Abstract

In Bipartite Correlation Clustering (BCC) we are given a complete bipartite graph $G$ with ‘+’ and ‘−’ edges, and we seek a vertex clustering that maximizes the number of agreements: the number of all ‘+’ edges within clusters plus all ‘−’ edges cut across clusters. BCC is known to be NP-hard [5].

We present a novel approximation algorithm for $k$-BCC, a variant of BCC with an upper bound $k$ on the number of clusters. Our algorithm outputs a $k$-clustering that provably achieves a number of agreements within a multiplicative $(1 - \delta)$-factor from the optimal, for any desired accuracy $\delta$. It relies on solving a combinatorially constrained bilinear maximization on the bi-adjacency matrix of $G$. It runs in time exponential in $k$ and $1/\delta$, but linear in the size of the input.

Further, we show that in the (unconstrained) BCC setting, an $(1 - \delta)$-approximation can be achieved by $O(\delta^{-1})$ clusters regardless of the size of the graph. In turn, our $k$-BCC algorithm implies an Efficient PTAS for the BCC objective of maximizing agreements.

1 Introduction

Correlation Clustering (CC) [6] considers the task of partitioning a set of objects into clusters based on their pairwise relationships. It arises naturally in several areas such as in network monitoring [1], document clustering [6] and textual similarity for data integration [9]. In its simplest form, objects are represented as vertices in a complete graph whose edges are labeled ‘+’ or ‘−’ to encode similarity or dissimilarity among vertices, respectively. The objective is to compute a vertex partition that maximizes the number of agreements with the underlying graph, i.e., the total number of ‘+’ edges in the interior of clusters plus the number of ‘−’ edges across clusters. The number of output clusters is itself an optimization variable—not part of the input. It may be meaningful, however, to restrict the number of output clusters to be at most $k$. The constrained version is known as $k$-CC and similarly to the unconstrained problem, it is NP-hard [6, 12, 14]. A significant volume of work has focused on approximately solving the problem of maximizing the number of agreements (MAXAGREE), or the equivalent—yet more challenging in terms of approximation—objective of minimizing the number of disagreements (MINDISAGREE) [6, 10, 8, 3, 18, 4].

Bipartite Correlation Clustering (BCC) Given a complete bipartite graph $G = (U, V, E)$ with edges labeled ‘+’ or ‘−’, the objective is once again to compute a clustering of the vertices that maximizes the number of agreements with the labeled pairs. BCC is a special case of the incomplete CC problem [8], where only a subset of vertices are connected and non-adjacent vertices do not affect the objective function. The output clusters may contain vertices from either one or both sides of the graph. Finally, we can define the $k$-BCC variant that enforces an upper bound on the number of output clusters, similar to $k$-CC for CC.

The task of clustering the vertices of a bipartite graph is common across many areas of machine learning. Applications include recommendation systems [17, 19], where analyzing the structure of a large set of pairwise interactions (e.g., among users and products) allows useful predictions about future interactions, gene expression data analysis [5, 15] and graph partitioning problems in data mining [11, 20].

Despite the practical interest in bipartite graphs, there is limited work on BCC and it is focused on the theoretically more challenging MINDISAGREE objective: [5] established an 11-approximation algorithm, while [2] achieved a 4-approximation, the currently best known guarantee. Algorithms for incomplete CC [10, 8, 16] can be applied to BCC, but they do not...
leverage the structure of the bipartite graph. Moreover, existing approaches for incomplete CC rely on LP or SDP solvers which scale poorly.

**Our contributions** We develop a novel approximation algorithm for $k$-BCC with provable guarantees for the MAXAGREE objective. Further, we show that under an appropriate configuration, our algorithm yields an Efficient Polynomial Time Approximation Scheme (EPTAS\(^1\)) for the unconstrained BCC problem. Our contributions can be summarized as follows:

1. **$k$-BCC**: Given a bipartite graph $G = (U, V, E)$, a parameter $k$, and any constant accuracy parameter $\delta \in (0, 1)$, our algorithm computes a clustering of $U \cup V$ into at most $k$ clusters and achieves a number of agreements that lies within a $(1 - \delta)$-factor from the optimal. It runs in time exponential in $k$ and $\delta^{-1}$, but linear in the size of $G$.

2. **BCC**: In the unconstrained BCC setting, the optimal number of clusters may be anywhere from 1 to $|U| + |V|$. We show that if one is willing to settle for a $(1 - \delta)$-approximation of the MAXAGREE objective, it suffices to use at most $O(\delta^{-2})$ clusters, regardless of the size of $G$. In turn, under an appropriate configuration, our $k$-BCC algorithm yields an EPTAS for the (unconstrained) BCC problem.

3. Our algorithm relies on formulating the $k$-BCC/ MAXAGREE problem as a combinatorially constrained bilinear maximization

\[
\max_{X \in \mathcal{X}, Y \in \mathcal{Y}} \text{Tr}(X^\top BY), \tag{1}
\]

where $B$ is the bi-adjacency matrix of $G$, and $\mathcal{X}$, $\mathcal{Y}$ are the sets of cluster assignment matrices for $U$ and $V$, respectively. In Sec. 3, we briefly describe our approach for approximately solving (1) and its guarantees under more general constraints.

We note that our $k$-BCC algorithm and its guarantees can be extended to *incomplete*, but dense BCC instances, where the input $G$ is *not* a complete bipartite graph, but $|E| = \Omega(|U| \cdot |V|)$ For simplicity, we restrict the description to the complete case. Finally, we supplement our theoretical findings with experimental results on synthetic and real datasets.

\(^1\)EPTAS refers to an algorithm that approximates the solution of an optimization problem within a multiplicative $(1 - \epsilon)$-factor, for any constant $\epsilon \in (0, 1)$, and has complexity that scales arbitrarily in $1/\epsilon$, but as a constant order polynomial (independent of $\epsilon$) in the input size $n$. EPTAS is more efficient than a PTAS; for example, a running time of $O(n^{1+\epsilon})$ is considered a PTAS, but not an EPTAS.

### 1.1 Related work

There is extensive literature on CC; see [7] for a list of references. Here, we focus on bipartite variants.

BCC was introduced by Amit in [5] along with an 11-approximation algorithm for the MinDisagree objective, based on a linear program (LP) with $O(|E|^3)$ constraints. In [2], Ailon et al. proposed two algorithms for the same objective: *i)* a deterministic 4-approximation based on an LP formulation and derandomization arguments by [18], and *ii)* a randomized 4-approximation combinatorial algorithm. The latter is computationally more efficient with complexity scaling linearly in the size of the input, compared to the LP that has $O((|V| + |U|)^3)$ constraints.

For the incomplete CC problem, which encompasses BCC as a special case, [10] provided an LP-based $O(\log n)$-approximation for MinDisagree. A similar result was achieved by [8]. For the MAXAGREE objective, [2, 16] provided an LP-based relaxation, similar to that for MAX k-CUT, and achieved a 0.7666-approximation. We are not aware of any results explicitly on the MAXAGREE objective for either BCC or $k$-BCC. For comparison, in the $k$-CC setting [10] provided a simple 3-approximation algorithm for $k = 2$, while for $k \geq 2$ [12] provided a PTAS for MAXAGREE and one for MinDisagree. [14] improved on the latter utilizing approximations schemes to the Gale-Berlekamp switching game. Table 1 summarizes the aforementioned results.

Finally, we note that our algorithm relies on adapting ideas from [7] for approximately solving a combinatorially constrained quadratic maximization.

### 2 $k$-BCC as a Bilinear Maximization

We describe the $k$-BCC problem and show that it can be formulated as a combinatorially constrained bilinear maximization on the bi-adjacency matrix of the input graph. Our $k$-BCC algorithm relies on approximately solving that bilinear maximization.

**Maximizing Agreements** An instance of $k$-BCC consists of an undirected, complete, bipartite graph $G = (U, V, E)$ whose edges have binary $\pm 1$ weights, and a parameter $k$. The objective, referred to as MAXAGREE$[k]$, is to compute a clustering of the vertices into at most $k$ clusters, such that the total number of positive edges in the interior of clusters plus the number of negative edges across clusters is maximized.

Let $E^+$ and $E^-$ denote the sets of positive and negative edges, respectively. Also, let $m = |U|$ and $n = |V|$. The bipartite graph $G$ can be represented by its weighted bi-adjacency matrix $B \in \{\pm 1\}^{m \times n}$, where
$B_{ij}$ is equal to the weight of edge $(i, j)$.

Consider a clustering $C$ of the vertices $U \cup V$ into at most $k$ clusters $C_1, \ldots, C_k$. Let $X \in \{0, 1\}^{m \times k}$ be the cluster assignment matrix for the vertices of $U$ associated with $C$, that is, $X_{ij} = 1$ if and only if vertex $i \in U$ is assigned to cluster $C_j$. Similarly, let $Y \in \{0, 1\}^{n \times k}$ be the cluster assignment matrix for $V$.

