Is the Algorithmic Kadison-Singer Problem Hard?

Ben Jourdan
University of Edinburgh, UK

Peter Macgregor
University of Edinburgh, UK

He Sun
University of Edinburgh, UK

Abstract

We study the following $\mathsf{KS}_2(c)$ problem: let $c \in \mathbb{R}^+$ be some constant, and $v_1, \ldots, v_m \in \mathbb{R}^d$ be vectors such that $\|v_i\|^2 \leq \alpha$ for any $i \in [m]$ and $\sum_{i=1}^m \langle v_i, x\rangle^2 = 1$ for any $x \in \mathbb{R}^d$ with $\|x\| = 1$. The $\mathsf{KS}_2(c)$ problem asks to find some $S \subset [m]$, such that it holds for all $x \in \mathbb{R}^d$ with $\|x\| = 1$ that

$$\left|\sum_{i\in S}\langle v_i, x\rangle^2 - \frac{1}{2}\right| \leq c \cdot \sqrt{\alpha},$$

or report no if such S doesn't exist. Based on the work of Marcus et al. [15] and Weaver [20], the $\mathsf{KS}_2(c)$ problem can be seen as the algorithmic Kadison-Singer problem with parameter $c \in \mathbb{R}^+$.

Our first result is a randomised algorithm with one-sided error for the $\mathsf{KS}_2(c)$ problem such that (1) our algorithm finds a valid set $S \subset [m]$ with probability at least $1 - 2/d$, if such S exists, or (2) reports no with probability 1, if no valid sets exist. The algorithm has running time

$$O\left(\binom{m}{n} \cdot \mathrm{poly}(m, d)\right) \quad \text{for} \quad n = O\left(\frac{d}{\epsilon^2}\log(d)\log\left(\frac{1}{c\sqrt{\alpha}}\right)\right),$$

where ϵ is a parameter which controls the error of the algorithm. This presents the first algorithm for the Kadison-Singer problem whose running time is quasi-polynomial in m in a certain regime, although having exponential dependency on d. Moreover, it shows that the algorithmic Kadison-Singer problem is easier to solve in low dimensions. Our second result is on the computational complexity of the $\mathsf{KS}_2(c)$ problem. We show that the $\mathsf{KS}_2\left(1/\left(4\sqrt{2}\right)\right)$ problem is FNP-hard for general values of d, and solving the $\mathsf{KS}_2\left(1/\left(4\sqrt{2}\right)\right)$ problem is as hard as solving the $\mathsf{NAE\text{-}3SAT}$ problem.

2012 ACM Subject Classification Mathematics of computing → Probabilistic algorithms

Keywords and phrases Kadison-Singer problem, spectral sparsification

Digital Object Identifier 10.4230/LIPIcs.ISAAC.2023.43

Related Version *Full Version*: https://arxiv.org/abs/2205.02161

Funding This work is supported by an EPSRC Early Career Fellowship (EP/T00729X/1).

1 Introduction

The Kadison-Singer problem [13] posed in 1959 asks whether every pure state on the (abelian) von Neumann algebra $\mathbb{D}$ of bounded diagonal operators on ℓ_2 has a unique extension to a pure state on $B(\ell_2)$, the von Neumann algebra of all bounded linear operators on the Hilbert space ℓ_2. The statement of the Kadison-Singer problem arises from work on the foundations of quantum mechanics done by Dirac in 1940s, and has been subsequently shown to be equivalent to numerous important problems in pure mathematics, applied mathematics, engineering and computer science [8]. Weaver [20] shows that the Kadison-Singer problem is equivalent to the following discrepancy question, which is originally posed as a conjecture.

▶ **Conjecture 1** (The KS_2 Conjecture). *There exist universal constants $\eta \geq 2$ and $\theta > 0$ such that the following holds. Let $v_1, \ldots, v_m \in \mathbb{C}^d$ satisfy $\|v_i\| \leq 1$ for all $i \in [m]$, and suppose $\sum_{i=1}^m |\langle u, v_i\rangle|^2 = \eta$ for every unit vector $u \in \mathbb{C}^d$. Then, there exists a partition S_1, S_2 of $[m]$ so that $\sum_{i \in S_j} |\langle u, v_i\rangle|^2 \leq \eta - \theta$, for every unit vector $u \in \mathbb{C}^d$ and every $j = \{1, 2\}$.*

As a major breakthrough in mathematics, Marcus, Spielman and Srivastava [15] prove that the KS_2 conjecture holds, and give an affirmative answer to the Kadison-Singer problem. Specifically, in this celebrated paper they show that, for any vectors $v_1, \ldots, v_m \in \mathbb{C}^d$ such that $\|v_i\|^2 \leq \alpha$ for any $i \in [m]$ and $\sum_{i=1}^m \langle v_i, x\rangle^2 = 1$ for any $x \in \mathbb{C}^d$ with $\|x\| = 1$, there is a partition S_1, S_2 of $[m]$ such that it holds for any $x \in \mathbb{C}^d$ with $\|x\| = 1$ and $j = 1, 2$ that $\left|\sum_{i \in S_j} \langle v_i, x\rangle^2 - 1/2\right| \leq 3 \cdot \sqrt{\alpha}$. The proof of this result is based on studying interlacing families of polynomials [14]. While analysing interlacing families of polynomials suffices to answer the KS_2 conjecture and, as a consequence, solve the Kadison-Singer problem, it is unclear if their existential proof on the partition guaranteed by the KS_2 conjecture can be turned into an efficient algorithmic construction; designing efficient algorithms for the Kadison-Singer problem is listed as a natural open question in [15]. This question is particularly interesting in theoretical computer science, since it is directly linked to constructing unweighted spectral sparsifiers [5] and spectrally thin trees [1], among many other applications in approximation algorithms. However, there has been little work on the algorithmic Kadison-Singer problem, and the complexity status of this problem is an important open question.

To address this question, we study the following KS_2 problem with some constant $c \in \mathbb{R}^+$:

▶ **Problem 2** (The $\mathsf{KS}_2(c)$ problem). *Given vectors $v_1, \ldots, v_m \in \mathbb{R}^d$ such that $\|v_i\|^2 \leq \alpha$ for any $i \in [m]$ and $\sum_{i=1}^m \langle v_i, x\rangle^2 = 1$ for any $x \in \mathbb{R}^d$ with $\|x\| = 1$, the $\mathsf{KS}_2(c)$ problem asks to*

- *find some $S \subset [m]$, such that it holds for all $x \in \mathbb{R}^d$ with $\|x\| = 1$ that*

$$\left|\sum_{i \in S} \langle v_i, x\rangle^2 - \frac{1}{2}\right| \leq c \cdot \sqrt{\alpha}, \tag{1}$$

- *or report no if such S doesn't exist.*

Notice that the KS_2 conjecture is equivalent to finding some subset $S \subset [m]$ as stated in Problem 2 for some constant c. Here we choose to formulate the discrepancy of any set $S \subset [m]$ in (1) as $c \cdot \sqrt{\alpha}$ for three reasons: first of all, Weaver [20] shows that the dependency on $O(\sqrt{\alpha})$ in (1) is tight, so the term $O(\sqrt{\alpha})$ is unavoidable when bounding the discrepancy; secondly, the KS_2 conjecture shows that the existence of any universal constant c in (1) suffices to prove the Kadison-Singer conjecture, and it is proven in [15] that the KS_2 conjecture holds for $c = 3$; however, studying the tightness of this constant remains an interesting open question on its own (Problem 8.1, [7]). Finally, as we will show shortly, the $\mathsf{KS}_2(c)$ problem belongs to different complexity classes with respect to different values of c, so introducing this parameter c allows us to better understand the complexity of the algorithmic Kadison-Singer problem.

1.1 Our Results

Our first result is an algorithm called RANDOMISED-KS$(\{v_i\}, c, \epsilon)$ for approximately solving the $\mathsf{KS}_2(c)$ problem for general values of c. For any constant c, $\epsilon < 1$, and any vectors $v_1, \ldots, v_m \in \mathbb{R}^d$ such that $\|v_i\|^2 \leq \alpha$ for all $i \in [m]$, we show that (i) if there exists an

S which satisfies (1), then with probability at least $(1 - 2/d)$ the algorithm returns a set $S' \subset \{v_i\}_{i=1}^m$ that satisfies

$$(1-\epsilon)\left(\frac{1}{2} - c\sqrt{\alpha}\right) \leq \sum_{v \in S'} \langle v, x\rangle^2 \leq (1+\epsilon)\left(\frac{1}{2} + c\sqrt{\alpha}\right) \tag{2}$$

for all unit vectors $x \in \mathbb{R}^d$, and (ii) if no set exists which satisfies (2), then with probability 1 the algorithm returns "no". Our result is summarised as follows:

▶ **Theorem 3.** *There is an algorithm,* RANDOMISED-$KS(\mathcal{I}, c, \epsilon)$*, such that for any instance* $\mathcal{I} \triangleq \{v_i\}_{i=1}^m$ *of the* $\mathsf{KS}_2(c)$ *problem with* $v_i \in \mathbb{R}^d$ *for* $d \geq 3$*, and for any* $\epsilon \in (0, 1)$*, the following holds:*

