Memory-Efficient Approximation Algorithms for MAX-K-CUT and Correlation Clustering

Anonymous Author(s)
Affiliation
Address
email

Abstract

MAX-K-CUT and correlation clustering are fundamental graph partitioning problems. The methods with the best approximation guarantees for MAX-K-CUT and the MAX-AGREE variant of correlation clustering involve solving SDPs with n^2 variables and n^2 constraints (where n is the number of vertices). Large-scale instances of SDPs, thus, present a memory bottleneck. In this paper, we develop a simple polynomial-time Gaussian sampling-based algorithms for these two problems that use $\mathcal{O}(n + |E|)$ memory and nearly achieve the best existing approximation guarantees. For dense graphs arriving in a stream, we eliminate the dependence on $|E|$ in the storage complexity at the cost of a slightly worse approximation ratio by combining our approach with sparsification.

1 Introduction

Semidefinite programs (SDPs) arise naturally as a relaxation of a variety of problems such as k-means clustering [5], correlation clustering [6] and MAX-K-CUT [14]. In each case, the decision variable is an $n \times n$ matrix and there are $d = \Omega(n^2)$ constraints. While reducing the memory bottleneck for large-scale SDPs has been studied quite extensively in literature [9, 11, 19, 36], all these methods use memory that scales linearly with the number of constraints and also depends on either the rank of the optimal solution or an approximation parameter. A recent Gaussian-sampling technique to 'solve' SDPs with smooth objective function involves replacing the decision variable X with a zero-mean random vector whose covariance is X [27]. This method uses at most $\mathcal{O}(n+d)$ memory, independent of the rank of the optimal solution. However, for SDPs with $\Omega(n^2)$ constraints, these algorithms still use $\Omega(n^2)$ memory and provide no advantage. In this paper, we show how to use adapt the Gaussian sampling-based approach of [27] to generate an approximate solution with provable approximation guarantees to MAX-K-CUT, and to the MAX-AGREE variant of correlation clustering on a graph $G = (V, E)$ with arbitrary edge weights using only $\mathcal{O}(|V| + |E|)$ memory.

1.1 MAX-K-CUT

MAX-K-CUT is the problem of partitioning the vertices of a weighted undirected graph $G = (V, E)$ into k distinct parts, such that the total weight of the edges across the parts is maximized. If w_{ij} is the edge weight corresponding to edge $(i, j) \in E$, then the cut value of a partition is $\texttt{CUT} = \sum_{i \text{ and } j \text{ are in different partitions}} w_{ij}$. Consider the standard SDP relaxation of MAX-K-CUT

$$\max_{X \succeq 0} \quad \langle C, X \rangle \quad \text{subject to} \quad \begin{cases} \operatorname{diag}(X) = \mathbb{1} \\ X_{ij} \geq -\frac{1}{k-1} \quad i \neq j, \end{cases} \tag{k-Cut-SDP}$$

where $C = \frac{k-1}{2k} L_G$ is a scaled Laplacian. Frieze and Jerrum [14] developed a randomized rounding scheme that takes an optimal solution $X^\star$ of (k-Cut-SDP) and produces a random partitioning

Submitted to 35th Conference on Neural Information Processing Systems (NeurIPS 2021). Do not distribute.

satisfying

$$\mathbb{E}[\texttt{CUT}] = \sum_{ij \in E, i<j} w_{ij} \Pr(i \text{ and } j \text{ are in different partitions}) \geq \alpha_k \langle C, X^\star \rangle \geq \alpha_k \text{opt}_k^G, \qquad (1)$$

where opt_k^G is the optimal k-cut value and α_k is the approximation ratio defined in (3). The rounding scheme proposed in [14], referred to as the FJ rounding scheme in the rest of the paper, generates k i.i.d. samples, $z_1, \ldots, z_k \sim \mathcal{N}(0, X^\star)$ and assigns vertex i to part p, if $[z_p]_i \geq [z_l]_i$ for $l \in \{1, \ldots, k\}$.

1.2 Correlation clustering

In correlation clustering, we are given a set of $|V|$ vertices together with the information indicating whether pairs of vertices are similar or dissimilar, modeled by the edges in the sets E^+ and E^- respectively. The MAX-AGREE variant of correlation clustering seeks to maximize

$$\mathcal{C} = \sum_{ij \in E^-} w_{ij}^- \mathbb{1}_{[i,j \text{ in different clusters}]} + \sum_{ij \in E^-} w_{ij}^+ \mathbb{1}_{[i,j \text{ in the same cluster}]}.$$

Define $G^+ = (V, E^+)$ and $G^=(V, E^-)$. A natural SDP relaxation of MAX-AGREE is

$$\max_{X \succeq 0} \quad \langle C, X \rangle \quad \text{subject to} \quad \begin{cases} \text{diag}(X) = \mathbb{1} \\ X_{ij} \geq 0 \quad i \neq j, \end{cases} \qquad \text{(MA-SDP)}$$

where $C = L_{G^-} + W^+$, L_{G^-} is the Laplacian of the graph G^- and W^+ is the weighted adjacency matrix of the graph G^+. Charikar et al. [10] (see also Swamy [30]) propose a rounding scheme that takes an optimal solution $X_G^\star$ of (MA-SDP) and produces a random clustering $\mathcal{C}$ satisfying

$$\mathbb{E}[\mathcal{C}] \geq 0.766 \langle C, X_G^\star \rangle \geq 0.766 \text{opt}_{CC}^G,$$

where opt_{CC}^G is the optimal clustering value. The rounding scheme proposed in [10], referred to as the CGW rounding scheme in the rest of the paper, generates up to 3 i.i.d. Gaussian samples $z_1, z_2, z_3 \sim \mathcal{N}(0, X_G^\star)$ and uses them to define up to 8 clusters.

1.3 Contributions

Gaussian sampling for MAX-K-CUT. Applying Gaussian sampling-based Frank-Wolfe [27] directly to (k-Cut-SDP) uses n^2 memory. We, however, show how to modify their approach (see Section 3), and combine it with FJ rounding scheme, to generate a k-cut with nearly the same approximation guarantees as stated in (1) using $\mathcal{O}(n + |E|)$ memory. A key highlight of our approach is that while the approximation ratio remains close to the state-of-the-art result in (1), reducing it by a factor of $1 - 4\epsilon$, the memory used is independent of ϵ. We summarize our result as follows.

Proposition 1. *For $\epsilon \in (0, 1/4)$, our $\mathcal{O}(n^2|E| \log(2n + |E|)/\epsilon^2)$-time method outlined in Section 3 uses $\mathcal{O}(n + |E|)$ memory and generates a k-cut for the graph $G = (V, E)$ whose expected value satisfies $\mathbb{E}[\texttt{CUT}] \geq \alpha_k(1 - 4\epsilon)\text{opt}_k^G$, where opt_k^G is the optimal k-cut value.*

Gaussian sampling for MAX-AGREE The structure of (MA-SDP) is similar to (k-Cut-SDP), however, the cost matrix in (MA-SDP) is no longer PSD or diagonally dominant, a property that plays an important role in our analysis in the case of MAX-K-CUT. Despite this, we show how to generate a $(1 - 6\epsilon)0.766$-optimal clustering using $\mathcal{O}(n + |E|)$ memory. Our approach makes a small sacrifice in the approximation ratio, however, the memory used remains independent of ϵ.