**Lemma 2.1.** For any instance $G = (U, V, E)$ of the $k$-BCC problem with bi-adjacency matrix $B \in \{\pm 1\}^{|U| \times |V|}$, and for any clustering $C$ of $U \cup V$ into $k$ clusters, the number of agreements achieved by $C$ is

$$\text{Agree}(C) = \text{Tr}(X^\top BY) + |E^-|,$$

where $X \in \{0, 1\}^{|U| \times k}$ and $Y \in \{0, 1\}^{|V| \times k}$ are the cluster assignment matrices corresponding to $C$.

**Proof.** Let $B^+$ and $B^- \in \{0, 1\}^{|U| \times |V|}$ be indicator matrices for $E^+$ and $E^-$. Then, $B = B^+ - B^-$. Given a clustering $C = \{C_j\}_{j=1}^k$, or equivalently the assignment matrices $X \in \{0, 1\}^{|U| \times k}$ and $Y \in \{0, 1\}^{|V| \times k}$, the number of pairs of similar vertices assigned to the same cluster is equal to $\text{Tr}(X^\top B^+ Y)$. Similarly, the number of pairs of dissimilar vertices assigned to different clusters is equal to $|E^-| - \text{Tr}(X^\top B^- Y)$. The total number of agreements achieved by $C$ is

$$\text{Agree}(C) = \text{Tr}(X^\top B^+ Y) + |E^-| - \text{Tr}(X^\top B^- Y) = \text{Tr}(X^\top BY) + |E^-|,$$

which is the desired result. \qed

It follows that computing a $k$-clustering that achieves the maximum number of agreements, boils down to a constrained bilinear maximization:

$\text{MaxAgree}[k] = \max_{X \in \mathcal{X}} \max_{Y \in \mathcal{Y}} \text{Tr}(X^\top BY) + |E^-|$, \hspace{1cm} (2)

where

$$\mathcal{X} \triangleq \{X \in \{0, 1\}^{m \times k} : \|X\|_{\infty, 1} = 1\},$$

$$\mathcal{Y} \triangleq \{Y \in \{0, 1\}^{n \times k} : \|Y\|_{\infty, 1} = 1\}. \hspace{1cm} (3)$$

Here, $\|X\|_{\infty, 1}$ denotes the maximum of the $\ell_1$-norm of the rows of $X$. Since $X \in \{0, 1\}^{|U| \times k}$, the constraint ensures that each row of $X$ has exactly one nonzero entry. Hence, $\mathcal{X}$ and $\mathcal{Y}$ describe the sets of valid cluster assignment matrices for $U$ and $V$, respectively.

In the sequel, we briefly describe our approach for approximately solving the maximization in (2), under more general constraints, and subsequently apply it to the $k$-BCC/MaxAgree problem.

## 3 Bilinear Maximization Framework

We describe a simple algorithm for computing an approximate solution to the constrained maximization

$$\max_{X \in \mathcal{X}, Y \in \mathcal{Y}} \text{Tr}(X^\top AY), \hspace{1cm} (4)$$

where the input argument $A$ is a real $m \times n$ matrix, and $\mathcal{X}$, $\mathcal{Y}$ are norm-bounded sets. Our approach is exponential in the rank of the argument $A$, which can be prohibitive in practice. To mitigate this effect, we solve the maximization on a low-rank approximation $\tilde{A}$ of $A$, instead. The quality of the output depends on the spectrum of $A$ and the rank $r$ of the surrogate matrix $\tilde{A}$. Alg. 1 outlines our approach for solving (4), operating directly on the rank-$r$ matrix $\tilde{A}$.

If we knew the optimal value for variable $Y$, then the optimal value for variable $X$ would be the solution to

$$P_{\mathcal{X}}(L) = \arg \max_{X \in \mathcal{X}} \text{Tr}(X^\top L), \hspace{1cm} (5)$$

\begin{table}[]
\centering
\begin{tabular}{|l|c|c|c|c|c|}
\hline
Ref. & Min./Max. & Guarantee & Complexity & D/R & Setting \\
\hline
\cite{1} & MaxAgree & (E)PTAS & $n/\delta \cdot k^O(\delta^{-2} \log k \log(1/\delta))$ & R & k-CC \\
\cite{4} & MinDisagree & PTAS & $n^O(1/\delta^2) \cdot \log n$ & R & k-CC \\
\cite{6} & MinDisagree & PTAS & $n^O(1/\delta^2) \cdot \log n$ & R & k-CC \\
\cite{10} & MinDisagree & $O(\log n)$-OPT & LP & D & Inc. CC \\
\cite{16} & MaxAgree & 0.766-OPT & SDP & D & Inc. CC \\
\cite{5} & MinDisagree & 11-OPT & LP & D & BCC \\
\cite{2} & MinDisagree & 4-OPT & LP & D & BCC \\
\cite{2} & MinDisagree & 4-OPT & $|E|$ & R & BCC \\
\hline
Ours & MaxAgree & (E)PTAS & $2^{O(\delta^{-3} \log \delta^{-3})} \cdot (\delta^{-2} + k) \cdot n + T_{svd}(\delta^{-2})$ & R & k-BCC \\
MaxAgree & (E)PTAS & $2^{O(\delta^{-3} \log \delta^{-3})} \cdot (\delta^{-2} \cdot n + T_{svd}(\delta^{-2})$ & R & BCC \\
\hline
\end{tabular}
\caption{Summary of results on BCC and related problems. For each scheme, we indicate the problem setting, the objective (MaxAgree/MinDisagree), the guarantees ($c$-OPT implies a multiplicative factor approximation), and its computational complexity ($n$ denotes the total number of vertices, LP/SDP denotes the complexity of a linear/semidefinite program, and $T_{svd}(r)$ the time required to compute a rank-$r$ truncated SVD of a $n \times n$ matrix). The D/R column indicates whether the scheme is deterministic or randomized.}
\end{table}
for $L = \tilde{A}Y$. Similarly, with $R = \tilde{A}^T X$ for a given $X$,
\[
P_Y(R) = \arg\max_{Y \in Y} \text{Tr}(R^T Y) \tag{6}
\]
is the optimal value of $Y$ for that $X$. Our algorithm requires that such a linear maximization oracles $P_X(\cdot)$ and $P_Y(\cdot)$ exist\footnote{In the case of the $k$-BCC problem \cite{BCC}, where $\mathcal{X}$ and $\mathcal{Y}$ correspond to the sets of cluster assignment matrices defined in \cite{BCC}, such optimization oracles exist (see Alg. 2).} Of course, the optimal value for either of the two variables is not known. It is known, however, that the columns of the $m \times k$ matrix $L = \tilde{A}Y$ lie in the $r$-dimensional range of $\tilde{A}$ for all feasible $Y \in Y$. Alg. \[1\] effectively operates by sampling candidate values for $L$. More specifically, it considers a large –exponential in $r \cdot k$– collection of $r \times k$ matrices $C$ whose columns are points of an $e$-net of the $\ell_2$-ball $B_{2}^{-1}$. Each matrix $C$ is used to generate a matrix $L$ whose $k$ columns lie in the range of the input matrix $\tilde{A}$ and in turn to produce a feasible solution pair $X, Y$ by successively solving (5) and (6). Due to the properties of the $e$-net, one of the computed feasible pairs is guaranteed to achieve an objective value in \[4\] close to the optimal one (for argument $\tilde{A}$).

**Lemma 3.2.** For any real $m \times n$, rank-$r$ matrix $\tilde{A}$, and sets $\mathcal{X} \subset \mathbb{R}^{m \times k}$ and $\mathcal{Y} \subset \mathbb{R}^{n \times k}$, let
\[
(\tilde{X}_*, \tilde{Y}_*) \triangleq \arg\max_{X \in \mathcal{X}, Y \in \mathcal{Y}} \text{Tr}(X^T \tilde{A}Y).
\]
If there exist linear maximization oracles $P_X$ and $P_Y$ as in \[5\] and \[6\], then Alg. \[1\] with input $\tilde{A}$ and accuracy $\epsilon \in (0, 1)$ outputs $X \in \mathcal{X}$ and $Y \in \mathcal{Y}$ such that
\[
\text{Tr}(X^T \tilde{A}Y) \geq \text{Tr}(\tilde{X}_* \tilde{A} \tilde{Y}_*) - 2e\sqrt{k} \cdot \|\tilde{A}\|_2 \cdot \mu_X \cdot \mu_Y,
\]
where $\mu_X \triangleq \max_{X \in \mathcal{X}} \|X\|_F$ and $\mu_Y \triangleq \max_{Y \in \mathcal{Y}} \|Y\|_F$, in time $O(2^\gamma (\sqrt{\epsilon}/r)^{k}: (T_X + T_Y + (m + n)r) + T_{\text{svd}}(r))$. Here, $T_{\text{svd}}(r)$ denotes the time required to compute the truncated SVD of $\tilde{A}$, while $T_X$ and $T_Y$ denote the running time of the two oracles.