- *if there exists a set* $S \subset \mathcal{I}$ *such that*

$$\left(\frac{1}{2} - c\sqrt{\alpha}\right) \leq \sum_{v \in S} \langle v, x\rangle^2 \leq \left(\frac{1}{2} + c\sqrt{\alpha}\right)$$

  for all unit vectors $x \in \mathbb{R}^d$*, then with probability at least* $(1 - 2/d)$*, the* RANDOMISED-$KS(\mathcal{I}, c, \epsilon)$ *algorithm returns a subset* $S' \subset \mathcal{I}$ *which satisfies* (2) *for all unit vectors* $x \in \mathbb{R}^d$*.*
- *if there is no set* $S \subset \mathcal{I}$ *which satisfies* (2)*, then with probability* 1*, the* RANDOMISED-$KS(\mathcal{I}, c, \epsilon)$ *algorithm reports that no such set exists.*

The algorithm has running time

$$O\left(\binom{m}{n} \cdot \mathrm{poly}(m, d)\right) \quad \text{for} \quad n \triangleq O\left(\frac{d}{\epsilon^2}\log(d)\max\left(\log\left(\frac{1}{c\sqrt{\alpha}}\right), \log\left(\frac{1}{(1/2) - c\sqrt{\alpha}}\right)\right)\right).$$

▶ Remark 4. Since the most interesting instances of the $\mathsf{KS}_2(c)$ problem are the cases in which $1/2 + c\sqrt{\alpha}$ is bounded away from 1, we can assume that $c\sqrt{\alpha} \leq 1/2 - \sigma$ for some constant σ which implies that

$$n = O\left(\frac{d}{\epsilon^2}\log(d)\log\left(\frac{1}{c\sqrt{\alpha}}\right)\right).$$

Combining this with $d = \sum_{i=1}^m \|v_i\|^2 \leq \alpha m$, a constraint due to the isotropic nature of the input, shows that our algorithm runs in quasi-polynomial time in m when $d = O(\mathrm{polylog}(m))$.

Compared with the state-of-the-art that runs in $d^{O(m^{1/3}\alpha^{-1/4})}$ time [2], the most appealing fact of Theorem 3 is that it shows the $\mathsf{KS}_2(c)$ problem can be approximately solved in quasi-polynomial time when $d = O(\mathrm{poly}\log m)$. Moreover, for small values of c where a subset $S \subset [m]$ satisfying (1) isn't guaranteed to exist, our algorithm, with the same time complexity, is still able to find an S satisfying (2) with high probability if it exists, or report no with probability 1 otherwise. These two facts together show that both determining the existence of a valid subset S and finding such S are computationally much easier in low dimensions, regardless of the range of c. In addition, our result is much stronger than a random sampling based algorithm, which only works in the regime of $\alpha = O(1/\log d)$ [19], while our algorithm works even when there are vectors with much larger norm, e.g., $\alpha = \Theta(1)$. On the other side, like many optimisation problems that involve the dimension of input items in their formulation (e.g., multi-dimensional packing [10], and vector scheduling [4]), Theorem 3 indicates that the order of d might play a significant role in the hardness of the $\mathsf{KS}_2(c)$ problem, and the hard instances of the problem might be in the regime of $m = O(d)$.

Inspired by this, we study the computational complexity of the $\mathsf{KS}_2(c)$ problem for general values of d, where the number of input vectors satisfies $m = O(d)$. In order to study the "optimal" partitioning, for a given instance of the problem $\mathcal{I} = \{v_1, \ldots, v_m\}$, let

$$\mathcal{W}(\mathcal{I}) \triangleq \min_{S \subset \mathcal{I}} \max_{\substack{x \in \mathbb{R}^d \\ \|x\|=1}} \left| \sum_{v \in S} \langle v, x \rangle^2 - \frac{1}{2} \right|.$$

Then, we choose $c = 1/(4\sqrt{2})$ and notice that, for any vectors that satisfy the conditions of the $\mathsf{KS}_2(c)$ problem, there could be no subset S satisfying (1) for such c. As our second result, we prove that, for any $c \leq 1/(4\sqrt{2})$, distinguishing between instances for which $\mathcal{W}(\mathcal{I}) = 0$ and those for which $\mathcal{W}(\mathcal{I}) \geq c \cdot \sqrt{\alpha}$ is NP-hard. Our result is as follows:

▶ **Theorem 5.** *The* $\mathsf{KS}_2\left(1/\left(4\sqrt{2}\right)\right)$ *problem is* FNP*-hard for general values of* d*. Moreover, it is* NP*-hard to distinguish between instances of the* $\mathsf{KS}_2(c)$ *problem with* $\mathcal{W}(\mathcal{I}) = 0$ *from instances with* $\mathcal{W}(\mathcal{I}) \geq \left(1/4\sqrt{2}\right) \cdot \sqrt{\alpha}$*.*

▶ Remark 6. It's important to note that, when d is constant, the decision problem in Theorem 5 can be solved in polynomial time. For example, the 1-dimensional problem is equivalent to the PARTITION problem, in which we are given a set of real numbers $\mathcal{I} = \{x_1, \ldots, x_m\}$ such that $\sum_i x_i = 1$ and need to determine whether there is a subset $S \subset \mathcal{I}$ such that $\sum_{x \in S} x = 1/2$. In this setting,

$$\mathcal{W}(\mathcal{I}) = \min_{S \subset \mathcal{I}} \left| \left(\sum_{x \in S} x \right) - 1/2 \right|.$$

There is a well-known FPTAS for PARTITION which can distinguish between instances for which $\mathcal{W}(\mathcal{I}) = 0$ and those for which $\mathcal{W}(\mathcal{I}) \geq \epsilon$, for any $\epsilon > 0$. Theorem 5 implies that there is no such FPTAS for the optimisation version of the $\mathsf{KS}_2(c)$ problem for general d.

Theorem 5 shows that the isotropic structure of the $\mathsf{KS}_2(c)$ instance is not sufficient to make finding a partition easy when compared with similar problems. As such, the design of a potential polynomial-time algorithm for the Kadison-Singer problem would need to take some range of c into account and cannot solve the optimisation version of the $\mathsf{KS}_2(c)$ problem, otherwise one would end up solving an NP-hard problem. We remark that Theorem 5 shares the same style as the one for Spencer's Discrepancy Problem: given any input on N elements, Charikar et al. [9] shows that it is NP-hard to distinguish between the input with discrepancy zero and the one with discrepancy $\Omega(\sqrt{N})$, although it is known that a solution with $O(\sqrt{N})$ approximation can be computed efficiently [3].

1.2 Our Techniques

In this subsection we sketch our main techniques used in proving Theorems 3 and 5.

Proof Sketch of Theorem 3. We start by sketching the ideas behind our algorithmic result. First of all, it is easy to see that we can solve the $\mathsf{KS}_2(c)$ problem for any $c \in \mathbb{R}^+$ in $O\left(2^m \cdot \mathrm{poly}(m, d)\right)$ time, since we only need to enumerate all the 2^m subsets $S \subseteq \mathcal{I}$ of the input set $\mathcal{I}$ and check if every possible set S satisfies the condition (1). To express all the subsets of $\mathcal{I}$, we inductively construct level sets $\{\mathcal{L}_i\}_{i=0}^m$ with $\mathcal{L}_i \subseteq 2^{\mathcal{I}}$ as follows:

- initially, level $i = 0$ consists of a single set $\emptyset$, and we set $\mathcal{L}_0 = \{\emptyset\}$;
- based on $\mathcal{L}_{i-1}$ for any $1 \leq i \leq m$, we define $\mathcal{L}_i$ by $\mathcal{L}_i \triangleq \{S, S \cup \{v_i\} : S \in \mathcal{L}_{i-1}\}$.

It is important to see that, although $|\mathcal{L}_i|$ could be as high as 2^m, there are only m such level sets $\mathcal{L}_i$, which are constructed inductively in an *online* manner, and it holds for any $S \subseteq \mathcal{I}$ that $S \in \mathcal{L}_m$.

The bottleneck for improving the efficiency of this simple enumeration algorithm is the number of sets in $\mathcal{L}_m$, which could be exponential in m. To overcome this bottleneck, we introduce the notion of *spectral equivalence classes* to reduce $|\mathcal{L}_i|$ for any $i \in [m]$. Informally speaking, if there are different $S_1, S_2 \in \mathcal{L}_i$ for any $i \in [m]$ such that[1]

$$(1-\epsilon)\sum_{j\in S_2} v_j v_j^\intercal \preceq \sum_{j\in S_1} v_j v_j^\intercal \preceq (1+\epsilon)\sum_{j\in S_2} v_j v_j^\intercal$$

for some small ϵ, then we view S_1 and S_2 to be "spectrally equivalent" to each other[2]. It suffices to use one set to represent all of its spectral equivalences; hence, we only need to store the subsets which aren't spectrally equivalent to each other[3]. Since there is a spectral sparsifier of any S with $O(d\log(d)/\epsilon^2)$ vectors [11, 17], we can reduce the total number of stored subsets (i.e., the number of spectral equivalence classes) in $\mathcal{L}_i$ for any $i \in [m]$ to $\binom{m}{n}$ where $n = O\big(d\log(d)/\epsilon^2\big)$ which is no longer exponential in m.