Proposition 2. *For $\epsilon \in (0, 1/6)$, our $\mathcal{O}\left(n^2|E| \log(2n + |E|)/\epsilon^2\right)$-time method outlined in Section 4 uses $\mathcal{O}(n + |E|)$ memory and generates a clustering of graph $G = (V, E)$ whose expected value satisfies $\mathbb{E}[\mathcal{C}] \geq 0.766(1 - 6\epsilon)\text{opt}_{CC}^G$, where opt_{CC}^G is the optimal clustering value.*

The constructive proof outline of Propositions 1 and 2 is given in Sections 3 and 4 respectively.

Memory reduction using graph sparsification. Propositions 1 and 2 state that the memory used by our approach is $\mathcal{O}(n + |E|)$. However, for dense graphs, the memory used by our method is $\Theta(n^2)$. In this setting, to reduce the memory used, we first need to change the way we access the problem instance. We assume that the input weighted graph G arrives edge-by-edge, eliminating the need to store the entire dense graph. We then replace it with a τ-spectrally close graph $\tilde{G}$ (see Definition 1) with $\mathcal{O}(n \log n/\tau^2)$ edges. Next, we compute an approximate solution to the new problem defined on the sparse graph using $\mathcal{O}(n \log n/\tau^2)$ memory. For MAX-K-CUT and MAX-AGREE, we show that this method generates a solution with provable approximation guarantees.

1.4 Literature review

Review of low memory algorithms for SDPs. Burer and Monteiro [9] proposed a nonlinear programming approach which replaces the PSD decision variable with its low-rank factorization in SDPs with d linear constraints. If the selected value of rank r satisfies $r(r+1) \geq 2d$ and the constraint set is a smooth manifold, then any second-order critical point of the nonconvex problem is a global optimum [8]. Another approach, that requires $\Theta(d+nr)$ working memory, is to first determine (approximately) the subspace in which the (low) rank-r solution to an SDP lies and then solve the problem over the low-dimensional subspace [11]. Randomized sketching to a low dimensional subspace is often used to as low-memory alternative to storing a matrix decision variable [31, 34]. Recently, such sketched variables have used to generate a low-rank approximation of a near-optimal solution to SDPs [37]. The working memory required to compute a near-optimal solution and generate its rank-r approximation using the algorithmic framework proposed by Yurtsever et al. [37] is $\mathcal{O}(d+rn/\zeta)$ for some sketching parameter $\zeta \in (0,1)$. Gaussian-sampling based Frank- Wolfe [27] requires $\mathcal{O}(n+d)$ memory as a near-optimal solution to the linearly constrained SDP is represented using only the samples with covariance equal to the solution. This eliminates the dependency on the accuracy to which the low rank approximation of a near-optimal solution is computed or the rank r of the approximation. The two problems considered in this paper have SDP relaxations with n^2 constraints, for which applying the existing low-memory techniques provide no benefit since the memory requirement of these techniques depends on the number of constraints in the problem.

MAX-K-CUT. MAX-K-CUT and its dual MIN-K-PARTITION have applications in frequency allocation [12] and lower bound on co-channel interference in cellular networks [7]. These problems have been studied extensively in the literature [20, 26, 29]. The SDP-based rounding scheme given in [14] has also been adapted for similar problems of capacitated MAX-K-CUT [16] and approximate graph coloring [21]. In each case, however, the SDP relaxation has $\Omega(n^2)$ constraints. Alternative heuristic methods have been proposed in [13, 17, 23, 25], however, these methods generate a feasible cut which only provides a lower bound on the optimal cut value.

Correlation clustering. Charikar et al. [10], Swamy [30] provide 0.766-approximation schemes for MAX-AGREE each of which involve solving (MA-SDP). For large-scale applications, data streaming techniques have been studied quite extensively for various clustering problems, such as k-means and k-median [3, 24]. Ahn et al. [1] propose a single-pass, $\tilde{\mathcal{O}}(|E| + n\epsilon^{-10})$-time $0.766(1-\epsilon)$-approximation algorithm for the MAX-AGREE variant of correlation clustering that uses $\tilde{\mathcal{O}}(n/\epsilon^2)$ memory. In contrast, to achieve the same approximation guarantee, our approach uses $\mathcal{O}(n \log n/\epsilon^2)$ memory and its computational complexity is $\mathcal{O}\left(n^2 \min\left\{|E|, \frac{n \log n}{\epsilon^2}\right\} \log(|E|+2n)\frac{1}{\epsilon^2}\right)$ which is at most $\tilde{\mathcal{O}}\left(\frac{n^3}{\epsilon^4}\right)$. Moreover, our approach is algorithmically very simple to implement.

1.5 Outline

In Section 2, we review the Gaussian sampling-based Frank-Wolfe method [27] to compute a near-feasible, near-optimal solution to SDPs with linear equality and inequality constraints. In Sections 3 and 4 respectively, we adapt the Gaussian sampling approach to give an approximation algorithm for MAX-K-CUT and MAX-AGREE respectively, that use only $\mathcal{O}(n+|E|)$ memory, proving Propositions 1 and 2 respectively. In Section 5, we show how to combine our methods with streaming spectral sparsification to reduce the memory required to $\mathcal{O}(n \log n/\epsilon^2)$ for dense graphs presented edge-by-edge in a stream. We provide some preliminary computational results for MAX-AGREE in Section 6 and conclude our work and discuss possible future directions in Section 7. All proofs are deferred to the Supplementary material.

Notations. The matrix inner product is denoted by $\langle A, B\rangle = \mathrm{Tr}\left(A^T B\right)$. The vector of diagonal entries of a matrix X is $\mathrm{diag}(X)$ and $\mathrm{diag}^*(x)$ is a diagonal matrix with the vector x on the diagonal. The notations $\mathcal{O}, \Omega, \Theta$ have the usual complexity interpretation and $\tilde{\mathcal{O}}$ suppresses the dependence on $\log n$. An undirected edge (i,j) in the set E is denoted using $(i,j) \in E$ and $ij \in E$ interchangably.

2 Gaussian Sampling-based Frank-Wolfe

Consider a smooth, concave function g and define the trace constrained SDP

$$\max_{X \in \mathcal{S}} \quad g(\mathcal{B}(X)), \tag{BoundedSDP}$$

where $\mathcal{S} = \{\mathrm{Tr}(X) \leq \alpha, X \succeq 0\}$ and $\mathcal{B}(\cdot) : \mathbb{S}^n \to \mathbb{R}^d$ is a linear mapping that projects the variable from $\binom{n+1}{2}$-dimensional space to a d-dimensional space. One algorithmic approach to solving (BoundedSDP) is to use the Frank-Wolfe algorithm [18] which, in this case, computes an ϵ-optimal solution by taking steps of the form $X_{t+1} = (1-\gamma_t)X_t + \gamma_t \alpha h_t h_t^T$, where $\gamma_t \in [0,1]$ and h_t's arise from approximately solving a symmetric eigenvalue problem that depends only on $\mathcal{B}(X_t)$ and $g(\cdot)$. Standard convergence results show that an ϵ-optimal solution is reached after $\mathcal{O}(C_g^u/\epsilon)$ iterations, C_g^u is an upper bound on the curvature constant of g [18].