The crux of Lemma 3.2 is that if there exists an efficient procedure to solve the simpler maximizations \[5\] and \[6\], then we can approximately solve the bilinear maximization \[4\] in time that depends exponentially on the intrinsic dimension $r$ of the input $\tilde{A}$, but polynomially on the size of the input. A formal proof is given in Appendix Sec. \[A\].

Recall that $\tilde{A}$ is only a low-rank approximation of the potentially full rank original argument $A$. The guarantees of Lemma 3.2 can be translated to guarantees for the original matrix introducing an additional error term to account for the extra level of approximation.

**Algorithm 1 BiLinearLowRankSolver**

\begin{algorithm}
\begin{algorithmic}
\STATE INPUT : $m \times n$ real matrix $\tilde{A}$ of rank $r$, $\epsilon \in (0, 1)$
\STATE OUTPUT $\tilde{X} \in \mathcal{X}$, $\tilde{Y} \in \mathcal{Y}$ (See Lemma 3.2)
\STATE \begin{algorithmic}
\STATE 1: $C \leftarrow \{\}$ \{Candidate solutions\}
\STATE 2: $U, \Sigma, V \leftarrow \text{svd}(\tilde{A})$ \{ $\Sigma \in \mathbb{R}^{r \times r}$ \}
\STATE 3: FOR each $C \in (\epsilon$-net of $B_{2}^{-1})^{r \times k}$ DO
\STATE 4: $L \leftarrow U \Sigma C$ \{$L \in \mathbb{R}^{m \times k}$\}
\STATE 5: $X \leftarrow P_X(L)$
\STATE 6: $R \leftarrow X^T \tilde{A}$ \{$R \in \mathbb{R}^{k \times n}$\}
\STATE 7: $Y \leftarrow P_Y(R)$
\STATE 8: $C \leftarrow C \cup \{(X, Y)\}$
\STATE END FOR
\STATE 10: $(\tilde{X}, \tilde{Y}) \leftarrow \arg\max_{(X, Y) \in C} \text{Tr}(X^T U \Sigma V^T Y)$
\STATE END FOR
\end{algorithmic}
\end{algorithm}

**Algorithm 2 $P_X(\cdot)$**

\begin{algorithm}
\begin{algorithmic}
\STATE INPUT : $m \times k$ real matrix $L$
\STATE OUTPUT $\tilde{X} = \arg\max_{X \in \mathcal{X}} \text{Tr}(X^T L)$
\STATE \begin{algorithmic}
\STATE 1: $\tilde{X} \leftarrow 0_{m \times k}$
\STATE 2: FOR $i = 1, \ldots, m$ DO
\STATE 3: $j_i \leftarrow \arg\max_{j \in [k]} L_{ij}$
\STATE 4: $\tilde{X}_{i,j_i} \leftarrow 1$
\STATE END FOR
\end{algorithmic}
\end{algorithm}
\end{algorithm}

**Lemma 3.3.** For any $A \in \mathbb{R}^{m \times n}$, let
\[
(\tilde{X}_*, \tilde{Y}_*) \triangleq \arg\max_{X \in \mathcal{X}, Y \in \mathcal{Y}} \text{Tr}(X^T AY),
\]
where $X$ and $Y$ satisfy the conditions of Lemma 3.2 Let $\tilde{A}$ be a rank-$r$ approximation of $A$, and $\tilde{X} \in \mathcal{X}$, $\tilde{Y} \in \mathcal{Y}$ be the output of Alg. \[1\] with input $\tilde{A}$ and accuracy $\epsilon$. Then,
\[
\text{Tr}(X^T AY) - \text{Tr}(\tilde{X}_* A \tilde{Y}_*) \\
\leq 2 \cdot (e\sqrt{K} \cdot \|A\|_2 + \|A - \tilde{A}\|_2) \cdot \mu_X \cdot \mu_Y.
\]

Lemma 3.3 follows from Lemma 3.2 A formal proof is deferred to Appendix Sec. \[A\] This concludes the brief discussion on our bilinear maximization framework.

## 4 Our $k$-BCC Algorithm

The $k$-BCC/MaxAgree problem on a bipartite graph $G = (U, V, E)$ can be written as a constrained bilinear maximization \[2\] on the bi-adjacency matrix $B$ over the sets of valid cluster assignment matrices \[3\] for $V$ and $U$. Adopting the framework of the previous section to approximately solve the maximization in \[2\] we obtain our $k$-BCC algorithm.

The missing ingredient is a pair of efficient procedures $P_X(\cdot)$ and $P_Y(\cdot)$, as described in Lemma 3.2 for solving \[5\] and \[6\], respectively, in the case where $\mathcal{X}$ and $\mathcal{Y}$ are the sets of valid cluster assignment matrices \[3\]. Such a procedure exists and is outlined in Alg. \[2\].
Lemma 4.4. For any \( L \in \mathbb{R}^{m \times k} \), Algorithm 3 outputs

\[
\arg \max_{X \in \mathcal{X}} \text{Tr}(X^\top L),
\]

where \( \mathcal{X} \) is defined in (3), in time \( O(k \cdot m) \).

A proof is provided in Appendix, Sec. B.

We provide an efficient polynomial time approximation scheme (PTAS) for BCC/\( \text{MAXAGREE} \), i.e., the unconstrained version with no upper bound on the number of output clusters. We rely on the following key observation: any constant factor approximation of the \( \text{MAXAGREE} \) objective, can be achieved by constant number of clusters. Hence, the desired approximation algorithm for BCC can be obtained by invoking the \( k \)-BCC algorithm under the appropriate configuration.

Recall that in the unconstrained BCC setting, the number of output clusters is an optimization variable; the optimal clustering may comprise any number of clusters, from a single cluster containing all vertices of

\[^{3}\] A similar result for CC/\( \text{MAXAGREE} \) exists in [6].
Bipartite Correlation Clustering – Maximizing Agreements

Algorithm 3 k-BCC via Low-rank Bilinear Maximization

input: • Bi-adjacency matrix $B \in \{\pm 1\}^{m,n}$ of bipartite graph $G = (U, V, E)$, • Target number of clusters $k$, • Approximation rank $r$, • Accuracy parameter $\epsilon \in (0, 1)$.

output Clustering $\tilde{C}^{(k)}$ of $U \cup V$ into $k$ clusters –

    Equivalently, cluster membership matrices $\tilde{X}^{(k)} \in \mathcal{X}$ and $\tilde{Y}^{(k)} \in \mathcal{Y}$ for $U$ and $V$, respectively.

1: $\tilde{B} \leftarrow \text{SVD}(B, r)$ \{Truncated SVD.\}
2: Approx. solve the bilinear maximization (2) on $\tilde{B}$, over the sets $\mathcal{X}$, $\mathcal{Y}$ of valid cluster assignment matrices (Eq. 3):

\[
\tilde{X}^{(k)}, \tilde{Y}^{(k)} \leftarrow \text{BiLinearLowRankSolver}(\tilde{B}, \epsilon, r) \quad \text{(Alg. 4 using Alg. 2 as P}_X(\cdot) \text{ and P}_Y(\cdot)).
\]

Algorithm 4 k-BCC/MAXAGREE

input: • Bi-adjacency matrix $B \in \{\pm 1\}^{m,n}$, • Target number of clusters $k$, • Accuracy $\delta \in (0, 1)$.

output Clustering $\tilde{C}^{(k)}$ of $U \cup V$ such that

\[\text{Agree}(\tilde{C}^{(k)}) \geq (1 - \delta) \cdot \text{Agree}(C^*)\].

1: Set up parameters:

\[\epsilon \leftarrow 2^{-\delta/2}, \quad r \leftarrow 2^\delta \cdot \delta^{-2} - 1\].
2: Return output of Alg. 5 for input $(B, k, r, \epsilon)$.

Algorithm 5 A PTAS for BCC/MAXAGREE

input: • Bi-adjacency matrix $B \in \{\pm 1\}^{m,n}$, • Accuracy $\delta \in (0, 1)$.

output Clustering $\tilde{C}$ of $U \cup V$ (into at most $2^3 \cdot 2^\delta \cdot \delta^{-1}$ clusters) such that

\[\text{Agree}(\tilde{C}) \geq (1 - \delta) \cdot \text{Agree}(C^*_\star)\]

1: Set up parameters:

\[k \leftarrow 2^3 \cdot 2^\delta \cdot \delta^{-1}, \quad \epsilon \leftarrow 2^{-6} \cdot \delta^2, \quad r \leftarrow 2^8 \cdot \delta^{-2} - 1\].
2: Return output of Alg. 5 for input $(B, k, r, \epsilon)$.

where $C^*_\star$ is an optimal clustering (with no constraint on the number of clusters), in time

\[2^{O(3/4 \cdot 3 \cdot \delta^{-2}) \cdot \delta 2 \cdot (m + n)} + T_{\text{svd}}(\delta^{-2})\].

