A Theory of Spectral CSP Sparsification

Sanjeev Khanna
School of Engineering and Applied Sciences, University of Pennsylvania, Philadelphia, PA, USA

Aaron Putterman
School of Engineering and Applied Sciences, Harvard University, Cambridge, MA, USA

Madhu Sudan
School of Engineering and Applied Sciences, Harvard University, Cambridge, MA, USA

Abstract

We initiate the study of spectral sparsification for instances of Constraint Satisfaction Problems (CSPs). In particular, we introduce a notion of the *spectral energy* of a fractional assignment for a Boolean CSP instance, and define a *spectral sparsifier* as a weighted subset of constraints that approximately preserves this energy for all fractional assignments. Our definition not only strengthens the combinatorial notion of a CSP sparsifier but also extends well-studied concepts such as spectral sparsifiers for graphs and hypergraphs.

Recent work by Khanna, Putterman, and Sudan [SODA 2024] demonstrated near-linear sized *combinatorial sparsifiers* for a broad class of CSPs, which they term *field-affine CSPs*. Our main result is a polynomial-time algorithm that constructs a spectral CSP sparsifier of near-quadratic size for all field-affine CSPs. This class of CSPs includes graph (and hypergraph) cuts, XORs, and more generally, any predicate which can be written as $P(x_1, \dots x_r) = \mathbf{1}[\sum a_i x_i \neq b \mod p]$.

Based on our notion of the spectral energy of a fractional assignment, we also define an analog of the second eigenvalue of a CSP instance. We then show an extension of Cheeger's inequality for all even-arity XOR CSPs, showing that this second eigenvalue loosely captures the "expansion" of the underlying CSP. This extension specializes to the case of Cheeger's inequality when all constraints are even XORs and thus gives a new generalization of this powerful inequality which converts the combinatorial notion of expansion to an analytic property.

Perhaps the most important effect of spectral sparsification is that it has led to certifiable sparsifiers for graphs and hypergraphs. This aspect remains open in our case even for XOR CSPs since the eigenvalues we describe in our Cheeger inequality are not known to be efficiently computable. Computing this efficiently, and/or finding other ways to certifiably sparsify CSPs are open questions emerging from our work. Another important open question is determining which classes of CSPs have near-linear size *spectral* sparsifiers.[1]

2012 ACM Subject Classification Theory of computation → Sketching and sampling

Keywords and phrases Sparsification, sketching, hypergraphs

Digital Object Identifier 10.4230/LIPIcs.ICALP.2025.107

Category Track A: Algorithms, Complexity and Games

Related Version *Full Version*: `https://arxiv.org/abs/2504.16206`

Funding *Sanjeev Khanna*: Supported in part by NSF award CCF-2402284 and AFOSR award FA9550-25-1-0107.
Aaron Putterman: Supported in part by the Simons Investigator Awards of Madhu Sudan and Salil Vadhan, NSF Award CCF 2152413 and AFOSR award FA9550-25-1-0112.
Madhu Sudan: Supported in part by a Simons Investigator Award, NSF Award CCF 2152413 and AFOSR award FA9550-25-1-0112.

[1] Find a full version of the paper at `https://arxiv.org/abs/2504.16206`.

1 Introduction

Spectral tools have played a powerful role in the analysis of graphs and hypergraphs and led to many celebrated successes including the use of Cheeger's inequality to approximate computationally intractable notions like expansion [2, 1] and the use of spectral methods to sparsify graphs and hypergraphs [4, 21]. Recent years have seen the use of CSPs to add greater richness to the study of hypergraphs by considering generalized notion of when an edge (or hyperedge) is deemed to be cut by a (multi-)partition of the vertex set. In this work we initiate the study of spectral methods for CSP analysis, by introducing a concrete notion of the spectral energy of a (fractional) assignment to variables, and giving analogs to Cheeger's inequality and spectral sparsification. We describe our work in more detail below after reviewing some of the classical works on spectral analysis, notably on sparsification.

