. Since y_u is a unit vector, we have that

$$y'_u \cdot y'_u = \sum_{k=0}^{\infty} |f_k| (y_u \cdot y_v)^k = g(1).$$

Since $c \in (0,1)$, we have that $\cos(\frac{c\pi}{2}) \geq 0$ and $\sin(\frac{c\pi}{2}) \geq 0$. Therefore, we can write g in the form:

$$g(x) = \sum_{k=0}^{\infty} |a_k| x^k = \sinh(c\pi x) \cos(\frac{c\pi}{2}) + \cosh(c\pi x) \sin(\frac{c\pi}{2}).$$

Hence, we can choose c to be the solution of the equation

$$\sum_{k=0}^{\infty} |f_k| = \sinh(c\pi) \cos(\frac{c\pi}{2}) + \cosh(c\pi) \sin(\frac{c\pi}{2}) = 1$$

in the range $(0,1)$, and the vectors $\{\tilde{y}_v\}_{v \in V}$ will be unit vectors. The constant c turns out to be at least 0.19829. ◀

Finally, we are ready to complete the analysis of Algorithm 1, which is given in the following theorem.

▶ **Theorem 14.** *There exists a polynomial-time* 0.198*-approximation algorithm for the problem of* MaxCorr *on bipartite graphs that clusters the graph to at most 2 clusters.*

Proof of Theorem 14. From Corollary 12, Lemma 13 and Lemma 3, it follows that:

$$\begin{aligned}\mathbb{E}[X_{ALG}] &= \sum_{(u,v)\in E^+} w_{u,v} \left(1 - 2\frac{\arccos(\tilde{y}_u \cdot \tilde{y}_v)}{\pi}\right) + \sum_{(u,v)\in E^-} w_{u,v} \left(2\frac{\arccos(\tilde{y}_u \cdot \tilde{y}_v)}{\pi} - 1\right) \\ &= \sum_{(u,v)\in E^+} w_{u,v} c(2y_u \cdot y_v - 1) + \sum_{(u,v)\in E^-} w_{u,v} c(1 - 2y_u \cdot y_v) \\ &= c \cdot \mathrm{OPT}_{\mathrm{SDP}} \geq c \cdot \mathrm{OPT},\end{aligned}$$

where c is at least 0.19828. ◀

B Max-k-Corr: Adapting Algorithm 2

As previously mentioned, given k, we need to present the combination of clustering techniques that will be used to approximate Max-k-Corr. Let us start by doing exactly this. In what follows, it is important to note that given k: (1) the mixing probability p appearing below depends on k; and (2) the transformations f and g also depend on k.

First, when $k = 3$, with a probability of p we produce two clusters by a single random hyperplane and with the remaining probability of $1-p$ we produce three clusters as described in Algorithm 2 (use one random hyperplane to produce two clusters, choose uniformly at random one of the two clusters, and further partition it using the second random hyperplane). Second, when $k \geq 4$, with a probability of p we produce two clusters by a single random hyperplane and with the remaining probability of $1-p$ we produce four clusters by using two random hyperplanes.

Let focus now on the analysis, specifically, how the transformations f and g are chosen. First, when $k = 3$, the probability that two vertices u and v are in the same cluster equals:

$$p\left(1-\frac{\arccos(\tilde{y}_u\cdot\tilde{y}_v)}{\pi}\right)+(1-p)\left(\frac{1}{2}\left(1-\frac{\arccos(\tilde{y}_u\cdot\tilde{y}_v)}{\pi}\right)+\frac{1}{2}\left(1-\frac{\arccos(\tilde{y}_u\cdot\tilde{y}_v)}{\pi}\right)^2\right)$$
$$=\frac{1}{2}(1+p)\left(1-\frac{\arccos(\tilde{y}_u\cdot\tilde{y}_v)}{\pi}\right)+\frac{1}{2}(1-p)\left(1-\frac{\arccos(\tilde{y}_u\cdot\tilde{y}_v)}{\pi}\right)^2.$$

Thus, we wish to find the optimal parameters p and c that will satisfy that:

$$2\Pr[u,v \text{ are in the same cluster}]-1=c\left(\frac{4}{3}y_u\cdot y_v-\frac{1}{3}\right)$$

and the transformed vectors remain unit vectors. Hence, the function f is chosen to be:

$$f(x)=\cos\left(\frac{\pi\left(\sqrt{9\left(p^2-2p+5\right)-12c(p-1)(4x-1)}+3p-9\right)}{6(p-1)}\right),$$

and g is chosen accordingly. Performing the computation of the parameters, similarly to the previous sections of the paper, we obtain that $p \approx 0.47$ and $c \approx 0.397$.