The proof of Theorem 2 follows from the guarantees of Alg. 3 in (8) substituting the values of the parameters $k$, $r$, and $\epsilon$ with the values specified by Alg 5. The core k-BCC Alg. 5 returns a clustering $\tilde{C}$ with at most $k_0 = 2^3 \cdot 2^\delta \cdot \delta^{-1}$ clusters that satisfies

\[\text{Agree}(\tilde{C}) \geq \text{Agree}(C^{*(k_0)}) - \frac{\delta}{4} \cdot mn, \quad (11)\]

where $C^{*(k_0)}$ is the best among the clusterings using at most $k_0$ clusters. Also, for $k = k_0$, Lemma 5.5 implies that $\text{Agree}(C^{*(k_0)}) \geq \text{Agree}(C^*_{\star}) - \delta/4 \cdot mn$, where $C_{\star}$ is the optimal clustering without any constraint on the number of output clusters. Hence,

\[\text{Agree}(\tilde{C}) \geq \text{Agree}(C^*_{\star}) - \frac{\delta}{2} \cdot mn. \quad (12)\]

Continuing from (12), and taking into account Fact 1, we conclude that the output $\tilde{C}$ of Alg. 5 satisfies

\[\text{Agree}(\tilde{C}) \geq (1 - \delta) \cdot \text{Agree}(C^*_{\star}), \quad (13)\]

which is the desired result. Finally, the computational complexity of Alg. 5 follows from that of Alg. 3 substituting the parameter values in Lemma 2, taking into account the running time $T_X = O(\delta^{-2} \cdot m)$ of Alg. 3 used as subroutine $P_X(\cdot)$ and similarly $T_Y = O(\delta^{-2} \cdot n)$ for $P_Y(\cdot)$. This concludes the proof of Theorem 2.

For any desired constant accuracy $\delta \in (0, 1)$, Alg. 5 outputs a clustering that achieves a number of agreements within a $(1 - \delta)$-factor from the optimal, in time

the graph, up to $|U| + |V|$ singleton clusters. In principle, one could solve MAXAGREE[$k$] for all possible values of the parameter $k$ to identify the best clustering, but this approach is computationally intractable.

On the contrary, if one is willing to settle for an approximately optimal solution, then a constant number of clusters may suffice. More formally,

Lemma 5.5. For any BCC instance, and $0 < \epsilon \leq 1$, there exists a clustering $\tilde{C}$ with at most $k = 2 \cdot \epsilon^{-1} + 2$ clusters such that $\text{Agree}(\tilde{C}) \geq \text{Agree}(C^*_\star) - \epsilon \cdot nm$, where $C^*_\star$ denotes the optimal clustering.

We defer the proof to the end of the section.

In conjunction with Fact 1 and Lemma 5.5, it suggests to obtain a constant factor approximation for the unconstrained BCC/MAXAGREE problem, for any constant arbitrarily close to 1, it suffices to solve a k-BCC instance for a sufficiently large but constant number of clusters $k$. In particular, to obtain an $(1 - \delta)$-factor approximation for any constant $\delta \in (0, 1)$, it suffices to solve the k-BCC problem with an upper bound $k = O(\delta^{-1})$ on the number of clusters.

Alg. 5 outlines the approximation scheme for BCC. For a given accuracy parameter $\delta$, it invokes Alg. 2 under an appropriate configuration of the parameters $k$, $r$ and $\epsilon$, yielding the following guarantees:

Theorem 2. (PTAS for BCC.) For any instance $G = (U, V, E)$ of the BCC problem with bi-adjacency matrix $B$ and for any desired accuracy parameter $\delta \in (0, 1)$, Algorithm 3 computes a clustering $\tilde{C}$ of $U \cup V$ into (at most) $2^3 \cdot 2^\delta \cdot \delta^{-1}$ clusters, such that

\[\text{Agree}(\tilde{C}) \geq (1 - \delta) \cdot \text{Agree}(C^*_\star), \quad \text{(10)}\]

where $C^*_\star$ is the best among the clusterings using at most $k_0$ clusters. Also, for $k = k_0$, Lemma 5.5 implies that $\text{Agree}(C^{*(k_0)}) \geq \text{Agree}(C^*_{\star}) - \delta/4 \cdot mn$, where $C_{\star}$ is the optimal clustering without any constraint on the number of output clusters. Hence,

\[\text{Agree}(\tilde{C}) \geq \text{Agree}(C^*_{\star}) - \frac{\delta}{2} \cdot mn. \quad \text{(12)}\]

Continuing from (12), and taking into account Fact 1, we conclude that the output $\tilde{C}$ of Alg. 5 satisfies

\[\text{Agree}(\tilde{C}) \geq (1 - \delta) \cdot \text{Agree}(C^*_{\star}), \quad \text{(13)}\]

which is the desired result. Finally, the computational complexity of Alg. 5 follows from that of Alg. 3 substituting the parameter values in Lemma 3, taking into account the running time $T_X = O(\delta^{-2} \cdot m)$ of Alg. 3 used as subroutine $P_X(\cdot)$ and similarly $T_Y = O(\delta^{-2} \cdot n)$ for $P_Y(\cdot)$. This concludes the proof of Theorem 2.
that grows exponentially in $\delta^{-1}$, but linearly in the size $mn$ of the input. In other words, Alg. 5 is an EPTAS for BCC/MaxAgree.

We complete the section with a proof for Lemma 5.5 which stated that a constant number of clusters suffices to obtain an approximately optimal solution.

5.1 Proof of Lemma 5.5

It suffices to consider $\epsilon > 2/(m + n - 2)$ as the lemma holds trivially otherwise. Without loss of generality, we focus on clusterings whose all but at most two clusters contain vertices from both $U$ and $V$; one of the two remaining clusters can contain vertices only from $U$ and the other only from $V$. To verify that, consider an arbitrary clustering $C$ and let $C'$ be the clustering obtained by merging all clusters of $C$ containing only vertices of $U$ into a single cluster, and those containing only vertices of $V$ into another. The number of agreements is not affected, i.e., $\text{Agree}(C') = \text{Agree}(C)$.

We show that a clustering $C$ with the properties described in the lemma exists by modifying $C_*$. Let $S_U \subseteq C_*$ be the set of clusters that contain at most $\epsilon mn/2$ vertices of $U$, and $S_V \subseteq C_*$ those containing at most $\epsilon mn/2$ vertices of $V$. ($S_U$ and $S_V$ may not be disjoint). Finally, let $B$ be the set of vertices contained in clusters of $S_U \cup S_V$. We construct $C$ as follows. Clusters of $C_*$ not in $S_U \cup S_V$ are left intact. Each such cluster has size at least $\epsilon(n + m)/2$. Further, all vertices in $B$ are rearranged into two clusters: one for the vertices in $B \cap U$ and one those in $B \cap V$.

The above rearrangement can reduce the number of agreements by at most $\epsilon mn$. To verify that, consider a cluster $C$ in $S_U$; the cluster contains at most $\epsilon mn/2$ vertices of $U$. Let $i \triangleq |V \cap C|$. Splitting $C$ into two smaller clusters $C \cap U$ and $C \cap V$ can reduce the number of agreements by $\frac{1}{2} \epsilon m i$, i.e., the total number of edges in $C$. Note that agreements on $-$ edges are not affected by splitting a cluster. Performing the same operation on all clusters in $S_U$ incurs a total reduction

$$\frac{1}{2} \epsilon m \sum_{C \in S_U} |V \cap C| \leq \frac{1}{2} \epsilon m |V| = \frac{1}{2} \epsilon m n.$$

Repeating on $S_V$ incurs an additional cost of at most $\epsilon mn/2$ agreements, while merging all clusters containing only vertices from $U$ (similarly for $V$) into a single cluster does not reduce the agreements. We conclude that $\text{Agree}(C) \geq \text{Agree}(C_*) - \epsilon mn$.

Finally, by construction all but at most two clusters in $C$ contain at least $\frac{1}{2} \epsilon (n + m)$. In turn,

$$n + m = \sum_{C \in C} |C| \geq (|C| - 2) \cdot \frac{1}{2} \epsilon (n + m),$$

from which the desired result follows.

6 Experiments

We evaluate our algorithm on synthetic and real data and compare with PivotBiCluster [2] and the SDP-based approach of [16]. Although [5, 8, 10] provide algorithms applicable to BCC, those rely on LP or SDP formulations with a high number of constraints which imposes limitations in practice.

We run our core $k$-BCC algorithm (Alg. 3) to obtain a $k$-clustering. Disregarding the theoretical guarantees, we apply a threshold on the execution time which can only hurt the performance; equivalently, the iterative procedure of Alg. 5 is executed for an arbitrary subset of the points in the $\epsilon$-net (Alg. 1 step 3).

6.1 Synthetic Data

We generate synthetic BCC instances as follows. We arbitrarily set $m = 100$, $n = 50$, and number of clusters $k' = 5$, and construct a complete bipartite graph $G = (U, V, E)$ with $|U| = m$, $|V| = n$ and assign binary $\pm 1$ labels to the edges according to a random $k'$-clustering of the vertices. Then, we modify the sign of each label independently with probability $p$.