Frank-Wolfe with Gaussian sampling. The Gaussian sampling technique of [27] replaces the matrix-valued iterates, X_t, with Gaussian random vectors $z_t \sim \mathcal{N}(0, X_t)$. The update, at the level of samples, is then $z_{t+1} = \sqrt{1-\gamma_t} z_t + \sqrt{\gamma_t \alpha}\, \zeta_t h_t$, where ζ_t are i.i.d. standard Gaussian vectors. Furthermore, to track the change in the objective function value, it is sufficient to track the value $v_t = \mathcal{B}(X_t)$. To compute the update direction and to track the objective value only require $\mathcal{B}(X_t)$, which can be updated without explicitly storing X_t and thereby, reducing the memory used.

Algorithm 1[27] describes, in detail, Frank-Wolfe algorithm with Gaussian sampling when applied to (BoundedSDP). It uses at most $\mathcal{O}(n+d)$ memory at each iteration, and after at most $\mathcal{O}(C_g^u/\epsilon)$ iterations, returns a sample $\widehat{z}_\epsilon \sim \mathcal{N}(0, \widehat{X}_\epsilon)$, where $\widehat{X}_\epsilon$ is an ϵ-optimal solution to (BoundedSDP).

Algorithm 1: (`FWGaussian`) Frank-Wolfe Algorithm with Gaussian Sampling [27]

Input : Input data for (BoundedSDP), stopping criteria ϵ, accuracy parameter η, upper bound C_g^u on the curvature constant

Output : $z \sim \mathcal{N}(0, \widehat{X}_\epsilon)$ and $v = \mathcal{B}(\widehat{X}_\epsilon)$, where $\widehat{X}_\epsilon$ is an ϵ-optimal solution of (BoundedSDP)

Function `FWGaussian`**:**
- Select initial point $X_0 \in \mathcal{S}$; set $v_0 \leftarrow \mathcal{B}(X_0)$ and sample $z_0 \sim \mathcal{N}(0, X_0)$
- $t \leftarrow 0, \gamma \leftarrow 2/(t+2)$
- $(h_t, q_t) \leftarrow \texttt{LMO}(\mathcal{B}^*(\nabla g(v_t)), \frac{1}{2}\eta\gamma C_g^u)$
- **while** $\langle q_t - v_t, \nabla g(v_t)\rangle > \epsilon$ **do**
  - $(z_{t+1}, v_{t+1}) \leftarrow \texttt{UpdateVariable}(z_t, v_t, h_t, q_t, \gamma)$
  - $t \leftarrow t+1, \gamma \leftarrow 2/(t+2)$
  - $(h_t, q_t) \leftarrow \texttt{LMO}(\mathcal{B}^*(\nabla g(v_t)), \frac{1}{2}\eta\gamma C_g^u, p)$
- **end**

return (z_t, v_t)

Function `LMO`(J, δ)**:**
- Find a unit vector h such that $\alpha\lambda = \alpha\langle hh^T, J\rangle \geq \max_{d \in \mathcal{S}} \alpha\langle d, J\rangle - \delta$
- **if** $\lambda \geq 0$ **then** $q \leftarrow \mathcal{B}(\alpha h h^T)$
- **else** $q \leftarrow 0, h \leftarrow 0$

return (h, q)

Function `UpdateVariable`(z, v, h, q, γ)**:**
- $z \leftarrow (\sqrt{1-\gamma})z + \sqrt{\gamma\alpha}\, h\zeta$ where $\zeta \sim \mathcal{N}(0,1)$
- $v \leftarrow (1-\gamma)v + \gamma q$

return (z, v)

2.1 SDP with linear equality and inequality constraints

Consider an SDP with linear objective function and a bounded feasible region,

$$\max_{X \succeq 0} \quad \langle C, X\rangle \quad \text{subject to} \quad \begin{cases} \mathcal{A}^{(1)}(X) = b^{(1)} \\ \mathcal{A}^{(2)}(X) \geq b^{(2)}, \end{cases} \tag{SDP}$$

where $\mathcal{A}^{(1)}(\cdot) : \mathbb{S}^n_+ \to \mathbb{R}^{d_1}$ and $\mathcal{A}^{(2)}(\cdot) : \mathbb{S}^n_+ \to \mathbb{R}^{d_2}$ are linear maps. To use Algorithm 1, the linear constraints are penalized using a smooth penalty function. Let $u_l = \langle A^{(1)}_l, X\rangle - b^{(1)}_l$ for $l = 1, \ldots, d_1$ and $v_l = b^{(2)}_l - \langle A^{(2)}_l, X\rangle$ for $l = 1, \ldots, d_2$. For $M > 0$, the smooth function $\phi_M(\cdot) : \mathbb{R}^{d_1+d_2} \to \mathbb{R}$,

$$\phi_M(u, v) = \frac{1}{M} \log \left(\sum_{i=1}^{d_1} e^{M(u_i)} + \sum_{i=1}^{d_1} e^{M(-u_i)} + \sum_{j=1}^{d_2} e^{M(v_j)} \right), \quad \text{satisfies} \tag{2}$$

$$\max \left\{ \|u\|_\infty, \max_i v_i \right\} \le \phi_M(u, v) \le \frac{\log(2d_1 + d_2)}{M} + \max \left\{ \|u\|_\infty, \max_i v_i \right\}.$$

We add this penalty term to the objective of (SDP) and define

$$\max_{X \succeq 0} \left\{ \langle C, X\rangle - \beta \phi_M(\mathcal{A}^{(1)}(X) - b^{(1)}, b^{(2)} - \mathcal{A}^{(2)}(X)) \ : \ \mathrm{Tr}(X) \le \alpha \right\}, \tag{SDP-LSE}$$

where α, β and M are appropriately chosen parameters. Algorithm 1 then generates a Gaussian sample with covariance $\widehat{X}_\epsilon$ which is an ϵ-optimal solution to (SDP-LSE). It is also a near-optimal, near-feasible solution to (SDP). This result is a slight modification of [27, Lemma 3.2] which only provides bounds for SDPs with linear equality constraints.