Second, when $k \geq 4$, the probability that two vertices u and v are in the same cluster equals:

$$p\cdot\left(1-\frac{\arccos(\tilde{y}_u\cdot\tilde{y}_v)}{\pi}\right)+(1-p)\cdot\left(1-\frac{\arccos(\tilde{y}_u\cdot\tilde{y}_v)}{\pi}\right)^2.$$

Thus, following the same footsteps as before we can conclude that f is chosen to be:

$$f(x)=\cos\left(\frac{\pi\left(\sqrt{-\frac{2c(p-1)(2kx-k-2x+2)}{k}+p^2-2p+2}+p-2\right)}{2(p-1)}\right),$$

and g is chosen accordingly. For every $k \geq 4$ we can repeat the process and find the best parameters p and c. Summarizing, we present these parameters (optimized numerically) for several values of k (refer to Table 3).

Table 3 Approximation factors obtained by adapting Algorithm 1 and Algorithm 2 for Max-k-Corr.

k	3	4	5	6	10
p	0.47	0.655	0.648	0.595	0.541
approximation	0.397	0.348	0.32	0.309	0.285

We note that when k is large it is not clear why one should not use more than two hyperplanes and obtain a clustering that might contain a large number of clusters. An interesting phenomenon arises when we want to split the hypersphere to m slices. If $m \in (2^t, 2^{t+1})$ for some integer t, then we can sample t random hyperplanes and split the hypersphere into 2^t slices. Then, we randomly choose $m - 2^t$ of the slices, and split each one of them with a random hyperplane. The result is m slices, and the probability that two vertices will be in the same cluster equals:

$$\frac{2^{t+1} - m}{2^t} \cdot \left(1 - \frac{\arccos(\tilde{y}_u \cdot \tilde{y}_v)}{\pi}\right)^t + \frac{m - 2^t}{2^t} \cdot \left(1 - \frac{\arccos(\tilde{y}_u \cdot \tilde{y}_v)}{\pi}\right)^{t+1}.$$

That is, this probability is a convex combination of the probabilities that two vertices will be in the same cluster when we randomly chose t or $t+1$ hyperplanes. Since the analysis of the algorithm is edge-wise, we can simply consider a distribution $\{p_t\}_{t>0}, \sum_{t=1}^{\infty} p_t = 1$, where p_t is the probability that we split the hypersphere into 2^t slices using t random independent hyperplanes. However, numeric calculations of splitting into more than four clusters did not yield better approximation factors. Thus, for a fixed k, the behavior of the value of c is unimodal, and reaches a peak when $p_1 + p_2 = 1$. Therefore, we focus on on algorithms that pick one hyperplane with a probability of p and two hyperplanes with the remaining probability of $1 - p$.

C The Integrality Gap of (1)

In this section, we wish to discuss the integrality gap of the relaxation presented in (1). We consider the following simple example: a 4-cycle graph that has 3 plus edges and 1 minus edge. We denote it by $V = \{1, 2, 3, 4\}$ and $E^+ = \{(1, 2), (2, 3), (3, 4)\}, E^- = \{(1, 4)\}$. Clearly, this graph is bipartite and the value of the optimal solution is 2. However, the optimum of (1) is obtained when the vectors y_1, y_4 are orthogonal, and $y_1 \cdot y_2 = y_2 \cdot y_3 = y_3 \cdot y_4 = \sqrt{3}/2$. One possible solution will be $y_1 = (1, 0), y_2 = (\sqrt{3}/2, 1/2), y_3 = (1/2, \sqrt{3}/2), y_4 = (0, 1)$, and the value of this solution is $3\sqrt{3} - 2$. Therefore, (1) admits an integrality gap of at most $\frac{2}{3\sqrt{3}-2} \approx 0.62575$.

In addition, we can give better bound on the integrality gap for general graphs. We consider the simple 3-cycle graph, that has two plus edges, $(1, 2), (2, 3)$, and one minus edge, $(1, 3)$. One can see that the value of the optimal integral solution is 1, whereas the fractional optimum is obtained when $y_1 \cdot y_3 = 0$ and $y_1 \cdot y_2 = y_2 \cdot y_3 = \sqrt{2}/2$. Therefore, the integrality gap on general graphs is at most $\frac{1}{2\sqrt{2}-1} \approx 0.54691$.