Turning this idea into an algorithm design, we need be careful that the small approximation error introduced by every constructed spectral sparsifier does not compound as we construct sparsifiers from one level to another. In order to avoid this, we employ the online vector sparsification algorithm presented in [11]. This allows us to construct sparsifiers in $\mathcal{L}_i$ from the ones in $\mathcal{L}_{i-1}$ and the vector v_i. In addition, the construction in each level preserves the same approximation error as the previous one.

We highlight that the design of our algorithm for solving the $\mathsf{KS}_2(c)$ problem is entirely different from the previous work, which is based on analysing the properties of interlacing polynomials [2]. Moreover, one can view our use of online spectral sparsifiers in constructing spectral equivalence classes as an *encoding* strategy to reduce the enumeration space of the $\mathsf{KS}_2(c)$ problem. From this aspect, our work sheds light on potential applications of other tools well-studied in algorithmic spectral graph theory and numerical linear algebra, such as sparsification and sketching.

Proof Sketch of Theorem 5. Our proof of the FNP-hardness of the $\mathsf{KS}_2\left(1/\left(4\sqrt{2}\right)\right)$ problem is based on a reduction from the well-known NAE-3SAT problem [12] to a decision version of the $\mathsf{KS}_2\left(1/\left(4\sqrt{2}\right)\right)$ problem, which asks whether $\mathcal{W}(\mathcal{I}) = 0$ or $\mathcal{W}(\mathcal{I}) \geq \left(1/\left(4\sqrt{2}\right)\right)\sqrt{\alpha}$. Our overall reduction consists of two steps: we first build a reduction from the NAE-3SAT problem to the so-called NAE-3SAT-KS problem, and then build a reduction from the NAE-3SAT-KS problem to the $\mathsf{KS}_2\left(1/\left(4\sqrt{2}\right)\right)$ problem.

To sketch the first reduction, we examine the so-called NAE-3SAT-KS problem, which can be viewed as a restricted version of the NAE-3SAT problem, and used only as a tool to build the reduction from the NAE-3SAT problem to the $\mathsf{KS}_2\left(1/\left(4\sqrt{2}\right)\right)$ problem. Informally, the NAE-3SAT-KS problem consists of the 3SAT Boolean formula ψ, in which the number of occurrences of both u and $\bar{u}$ for every variable u in any ψ is limited with respect to some additional constraints and any two clauses of ψ share at most one literal; the NAE-3SAT-KS problem asks if there is a satisfying assignment for ψ such that every clause of ψ has at

[1] For any two matrices A and B of the same dimension, we write $A \preceq B$ if $B - A$ is positive semi-definite.
[2] Although this relationship is not symmetric, this informal definition is sufficient for the proof sketch and is not used directly in our analysis.
[3] The list of stored subsets can be thought of as an epsilon cover of all possible subsets.

least one true literal and at least one false literal; we refer the reader to Problem 11 in Section 3 for the formal definition of the NAE-3SAT-KS problem. Based on a reduction from the NAE-3SAT problem, we show that the NAE-3SAT-KS problem is NP-complete.

For the second and main reduction of our analysis, we build a reduction from the NAE-3SAT-KS problem to the $\mathsf{KS}_2\left(1/\left(4\sqrt{2}\right)\right)$ problem. Specifically, for an NAE-3SAT-KS instance ψ of n variables and m clauses, we construct a set A of $\Theta(n+m)$ vectors as a $\mathsf{KS}_2\left(1/\left(4\sqrt{2}\right)\right)$ instance, and each $v \in A$ has dimension $n+m$, such that the following properties hold:

- every vector v has norm $\|v\|^2 \leq 1/4$ and $\sum_{v\in A} vv^\intercal = I$;
- if ψ is a satisfiable instance of NAE-3SAT-KS, then there is a subset $S \subset A$ such that $\sum_{v\in S} vv^\intercal = (1/2)\cdot I$;
- if ψ is not a satisfiable instance of NAE-3SAT-KS, then for any subset $S \subset A$ there is always some $y \in \mathbb{R}^n$ with $\|y\| = 1$ such that $\left|\sum_{v\in S}\langle v, y\rangle^2 - 1/2\right| \geq 1/\left(8\sqrt{2}\right)$.

The key to proving these properties is the construction of a KS instance $\mathcal{I}$ from any formula ψ, and an analysis of the properties of $\sum_{v\in S} vv^\intercal$ for any $S \subseteq \mathcal{I}$ if ψ is an unsatisfiable instance of NAE-3SAT-KS. We think that such a reduction from any SAT instance to a KS instance is quite novel, and might be further employed to sharpen the constant $1/(4\sqrt{2})$.

1.3 Related Work

There has been little work on the algorithmic Kadison-Singer problem. Anari et al. [2] studies approximating the largest root of a real rooted polynomial and its applications to interlacing families, which are the main tool developed in [15] to prove the Kadison-Singer conjecture. They show that a valid partition promised by Weaver's KS_2 conjecture can be found in $d^{O(m^{1/3}\alpha^{-1/4})}$ time, suggesting that exhaustive search of all possibilities is not required for the algorithmic Kadison-Singer problem. Becchetti et al. [6] studies the algorithmic Kadison-Singer problem for graphs under some restricted condition. Specifically, they show that, if $G = (V, E)$ is an n-vertex and Δ-regular graph of $\Delta = \Omega(n)$ and the second eigenvalue of the adjacency matrix of G is at most a sufficient small constant times Δ, then an unweighted spectral sparsifier of G can be constructed efficiently.

Weaver [21] shows that the BSS-framework for constructing linear-sized spectral sparsifiers [5] can be adapted for the one-sided Kadison-Singer problem, where the term "one-sided" refers to the fact that the discrepancy of the algorithm's output can be only upper bounded.

Finally, independent of our work, Spielman and Zhang [18] studies the same complexity problem as ours. Different from our approach, their analysis starts with the $(3, 2\text{-}2)$ Set Splitting problem, which is a variant of the 2-2 Set Splitting problem. They prove that the $(3, 2\text{-}2)$ Set Splitting problem remains NP-hard even if no pair of sets intersects in more than one variable. Applying this, they show that the $\mathsf{KS}_2(c)$ problem is NP-hard for $c = 1/4$. While their result is slightly tighter than ours with respect to the value of c, the conclusions of the two works are essentially the same.

1.4 Notation

Let $[m] \triangleq \{1, \ldots, m\}$. For any integer j, we define vector $\mathbf{1}_j$, in which $\mathbf{1}_j(j) = 1$ and all of $\mathbf{1}_j$'s other entries are 0. For any integer $d \geq 1$, let $\mathbf{0}_{d\times d} \in \mathbb{R}^{d\times d}$ be the matrix in which every entry is equal to 0. We call a matrix A positive semi-definite (PSD) if $x^\intercal Ax \geq 0$ holds for any $x \in \mathbb{R}^d$. For any two matrices A and B, we write $A \preceq B$ if $B - A$ is PSD. The spectral norm of any matrix A is expressed by $\|A\|$.

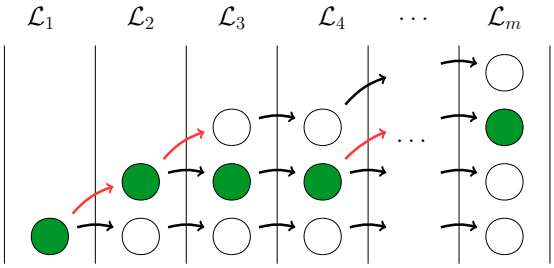

Figure 1 The construction of the sets $\mathcal{L}_i$ in Algorithm 1. Each $\mathcal{L}_{i-1}$ contains sparsifiers representing the spectral equivalence classes of the vectors $\{v_1, \ldots, v_{i-1}\}$. Then, $\mathcal{L}_i$ contains either one or two "children" of each sparsifier in $\mathcal{L}_{i-1}$, where the second child is added with some small probability which prevents $|\mathcal{L}_m|$ from growing exponentially with m. For a particular target subset $S \subseteq \{v_1, \ldots v_m\}$, there is some sequence of constructed sparsifiers which corresponds to the process of the online algorithm for constructing spectral sparsifiers [11], applied to S.

2 Algorithm Based on Spectral Equivalence Classes

This section discusses in detail the construction of spectral equivalence classes, and its application in designing a randomised algorithm for the $\mathsf{KS}_2(c)$ problem. We analyse the presented algorithm, and prove Theorem 3. All the proofs omitted from this section can be found in Appendix A.

2.1 Algorithm

Our algorithm consists of m iterations: in iteration i, the algorithm constructs the set $\mathcal{L}_i$ of spectral equivalence classes for the subsets $S \subseteq \{v_1, \ldots, v_i\}$. For each equivalence class, $\mathcal{L}_i$ contains a pair (S, B) where $S \subseteq \{v_1, \ldots v_i\}$ is a representative set in the equivalence class and $B \in \mathbb{R}^{d \times d}$ is a spectral sparsifier representing the equivalence class. Moreover, the algorithm constructs the representations of spectral equivalence classes in iteration i based on the ones maintained in iteration $i-1$. That is, instead of constructing all the subsets of $\{v_1, \ldots, v_i\}$ and grouping them into different spectral equivalence classes, the algorithm directly constructs the representations of the spectral equivalence classes of $\{v_1, \ldots, v_i\}$ based on its constructed equivalence classes of $\{v_1, \ldots, v_{i-1}\}$. This can be achieved by applying an online algorithm for constructing spectral sparsifiers, since, if we assume that in iteration $i-1$ every subset $S \subseteq \{v_1, \ldots, v_{i-1}\}$ is spectrally equivalent to some $(S', B') \in \mathcal{L}_{i-1}$ maintained by the algorithm, then both of S and $S \cup \{v_i\}$ are spectrally equivalent to S' and $S' \cup \{v_i\}$ in iteration i as well. As such, in iteration i we only need to ensure that the sets S' and $S' \cup \{v_i\}$ are still represented by some sparsifiers in $\mathcal{L}_i$.