Lemma 1. *For $\epsilon > 0$, let $(X^\star, \vartheta^\star, \mu^\star)$ be an optimal primal-dual solution to* (SDP) *and its dual, and let $\widehat{X}_\epsilon$ be an ϵ-optimal solution to* (SDP-LSE)*. If $\beta > \|\vartheta^\star\|_1 + \|\mu^\star\|_1$ and $M > 0$, then*

$$\langle C, X^\star\rangle - \frac{\beta \log(2d_1 + d_2)}{M} - \epsilon \le \langle C, \widehat{X}_\epsilon\rangle \le \langle C, X^\star\rangle + (\|\vartheta^\star\|_1 + \|\mu^\star\|_1) \frac{\beta \frac{\log(2d_1+d_2)}{M} + \epsilon}{\beta - \|\vartheta^\star\|_1 - \|\mu^\star\|_1},$$

$$\max \left\{ \|\mathcal{A}^{(1)}(X) - b^{(1)}\|_\infty, \max_i \left(b^{(2)}_i - \mathcal{A}^{(2)}_i(X) \right) \right\} \le \frac{\beta \frac{\log(2d_1+d_2)}{M} + \epsilon}{\beta - \|\vartheta^\star\|_1 - \|\mu^\star\|_1}.$$

3 Application of Gaussian Sampling to (k-Cut-SDP)

In this section, we look at the application of Gaussian sampling to MAX-K-CUT. Since Algorithm 1 uses $\mathcal{O}(n^2)$ memory when solving (k-Cut-SDP), we define a new SDP relaxation of MAX-K-CUT with the same approximation guarantee, but with $\mathcal{O}(|E|)$ constraints. We then apply Algorithm 1 to this new relaxation, and show how to round the solution to achieve nearly the same approximation ratio as given in (1). Let

$$\alpha_k = \min_{-1/(k-1) \le \rho \le 1} \frac{kp(\rho)}{(k-1)(1-\rho)}, \tag{3}$$

where $p(X_{ij})$ is the probability that vertices i and j are in different partitions. If X is feasible for (k-Cut-SDP) and $\texttt{CUT}$ is the value of the k-cut generated by the FJ rounding scheme, then

$$\begin{aligned} \mathbb{E}[\texttt{CUT}] &= \sum_{ij \in E, i<j} w_{ij} p(X_{ij}) \\ &\ge \sum_{ij \in E, i<j} \frac{k-1}{k} w_{ij}(1 - X_{ij})\alpha_k = \alpha_k \langle C, X\rangle. \end{aligned} \tag{4}$$

Frieze and Jerrum [14] derive a lower bound on α_k, showing that the method gives a nontrivial approximation guarantee. Observe that (4) depends only on the values X_{ij} if $(i, j) \in E$.

A new SDP relaxation to MAX-K-CUT We relax the constraints in (k-Cut-SDP) to define

$$\max_{X \succeq 0} \quad \langle C, X\rangle \quad \text{subject to} \quad \begin{cases} \mathrm{diag}(X) = \mathbb{1} \\ X_{ij} \ge -\frac{1}{k-1} \quad (i,j) \in E, i < j. \end{cases} \tag{k-Cut-Rel}$$

Since (k-Cut-Rel) is a relaxation of (k-Cut-SDP), its optimal objective function value provides an upper bound on $\langle C, X^\star\rangle$ and hence, on the optimal MAX-K-CUT value opt^G_k. Note that the bound in (4) holds true even if we replace $X^\star$ by an optimal solution to (k-Cut-Rel) since it depends on the value of X_{ij} only if $(i, j) \in E$. Furthermore, when the FJ rounding scheme is applied to the solution of (k-Cut-Rel), it satisfies the approximation guarantee on the expected value of the generated k-cut given in (1), i.e., $\mathbb{E}[\texttt{CUT}] \ge \alpha_k \mathrm{opt}^G_k$.

Using Algorithm 1. We now have an SDP relaxation of MAX-K-CUT that has $n + |E|$ constraints. Penalizing the linear constraints using the function $\phi_M(\cdot)$ (2), Algorithm 1 can now be used to generate k samples with covariance $\widehat{X}_\epsilon$ which is an ϵ-optimal solution to

$$\max_{X \succeq 0} \left\{ \langle C, X \rangle - \beta \phi_M \left(\text{diag}(X) - \mathbb{1}, -\tfrac{1}{k-1} - e_i^T X e_j \right) : (i,j) \in E, \text{Tr}(X) \leq n \right\}. \quad \text{(k-Cut-LSE)}$$

Optimality and feasibility results for (k-Cut-Rel)**.** Given an ϵ-optimal solution to (k-Cut-LSE), we show that it is also a near-optimal, near-feasible solution to (k-Cut-Rel).

Lemma 2. *For $\epsilon \in (0, 1/2)$, let $X_R^\star$ be an optimal solution to* (k-Cut-Rel) *and let $\widehat{X}_\epsilon$ be an $\epsilon\text{Tr}(C)$-optimal solution to* (k-Cut-LSE)*. For $\beta = 6\text{Tr}(C)$ and $M = 6\frac{\log(2n+|E|)}{\epsilon}$, we have*

$$(1 - 2\epsilon)\langle C, X_R^\star \rangle \leq \langle C, \widehat{X}_\epsilon \rangle \leq (1 + 4\epsilon)\langle C, X_R^\star \rangle \quad \textit{and} \tag{5}$$

$$\|\text{diag}(\widehat{X}_\epsilon) - 1\|_\infty \leq \epsilon, \quad [\widehat{X}_\epsilon]_{ij} \geq -\frac{1}{k-1} - \epsilon, \quad (i,j) \in E, i < j. \tag{6}$$

Generating a feasible solution to MAX-K-CUT. Since $\widehat{X}_\epsilon$ might not necessarily be feasible to (k-Cut-Rel), we cannot apply the FJ rounding scheme to the samples $z_i \sim \mathcal{N}(0, \widehat{X}_\epsilon)$. We, therefore, generate samples $z_i^f \sim \mathcal{N}(0, X^f)$ using the procedure given in Algorithm 2 such that X^f is feasible to (k-Cut-Rel) and $\langle C, X^f \rangle$ is close to $\langle C, \widehat{X}_\epsilon \rangle$.

Algorithm 2: Generate Gaussian samples with covariance feasible to (k-Cut-Rel)

Input : Sample $z_i \sim \mathcal{N}(0, \widehat{X}_\epsilon)$ for $i = 1, \ldots, k$ and $\text{diag}(\widehat{X}_\epsilon)$
Output : $z_i^f \sim \mathcal{N}(0, X^f)$ for $i = 1, \ldots, k$ with X^f feasible to (k-Cut-Rel)

Function `GeneratekSamples`**:**
  for $i = 1, \ldots, k$ **do**
    Set $\text{err} = \max\{0, \max_{(i,j)\in E, i<j} \; -1/(k-1) - [\widehat{X}_\epsilon]_{ij}\}$
    Set $\overline{z}_i = z_i + \sqrt{\text{err}}\, y\, \mathbb{1}$, where $y \sim \mathcal{N}(0,1)$
    Generate $\zeta \sim \mathcal{N}\left(0, I - \text{diag}^*\left(\frac{\text{diag}(\widehat{X}_\epsilon)+\text{err}}{\max(\text{diag}(\widehat{X}_\epsilon))+\text{err}}\right)\right)$
    Set $z_i^f = \frac{\overline{z}_i}{\sqrt{\max(\text{diag}(\widehat{X}_\epsilon))+\text{err}}} + \zeta$
  end
return $z_1^f, \ldots, z_k^f$

We can now apply the FJ rounding scheme to $z_1^f, \ldots, z_k^f$ as given in Lemma 3.