1.1 Background

Given a graph $G = (V, E)$, there are many important combinatorial properties of the graph that are hard to explicitly calculate. Perhaps most notably is the notion of *conductance*: given a graph $G = (V, E)$, we can define its conductance as

$$\phi_G = \min_{S \subset V} \frac{|\delta S|}{\min(\mathrm{Vol}(S), \mathrm{Vol}(V - S))}.$$

Here, $\mathrm{Vol}(S)$ refers to the sum of degrees of vertices in S, and δS refers to the edges crossing from S to $\bar{S}$. In a celebrated result [2, 1], it was shown that ϕ_G can actually be well-approximated by the second eigenvalue of the normalized Laplacian of the graph:

$$\frac{\lambda_2}{2} \leq \phi_G \leq \sqrt{2\lambda_2}.$$

The normalized Laplacian is simply $\mathcal{L} = W^{-1/2}(W - A)W^{-1/2}$, where W is the diagonal matrix of vertex degrees, and A is the (weighted) adjacency matrix of the graph. Subsequent to this discovery, spectral theory has emerged as a central topic in graph theory with wide-reaching implications and connections (see [22] for further discussion). In particular, one line of research emanating from spectral theory has been *spectral sparsification*, which is the key focus of this paper.

1.1.1 (Spectral) Graph Sparsification

Graph sparsification has emerged as a key topic in algorithmic graph theory. First proposed in the works of Karger [14] and Benczùr and Karger [5], graph sparsification takes as input a graph $G = (V, E)$ and a parameter $\varepsilon \in (0, 1)$, and returns a reweighted sub-graph G' such that the size of every cut in G is simultaneously preserved to a $(1 \pm \varepsilon)$ multiplicative factor in G'. Although the algorithm presented in [5] is nearly-optimal for cut sparsification, these works spurred a flurry of research in sparsification in many different directions: for instance, the design of *deterministic* algorithms for graph sparsification [4], generalizing notions of sparsification to hypergraphs [17, 8, 12] and to norms [9], to CSPs [17, 15], and to arbitrary sets of vectors [6]. Along the way, one of the core advances in the study of sparsification has been the notion of a *spectral* sparsifier [23]. Here, instead of only preserving the *cut-sizes* of a graph, the sparsifier instead preserves the *energy* of the graph, which is defined for a graph $G = (V, E)$, and a vector $x \in [0, 1]^V$ as $Q_G(x) = \sum_{e=(u,v)\in E} w_e \cdot (x_u - x_v)^2$. Specifically, a spectral sparsifier is a re-weighted subgraph G' such that *for every* $x \in [0, 1]^V$, it is the case that $Q_{G'}(x) \in (1 \pm \varepsilon)Q_G(x)$. This energy can be written as a quadratic form involving the graph's (un-normalized) Laplacian, which is denoted by L_G, and satisfies $(L_G)_{i,j} = \begin{cases} \deg(i) & \text{if } i = j \\ -w_{i,j} & \text{else} \end{cases}$.

In graphs, the benefits from the use of spectral sparsification are several-fold:

1. Spectral sparsification is a stronger condition than cut sparsification, as it preserves the eigenvalues of the graph's Laplacian in addition to the cut sizes.
2. The energy of a graph allows for an interpretation of the graph as an electrical network, which allows for well-motivated sampling schemes for designing sparsifiers.
3. The energy of the graph has a convenient form as the quadratic form of the Laplacian of the graph. This means that spectral sparsifiers are *efficiently certifiable* (in the sense that one can verify in polynomial time whether a graph G' is a $(1 \pm \varepsilon)$ spectral-sparsifier of a graph G), a key fact that is used when designing deterministic sparsification algorithms [4].
4. This property of efficient (and deterministic) certification has also contributed to *linear-size* spectral sparsifiers of graphs [4], shaving off the $\log(n)$ factor that is inherent to the sampling-based approaches that create cut-sparsifiers.