Based on this, we can view all the vectors $v_1, \ldots, v_m$ as arriving *online* and, starting with the trivial spectral equivalence class defined by $\mathcal{L}_0 = \{(\emptyset, \mathbf{0}_{d \times d})\}$, the algorithm constructs the representations of spectral equivalence classes of $\{v_1, \ldots v_i\}$ in iteration i. Our algorithm applies the online algorithm for constructing spectral sparsifiers [11] (Lines 13-18 of Algorithm 1) to construct the representations of spectral equivalence classes of $\{v_1, \ldots, v_i\}$ based on those of $\{v_1, \ldots, v_{i-1}\}$. Since any subset of $\{v_1, \ldots v_m\}$ is spectrally equivalent to some set of vectors with size n where n is nearly linear in d [11], the number of spectral equivalence classes in any set $\mathcal{L}_i$ will be at most $\binom{m}{n}$. See Figure 1 for an illustration of the construction of the sets $\mathcal{L}_i$ and Algorithm 1 for the formal description of the algorithm.

▶ Remark 7. The if-condition on Line 10 of Algorithm 1 can be checked in polynomial time while introducing an arbitrarily small error, by constructing the matrix $\sum_{v \in S'} vv^\intercal$ and computing its eigenvalues.

Algorithm 1 RANDOMISED-KS($\mathcal{I} = \{v_i\}_{i=1}^m, c, \epsilon$), where $v_i \in \mathbb{R}^d$ and $\|v_i\|^2 \le \alpha$.

$\mu \leftarrow \epsilon/6$
$\lambda \leftarrow \min\left(c\sqrt{\alpha}, 1/2 - c\sqrt{\alpha}\right)$
$b \leftarrow 8\log(d)/\mu^2$
$n \leftarrow O\big(d\log(d)\log(1/\lambda)/\mu^2\big)$
$\mathcal{L}_0 \leftarrow \{(\emptyset, \mathbf{0}_{d\times d})\}$
for $i \leftarrow 1$ to m **do**
  $\mathcal{L}_i \leftarrow \emptyset$
  for $(S, B) \in \mathcal{L}_{i-1}$ *and* B *constructed with at most* n *vectors* **do**
    $S' \leftarrow S \cup \{v_i\}$
    if S' *satisfies* (2) **then**
      return S'
    end
    $p \leftarrow \min\left(b\,(1+\mu)\,v_i^\intercal\,(B+\lambda I)^{-1}\,v_i, 1\right)$
    if $X \le p$ *where* $X \sim \mathrm{Uniform}[0,1]$ **then**
      $B' \leftarrow B + \frac{1}{p}v_i v_i^\intercal$
      $\mathcal{L}_i \leftarrow \mathcal{L}_i \cup \{(S, B), (S', B')\}$
    else
      $\mathcal{L}_i \leftarrow \mathcal{L}_i \cup \{(S', B)\}$
    end
  end
end
return FAILURE

2.2 Analysis

First of all, notice that sparsifying $\sum_{v\in S} vv^\intercal$ for any $S \subseteq \mathcal{I}$ is equivalent to sparsifying the $|S| \times d$ matrix whose rows are defined by all the $v \in S$. Based on this, our proof uses the result from the online matrix sparsification algorithm [11] as a black box. Specifically, we apply the following lemma in our analysis, which is a special case of Theorem 2.3 from [11]. Notice that the algorithm described in Lemma 8 below corresponds to the sampling scheme used in Algorithm 1.

▶ **Lemma 8** ([11], Theorem 2.3). *Let S be a set of vectors $v_1, \ldots, v_m \in \mathbb{R}^d$, and let $A = \sum_{v\in S} vv^\intercal$. With $\mu, \delta \in [0,1]$, $b \triangleq 8\log(d)/\mu^2$ and $B_0 = \mathbf{0}_{d\times d}$, construct B_i inductively for $i \in [m]$ such that with probability*

$$p_i = \min\left(b(1+\mu)v_i^\intercal\left(B_{i-1} + \frac{\delta}{\mu}I\right)^{-1} v_i, 1\right),$$

we have

$$B_i = B_{i-1} + \frac{1}{p_i}v_i v_i^\intercal,$$

and with probability $1 - p_i$, we have $B_i = B_{i-1}$. Then, it holds with probability $(1 - 1/d)$ that

$$(1-\mu)A - \delta I \preceq B_m \preceq (1+\mu)A + \delta I,$$

and the number of vectors added to B_m is $O\left(d\log d\log\left(\mu\|A\|^2/\delta\right)/\mu^2\right)$.

Now, we analyse Algorithm 1. We begin by showing that, for each pair (S, B) constructed by Algorithm 1, B is a spectral sparsifier of S with high probability.

▶ **Lemma 9.** *Let $\mathcal{L}_i$ be the set constructed by Algorithm 1 at iteration i. Then, for any $(S^\star, B^\star) \in \mathcal{L}_i$, it holds with probability $(1 - 1/d)$ that*

$$(1-\mu)A_{S^\star} - \delta I \preceq B^\star \preceq (1+\mu)A_{S^\star} + \delta I$$

where $A_{S^\star} = \sum_{v \in S^\star} vv^\intercal$, and the parameters are set in Algorithm 1 to be $\mu = \epsilon/6$ and $\delta = \mu \min(c\sqrt{\alpha}, 1/2 - c\sqrt{\alpha})$.

Next we show that *any* set $S \subset \{v_1, \ldots, v_m\}$ is well approximated by one of the sparsifiers constructed in Algorithm 1.

▶ **Lemma 10.** *Let $\mathcal{I} = \{v_i\}_{i=1}^m$ be the input to Algorithm 1. Let $S \subseteq \mathcal{I}$ be any fixed set, and $A = \sum_{v \in S} vv^\intercal$. Then, with probability $(1 - 1/d)$, there is a matrix B constructed by Algorithm 1 such that*

$$(1-\mu)A - \delta I \preceq B \preceq (1+\mu)A + \delta I,$$

where $\mu = \epsilon/6$ and $\delta = \mu \min(c\sqrt{\alpha}, 1/2 - c\sqrt{\alpha})$.

Finally, to prove Theorem 3, we need only apply Lemma 10 for the target set $S \subset \mathcal{I}$, and Lemma 9 for one of the pairs (S', B) constructed by the algorithm. In particular, we do not need to take the union bound over all sparsifiers constructed by the algorithm; rather, it is sufficient that an accurate sparsifier is constructed for one specific target set.

Proof of Theorem 3. We first look at the case in which there is some $S \subset \mathcal{I}$, such that for $A_S = \sum_{i \in S} v_i v_i^\intercal$ it holds that $1/2 - c\sqrt{\alpha} \le x^\intercal A_S x \le 1/2 + c\sqrt{\alpha}$, for all unit vectors $x \in \mathbb{R}^d$. By Lemma 10, with probability greater than or equal to $1 - 1/d$, there exists some pair $(S', B) \in \mathcal{L}_m$ such that

$$(1-\mu)A_S - \delta I \preceq B \preceq (1+\mu)A_S + \delta I, \tag{3}$$

where $\mu = \epsilon/6$ and $\delta = \mu \min(c\sqrt{\alpha}, 1/2 - c\sqrt{\alpha}) \le \mu$. By Lemma 9, with probability $1 - 1/d$, we have $(1-\mu)A_{S'} - \delta I \preceq B \preceq (1+\mu)A_{S'} + \delta I$, where S' is the set constructed alongside B. Taking the union bound, with probability at least $1 - 2/d$, we have for any unit vector $x \in \mathbb{R}^d$ that

$$\begin{aligned} x^\intercal A_{S'} x &\le \frac{1+\mu}{1-\mu}\left(\frac{1}{2} + c\sqrt{\alpha}\right) + \frac{2\delta}{1-\mu} \\ &\le \frac{1+3\mu}{1-\mu}\left(\frac{1}{2} + c\sqrt{\alpha}\right) \\ &\le (1+\epsilon)\left(\frac{1}{2} + c\sqrt{\alpha}\right) \end{aligned} \qquad \text{and} \qquad \begin{aligned} x^\intercal A_{S'} x &\ge \frac{1-\mu}{1+\mu}\left(\frac{1}{2} - c\sqrt{\alpha}\right) - \frac{2\delta}{1-\mu} \\ &\ge \frac{1-3\mu}{1-\mu}\left(\frac{1}{2} - c\sqrt{\alpha}\right) \\ &\ge (1-\epsilon)\left(\frac{1}{2} - c\sqrt{\alpha}\right), \end{aligned}$$

where we use the definition of δ and the fact that $\epsilon = 6\mu \le 1$. Therefore, the set S' satisfies (2) and will be returned by Algorithm 1.