We consider multiple values of $p$ in the range $[0, 0.5]$. For each value, we generate 10 random instances as described above and compute a vertex clustering using all algorithms. PivotBiCluster returns a clustering of arbitrary size; no parameter $k$ is specified and the algorithm determines the number of clusters in an attempt to maximize the number of agreements. The majority of the output clusters are singletons which can in principle be merged into two clusters (see Subsec. 5.1). The algorithm is substantially faster that the other approaches on examples of this scale. Hence, we run PivotBiCluster with 3-10$^4$ random restarts on each instance and depict best results to allow for comparable running times. Contrary to PivotBiCluster, for our algorithm, which is referred to as BccBilinear, we need to specify the target number $k$ of clusters. We run it for $k = 5, 10$ and 15 and arbitrarily set the parameter $r = 5$ and terminate our algorithm after 10$^4$ random samples/rounds. Finally, the SDP-based approach returns 4 clusters by construction, and is configure to output the best results among 100 random pivoting steps. Fig. 1 depicts the number of agreements achieved by each algorithm for each value of $p$, the number of output clusters, and the execution times. All numbers are averages over multiple random instances.

We note that in some cases and contrary to intuition, our algorithm performs better for lower values of $k$, the target number of clusters. We attribute this phenomenon to the fact that our algorithm initially (in Alg. 1) aims to maximize the number of agreements. The majority of the output clusters are singletons which can in principle be merged into two clusters (see Subsec. 5.1). The algorithm is substantially faster than the other approaches on examples of this scale. Hence, we run PivotBiCluster with 3-10$^4$ random restarts on each instance and depict best results to allow for comparable running times. Contrary to PivotBiCluster, for our algorithm, which is referred to as BccBilinear, we need to specify the target number $k$ of clusters. We run it for $k = 5, 10$ and 15 and arbitrarily set the parameter $r = 5$ and terminate our algorithm after 10$^4$ random samples/rounds. Finally, the SDP-based approach returns 4 clusters by construction, and is configured to output the best results among 100 random pivoting steps. Fig. 1 depicts the number of agreements achieved by each algorithm for each value of $p$, the number of output clusters, and the execution times. All numbers are averages over multiple random instances.

We note that in some cases and contrary to intuition, our algorithm performs better for lower values of $k$, the target number of clusters. We attribute this phenomenon to the fact that our algorithm initially (in Alg. 1) aims to maximize the number of agreements. The majority of the output clusters are singletons which can in principle be merged into two clusters (see Subsec. 5.1). The algorithm is substantially faster than the other approaches on examples of this scale. Hence, we run PivotBiCluster with 3-10$^4$ random restarts on each instance and depict best results to allow for comparable running times. Contrary to PivotBiCluster, for our algorithm, which is referred to as BccBilinear, we need to specify the target number $k$ of clusters. We run it for $k = 5, 10$ and 15 and arbitrarily set the parameter $r = 5$ and terminate our algorithm after 10$^4$ random samples/rounds. Finally, the SDP-based approach returns 4 clusters by construction, and is configured to output the best results among 100 random pivoting steps. Fig. 1 depicts the number of agreements achieved by each algorithm for each value of $p$, the number of output clusters, and the execution times. All numbers are averages over multiple random instances.

We note that in some cases and contrary to intuition, our algorithm performs better for lower values of $k$, the target number of clusters. We attribute this phenomenon to the fact that our algorithm initially (in Alg. 1) aims to maximize the number of agreements.

4 All algorithms are prototypically implemented in Matlab. [16] was implemented using CVX.
Bipartite Correlation Clustering – Maximizing Agreements

Figure 1: Synthetic data. We generate an $m \times n$ bipartite graph with edge weights $\pm 1$ according to a random vertex $k$-clustering and subsequently flip the sign of each edge with probability $p$. We plot the average number of agreements achieved by each method over 10 random instances for each $p$ value. The bar plot depicts the average number of output clusters for each scheme $p$-value pair. Within each bar, a horizontal line marks the number of singleton clusters.

nomenon to the fact that for higher values of $k$, we should typically use higher approximation rank and consider larger number of samples.

6.2 MovieLens Dataset

The MovieLens datasets [13] are sets of movie ratings; each of $m$ users assigns scores in $\{1, \ldots, 5\}$ to a small subset of the $n$ movies. Table 2 lists the dimensions of the datasets. From each dataset, we generate an instance of the (incomplete) BCC problem as follows: we construct the bipartite graph on the user-movie pairs using the ratings as edge weights, compute the average weight and finally set weights higher than the average to $+1$ and the others to $-1$.

We run our algorithm to obtain a clustering of the vertices. For a reference, we compare to PivotBiCluster with 50 random restarts. Note, however, that PivotBiCluster is not designed for the incomplete BCC problem; to apply the algorithm, we effectively treat missing edges as edges of negative weight. Finally, the SDP approach of [16], albeit suitable for the incomplete CC problem, does not scale to this size of input. Table 3 lists the number of agreements achieved by each method on each one of the three datasets and the corresponding execution times.

7 Conclusions

We presented the first algorithm with provable approximation guarantees for $k$-BCC/MaxAgree. Our approach relied on formulating $k$-BCC as a constrained bilinear maximization over the sets of cluster assignment matrices and developing a simple framework to approximately solve that combinatorial optimization.

In the unconstrained BCC setting, with no bound on the number of output clusters, we showed that any constant multiplicative factor approximation for the MaxAgree objective can be achieved using a constant number of clusters. In turn, under the appropriate configuration, our $k$-BCC algorithm yields an Efficient PTAS for BCC/MaxAgree.

Acknowledgments This research has been supported by NSF Grants CCF 1344179, 1344364, 1407278, 1422549 and ARO YIP W911NF-14-1-0258. DP is supported by NSF awards CCF-1217058 and CCF-1116404 and MURI AFOSR grant 556016.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>$m$ (Users)</th>
<th>$n$ (Movies)</th>
<th>Ratings</th>
</tr>
</thead>
<tbody>
<tr>
<td>MovieLens100K</td>
<td>1000</td>
<td>1700</td>
<td>$10^5$</td>
</tr>
<tr>
<td>MovieLens1M</td>
<td>6000</td>
<td>4000</td>
<td>$10^6$</td>
</tr>
<tr>
<td>MovieLens10M</td>
<td>72000</td>
<td>10000</td>
<td>$10^7$</td>
</tr>
</tbody>
</table>

Table 2: Summary of MovieLens datasets [13].

<table>
<thead>
<tr>
<th>Dataset</th>
<th>$m$ (Users)</th>
<th>$n$ (Movies)</th>
<th>Ratings</th>
</tr>
</thead>
<tbody>
<tr>
<td>MovieLens</td>
<td>100K</td>
<td>1M</td>
<td>10M</td>
</tr>
<tr>
<td>PivotBiCluster</td>
<td>46134</td>
<td>429277</td>
<td>5008577</td>
</tr>
<tr>
<td></td>
<td>(27.95)</td>
<td>(651.13)</td>
<td>(1.5 $\cdot 10^6$)</td>
</tr>
<tr>
<td>BccBilinear</td>
<td>68141</td>
<td>694366</td>
<td>6857509</td>
</tr>
<tr>
<td></td>
<td>(6.65)</td>
<td>(19.50)</td>
<td>(1.2 $\cdot 10^6$)</td>
</tr>
</tbody>
</table>

Table 3: Number of agreements achieved by the two algorithms on incomplete $(k)$-BCC instances obtained from the MovieLens datasets [13]. For PivotBiCluster we present best results over 50 random restarts. Our algorithm was arbitrarily configured with $r = 4$, $k = 10$ and a limit of $10^4$ samples (iterations). We also note in parentheses the runtimes in seconds.
References


A Bilinear Maximization Guarantees

Lemma 3.2. For any real $m \times n$, rank-$r$ matrix $\tilde{A}$ and arbitrary norm-bounded sets $\mathcal{X} \subset \mathbb{R}^{m \times k}$ and $\mathcal{Y} \subset \mathbb{R}^{n \times k}$, let

$$(\tilde{X}_*, \tilde{Y}_*) \triangleq \arg\max_{X \in \mathcal{X}, Y \in \mathcal{Y}} \text{Tr}(X^\top \tilde{A} Y).$$

If there exist operators $P_X : \mathbb{R}^{m \times k} \to \mathcal{X}$ such that

$$P_X(L) = \arg\max_{X \in \mathcal{X}} \text{Tr}(X^\top L)$$

and similarly, $P_Y : \mathbb{R}^{n \times k} \to \mathcal{Y}$ such that

$$P_Y(R) = \arg\max_{Y \in \mathcal{Y}} \text{Tr}(R^\top Y)$$

with running times $T_X$ and $T_Y$, respectively, then Algorithm 1 outputs $\tilde{X} \in \mathcal{X}$ and $\tilde{Y} \in \mathcal{Y}$ such that

$$\text{Tr}(\tilde{X}^\top \tilde{A} \tilde{Y}) \geq \text{Tr}(X_*^\top \tilde{A} Y_*) - 2c\sqrt{k} \cdot \|\tilde{A}\|_2 \cdot \mu_X \cdot \mu_Y,$$

where $\mu_X \triangleq \max_{X \in \mathcal{X}} \|X\|_F$ and $\mu_Y \triangleq \max_{Y \in \mathcal{Y}} \|Y\|_F$, in time $O\left(\left(2\sqrt{r}/\epsilon\right)^{r^2} \cdot (T_X + T_Y + (m+n)r)\right) + T_{soa}(r)$.