1.1.2 (Spectral) Hypergraph Sparsification

Given the wide-reaching benefits of spectral sparsification, as research on graph sparsification pivoted to *hypergraph* sparsification, spectral notions of sparsification in hypergraphs were quick to be proposed. In a hypergraph $H = (V, E)$, each hyperedge $e \in E$ is an arbitrary subset of vertices. A cut in the hypergraph H is given by a subset $S \subseteq V$ of the vertices, and we say that a hyperedge e is cut by S if $S \cap e \neq \emptyset$ and $\bar{S} \cap e \neq \emptyset$. A $(1 \pm \varepsilon)$ cut-sparsifier of a hypergraph is then just a re-weighted subset of hyperedges which simultaneously preserves the weight of every cut to a $(1 \pm \varepsilon)$ factor. Analogous to the graph case, spectral sparsifiers of hypergraphs instead operate with an *energy* formulation. Given a hypergraph $H = (V, E)$ and a vector of potentials $x \in [0, 1]^V$, the energy of the hypergraph is given by $Q_H(x) = \sum_{e \in E} w_e \cdot \max_{u,v \in e}(x_u - x_v)^2$, and, as with graphs, a spectral sparsifier is simply a re-weighted subset of hyperedges which preserves this energy to a $(1 \pm \varepsilon)$ factor *simultaneously* for every x.

A line of works by Louis and Makarychev [20], Louis [19], and Chan, Louis, Tang, and Zhang [7] showed that this energy definition in hypergraphs enjoys many of the same properties as the energy definition in graphs: in particular, that it can be viewed as a random walk operator on the hypergraph, that it generalizes the cuts of a hypergraph, and that it admits Cheeger's inequalities in a manner similar to graphs. Because of these connections, a long line of works simultaneously studied the ability to *cut-sparsify* ([17, 8]) and *spectrally-sparsify* ([21, 3, 13, 12, 11, 18]) hypergraphs.

1.1.3 CSP Sparsification

However, subsequent to this unified effort to understand hypergraph sparsification in the cut and spectral settings, the directions of focus from the sparsification community have largely been separate; on the one hand, combinatorial generalizations of cut-sparsification have been studied for more general objects like linear (and non-linear) codes as well as CSPs [15, 16, 6], while on the other hand, continuous generalizations of spectral sparsification have been studied in the context of semi-norms and generalized linear models [9, 10]. The techniques used by these respective research efforts are likewise separate, with combinatorial sparsifiers driven mainly by progress on counting bounds, sampling, and union bounds, while continuous sparsifiers have relied on a deeper understanding of chaining.

In this work, we aim to close this gap between recent work on combinatorial and continuous notions of sparsification by introducing the model of *spectral CSP sparsification*. This framework simultaneously generalizes cut and spectral sparsification of graphs, cut and spectral sparsification of hypergraphs, and the (combinatorial) sparsification of codes and CSPs. We then show that even under this more general definition, there still exist sparsifiers of small sizes, and moreover, these sparsifiers can even be computed efficiently. We summarize our contributions more explicitly in the following subsection.

1.2 Our Contributions

To start, we formally define CSP sparsification, as it will be a key building block in our discussion of spectral CSPs.

▶ **Definition 1.** *A CSP C on n variables and m constraints is given by a predicate $P : \{0,1\}^r \to \{0,1\}$, along with m ordered subsets of variables, denoted $S_1, \dots S_m$ where each $S_j \in [n]^r$. The jth constraint is then denoted by $P(x_{S_j})$, where the arguments in P are understood to be $\{x_\ell : \ell \in S_j\}$. The CSP C can also be accompanied by weights w_c for each constraint $c \in C$. Often, for a constraint $c \in C$, we will use $c(x)$ to denote the evaluation of the constraint c on assignment x. Here it is understood that c may only be acting on a* subset *of the variables in the assignment x.*

▶ **Definition 2.** *Let C be a CSP on n variables and m constraints. For an assignment $x \in \{0,1\}^n$, the value of the assignment is*

$$|C(x)| = \sum_{j=1}^{m} w_j \cdot P(x_{S_j}).$$

In words, this is simply the weight of all satisfied constraints.