On the other side, notice that, by the condition on Line 10 of Algorithm 1, any set returned by the algorithm satisfies (2). Therefore, with probability 1 the algorithm will correctly report that there is no set $S \subset \mathcal{I}$ satisfying (2) if it is the case.

Finally, we analyse the running time of the algorithm. By Lemma 8, it holds that B is constructed from $O(n)$ vectors with probability at least $1 - 1/d$. For this reason, on Line 8 of Algorithm 1 we consider only the sparsifiers of size $O(n)$. The remaining part of the algorithm contributes only polynomial factors to its running time, so the total running time of the algorithm is $O\left(\binom{m}{n} \cdot \mathrm{poly}(m, d)\right)$. ◀

3 FNP-Hardness of $\mathsf{KS}_2\left(1/(4\sqrt{2})\right)$

This section studies the computational complexity of the $\mathsf{KS}_2(c)$ problem, and is organised as follows. In Section 3.1 we introduce the FNP complexity class. We formally define the NAE-3SAT-KS problem in Section 3.2, and prove that this problem is NP-hard. In Section 3.3, we build a reduction from the NAE-3SAT-KS problem to the $\mathsf{KS}_2\left(1/\left(4\sqrt{2}\right)\right)$ problem.

3.1 The FNP Complexity Class

In contrast with the complexity classes P and NP, the class FNP is used to study problems with output which is more complex than simply "yes" or "no". Formally, given a binary relation R and an input X, the corresponding *function problem* is to find Y such that $R(X,Y)$ holds or report "no" if no such Y exists. For example, we can take X to be an instance $\mathcal{I} = \{v_i\}_{i=1}^m$ of the $\mathsf{KS}_2(c)$ problem, and $Y \subseteq \mathcal{I}$ to be a candidate solution. Then, the relation $R_{\mathsf{KS}_2(c)}(\mathcal{I}, Y)$ holds if and only if Y satisfies (1). Any given binary relation R is in the class FNP iff there is a deterministic polynomial-time algorithm which can determine whether $R(X,Y)$ holds for a given pair (X,Y) [16]. Notice that every function problem has a natural corresponding decision problem. Specifically, given a binary relation R and a value of X, the decision problem asks whether there exists some Y such that $R(X,Y)$ holds. A function problem F is FNP-hard if there is a polynomial-time reduction from all problems in FNP to F. It is known that if the decision problem corresponding to F is NP-hard, then F is FNP-hard [16], and we will use this fact in our proof of Theorem 5.

3.2 NP-Completeness of NAE-3SAT-KS

In this subsection, we study the following NAE-3SAT-KS problem, and prove that the problem is NP-complete. We remark that we restrict ourselves to study SAT instances of a specific form here, as these SAT instances will be employed to prove the NP-hardness of the $\mathsf{KS}_2\left(1/(4\sqrt{2})\right)$ problem.

▶ **Problem 11** (NAE-3SAT-KS). *Given a 3SAT instance ψ consisting of a collection C of clauses over the set U of variables such that*

1. *every $c \in C$ has 3 literals,*
2. *for every $u \in U$, both of u and $\bar{u}$ appear in at most 2 clauses of C,*
3. *for every $u \in U$, at least one of u or $\bar{u}$ appears in exactly 2 clauses of C, and*
4. *any two clauses share at most one literal and no variable appears twice in the same clause,*

the NAE-3SAT-KS problem asks if there is a satisfying assignment for ψ such that every clause of ψ has at least one true literal and at least one false literal.

Our reduction is from the following well-known NP-complete problem.

▶ **Problem 12** (NAE-3SAT, [12]). *Given a 3SAT instance ψ that consists of a collection C of clauses over the set U of variables such that every clause $c \in C$ has 3 literals, the NAE-3SAT problem asks if there is a satisfying assignment for ψ such that every clause of ψ has at least one true literal and at least one false literal.*

▶ **Theorem 13.** *The NAE-3SAT-KS problem is NP-complete.*

Proof. Given any NAE-3SAT-KS instance ψ and an assignment to ψ's variables, it's straightforward to check in polynomial time if this is a satisfying assignment, and every clause of ψ has at least one true literal and at least one false literal. Hence, the NAE-3SAT-KS problem is in NP.

To prove that the NAE-3SAT-KS problem is NP-complete, we build a reduction from the NAE-3SAT problem to the NAE-3SAT-KS problem. Specifically, for any NAE-3SAT instance (U, C), where U is the set of variables and C is a collection of clauses, we construct an NAE-3SAT-KS instance (U', C') such that (U, C) is satisfiable in NAE-3SAT if and only if (U', C') is satisfiable in NAE-3SAT-KS. Our construction of (U', C') is as follows. Initially, we set $U' = U$ and $C' = C$. Then, for any variable x which appears only once in C, we remove x from U' and the corresponding clause from C' since the clause can always be satisfied by setting x appropriately and so removing the clause does not change the satisfiability of (U', C'). Then, for every remaining variable x, we replace the instances of x and $\bar{x}$ with new variables and add additional clauses to ensure that the satisfiability is unchanged. Specifically, for each x left in U' let $n_1 = |\{c \in C : x \in c\}|$, $n_2 = |\{c \in C : \bar{x} \in c\}|$, and set $n = n_1 + n_2$. Then, we introduce new variables $x_1, \ldots, x_n$ and replace the instances of x in C' with $x_1, \ldots, x_{n_1}$. Similarly, we replace the instances of $\bar{x}$ with $\bar{x}_{n_1+1}, \ldots, \bar{x}_n$.

Now, in order to ensure that (U', C') is satisfiable if and only if (U, C) is satisfiable, we introduce new clauses to C' which have the effect of constraining the variables $x_1, \ldots, x_n$ to have the same truth value in any satisfying assignment. To achieve this, let $n' \geq n$ be an odd number, and we introduce additional new variables $y_1, \ldots, y_{n'}$ and clauses

$$(\bar{y}_i \vee \bar{y}_{i+1} \vee y_{i+2}) \quad \text{for any } i \in [1, n'], \tag{4}$$

where the indices are taken modulo n'. We will see that these clauses ensure that the y_i variables must all have the same value in a satisfying assignment. We see this by a simple case distinction.

- Case 1: $y_1 = y_2$ in a satisfying assignment. Then, by the first clause in (4) it must be that $y_2 = y_3$ since there must be at least one true literal and one false literal in each satisfied clause. Proceeding inductively through the clauses in (4), we establish that $y_1 = y_2 = \ldots = y_{n'}$.
- Case 2: $y_1 \neq y_2$ in a satisfying assignment. We will show that this leads to a contradiction. By the last clause in (4), $y_{n'} \neq y_1$ since there must be at least one true literal and one false literal. Again, we proceed inductively from the $(n'-1)$th clause in (4) down to establish that $y_1 \neq y_2, y_2 \neq y_3, \ldots, y_{n'-1} \neq y_{n'}$. As such, we have $y_1 = y_3 = \ldots = y_{2i+1}$ which is a contradition since n' is odd and we have already established that $y_1 \neq y_{n'}$.

As such, we can use the variables $y_1, \ldots, y_{n'}$ with the assumption that they have the same value in any satisfying assignment of (U', C'). It remains to construct clauses to guarantee that the variables $x_1, \ldots, x_n$ have the same value in any satisfying assignment. We add the clauses

$$(x_i \vee \bar{x}_{i+1} \vee y_i) \quad \text{for any } i \in [1, n], \tag{5}$$

where the indices are taken modulo n. We will show that $x_1 = x_2 = \ldots = x_n$ in a satisfying assignment by case distinction.

- Case 1: $x_1 = y_i$ for all i. By the first clause in (5), it must be that $x_1 = x_2$ since we cannot have $x_1 = \bar{x}_2 = y_i$ in a satisfying assignment. Then, proceeding inductively using each clause in turn we establish that $x_1 = x_2 = \ldots = x_n$.
- Case 2: $\bar{x}_1 = y_i$ for all i. By the last clause in (5), it must be that $\bar{x}_n = \bar{x}_1$ since we cannot have $x_n = \bar{x}_1 = y_n$ in a satisfying assignment. Then, proceeding inductively from the $(n-1)$th clause down, we establish that $\bar{x}_1 = \bar{x}_2 = \ldots = \bar{x}_n$.

Notice that by this construction, each literal x_i, $\bar{x}_i$, y_i, and $\bar{y}_i$ now appears at most twice in C', no two clauses share more than one literal and no literal appears twice in the same clause. Additionally, every x_i and $\bar{x}_i$ appears exactly once in the clauses added by (5).