Lemma 3. *For $\epsilon \in \left(0, \frac{1}{4}\right)$, let $\widehat{X}_\epsilon \succeq 0$ satisfy* (5) *and* (6)*. Let $z_1^f, \ldots, z_k^f$ be random vectors generated by Algorithm 2 with input $z_i, \ldots, z_k \sim \mathcal{N}(0, \widehat{X}_\epsilon)$ and let `CUT` denote the value of a k-cut generated by applying the FJ rounding scheme to $z_1^f, \ldots, z_k^f$. Then*

$$\alpha_k(1 - 4\epsilon)\text{opt}_k^G \leq \alpha_k(1 - 4\epsilon)\langle C, X^\star \rangle \leq \mathbb{E}[\texttt{CUT}] \leq \text{opt}_k^G, \tag{7}$$

where opt_k^G is the optimal k-cut value and α_k is defined by (3)*.*

Convergence guarantees. We can generate an $\epsilon\text{Tr}(C)$-optimal solution to (k-Cut-LSE) after at most $\mathcal{O}\left(\frac{C_f^u}{\epsilon\text{Tr}(C)}\right) = \mathcal{O}\left(\frac{\beta M n^2}{\epsilon\text{Tr}(C)}\right)$ iterations of `LMO` in Algorithm 1 using the convergence of Frank-Wolfe [18], together with an upper bound C_f^u on the curvature constant of the objective of (k-Cut-LSE) of $\beta M n^2$ [27, Lemma 3.1]. Furthermore, `LMO` involves matrix-vector multiplication with the gradient of the objective and so, its complexity depends on the sparsity of the matrix, i.e., $\mathcal{O}(|E|)$ leading to a $\mathcal{O}(n^2|E|\log(2n + |E|)/\epsilon^2)$-time algorithm and proving Proposition 1.

4 Application of Gaussian Sampling to (MA-SDP)

We now look at the application of our Gaussian sampling-based method to MAX-AGREE. Algorithm 1 uses $\mathcal{O}(n^2)$ memory to generate samples whose covariance is an ϵ-optimal solution to (MA-SDP).

However, with the similar observation as MAX-K-CUT, we note that for any X feasible to (MA-SDP), the proof of the inequality $\mathbb{E}[\mathcal{C}] \geq 0.766\langle C, X\rangle$, given in [10, Theorem 3], requires $X_{ij} \geq 0$ only if $(i,j) \in E$. We therefore, write a new relaxation of (MA-SDP),

$$\max_{X \succeq 0} \langle C, X\rangle = \langle W^+ + L_{G^-}, X\rangle \quad \text{subject to} \begin{cases} X_{ii} = 1 \ \ \forall\, i \in \{1, \ldots, n\} \\ X_{ij} \geq 0 \ \ (i,j) \in E, i < j, \end{cases} \tag{MA-Rel}$$

with only $n + |E|$ constraints. The bound $\mathbb{E}[\mathcal{C}] \geq 0.766\langle C, X^\star\rangle \geq 0.766\text{opt}_{CC}^G$ on the expected value of the clustering holds even if the clustering is generated by applying the CGW rounding scheme to an optimal solution $X^\star$ of (MA-Rel). To use Algorithm 1, we penalize the constraints in (MA-Rel) and define

$$\max_{X \succeq 0} \left\{ \langle C, X\rangle - \beta\phi_M(\text{diag}(X) - \mathbb{1}, -e_i^T X e_j) : (i,j) \in E, \text{Tr}(X) \leq n \right\}. \tag{MA-LSE}$$

Optimality and feasibility results for (MA-Rel)**.** Algorithm 1 generates $z \sim \mathcal{N}(0, \widehat{X}_\epsilon)$, where $\widehat{X}_\epsilon$ is an ϵ-optimal solution to (MA-LSE) and a near-optimal, near-feasible solution to (MA-Rel).

Lemma 4. *For* $\Delta = \text{Tr}(L_{G^-}) + \sum_{ij \in E^+} w_{ij}^+$, $\epsilon \in \left(0, \frac{1}{4}\right)$, *let* $X_G^\star$ *be an optimal solution to* (MA-Rel) *and* $\widehat{X}_\epsilon$ *be an* $\epsilon\Delta$*-optimal solution to* (MA-LSE)*. Setting* $\beta = 4\Delta$ *and* $M = 4\frac{\log(2n+|E|)}{\epsilon}$*, we have*

$$(1 - 4\epsilon)\langle C, X_G^\star\rangle \leq \langle C, \widehat{X}_\epsilon\rangle \leq (1 + 4\epsilon)\langle C, X_G^\star\rangle \quad \text{and} \tag{8}$$

$$\|[\widehat{X}_\epsilon]_{ii} - 1\|_\infty \leq \epsilon, [\widehat{X}_\epsilon]_{ij} \geq -\epsilon, \quad (i,j) \in E, i < j. \tag{9}$$

Using the convergence of Frank-Wolfe [18] and $C_f^u = \beta M n^2$ [27, Lemma 3.1], we conclude that after $\mathcal{O}\left(C_u^f/\epsilon\Delta\right)$, i.e., at most $\mathcal{O}\left(n^2 \log(2n + |E|)/\epsilon^2\right)$ iterations of `LMO`, Algorithm 1 generates an $\epsilon\Delta$-optimal solution to (MA-LSE).

Generate an approximate clustering. We can use the modified Algorithm 2 (Step 3 replaced with $\text{err} = \max\{0, \max_{(i,j)\in E, i<j} -[\widehat{X}_\epsilon]_{ij}\}$) to generate feasible samples and apply the CGW rounding scheme to the feasible samples which concludes the proof of Proposition 2.

Lemma 5. *Let* $X_G^\star$ *be an optimal solution to* (MA-Rel)*. For* $\epsilon \in (0, 1/6)$*, let* $\widehat{X}_\epsilon \succeq 0$ *satisfy* (8) *and* (9)*, and let* z_1^f, z_2^f, z_3^f *be random vectors generated by Algorithm 2 with input* $z_1, z_2, z_3 \sim \mathcal{N}(0, \widehat{X}_\epsilon)$*. Let* opt_{CC}^G *denote the optimal clustering value and let* $\mathcal{C}$ *denote the value of the clustering generated by* z_1^f, z_2^f, z_3^f *using the CGW rounding scheme. Then*

$$\mathbb{E}[\mathcal{C}] \geq 0.766(1 - 6\epsilon)\langle C, X_G^\star\rangle \geq 0.766(1 - 6\epsilon)\text{opt}_{CC}^G. \tag{10}$$

5 Sparsifying the Laplacian Cost Matrix

As seen in Sections 3 and 4, the memory requirement for generating and representing an ϵ-optimal solution to (k-Cut-LSE) and (MA-LSE) is bounded by $\mathcal{O}(n + |E|)$. If the input graph G is dense, the cost matrix will be dense and the number of inequality constraints will still be high. In this section, we consider the situation in which the dense weighted graph arrives in a stream, and we first build a sparse approximation with similar spectral properties.