▶ **Definition 3.** *For a CSP C on n variables and m constraints, a $(1 \pm \varepsilon)$ sparsifier $\hat{C}$ of C is a re-weighted CSP $\hat{C}$, with weights $\hat{w}_j : j \in [m]$ such that $\forall x \in \{0,1\}^n$:*

$$|\hat{C}(x)| \in (1 \pm \varepsilon) \cdot |C(x)|.$$

The sparsity of $\hat{C}$ is then given by $|\hat{w}|_0$.

As remarked in many previous works, when the predicate $P : \{0,1\}^2 \to \{0,1\}$ is the 2-XOR function, then CSP sparsification captures the notion of cut-sparsification in ordinary graphs. This is because a constraint $x_i \oplus x_j$ simulates an edge (i,j) and will evaluate to 1 if and only if $x_i \neq x_j$ (equivalently, if i,j are on different sides of the cut - see [15] for further discussion). Likewise, when the predicate $P : \{0,1\}^r \to \{0,1\}$ is the *not-all-equal* function (i.e., evaluates to 0 on 0^r and 1^r), then this exactly captures the notion of a hyperedge of arity r being cut. Now, recall that in graphs, the corresponding energy of an edge (i,j) is $(x_i - x_j)^2$. Spectral sparsification of graphs thereby captures cut sparsification as when $x \in \{0,1\}^n$, then $(x_i - x_j)^2 = x_i \oplus x_j$. For arbitrary CSPs, we capture this behavior as follows:

▶ **Definition 4.** *For a vector $x \in [0,1]^n$, and a value $\theta \in [0,1]$, the (deterministic) rounding of x with respect to θ is the vector $x^{(\theta)}$ such that*

$$x_i^{(\theta)} = \mathbf{1}[x_i \geq \theta].$$

This leads to a straightforward definition of the energy of a CSP:

▶ **Definition 5.** *For a CSP C on n variables and m constraints and a vector $x \in [0,1]^n$, the energy of C is defined as*

$$Q_C(x) = \sum_{c \in C} w_c \cdot \left(\Pr_{\theta \in [0,1]} [c(x^{(\theta)}) = 1] \right)^2 .$$

Our notion of spectral sparsification is then a natural continuation of this idea:

▶ **Definition 6.** *For a CSP C on n variables and m constraints, we say that a re-weighted sub-CSP $\hat{C}$ is a $(1 \pm \varepsilon)$ spectral-sparsifier of C if $\forall x \in [0,1]^n$*

$$Q_{\hat{C}}(x) \in (1 \pm \varepsilon) Q_C(x).$$

For instance, let us consider an assignment of potentials $x \in [0,1]^n$, and a constraint $c(x) = x_i \oplus x_j$. Now, our goal is to understand what the expression $\Pr_{\theta \in [0,1]}[c(x^{(\theta)}) = 1]$ looks like; indeed, the only case where the constraint evaluates to 1 is when θ falls in the interval $[x_i, x_j]$. If θ is smaller than $\min\{x_i, x_j\}$, then both become 1 in the rounded vector and the XOR becomes 0, and if θ is larger than $\max\{x_i, x_j\}$, then they both become 0. Thus, the only way for the XOR constraint to become 1 is if θ is larger than exactly one of x_i, x_j: in this case one of the variables gets rounded to 1, and the other gets rounded to 0. Thus, $\Pr_{\theta \in [0,1]}[c(x^{(\theta)}) = 1] = |x_i - x_j|$, and when we consider this expression squared, we recover $(x_i - x_j)^2$, which exactly mirrors the energy for ordinary graphs. A similar analysis shows that when we consider r-NAE constraints, we recover exactly the same energy expression as in hypergraphs of arity r.