Since the variable x_i also appears exactly once in the clauses corresponding directly to C, requirement (3) of the NAE-3SAT-KS problem is satisfied. Moreover, we have that (U', C') has a satisfying assignment if (U, C) has a satisfying assignment; this follows by setting the values of $x_1, \ldots, x_n$ in U' to the value of their corresponding $x \in U$. On the other hand, any satisfying assignment of (U', C') corresponds to a satisfying assignment of (U, C), since we must have that $x_1 = \ldots = x_n$ and can set the value of $x \in U$ to be the same value to get a satisfying assignment of (U, C). Finally, notice that our new instance (U', C') of NAE-3SAT-KS can be constructed in polynomial time in the size of the instance (U, C) of NAE-3SAT. This completes the proof. ◀

3.3 FNP-Hardness of $\mathsf{KS}_2\left(1/\left(4\sqrt{2}\right)\right)$

We now show that the $\mathsf{KS}_2(c)$ problem is FNP-hard for any $c \leq 1/(4\sqrt{2})$, i.e., Theorem 5. At a high level, our proof is by reduction from the NAE-3SAT-KS problem. Given an instance of the NAE-3SAT-KS problem, we will construct an instance $\mathcal{I}$ of $\mathsf{KS}_2(c)$ such that

1. if the NAE-3SAT-KS instance is satisfiable, then there is a set $S \subset \mathcal{I}$ with $\sum_{v \in S} vv^\intercal = (1/2) \cdot I$, and
2. if the NAE-3SAT-KS instance is not satisfiable, then for all sets $S \subset \mathcal{I}$ we have

$$\left\| \sum_{v \in S} vv^\intercal - \frac{1}{2} I \right\| \geq \frac{1}{4\sqrt{2}} \cdot \sqrt{\alpha}.$$

This will establish that the $\mathsf{KS}_2\left(1/\left(4\sqrt{2}\right)\right)$ problem is FNP-complete, and that it is NP-hard to distinguish between instances of $\mathsf{KS}_2(c)$ with $\mathcal{W}(\mathcal{I}) = 0$ and those for which $\mathcal{W}(\mathcal{I}) \geq \left(1/4\sqrt{2}\right)\sqrt{\alpha}$.

Proof of Theorem 5. We prove that $\mathsf{KS}_2(1/(4\sqrt{2}))$ is NP-hard by a reduction from the NAE-3SAT-KS problem to the decision version of the $\mathsf{KS}_2\left(1/\left(4\sqrt{2}\right)\right)$ problem. We are given an instance (U, C) of the NAE-3SAT-KS problem, and construct an instance of $\mathsf{KS}_2(c)$. Let us refer to

- the clauses in C as $c_1, \ldots c_m$;
- the variables in U as $x_1, \ldots, x_n$; we sometimes write x_i and $\bar{x}_i$ for the un-negated and negated literals.

Our constructed $\mathsf{KS}_2(c)$ instance has $O(n+m)$ dimensions. Specifically, there is one dimension for each clause in C and one dimension for each variable in U which appears both negated and un-negated in C. We use d_j^c to refer to the dimension corresponding to clause c_j, and d_j^x to refer to the dimension corresponding to variable x_j. We add $O(m+n)$ vectors to our $\mathsf{KS}_2(c)$ instance. Conceptually, we add one vector for each clause and 4 vectors for each literal. We use v_j^c to refer to the vector corresponding to clause c_j, and $v_{j,1}^x$ to $v_{j,4}^x$ or $v_{j,1}^{\bar{x}}$ to $v_{j,4}^{\bar{x}}$ to refer to the vectors corresponding to the literal x_j or $\bar{x}_j$. For each clause c_j, we set $v_j^c(d_j^c) = 1/2$, and set the other entries of v_j^c to be 0. Table 1 completes the definition of the vectors corresponding to literals. For each literal, we define only the value on the dimensions corresponding to the variable and the clauses containing the literal; all other entries in the vector are 0. Let A be the set of vectors defined above. Notice that the squared norms of the vectors in A are bounded above by $1/4$ and so $\alpha = 1/4$ in the constructed $\mathsf{KS}_2(c)$ instance.

To complete the reduction, we'll show the following:

1. It holds that $\sum_{v \in A} vv^\intercal = I$.
2. If the original NAE-3SAT-KS instance has a satisfying assignment, then there's a set $S \subset A$ such that $\sum_{v \in S} vv^\intercal = \frac{1}{2} \cdot I$.

Table 1 The construction of the vectors in the $\mathsf{KS}_2(c)$ instance for a literal x_i appearing in clause c_j and possibly also in c_k. If a literal appears in only one clause, c_j, we ignore the middle column corresponding to c_k; i.e., the vectors corresponding to x_i are non-zero only on dimensions d_j^c and d_i^x.

Vector	Value on d_j^c	Value on d_k^c	Value on d_i^x
$v_{i,1}^x$	$1/4$	$1/4$	$1/\sqrt{8}$
$v_{i,2}^x$	$1/4$	$1/4$	$-1/\sqrt{8}$
$v_{i,3}^x$	$1/4$	$-1/4$	$1/\sqrt{8}$
$v_{i,4}^x$	$1/4$	$-1/4$	$-1/\sqrt{8}$

3. Any set $S \subset A$ with

$$\left\| \sum_{v \in S} vv^\intercal - \frac{1}{2} I \right\| < \frac{1}{8\sqrt{2}} = \frac{1}{4\sqrt{2}} \sqrt{\alpha}$$

corresponds to a satisfying assignment of the original NAE-3SAT-KS instance.

Vectors in A are isotropic. Let $B = \sum_{v \in A} vv^\intercal$. Then, for any variable x_i, we have that

$$B(d_i^x, d_i^x) = \sum_{v \in A} v(d_i^x)^2 = \sum_{j=1}^{4} v_{i,j}^x (d_i^x)^2 + \sum_{j=1}^{4} v_{i,j}^{\bar{x}} (d_i^x)^2 = 1.$$

Additionally, for any clause c_i we have that

$$B(d_i^c, d_i^c) = \sum_{v \in A} v(d_i^c)^2 = v_i^c (d_i^c)^2 + \sum_{x_j \in c} \sum_{k=1}^{4} v_{j,k}^x (d_i^c)^2 = \frac{1}{4} + 3 \cdot \frac{1}{4} = 1.$$

This demonstrates that the diagonal entries of B are all 1. We now see that the off-diagonal entries are all 0. First, notice that for any two dimensions relating to variables, d_i^x and d_j^x, we have

$$B(d_i^x, d_j^x) = \sum_{v \in A} v(d_i^x) v(d_j^x) = 0,$$

since there is no vector in A with a non-zero contribution to more than one dimension corresponding to a variable. Now, let us consider two dimensions corresponding to different clauses c_i and c_j. We have

$$B(d_i^c, d_j^c) = \sum_{v \in A} v(d_i^c) v(d_j^c) = \sum_{x_k \in c_i \cap c_j} \sum_{\ell=1}^{4} v_{k,\ell}^x (d_i^c) \cdot v_{k,\ell}^x (d_j^c) = 0,$$

where we use the fact that c_i and c_j share at most one literal. Finally, consider the case when one dimension corresponds to the clause c_i and the other dimension corresponds to the variable x_j. If the variable x_j does not appear in c_i, then there are no vectors with a non-zero contribution to the two dimensions and so the entry is 0. Otherwise, we have

$$B(d_i^c, d_j^x) = \sum_{v \in A} v(d_i^c) v(d_j^x) = \sum_{k=1}^{4} v_{i,k}^x (d_i^c) v_{i,k}^x (d_j^x) = \frac{1}{4\sqrt{8}} + \frac{1}{4\sqrt{8}} - \frac{1}{4\sqrt{8}} - \frac{1}{4\sqrt{8}} = 0,$$

where we use the fact that no variable appears twice in the same clause. This completes the proof that $\sum_{v \in A} vv^\intercal = I$.

If the NAE-3SAT-KS instance is satisfiable, then there is a solution to $\mathsf{KS}_2(1/(4\sqrt{2}))$. Given a satisfying assignment to NAE-3SAT-KS, let $T \subset U$ be the set of variables which are set to be TRUE and let $F \subset U$ be the set of variables which are set to be FALSE. Recall that in a satisfying assignment, each clause in C contains either 1 or 2 true literals. Let $C' \subset C$ be the set of clauses with exactly 1 true literal in the satisfying assignment. Then, we define S to be

$$S \triangleq \{v^x_{i,1}, v^x_{i,2}, v^x_{i,3}, v^x_{i,4} : x_i \in T\} \cup \{v^{\bar{x}}_{i,1}, v^{\bar{x}}_{i,2}, v^{\bar{x}}_{i,3}, v^{\bar{x}}_{i,4} : x_i \in F\} \cup \{v^c_i : c_i \in C'\},$$

and we show that $\sum_{v\in S} vv^\intercal = \frac{1}{2}I$. We can repeat the previous calculations, this time setting $B = \sum_{v\in S} vv^\intercal$ to show that $B = (1/2)I$. Specifically, for any variable x_i, it holds that $B(d^x_i, d^x_i) = \frac{1}{2}$ since only the vectors corresponding to the negated *or* un-negated variable are included. For any clause $c_i \in C'$, we have

$$B(d^c_i, d^c_i) = v^c_i(d^c_i)^2 + \sum_{k=1}^{4} v^x_{j,k}(d^c_i)^2 = 1/4 + 1/4 = 1/2,$$

where x_j is the literal which is set to be true in the clause c_i. Similarly, for any clause in $c_i \in C \setminus C'$, we have

$$B(d^c_i, d^c_i) = \sum_{k=1}^{4} v^x_{j,k}(d^c_i)^2 + \sum_{k=1}^{4} v^x_{\ell,k}(d^c_i)^2 = 1/4 + 1/4 = 1/2,$$

where the literals x_j and x_ℓ are set to be true in the clause c_i. Then, notice that the calculations for the off-diagonal entries follow in the same way as before. This completes the proof that a satisfying assignment for NAE-3SAT-KS implies a solution to the $\mathsf{KS}_2(1/(4\sqrt{2}))$ problem.