Proof. In the sequel, $\tilde{U}$, $\tilde{\Sigma}$ and $\tilde{V}$ are used to denote the $r$-truncated singular value decomposition of $\tilde{A}$.

Without loss of generality, we assume that $\mu_X = \mu_Y = 1$ since the variables in $\mathcal{X}$ and $\mathcal{Y}$ can be normalized by $\mu_X$ and $\mu_Y$, respectively, while simultaneously scaling the singular values of $\tilde{A}$ by a factor of $\mu_X \cdot \mu_Y$. Then, $\|Y\|_{\infty,2} \leq 1, \forall Y \in \mathcal{Y}$, where $\|Y\|_{\infty,2}$ denotes the maximum of the $\ell_2$-norm of the columns of $Y$.

Let $\tilde{X}_*, \tilde{Y}_*$ be the optimal pair on $\tilde{A}$, i.e.,

$$(\tilde{X}_*, \tilde{Y}_*) \triangleq \arg\max_{X \in \mathcal{X}, Y \in \mathcal{Y}} \text{Tr}(X^\top \tilde{A} Y)$$

and define the $r \times k$ matrix $\tilde{C} \triangleq \tilde{V}^\top \tilde{Y}_*$. Note that

$$\|\tilde{C}\|_{\infty,2} = \|\tilde{V}^\top \tilde{Y}_*\|_{\infty,2} = \max_{1 \leq i \leq k} \|\tilde{V}^\top [\tilde{Y}_*]_{.,i}\|_2 \leq 1,$$

(14)

with the last inequality following from the facts that $\|Y\|_{\infty,2} \leq 1 \forall Y \in \mathcal{Y}$ and the columns of $\tilde{V}$ are orthonormal. Alg. 1 iterates over the points in $(\mathbb{B}^{n-1}_2)^\otimes k$. The latter is used to describe the set of $r \times k$ matrices whose columns have $\ell_2$ norm at most equal to 1. At each point, the algorithm computes a candidate solution. By (14), the $\epsilon$-net contains an $r \times k$ matrix $C_* \triangleq \tilde{V}^\top \tilde{Y}_*$ such that

$$\|C_\epsilon - \tilde{C}\|_{\infty,2} \leq \epsilon.$$

Let $X_\epsilon, Y_\epsilon$ be the candidate pair computed at $C_\epsilon$ by the two step maximization, i.e.,

$$X_\epsilon \triangleq \arg\max_{X \in \mathcal{X}} \text{Tr}(X^\top \tilde{U} \tilde{\Sigma} C_\epsilon)$$

and

$$Y_\epsilon \triangleq \arg\max_{Y \in \mathcal{Y}} \text{Tr}(X_\epsilon^\top \tilde{A} Y).$$

(15)

We show that the objective values achieved by the candidate pair $X_\epsilon, Y_\epsilon$ satisfies the inequality of the lemma implying the desired result.

By the definition of $\tilde{C}$, and the linearity of the trace,

$$\text{Tr}(\tilde{X}_*^\top \tilde{A} \tilde{Y}_*) = \text{Tr}(\tilde{X}_*^\top \tilde{U} \tilde{\Sigma} \tilde{C}_*)$$

$$\leq \text{Tr}(\tilde{X}_*^\top \tilde{U} \tilde{\Sigma} (\tilde{C}_* - C_*)) \leq \text{Tr}(\tilde{X}_*^\top \tilde{U} \tilde{\Sigma} (C_\epsilon - C_*))$$

(16)

The inequality follows from the fact that (by definition (15)) $X_\epsilon$ maximizes the first term over all $X \in \mathcal{X}$. We compute an upper bound on the right hand side of (16). Define

$$\hat{Y} \triangleq \text{arg\ min}_{Y \in \mathcal{Y}} \|\tilde{V}^\top Y - C_*\|_{\infty,2}.$$

(We note that $\hat{Y}$ is used for the analysis and is not explicitly calculated.) Further, define the $r \times k$ matrix $\hat{C} \equiv \tilde{V}^\top \hat{Y}$. By the linearity of the trace operator

$$\text{Tr}(X_\epsilon^\top \tilde{U} \tilde{\Sigma} C_* )$$

$$\leq \text{Tr}(X_\epsilon^\top \tilde{U} \tilde{\Sigma} \hat{C} ) + \text{Tr}(X_\epsilon^\top \tilde{U} \tilde{\Sigma} (C_* - \hat{C} ))$$

$$\leq \text{Tr}(X_\epsilon^\top \tilde{U} \tilde{\Sigma} (C_\epsilon - \hat{C} )) + \text{Tr}(X_\epsilon^\top \tilde{U} \tilde{\Sigma} (\hat{C} - C_* ))$$

(17)

The inequality follows from the fact that (by definition (15)) $Y_\epsilon$ maximizes the first term over all $Y \in \mathcal{Y}$. Combining (17) and (16), and rearranging the terms, we obtain

$$\text{Tr}(\tilde{X}_*^\top \tilde{A} \tilde{Y}_*) - \text{Tr}(X_\epsilon^\top \tilde{A} Y_\epsilon)$$

$$\leq \text{Tr}(\tilde{X}_*^\top \tilde{U} \tilde{\Sigma} (C_* - C_\epsilon)) + \text{Tr}(X_\epsilon^\top \tilde{U} \tilde{\Sigma} (C_\epsilon - \hat{C} ))$$

(18)

By Lemma C.10

$$|\text{Tr}(\tilde{X}_*^\top \tilde{U} \tilde{\Sigma} (\tilde{C}_* - C_\epsilon ))|$$

$$\leq \|\tilde{X}_*\|_F \cdot \|\tilde{\Sigma}\|_2 \cdot \|\tilde{C}_* - C_\epsilon\|_F$$

$$\leq \|\tilde{X}_*\|_F \cdot \sigma_1(\tilde{A}) \cdot \sqrt{k} \cdot \epsilon$$

$$\leq \max_{X \in \mathcal{C}} \|X\|_F \cdot \sigma_1(\tilde{A}) \cdot \sqrt{k} \cdot \epsilon$$

(19)
Similarly,
\[
\left| \text{Tr}(X_x^T U \Sigma (C_x - \hat{C})) \right| \leq \|X_x U\|_F \cdot \|\Sigma\|_2 \cdot \|C_x - \hat{C}\|_F \\
\leq \max_{X \in \mathcal{X}} \|X\|_F \cdot \sigma_1(\hat{A}) \cdot \sqrt{k} \cdot \epsilon \\
\leq \sigma_1(\hat{A}) \cdot \sqrt{k} \cdot \epsilon. \quad (20)
\]

The second inequality follows from the fact that by the definition of \( \hat{C} \),
\[
\|\hat{C} - C_x\|_{\infty,2} = \|\hat{V}^\top \hat{Y} - C_x\|_{\infty,2} \leq \|\hat{V}^\top \hat{Y} - C_x\|_{\infty,2} \\
= \|\hat{C}_x - C_x\|_{\infty,2} \leq \epsilon,
\]
which implies that
\[
\|\hat{C} - C_x\|_F \leq \sqrt{k} \cdot \epsilon.
\]
Continuing from (18) under (19) and (20),
\[
\text{Tr}(X_x^T A Y_x) \geq \text{Tr}(\hat{X}_x^T \hat{A} Y_x) - 2 \cdot \epsilon \cdot \sqrt{k} \cdot \sigma_1(A).
\]
Recalling that the singular values of \( \hat{A} \) have been scaled by a factor of \( \mu \cdot \mu_y \) yields the desired result.