Definition 1 (τ-spectral closeness)**.** *Two graphs* G *and* $\tilde{G}$ *defined on the same set of vertices are said to be* τ-spectrally close *if, for any* $x \in \mathbb{R}^n$ *and* $\tau \in (0, 1)$*,*

$$(1 - \tau)x^T L_G x \leq x^T L_{\tilde{G}} x \leq (1 + \tau)x^T L_G x. \tag{11}$$

We refer to this additional step as *sparsifying* the cost. Spectral graph sparsification has been studied quite extensively (see, e.g., [2, 15, 28]). Kyng et al. [22] propose a framework to replace a dense graph $G = (V, E)$ with $|E| = m$ by a sparser graph $\tilde{G} = (V, \tilde{E})$ such that $|\tilde{E}| = \mathcal{O}(n \log n/\tau^2)$ and $\tilde{G}$ satisfies (11) with probability $1 - \frac{1}{\text{poly}(n)}$ in $\mathcal{O}(m \log^2 n)$ time. Their algorithm assumes that the edges of the graph arrive one at a time, so that the total memory requirement is $\mathcal{O}(n \log n/\tau^2)$ rather than $\mathcal{O}(m)$. Furthermore, a sparse cost matrix will decrease the computation time of the subproblem in Algorithm 1 since it involves matrix-vector multiplication with the cost matrix.

MAX-K-CUT with sparsification. Let $\tilde{G}$ be a sparse graph that is τ-spectrally close to the input graph G with $\mathcal{O}(n \log n/\tau^2)$ edges. By applying the method outlined in Section 3, we can generate a k-cut for the graph $\tilde{G}$ whose value satisfies the bound (7) using $\mathcal{O}(n \log n/\tau^2)$ memory. This generated k-cut still enjoys a similar approximation guarantee with respect to the original graph G.

Lemma 6. *For $\epsilon, \tau \in (0, 1/5)$, let $\widehat{X}_\epsilon$ be a near-feasible, near-optimal solution to* (k-Cut-Rel) *defined on the graph $\tilde{G}$ that satisfies* (5) *and* (6)*. Let $\mathtt{CUT}$ denote the value of the k-cut generated by applying Algorithm 2 followed by the FJ rounding scheme to $\widehat{X}_\epsilon$. Then the generated cut satisfies*

$$\alpha_k(1 - 4\epsilon - \tau)\text{opt}_k^G \leq \mathbb{E}[\mathtt{CUT}] \leq \text{opt}_k^G, \tag{12}$$

where opt_k^G is the value of the optimal k-cut for the original graph G.

MAX-AGREE with sparsification. The number of edges $|E^+|$ and $|E^-|$ in graphs G^+ and G^- respectively determine the working memory of Algorithm 1. We now sparsify the graphs G^+ and G^- to generate graphs $\tilde{G}^+$ and $\tilde{G}^-$ with at most $\mathcal{O}(n \log n/\tau^2)$ edges and define

$$\max_{X \in \mathcal{S}} \tilde{f}(X) = \langle L_{\tilde{G}^-} + \tilde{W}^+, X\rangle - \beta\phi_M(\text{diag}(X) - \mathbb{1}, -[e_i^T X e_j]_{(i,j)\in E}), \tag{MA-Sparse}$$

where $\mathcal{S} = \{X : \text{Tr}(X) \leq n, X \succeq 0\}$, $L_{\tilde{G}^-}$ is the Laplacian of the graph $\tilde{G}^-$ and $\tilde{W}^+$ is matrix with nonnegative entries denoting the weight of each edge $(i, j) \in \tilde{E}^+$. Algorithm 1 then generates an $\epsilon(1 + \tau)(\text{Tr}(L_{G^-}) + \sum_{ij\in E^+} w_{ij}^+)$-optimal solution, $\widehat{X}_\epsilon$, to (MA-Sparse) using $\mathcal{O}(n \log n/\tau^2)$ memory. We can now use the method given in Section 4 to generate a clustering of graph $\tilde{G}$ whose expected value, $\mathbb{E}[\mathcal{C}]$, satisfies (10). The following lemma shows that $\mathcal{C}$ also represents a clustering for the original graph G with provable guarantees.

Lemma 7. *For $\epsilon, \tau \in (0, 1/9)$, let $\widehat{X}_\epsilon$ be a near-feasible, near-optimal solution to* (MA-Sparse) *defined on the graph $\tilde{G}$ that satisfies* (8) *and* (9)*. Let $\mathcal{C}$ denote the value of the clustering generated by applying Algorithm 2 followed by the CGW rounding scheme to $\widehat{X}_\epsilon$. Then, $\mathbb{E}[\mathcal{C}]$ satisfies*

$$0.766(1 - 6\epsilon - 3\tau)(1 - \tau^2)\text{opt}_{CC}^G \leq \mathbb{E}[\mathcal{C}] \leq \text{opt}_{CC}^G, \tag{13}$$

where opt_{CC}^G is the value of the optimal clustering of the original graph G.

We summarize our results in the following proposition.

Proposition 3. *Assume that the edges of the input graph $G = (V, E)$ arrive one at a time in a stream. The procedure given in this section reads the input graph edge-by-edge and using at most $\mathcal{O}(n \log n/\tau^2)$ memory, generates approximate solutions to* MAX-K-CUT *and* MAX-AGREE *in $\mathcal{O}(n^2|E| \log(2n + |E|)/\epsilon^2)$ time that satisfy the bounds* (12) *and* (13) *respectively.*

6 Computational Results

We now discuss the results of preliminary computations to cluster the vertices of a graph G using the approach outlined in Section 4. Correlation clustering generates clustering of vertices for graphs where each edge has a label indicating 'similarity' or 'dissimilarity'. We converted the undirected, unweighted graphs from the GSET dataset [35] into the instances of correlation clustering using an adaptation of the approach used in [32, 33]. We note that GSET consists of randomly generated graphs. The adaptation of the approach given in [32, 33] generated the label and the weight indicating the amount of 'similarity' or 'dissimilarity' for each edge $(i, j) \in E$. First, we computed the Jaccard coefficient $J_{ij} = |N(i) \cap N(j)|/|N(i) \cup N(j)|$, where $N(i)$ is the set of neighbours of i for each edge $(i, j) \in E$. Next, we computed the quantity $S_{ij} = \log((1 - J_{ij} + \delta)/(1 + J_{ij} - \delta))$ with $\delta = 0.05$ which is a measure of the amount of 'similarity' or 'dissimilarity'. The edge (i, j) was labelled as 'dissimilar' if $S_{ij} < 0$ with $w_{ij}^- = -S_{ij}$ and labelled as 'similar' with $w_{ij}^+ = S_{ij}$ otherwise.

The input parameters were set to $\epsilon = 0.05$, $\Delta = \text{Tr}(L_{G^-}) + \sum_{ij\in E^+} w_{ij}^+$, $\beta = 4\Delta$, $M = 4\frac{\log(2n)+|E|}{\epsilon}$. Using Algorithm 1, (MA-LSE) was solved to $\epsilon\Delta$-optimality and, we computed feasible samples using Algorithm 2. Finally, we generated two Gaussian samples and created at most four clusters by applying the 0.75-rounding scheme proposed by Swamy [30, Theorem 2.1], for simplicity.