If there is a solution to $\mathsf{KS}_2(c)$, then the NAE-3SAT-KS instance is satisfiable. We prove this by a contrapositive argument. That is, we show that for any set S' which does not correspond to a satisfying assignment of the NAE-3SAT-KS problem, there must be some vector y with $\|y\| = 1$ such that

$$\left| y^\intercal \left(\sum_{v\in S'} vv^\intercal \right) y - \frac{1}{2} \right| \geq \epsilon \tag{6}$$

for $\epsilon = \frac{1}{8\sqrt{2}}$. Specifically, we will analyse three cases, and show that

1. if there is some variable x_i such that S' does not contain exactly 4 of the vectors

   $$\{v^x_{i,1}, v^x_{i,2}, v^x_{i,3}, v^x_{i,4}, v^{\bar{x}}_{i,1}, v^{\bar{x}}_{i,2}, v^{\bar{x}}_{i,3}, v^{\bar{x}}_{i,4}\},$$

   then there is a vector y satisfying (6) for $\epsilon = 1/8$;
2. if Case (1) does not apply, then if there is some literal x_i such that S' contains 1, 2, or 3 of the vectors $\{v^x_{i,1}, v^x_{i,2}, v^x_{i,3}, v^x_{i,4}\}$, then there is a vector y satisfying (6) for $\epsilon = 1/(8\sqrt{2})$;
3. if neither Case (1) nor (2) applies, then if S' does not correspond to a satisfying assignment of the original NAE-3SAT-KS instance, there must be a vector y satisfying (6) for $\epsilon = 1/4$.

For the first case, suppose that there is some variable x_i such that S' does not contain exactly 4 vectors corresponding to the variable x_i. Let $k \neq 4$ be the number of such vectors, and let y be the vector with all zeros except for $y(d^x_i) = 1$. Notice that

$$\left| y^\intercal \left(\sum_{v\in S'} vv^\intercal \right) y - \frac{1}{2} \right| = \left| \sum_{v\in S'} v(d^x_i)^2 - \frac{1}{2} \right| = \left| \frac{k}{8} - \frac{1}{2} \right| \geq \frac{1}{8}.$$

Table 2 The absolute values of off-diagonal entries in $B = \sum_{v \in S'} vv^\intercal$, based on which vectors corresponding to the literal x_i are included in S'. We assume x_i appears in the clauses c_j and c_k.

Vectors in S'	$\lvert B(d_i^x, d_j^c)\rvert$	$\lvert B(d_i^x, d_k^c)\rvert$	$\lvert B(d_j^c, d_k^c)\rvert$
One vector $v_{i,\ell}^x$	$1/\left(8\sqrt{2}\right)$	$1/\left(8\sqrt{2}\right)$	$1/16$
Vectors $v_{i,1}^x$ and $v_{i,2}^x$	0	0	$1/8$
Vectors $v_{i,1}^x$ and $v_{i,3}^x$	$1/\left(4\sqrt{2}\right)$	0	0
Vectors $v_{i,1}^x$ and $v_{i,4}^x$	0	$1/\left(4\sqrt{2}\right)$	0
Vectors $v_{i,2}^x$ and $v_{i,3}^x$	0	$1/\left(4\sqrt{2}\right)$	0
Vectors $v_{i,2}^x$ and $v_{i,4}^x$	$1/\left(4\sqrt{2}\right)$	0	0
Vectors $v_{i,3}^x$ and $v_{i,4}^x$	0	0	$1/8$
Three vectors $v_{i,\ell}^x$	$1/\left(8\sqrt{2}\right)$	$1/\left(8\sqrt{2}\right)$	$1/16$

For the second case, suppose that the set S' contains 4 vectors for each variable, but there is some literal x_i such that S' contains some but not all of the vectors corresponding to x_i. By Condition 3 of the NAE-3SAT-KS problem (Problem 11) we can assume that x_i appears in two clauses c_j and c_k. Otherwise, this is the case for $\bar{x}_i$ and S' contains some, but not all, of the vectors corresponding to $\bar{x}_i$ since it contains exactly 4 vectors corresponding to the variable x_i. Now, we define $B = \sum_{v \in S'} vv^\intercal$ and we consider the absolute values of certain off-diagonal entries in B, which are summarised in Table 2. Notice that, regardless of which vectors corresponding to x_i are included, there are two indices $\hat{d}_1$ and $\hat{d}_2$ such that $\left|B(\hat{d}_1, \hat{d}_2)\right| \geq \frac{1}{8\sqrt{2}}$. Using the indices $\hat{d}_1$ and $\hat{d}_2$, define the unit vector

$$y = \begin{cases} \frac{1}{\sqrt{2}}(\mathbf{1}_{\hat{d}_1} + \mathbf{1}_{\hat{d}_2}) & \text{if } \operatorname{sgn}(B(\hat{d}_1, \hat{d}_1) + B(\hat{d}_2, \hat{d}_2) - 1) = \operatorname{sgn}(B(\hat{d}_1, \hat{d}_2)) \\ \frac{1}{\sqrt{2}}(\mathbf{1}_{\hat{d}_1} - \mathbf{1}_{\hat{d}_2}) & \text{otherwise} \end{cases}$$

where $\operatorname{sgn}(\cdot)$ is the sign function. Then we have

$$\begin{aligned} \left|y^\intercal B y - \frac{1}{2}\right| &= \left|\frac{1}{2}\Big(B(\hat{d}_1, \hat{d}_1) + B(\hat{d}_2, \hat{d}_2) \pm B(\hat{d}_1, \hat{d}_2) \pm B(\hat{d}_2, \hat{d}_1)\Big) - \frac{1}{2}\right| \\ &= \frac{1}{2}\left|B(\hat{d}_1, \hat{d}_1) + B(\hat{d}_2, \hat{d}_2) - 1 \pm 2B(\hat{d}_1, \hat{d}_2)\right| \\ &= \frac{1}{2}\left(\left|B(\hat{d}_1, \hat{d}_1) + B(\hat{d}_2, \hat{d}_2) - 1\right| + 2\left|B(\hat{d}_1, \hat{d}_2)\right|\right) \\ &\geq \left|B\left(\hat{d}_1, \hat{d}_2\right)\right| \\ &\geq \frac{1}{8\sqrt{2}}, \end{aligned}$$

where the third equality follows by the construction of y.

Finally, we consider the third case, in which there are 4 vectors in S' for each variable, and all 4 vectors correspond to the same literal. It is clear that such a set S' corresponds unambiguously to an assignment for the original variables in the NAE-3SAT-KS instance: specifically, one can set a variable x_i to be TRUE if S' contains $\{v_{i,1}^x, v_{i,2}^x, v_{i,3}^x, v_{i,4}^x\}$, and set x_i to be FALSE if S' contains $\{v_{i,1}^{\bar{x}}, v_{i,2}^{\bar{x}}, v_{i,3}^{\bar{x}}, v_{i,4}^{\bar{x}}\}$. Then, suppose that there is some clause $c_j \in C$ which is not satisfied by this assignment. This implies that either all 12 of the vectors corresponding to literals in c_j are included in S', or none of the vectors corresponding to literals in c_j are included in S'. In either case, we can set y to be the indicator vector of the dimension d_j^c, and have that

$$|y^\intercal By - 1/2| = \left| \sum_{v \in S'} v(d_j^c)^2 - 1/2 \right| \geq 1/4$$

since we can either include v_j^c or not in order to set $\sum_{v \in S'} v(d_j^c)^2$ equal to either $1/4$ or $3/4$.

This completes the reduction from the NAE-3SAT-KS problem to the decision version of the $\mathsf{KS}_2(c)$ problem for $c \leq 1/(4\sqrt{2})$, which implies that $\mathsf{KS}_2\left(1/\left(4\sqrt{2}\right)\right)$ is FNP-hard. Furthermore, notice that by the reduction in this proof,

- if the NAE-3SAT-KS instance is satisfiable, then the constructed instance $\mathcal{I}$ of the $\mathsf{KS}_2(c)$ problem satisfies $\mathcal{W}(\mathcal{I}) = 0$, and
- if the NAE-3SAT-KS instance is not satisfiable, then the constructed instance $\mathcal{I}$ of the $\mathsf{KS}_2(c)$ problem satisfies $\mathcal{W}(\mathcal{I}) \geq 1/\left(4\sqrt{2}\right) \cdot \sqrt{\alpha}$.