The runtime of Alg. 1 follows from the cost per iteration and the cardinality of the \( \epsilon \)-net. Matrix multiplications can exploit the truncated singular value decomposition of \( \hat{A} \) which is performed only once. \( \square \)

**Lemma A.6.** For any \( A, \hat{A} \in \mathbb{R}^{m \times n} \), and normal bounded sets \( \mathcal{X} \subseteq \mathbb{R}^{m \times k} \) and \( \mathcal{Y} \subseteq \mathbb{R}^{n \times k} \), let
\[
(X_*, Y_*) \triangleq \arg\max_{X \in \mathcal{X}, Y \in \mathcal{Y}} \text{Tr}(X^T A Y),
\]
and
\[
(\hat{X}_*, \hat{Y}_*) \triangleq \arg\max_{X \in \hat{\mathcal{X}}, Y \in \hat{\mathcal{Y}}} \text{Tr}(X^T \hat{A} Y).
\]
For any \((\tilde{X}, \tilde{Y}) \in \mathcal{X} \times \mathcal{Y}\) such that
\[
\text{Tr}(\tilde{X}^T \hat{A} Y) \geq \gamma \cdot \text{Tr}(\tilde{X}_*^T \hat{A} Y_*) - C
\]
for some \(0 < \gamma \leq 1\), we have
\[
\text{Tr}(\tilde{X}^T \hat{A} Y) \geq \gamma \cdot \text{Tr}(\tilde{X}_*^T \hat{A} Y_*) - C \\
- 2 \cdot \|A - \hat{A}\|_2 \cdot \mu_x \cdot \mu_y,
\]
where \( \mu_x \triangleq \max_{X \in \mathcal{X}} \|X\|_F \) and \( \mu_y \triangleq \max_{Y \in \mathcal{Y}} \|Y\|_F \).

**Proof.** By the optimality of \( \tilde{X}_*, \tilde{Y}_* \), for \( \hat{A} \), we have
\[
\text{Tr}(\tilde{X}_*^T \hat{A} Y_*) \geq \text{Tr}(\hat{X}_*^T \hat{A} Y_*).
\]
In turn, for any \((\tilde{X}, \tilde{Y}) \in \mathcal{X} \times \mathcal{Y}\) such that
\[
\text{Tr}(\tilde{X}^T \hat{A} Y) \geq \gamma \cdot \text{Tr}(\tilde{X}_*^T \hat{A} Y_*) - C
\]
for some \(0 < \gamma < 1\) (if such pairs exist), we have
\[
\text{Tr}(\tilde{X}^T \hat{A} Y) \geq \gamma \cdot \text{Tr}(\tilde{X}_*^T \hat{A} Y_*) - C. \quad (21)
\]
By the linearity of the trace operator,
\[
\text{Tr}(\tilde{X}^T \hat{A} Y) \\
= \text{Tr}(\tilde{X}^T A Y) - \text{Tr}(\tilde{X}^T (A - \hat{A}) Y) \\
\leq \text{Tr}(\tilde{X}^T A Y) + |\text{Tr}(\tilde{X}^T (A - \hat{A}) Y)|. \quad (22)
\]
By Lemma C.10
\[
|\text{Tr}(\tilde{X}^T (A - \hat{A}) Y)| \\
\leq \|\tilde{X}\|_F \cdot \|\tilde{Y}\|_F \cdot \|A - \hat{A}\|_2 \\
\leq \|A - \hat{A}\|_2 \cdot \max_{X \in \mathcal{X}} \|X\|_F \cdot \max_{Y \in \mathcal{Y}} \|Y\|_F \triangleq R. \quad (23)
\]
Continuing from (22),
\[
\text{Tr}(\tilde{X}^T A Y) \leq \text{Tr}(\tilde{X}^T A \hat{Y}) + R. \quad (24)
\]
Similarly,
\[
\text{Tr}(\tilde{X}^T \hat{A} Y_*) \\
= \text{Tr}(\tilde{X}^T A Y_*) - \text{Tr}(\tilde{X}^T (A - \hat{A}) Y_*) \\
\geq \text{Tr}(\tilde{X}^T A Y_*) - |\text{Tr}(\tilde{X}^T (A - \hat{A}) Y_*)| \\
\geq \text{Tr}(\tilde{X}^T A Y_*) - R. \quad (25)
\]
Combining the above, we have
\[
\text{Tr}(\tilde{X}^T A Y) \geq \text{Tr}(\tilde{X}^T A \hat{Y}) - R \\
\geq \gamma \cdot \text{Tr}(\tilde{X}_*^T \hat{A} Y_*) - R - C \\
\geq \gamma \cdot (\text{Tr}(\tilde{X}_*^T \hat{A} Y_*) - R) - R - C \\
= \gamma \cdot \text{Tr}(\tilde{X}_*^T \hat{A} Y_*) - (1 + \gamma) \cdot R - C \\
\geq \gamma \cdot \text{Tr}(\tilde{X}_*^T \hat{A} Y_*) - 2 \cdot R - C,
\]
where the first inequality follows from (24) the second from (21), the third from (25), and the last from the fact that \( R \geq 0 \). This concludes the proof. \( \square \)

**Lemma 3.3.** For any \( A \in \mathbb{R}^{m \times n} \), let
\[
(X_*, Y_*) \triangleq \arg\max_{X \in \mathcal{X}, Y \in \mathcal{Y}} \text{Tr}(X^T A Y),
\]
where \( \mathcal{X} \subseteq \mathbb{R}^{m \times k} \) and \( \mathcal{Y} \subseteq \mathbb{R}^{n \times k} \) are sets satisfying the conditions of Lemma 3.2. Let \( \bar{A} \) be a rank-\( r \) approximation of \( A \), and \( \tilde{X} \in \mathcal{X}, \tilde{Y} \in \mathcal{Y} \) be the output of Alg. 1 with input \( A \) and accuracy \( \epsilon \). Then,
\[
\text{Tr}(\tilde{X}^T A Y_*) - \text{Tr}(\bar{X}^T \bar{A} \tilde{Y}) \\
\leq 2 \cdot (\epsilon \sqrt{k} \cdot \|A\|_2 + \|A - \bar{A}\|_2) \cdot \mu_x \cdot \mu_y,
\]
where \( \mu_x \triangleq \max_{X \in \mathcal{X}} \|X\|_F \) and \( \mu_y \triangleq \max_{Y \in \mathcal{Y}} \|Y\|_F \).

**Proof.** The proof follows the approximation guarantees of Alg. 1 in Lemma 3.2 and Lemma A.6 \( \square \)
B Correctness of Algorithm 2

In the sequel, we use \( \| \mathbf{X} \|_{\infty,1} \) to denote the maximum of the \( \ell_1 \) norm of the rows of \( \mathbf{X} \). When \( \mathbf{X} \in \{0,1\}^{d \times k} \), the constraint \( \| \mathbf{X} \|_{\infty,1} = 1 \) effectively implies that each row of \( \mathbf{X} \) has exactly one nonzero entry.

**Lemma 4.4.** Let \( \mathcal{X} \triangleq \{ \mathbf{X} \in \{0,1\}^{d \times k} : \| \mathbf{X} \|_{\infty,1} = 1 \} \). For any \( d \times k \) real matrix \( \mathbf{L} \), Algorithm 2 outputs

\[
\tilde{\mathbf{X}} = \arg \max_{\mathbf{X} \in \mathcal{X}} \text{Tr}(\mathbf{X}^\top \mathbf{L}),
\]

in time \( O(k \cdot d) \)

**Proof.** By construction, each row of \( \mathbf{X} \) has exactly one nonzero entry. Let \( j_i \in [k] \) denote the index of the nonzero entry in the \( i \)-th row of \( \mathbf{X} \). For any \( \mathbf{X} \in \mathcal{X} \),

\[
\text{Tr}(\mathbf{X}^\top \mathbf{L}) = \frac{1}{k} \sum_{j=1}^{k} x_j^\top \mathbf{L} = \frac{1}{k} \sum_{j=1}^{k} \sum_{i \in \text{supp}(x_j)} 1 \cdot L_{ij}
\]

\[
= \frac{d}{k} \sum_{i=1}^{d} L_{ij} \leq \frac{d}{k} \max_{i \in [k]} L_{ij}.
\]

(26)

Algorithm 2 achieves equality in (26) due to the choice of \( j_i \) in line 3. Finally, the running time follows immediately from the \( O(k) \) time required to determine the maximum entry of each of the \( d \) rows of \( \mathbf{L} \).

**C Auxiliary Lemmas**

**Lemma C.7.** Let \( a_1, \ldots, a_n \) and \( b_1, \ldots, b_n \) be \( 2n \) real numbers and let \( p \) and \( q \) be two numbers such that \( 1/p + 1/q = 1 \) and \( p > 1 \). We have

\[
\sum_{i=1}^{n} a_i b_i \leq \left( \sum_{i=1}^{n} |a_i|^p \right)^{1/p} \cdot \left( \sum_{i=1}^{n} |b_i|^q \right)^{1/q}.
\]

**Lemma C.8.** For any \( \mathbf{A}, \mathbf{B} \in \mathbb{R}^{n \times k} \),

\[
|\langle \mathbf{A}, \mathbf{B} \rangle| \triangleq |\text{Tr}(\mathbf{A}^\top \mathbf{B})| \leq \| \mathbf{A} \|_F \cdot \| \mathbf{B} \|_F.
\]

**Proof.** Treating \( \mathbf{A} \) and \( \mathbf{B} \) as vectors, the lemma follows immediately from Lemma C.7 for \( p = q = 2 \).