The computations were performed using MATLAB R2021a on a machine with 8GB RAM. The peak memory used was noted using the `profiler` command in MATLAB. We have included the code and the GSET dataset in the Supplementary material.

We then used the approach given in Section 4 to generate a clustering of each graph in GSET dataset. The result for some randomly selected instances from the dataset is given in Table 1. We have provided the result for the rest of the graphs from GSET in the Supplementary material. First, we observed that the number of iterations of `LMO` for $\epsilon\Delta$ convergence satisfies the bound given in Proposition 1 and the infeasibility of the covariance $\widehat{X}_\epsilon$ of the generated samples was less than ϵ satisfying (9). We generated 10 pairs of samples and each in turn was used to generate a clustering for the input graph. Amongst the 10 clusterings generated for each graph, we picked the clustering with largest value denoted by $\mathcal{C}_{\text{best}}$. Note that, $\mathcal{C}_{\text{best}} \geq \mathbb{E}[\mathcal{C}] \geq 0.75(1-6\epsilon)\langle C, X_G^\star\rangle \geq 0.75\frac{1-6\epsilon}{1+4\epsilon}\langle C, \widehat{X}_\epsilon\rangle$, where the last inequality follows from combining (10) with (8). Since we were able to generate the values, $\mathcal{C}_{\text{best}}$ and $\langle C, \widehat{X}_\epsilon\rangle$, we noted that the weaker bound $\mathcal{C}_{\text{best}}/\langle C, \widehat{X}_\epsilon\rangle = \text{AR} \geq 0.75(1-6\epsilon)/(1+4\epsilon)$ was satisfied by every input graph when $\epsilon = 0.05$.

Furthermore, Table 1 also shows the memory used by our method. Consider the dataset G1, for which the memory used by our method was $1526.35\text{kB} \approx 9.8 \times (|V| + |E^+| + |E^-|) \times 8$, where a factor of 8 denotes that MATLAB requires 8 bytes to store a real number. Similarly, we observed that our method used at most $c \times (|V| + |E^+| + |E^-|) \times 8$ memory to generate clusters for other instances from GSET, where $c \leq 33$, showing that the memory used was linear in the size of the input graph.

Table 1: Result of generating a clustering of graphs from GSET using the method outlined in Section 4. We have, infeas $= \max\{\|\text{diag}(X) - 1\|_\infty, \max\{0, -[\widehat{X}_\epsilon]_{ij}\}\}$, AR $= \mathcal{C}_{\text{best}}/\langle C, \widehat{X}_\epsilon\rangle$ and $0.75(1-6\epsilon)/(1+4\epsilon) = 0.4375$ for $\epsilon = 0.05$.

Dataset	$\lvert V\rvert$	$\lvert E^+\rvert$	$\lvert E^-\rvert$	# Iterations ($\times 10^3$)	infeas	$\langle C, \widehat{X}_\epsilon\rangle$	$\mathcal{C}$	AR	Memory required (in kB)
G1	800	2453	16627	669.46	10^{-3}	849.48	643	0.757	1526.35
G11	800	817	783	397.2	6×10^{-4}	3000.3	2080	0.693	448.26
G14	800	3861	797	330.02	8×10^{-4}	542.55	469.77	0.866	423.45
G22	2000	115	19849	725.66	10^{-3}	1792.9	1371.1	0.764	1655.09
G32	2000	2011	1989	571.42	9×10^{-4}	7370	4488	0.609	1124
G43	1000	248	9704	501.31	10^{-3}	803.8	616.05	0.766	654.46
G48	3000	0	6000	9806.22	0.004	599.64	461.38	0.769	736.09
G51	1000	4734	1147	1038.99	0.001	676.21	446.29	0.66	517.09
G55	5000	66	12432	2707.07	0.002	1244.2	901.74	0.724	1281.03
G57	5000	4981	5019	574.5	0.005	18195	10292	0.565	812.78

7 Discussion

In this paper, we proposed a Gaussian sampling-based optimization algorithm to generate approximate solutions to MAX-K-CUT and the MAX-AGREE variant of correlation clustering using $\mathcal{O}\left(n + \min\left\{|E|, \frac{n\log n}{\tau^2}\right\}\right)$ memory. The key observation was that the approximation guarantees are preserved for a weaker SDP relaxation of these two problems with only $\mathcal{O}(n + |E|)$ constraints, and are the same as the original SDP relaxation with n^2 constraints. We observed that for MAX-K-CUT and the MAX-AGREE variant of correlation clustering, our approach nearly preserves the quality of the solution as given in [10, 14]. We also implemented the method outlined in Section 4 to generate approximate clustering for random graphs with provable guarantees. We observed that while the method was simple to implement, it was slow in practice. However, there is scope for improving the convergence rate of the method so that it can potentially be applied to the large-scale instances of various real-life applications of clustering.

Problems with triangle inequalities The known nontrivial approximation guarantees for sparsest cut and involve solving an SDP relaxation that has n^3 triangle inequalities [4]. It would be interesting to see whether it is possible to simplify these SDPs in such a way that they can be combined nicely with memory efficient algorithms, and still maintain good approximation guarantees.

References

[1] K. Ahn, G. Cormode, S. Guha, A. McGregor, and A. Wirth. Correlation clustering in data streams. In *International Conference on Machine Learning*, pages 2237–2246. PMLR, 2015.

[2] K. J. Ahn and S. Guha. Graph sparsification in the semi-streaming model. In *International Colloquium on Automata, Languages, and Programming*, pages 328–338. Springer, 2009.

[3] N. Ailon, R. Jaiswal, and C. Monteleoni. Streaming k-means approximation. In *Proceddings of the Twenty-Third Annual Conference on Neural Information Processing Systems*, volume 4, page 2, 2009.

[4] S. Arora, S. Rao, and U. Vazirani. Expander flows, geometric embeddings and graph partitioning. *Journal of the ACM (JACM)*, 56(2):1–37, 2009.

[5] P. Awasthi, A. S. Bandeira, M. Charikar, R. Krishnaswamy, S. Villar, and R. Ward. Relax, no need to round: Integrality of clustering formulations. In *Proceedings of the 2015 Conference on Innovations in Theoretical Computer Science*, pages 191–200, 2015.

[6] N. Bansal, A. Blum, and S. Chawla. Correlation clustering. *Machine Learning*, 56(1-3):89–113, 2004.

[7] R. Borndörfer, A. Eisenblätter, M. Grötschel, and A. Martin. Frequency assignment in cellular phone networks. *Annals of Operations Research*, 76:73–93, 1998.

[8] N. Boumal, V. Voroninski, and A. Bandeira. The non-convex Burer-Monteiro approach works on smooth semidefinite programs. In *Advances in Neural Information Processing Systems*, pages 2757–2765, 2016.

[9] S. Burer and R. D. Monteiro. A nonlinear programming algorithm for solving semidefinite programs via low-rank factorization. *Mathematical Programming*, 95(2):329–357, 2003.

[10] M. Charikar, V. Guruswami, and A. Wirth. Clustering with qualitative information. *Journal of Computer and System Sciences*, 71(3):360–383, 2005.