This shows that distinguishing between instances with $\mathcal{W}(\mathcal{I}) = 0$ and $\mathcal{W}(\mathcal{I}) \geq 1/\left(4\sqrt{2}\right) \cdot \sqrt{\alpha}$ is NP-hard, and completes the proof. ◀

4 Conclusion

This paper studies the algorithms and complexity of the Kadison-Singer problem through the $\mathsf{KS}_2(c)$ problem, and presents two results. On one side, we prove that the $\mathsf{KS}_2(c)$ problem for any $c \in \mathbb{R}^+$ can be solved in quasi-polynomial time when $d = O(\log m)$, which suggests that the problem is much easier to solve in low dimensions. The key to our algorithm design is a novel application of online spectral sparsification subroutines, with which we are able to efficiently construct representations of all spectral equivalence classes over time and reduce the enumeration space of the candidate solutions. We expect that our work could motivate more research on the applications of spectral sparsification and related problems in numerical linear algebra to the algorithmic Kadison-Singer problem.

On the other side, our NP-hardness result shows that the Kadison-Singer type problem for arbitrary dimensions can be as hard as solving the SAT problem, and the $\mathsf{KS}_2(c)$ problem belongs to different complexity classes for different values of c. Hence, more refined studies on the classification of its computational complexity would help us better understand the complexity of the algorithmic Kadison-Singer problem. In our point of view, both directions left from the paper are very interesting, and we leave these for future work.

References

1. Nima Anari and Shayan Oveis Gharan. The Kadison-Singer problem for strongly Rayleigh measures and applications to asymmetric TSP. *CoRR*, abs/1412.1143, 2014. `arXiv:1412.1143`.
2. Nima Anari, Shayan Oveis Gharan, Amin Saberi, and Nikhil Srivastava. Approximating the largest root and applications to interlacing families. In *29th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'18)*, pages 1015–1028, 2018.
3. Nikhil Bansal. Constructive algorithms for discrepancy minimization. In *51th Annual IEEE Symposium on Foundations of Computer Science (FOCS'10)*, pages 3–10, 2010.
4. Nikhil Bansal, Tim Oosterwijk, Tjark Vredeveld, and Ruben van der Zwaan. Approximating vector scheduling: Almost matching upper and lower bounds. *Algorithmica*, 76(4):1077–1096, 2016.
5. Joshua D. Batson, Daniel A. Spielman, and Nikhil Srivastava. Twice-Ramanujan sparsifiers. *SIAM Journal on Computing*, 41(6):1704–1721, 2012.
6. Luca Becchetti, Andrea E. F. Clementi, Emanuele Natale, Francesco Pasquale, and Luca Trevisan. Finding a bounded-degree expander inside a dense one. In *31st Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'20)*, pages 1320–1336, 2020.

7 Peter G. Casazza. Consequences of the Marcus/Spielman/Stivastava solution to the Kadison-Singer problem. *CoRR*, abs/1407.4768, 2014. arXiv:1407.4768.

8 Peter G Casazza, Matthew Fickus, Janet C Tremain, and Eric Weber. The Kadison-Singer problem in mathematics and engineering: a detailed account. *Contemporary Mathematics*, 414:299, 2006.

9 Moses Charikar, Alantha Newman, and Aleksandar Nikolov. Tight hardness results for minimizing discrepancy. In *22nd Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'11)*, pages 1607–1614, 2011.

10 Chandra Chekuri and Sanjeev Khanna. On multidimensional packing problems. *SIAM Journal on Computing*, 33(4):837–851, 2004.

11 Michael B. Cohen, Cameron Musco, and Jakub Pachocki. Online row sampling. *Theory of Computing*, 16(15):1–25, 2020.

12 Michael R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, 1979.

13 Richard Kadison and Isadore Singer. Extensions of pure states. *American Journal of Mathematics*, 81:383–400, 1959.

14 Adam Marcus, Daniel A. Spielman, and Nikhil Srivastava. Interlacing families I: bipartite Ramanujan graphs of all degrees. In *54th Annual IEEE Symposium on Foundations of Computer Science (FOCS'13)*, pages 529–537, 2013.

15 Adam W. Marcus, Daniel A. Spielman, and Nikhil Srivastava. Interlacing families II: mixed characteristic polynomials and the Kadison-Singer problem. *Annals of Mathematics*, 182(1):327–350, 2015.

16 Elaine Rich. *Automata, computability and complexity: theory and applications*. Pearson Prentice Hall Upper Saddle River, 2008.

17 Daniel A. Spielman and Nikhil Srivastava. Graph sparsification by effective resistances. *SIAM Journal on Computing*, 40(6):1913–1926, 2011.

18 Daniel A. Spielman and Peng Zhang. Hardness results for Weaver's discrepancy problem. In *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques (APPROX/RANDOM '22)*, pages 40:1–40:14, 2022.

19 Joel A Tropp. User-friendly tail bounds for sums of random matrices. *Foundations of computational mathematics*, 12(4):389–434, 2012.

20 Nicholas Weaver. The Kadison-Singer problem in discrepancy theory. *Discrete Mathematics*, 278(1-3):227–239, 2004.

21 Nik Weaver. The Kadison-Singer problem in discrepancy theory, II. *CoRR*, abs/1303.2405, 2013. arXiv:1303.2405.

A Omitted Proofs from Section 2

In this section we present the proofs omitted from Section 2.

Proof of Lemma 9. We will show that for any pair $(S^\star, B^\star)$ constructed by Algorithm 1, $B^\star$ is equivalent to the output of the algorithm described in Lemma 8 when applied to $S^\star$. We prove this by induction on i. The base case $i = 0$ follows immediately from the initialisation of $\mathcal{L}_0 = \{(\emptyset, \mathbf{0}_{d\times d})\}$. For the inductive step we show that the conclusion holds for every pair in $\mathcal{L}_i$, assuming it holds for every pair in $\mathcal{L}_{i-1}$. For each pair $(S^\star, B^\star) \in \mathcal{L}_i$, the proof proceeds by a case distinction.

Case 1: $(S^\star, B^\star) \in \mathcal{L}_{i-1}$. This case corresponds to the pairs (S, B) added on Line 16 of Algorithm 1. Accordingly, by the inductive hypothesis, we have that $B^\star$ is equivalent to the output of the algorithm described in Lemma 8 applied to $S^\star$.

Case 2: $(S^\star, B^\star) \notin \mathcal{L}_{i-1}$**.** This case covers the pairs involving S' added on Lines 16 and 18 of Algorithm 1. Let (S, B_{i-1}) be the pair in $\mathcal{L}_{i-1}$ from which $(S^\star, B^\star)$ is constructed. Notice that $S^\star = S \cup \{v_i\}$. Then, by the construction of Algorithm 1, with probability p_i, we have

$$B^\star = B_{i-1} + \frac{1}{p_i} v_i v_i^\intercal$$

and with probability $1 - p_i$, we have $B^\star = B_{i-1}$, where p_i is the probability defined in Lemma 8. As such, $B^\star$ is the result of applying an iteration of the algorithm defined in Lemma 8, for the new vector v_i. This maintains that $B^\star$ is equivalent to the output of the Lemma 8 algorithm applied to $S^\star$ and completes the inductive argument. ◀

Proof of Lemma 10. We prove that one of the matrices B constructed by Algorithm 1 is equivalent to the output of the algorithm defined in Lemma 8 applied to the set S. Although the matrices constructed in Algorithm 1 are always part of a pair (S', B), in this proof we consider only the matrices B, and ignore the sets S' which are constructed alongside them.

We now inductively define a sequence $B_0, B_1, \ldots, B_m$, such that B_i is a matrix constructed by the algorithm in iteration i and $B_i \in \mathcal{L}_i$ corresponds to the output of the Lemma 8 algorithm applied to $S \cap \{v_1, \ldots, v_i\}$. Firstly, let $B_0 = \mathbf{0}_{d \times d}$, which is the initial condition for the algorithm in Lemma 8 and is constructed by Algorithm 1 on Line 5. Then, for the inductive step, we assume that B_{i-1} is the output of the Lemma 8 algorithm applied to $S \cap \{v_1, \ldots, v_{i-1}\}$ and we define B_i by case distinction.

Case 1: $v_i \notin S$**.** In this case, we set $B_i = B_{i-1}$, and notice that if B_{i-1} is in the set $\mathcal{L}_{i-1}$ constructed by Algorithm 1, then B_i must be in the set $\mathcal{L}_i$ since every matrix B in $\mathcal{L}_{i-1}$ is included in $\mathcal{L}_i$ on either Line 16 or Line 18. Since $S \cap \{v_1, \ldots, v_{i-1}\} = S \cap \{v_1, \ldots, v_i\}$, we have that B_i is the output of the algorithm defined in Lemma 8 applied to $S \cap \{v_1, \ldots, v_i\}$ by the inductive hypothesis.

Case 2: $v_i \in S$**.** In this case, we set B_i to be either B_{i-1} or $B_{i-1} + (1/p) v_i v_i^\intercal$, according to the result of the condition on Line 14 of Algorithm 1. Notice that, since the definition of p in Algorithm 1 is the same as the definition in Lemma 8, B_i corresponds to the result of applying an iteration of the algorithm in Lemma 8 with B_{i-i} and v_i. Therefore, by the induction hypothesis, B_i is equivalent to the output of the Lemma 8 algorithm applied to $S \cap \{v_1, \ldots, v_i\}$, which completes the inductive construction of $B_1, \ldots, B_m$.

Finally, since our defined B_m corresponds to the output of the algorithm in Lemma 8 applied to S, we can apply Lemma 8 to S and B_m which completes the proof. ◀