**Lemma C.9.** For any two real matrices \( \mathbf{A} \) and \( \mathbf{B} \) of appropriate dimensions,

\[
\| \mathbf{A} \mathbf{B} \|_F \leq \min \{ \| \mathbf{A} \|_2 \| \mathbf{B} \|_F, \| \mathbf{A} \|_F \| \mathbf{B} \|_2 \}.
\]

**Proof.** Let \( \mathbf{b}_i \) denote the \( i \)-th column of \( \mathbf{B} \). Then,

\[
\| \mathbf{A} \mathbf{B} \|_F^2 = \sum_{i} \| \mathbf{A} \mathbf{b}_i \|_2^2 \leq \sum_{i} \| \mathbf{A} \|_2^2 \| \mathbf{b}_i \|_2^2
\]

\[
= \| \mathbf{A} \|_2^2 \sum_{i} \| \mathbf{b}_i \|_2^2 = \| \mathbf{A} \|_2^2 \| \mathbf{B} \|_F^2.
\]

Similarly, using the previous inequality,

\[
\| \mathbf{A} \mathbf{B} \|_F^2 = \| \mathbf{B}^\top \mathbf{A}^\top \|_F^2 \leq \| \mathbf{B}^\top \|_2^2 \| \mathbf{A}^\top \|_2^2 = \| \mathbf{B} \|_2^2 \| \mathbf{A} \|_F^2.
\]

The desired result follows combining the two upper bounds.

**Lemma C.10.** For any real \( m \times k \) matrix \( \mathbf{X} \), \( m \times n \) matrix \( \mathbf{A} \), and \( n \times k \) matrix \( \mathbf{Y} \),

\[
|\text{Tr}(\mathbf{X}^\top \mathbf{A} \mathbf{Y})| \leq \| \mathbf{X} \|_F \cdot \| \mathbf{A} \|_2 \cdot \| \mathbf{Y} \|_F.
\]

**Proof.** We have

\[
|\text{Tr}(\mathbf{X}^\top \mathbf{A} \mathbf{Y})| \leq \| \mathbf{X} \|_F \cdot \| \mathbf{A} \|_F \cdot \| \mathbf{Y} \|_F \leq \| \mathbf{X} \|_F \cdot \| \mathbf{A} \|_2 \cdot \| \mathbf{Y} \|_F,
\]

with the first inequality following from Lemma C.8 on \( |\langle \mathbf{X}, \mathbf{AY} \rangle| \) and the second from Lemma C.9.

**Lemma C.11.** For any real \( m \times n \) matrix \( \mathbf{A} \), and pair of \( m \times k \) matrix \( \mathbf{X} \) and \( n \times k \) matrix \( \mathbf{Y} \) such that \( \mathbf{X}^\top \mathbf{X} = \mathbf{I}_k \) and \( \mathbf{Y}^\top \mathbf{Y} = \mathbf{I}_k \) with \( k \leq \min \{m, n\} \), the following holds:

\[
|\text{Tr}(\mathbf{X}^\top \mathbf{A} \mathbf{Y})| \leq \sqrt{k} \cdot \left( \sum_{i=1}^{k} \sigma_i^2(\mathbf{A}) \right)^{1/2}.
\]

**Proof.** By Lemma C.8

\[
|\langle \mathbf{X}, \mathbf{AY} \rangle| = |\text{Tr}(\mathbf{X}^\top \mathbf{A} \mathbf{Y})| \leq \| \mathbf{X} \|_F \cdot \| \mathbf{AY} \|_F = \sqrt{k} \cdot \| \mathbf{AY} \|_F.
\]

where the last inequality follows from the fact that \( \| \mathbf{X} \|_F^2 = \text{Tr}(\mathbf{X}^\top \mathbf{X}) = \text{Tr}(\mathbf{I}_k) = k \). Further, for any \( \mathbf{Y} \) such that \( \mathbf{Y}^\top \mathbf{Y} = \mathbf{I}_k \),

\[
\| \mathbf{AY} \|_F^2 \leq \max_{\hat{\mathbf{Y}} \in \mathbb{R}^{n \times k}} \| \mathbf{A} \hat{\mathbf{Y}} \|_F^2 = \sum_{i=1}^{k} \sigma_i^2(\mathbf{A}).
\]

(27)

Combining the two inequalities, the result follows.

**Lemma C.12.** For any real \( m \times n \) matrix \( \mathbf{A} \), and any \( k \leq \min \{m, n\} \),

\[
\max_{\mathbf{Y} \in \mathbb{R}^{n \times k}} \| \mathbf{AY} \|_F = \left( \sum_{i=1}^{k} \sigma_i^2(\mathbf{A}) \right)^{1/2}.
\]

The above equality is realized when the \( k \) columns of \( \mathbf{Y} \) coincide with the \( k \) leading right singular vectors of \( \mathbf{A} \).
Proof. Let $U \Sigma V^T$ be the singular value decomposition of $A$, with $\sigma_j$ being the $j$th largest singular value of $A$, $j = 1, \ldots, d$, where $d \triangleq \min\{m, n\}$. Due to the invariance of the Frobenius norm under unitary multiplication,

$$\|AY\|_F^2 = \|U \Sigma V^T Y\|_F^2 = \|\Sigma Y\|_F^2. \quad (28)$$

Continuing from (28),

$$\|\Sigma Y\|_F^2 = \text{Tr}(Y^T V \Sigma^2 V^T Y) = \sum_{i=1}^k y_i^T \left( \sum_{j=1}^d \sigma_j \cdot v_j v_j^T \right) y_i = \sum_{j=1}^d \sigma_j^2 \cdot \sum_{i=1}^k (v_j^T y_i)^2.$$

Let $z_j \triangleq \sum_{i=1}^k (v_j^T y_i)^2$, $j = 1, \ldots, d$. Note that each individual $z_j$ satisfies

$$0 \leq z_j \triangleq \sum_{i=1}^k (v_j^T y_i)^2 \leq \|v_j\|^2 = 1,$$

where the last inequality follows from the fact that the columns of $Y$ are orthonormal. Further,

$$\sum_{j=1}^d z_j = \sum_{j=1}^d \sum_{i=1}^k (v_j^T y_i)^2 = \sum_{i=1}^k \sum_{j=1}^d (v_j^T y_i)^2 = \sum_{i=1}^k \|y_i\|^2 = k.$$

Combining the above, we conclude that

$$\|AY\|_F^2 = \sum_{j=1}^d \sigma_j^2 \cdot z_j \leq \sigma_1^2 + \ldots + \sigma_k^2. \quad (29)$$

Finally, it is straightforward to verify that if $y_i = v_i$, $i = 1, \ldots, k$, then (29) holds with equality. \qed

Lemma C.13. For any real $m \times n$ matrix $A$, let $\sigma_i(A)$ be the $i$th largest singular value. For any $r, k \leq \min\{m, n\}$,

$$\sum_{i=r+1}^{r+k} \sigma_i(A) \leq \frac{k}{\sqrt{r+k}} \|A\|_F.$$

Proof. By the Cauchy-Schwartz inequality,

$$\begin{align*}
\sum_{i=r+1}^{r+k} \sigma_i(A) &\leq \sum_{i=r+1}^{r+k} |\sigma_i(A)| \leq \left( \sum_{i=r+1}^{r+k} \sigma_i^2(A) \right)^{1/2} \|1_k\|_2 \\
&= \sqrt{k} \cdot \left( \sum_{i=r+1}^{r+k} \sigma_i^2(A) \right)^{1/2}.
\end{align*}$$

Note that $\sigma_{r+1}(A), \ldots, \sigma_{r+k}(A)$ are the $k$ smallest among the $r + k$ largest singular values. Hence,

$$\begin{align*}
\sum_{i=r+1}^{r+k} \sigma_i^2(A) &\leq \frac{k}{r + k} \sum_{i=1}^r \sigma_i^2(A) \leq \frac{k}{r + k} \sum_{i=1}^l \sigma_i^2(A) \\
&= \frac{k}{r + k} \|A\|_F^2.
\end{align*}$$

Combining the two inequalities, the desired result follows. \qed

Corollary 1. For any real $m \times n$ matrix $A$, the $r$th largest singular value $\sigma_r(A)$ satisfies $\|A\|_F / \sqrt{r}$. \qed

Proof. It follows immediately from Lemma C.13. \qed

First, we define the $\|\cdot\|_{\infty,2}$ norm of a matrix as the $l_2$ norm of the column with the maximum $l_2$ norm, i.e., for an $r \times k$ matrix $C$

$$\|C\|_{\infty,2} = \max_{1 \leq i \leq k} \|c_i\|_2.$$

Note that

$$\|C\|_F^2 = \sum_{i=1}^k \|c_i\|_2^2 \leq k \cdot \max_{1 \leq i \leq k} \|c_i\|_2^2 = k \cdot \|C\|_{\infty,2}^2,$$

and

$$\sum_{i=r+1}^{r+k} \sigma_i(A) \leq \frac{k}{\sqrt{r+k}} \|A\|_F. \quad (30)$$