[11] L. Ding, A. Yurtsever, V. Cevher, J. A. Tropp, and M. Udell. An optimal-storage approach to semidefinite programming using approximate complementarity. *arXiv preprint arXiv:1902.03373*, 2019.

[12] A. Eisenblätter, M. Grötschel, and A. Koster. Frequency planning and ramifications of coloring. *Discussiones Mathematicae Graph Theory*, 22(1):51–88, 2002.

[13] T. A. Feo and M. G. Resende. Greedy randomized adaptive search procedures. *Journal of Global Optimization*, 6(2):109–133, 1995.

[14] A. Frieze and M. Jerrum. Improved approximation algorithms for MAX-K-CUT and max bisection. *Algorithmica*, 18(1):67–81, 1997.

[15] W.-S. Fung, R. Hariharan, N. J. Harvey, and D. Panigrahi. A general framework for graph sparsification. *SIAM Journal on Scientific Computing*, 48(4):1196–1223, 2019.

[16] D. R. Gaur, R. Krishnamurti, and R. Kohli. The capacitated MAX-K-CUT problem. *Mathematical Programming*, 115(1):65–72, 2008.

[17] B. Ghaddar, M. F. Anjos, and F. Liers. A branch-and-cut algorithm based on semidefinite programming for the minimum k-partition problem. *Annals of Operations Research*, 188(1):155–174, 2011.

[18] M. Jaggi. Revisiting Frank-Wolfe: Projection-free sparse convex optimization. In *Proceedings of the 30th International Conference on Machine Learning*, pages 427–435, 2013.

[19] M. Journée, F. Bach, P.-A. Absil, and R. Sepulchre. Low-rank optimization on the cone of positive semidefinite matrices. *SIAM Journal on Optimization*, 20(5):2327–2351, 2010.

[20] V. Kann, S. Khanna, J. Lagergren, and A. Panconesi. *On the Hardness of Approximating* MAX-K-CUT *and Its Dual.* Citeseer, 1995.

[21] D. Karger, R. Motwani, and M. Sudan. Approximate graph coloring by semidefinite programming. *Journal of the ACM (JACM)*, 45(2):246–265, 1998.

[22] R. Kyng, J. Pachocki, R. Peng, and S. Sachdeva. A framework for analyzing resparsification algorithms. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 2032–2043. SIAM, 2017.

[23] F. Ma and J.-K. Hao. A multiple search operator heuristic for the MAX-K-CUT problem. *Annals of Operations Research*, 248(1-2):365–403, 2017.

[24] R. M. McCutchen and S. Khuller. Streaming algorithms for k-center clustering with outliers and with anonymity. In *Approximation, Randomization and Combinatorial Optimization. Algorithms and Techniques*, pages 165–178. Springer, 2008.

[25] N. Mladenović and P. Hansen. Variable neighborhood search. *Computers & Operations Research*, 24(11): 1097–1100, 1997.

[26] A. Newman. Complex semidefinite programming and MAX-K-CUT. *arXiv preprint arXiv:1812.10770*, 2018.

[27] N. Shinde, V. Narayanan, and J. Saunderson. Memory-efficient structured convex optimization via extreme point sampling. *arXiv preprint arXiv:2006.10945*, 2020.

[28] D. A. Spielman and S.-H. Teng. Spectral sparsification of graphs. *SIAM Journal on Scientific Computing*, 40(4):981–1025, 2011.

[29] A. P. Subramanian, H. Gupta, S. R. Das, and J. Cao. Minimum interference channel assignment in multiradio wireless mesh networks. *IEEE Transactions on Mobile Computing*, 7(12):1459–1473, 2008.

[30] C. Swamy. Correlation clustering: maximizing agreements via semidefinite programming. In *SIAM Symposium on Discrete Algorithms*, volume 4, pages 526–527. Citeseer, 2004.

[31] J. A. Tropp, A. Yurtsever, M. Udell, and V. Cevher. Streaming low-rank matrix approximation with an application to scientific simulation. *SIAM Journal on Scientific Computing*, 41(4):A2430–A2463, 2019.

[32] N. Veldt, D. F. Gleich, A. Wirth, and J. Saunderson. Metric-constrained optimization for graph clustering algorithms. *SIAM Journal on Mathematics of Data Science*, 1(2):333–355, 2019.

[33] Y. Wang, L. Xu, Y. Chen, and H. Wang. A scalable approach for general correlation clustering. In *International Conference on Advanced Data Mining and Applications*, pages 13–24. Springer, 2013.

[34] D. P. Woodruff. Sketching as a tool for numerical linear algebra. *Foundations and Trends® in Theoretical Computer Science*, 10(1–2):1–157, 2014.

[35] Y. Ye. GSET random graphs. `https://www.cise.ufl.edu/research/sparse/matrices/Gset/`, 2003. [accessed April-2021].

[36] A. Yurtsever, M. Udell, J. A. Tropp, and V. Cevher. Sketchy decisions: Convex low-rank matrix optimization with optimal storage. In *Artificial Intelligence and Statistics*, pages 1188–1196, 2017.

[37] A. Yurtsever, J. A. Tropp, O. Fercoq, M. Udell, and V. Cevher. Scalable semidefinite programming. *SIAM Journal on Mathematics of Data Science*, 3(1):171–200, 2021.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes] The abstract provides an overview of the results. See Section 1.3 (Contributions) for the summary of the results.
   (b) Did you describe the limitations of your work? [Yes] See Section 7
   (c) Did you discuss any potential negative societal impacts of your work? [No] Our method, while memory-efficient and simple to implement, is not yet a practical algorithm, owing to slow convergence. While clustering algorithms can potentially be misused, our contributions do not seem to have direct impact on large-scale applications.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes] The assumptions are provided in the statements of the results.
   (b) Did you include complete proofs of all theoretical results? [Yes] The proofs of all the results are provided in the supplementary material.
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] The code and data are included in the Supplementary Material. We will include the de-anonymized URL to the github repository containing the code and the data in the camera-ready version of the accepted paper.

(b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] All parameters choices for our experiments are explicitly given and they are chosen according to theory in the paper.

(c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [No] We have reported the best solution value and explicitly stated the results with reference to the best value as this appropriate from the context.

(d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] We have used MATLAB 2021b and performed the experiments on a local machine with 8GB RAM and 4 cores. We have also included this information in Section 6.

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...

(a) If your work uses existing assets, did you cite the creators? [Yes] We have used graphs from GSET dataset [35] for numerical experiments. We have added the citation accordingly in Section 6 (Computational results).

(b) Did you mention the license of the assets? [No] The license situation of the assets is not clear. However, the dataset has been widely used for over 18 years, and is posted publicly in multiple places.

(c) Did you include any new assets either in the supplemental material or as a URL? [Yes] We have added a link (URL) to the code which we created and used to run the experiment.

(d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A] GSET comprises of collection of randomly generated graphs, and does not contain personal information/data.

(e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A] GSET comprises of randomly generated graphs and as such, is not deemed to have personal data in any way.

5. If you used crowdsourcing or conducted research with human subjects...

(a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]

(b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]

(c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]