Naturally, in the context of sparsification, the next question to ask is whether this more general notion of CSP sparsification still allows for sparsifiers of small sizes. Our first result shows that for a broad class of CSPs, this is indeed the case. Specifically, we build upon the work of [15], and consider *field-affine* CSPs; namely CSPs using a predicate P where $P(x_1, \dots x_r) = \mathbf{1}[\sum_i a_i x_i \neq b_i \mod q]$ for a prime q. For these CSPs, we show the following:

▶ **Theorem 7.** *Let C be any CSP on n variables and m constraints using a predicate $P : \{0,1\}^r \to \{0,1\}$ such that $P(y) = 1 \iff \mathbf{1}[a_i y_i \neq b \mod p]$, for some prime p. Then, there is a randomized, polynomial time algorithm which with high probability computes a $(1 \pm \varepsilon)$ spectral-sparsifier of C that retains only $\widetilde{O}(n^2 \log^2(p)/\varepsilon^2)$ re-weighted constraints.*

Note that this theorem encapsulates a large variety of predicates, including arbitrary arity XORs, graph and hypergraph cut functions, and hedge-graph cut functions [15]. Additionally, while there has been a large research effort for sparsifying continuous functions [9, 10], many classes of CSPs do not fit into the established frameworks. For instance, if we consider a 4-XOR constraint $c(x_1, x_2, x_3, x_4) = x_1 \oplus x_2 \oplus x_3 \oplus x_4$, then we can immediately observe that for the assignment $x_1 = x_2 = 0, x_3 = x_4 = 1/2$, $\Pr_{\theta \in [0,1]}[c(x^{(\theta)}) = 1] = 0$, as there is no choice of θ for which an odd number of the x_i's round to 1. The same holds true when we consider the assignment $x_1 = x_3 = 0, x_2 = x_4 = 1/2$. However, when we *add* these assignments together, yielding $x_1 = 0, x_2 = x_3 = 1/2, x_4 = 1$, then in fact $\Pr_{\theta \in [0,1]}[c(x^{(\theta)}) = 1] = 1$, as there is always an odd number of x_i's that round to 1, for any choice of θ. All together, this means that the XOR functions *strongly disobey* the triangle inequality, which is one of the key properties that results on sparsifying sums of continuous functions rely on [9, 10]. Nevertheless, Theorem 7 shows that these functions still admit sparsifiers of small size, suggesting that there may be other avenues towards showing the sparsifiability of sums of continuous functions.

Finally, we show that despite the generality of our definition of the spectrum of a CSP, our definitions still admit many of the same properties enjoyed by the spectra of graphs and hypergraphs. In particular, we prove that a type of *Cheeger inequality* holds, relating eigenvalues of the CSP to a combinatorial notion of the *expansion* of the CSP. Our Cheeger inequality is defined with respect to the eigenvalues of a discrepancy ratio (defined analogously to [7]), and we use γ_2 to denote the second smallest eigenvalue. Likewise, we define the expansion in an intuitive way to generalize graphs and hypergraphs: the expansion is typically measured as a ratio of the number of edges *leaving* a set S of vertices divided by the number of edges *inside* this same set. Under a CSP perspective, a constraint c is considered to be *leaving* a set S if the constraint c evaluates to 1 when we apply the assignment $\mathbf{1}_S$. For example, in graphs, each edge (u, v) can be modeled by an XOR of the variables x_u and x_v. A simple check shows that the edge (u, v) is only crossing between S and $\bar{S}$ if $(\mathbf{1}_S)_u \oplus (\mathbf{1}_S)_v = 1$. We defer formal definitions of these notions to the full version. Assuming we denote the expansion of a CSP C by Φ_C, we establish the following theorem:

▶ **Theorem 8.** *Given any XOR CSP C where each constraint is of even size and maximum arity ℓ,*

$$\frac{\gamma_2}{2} \leq \Phi_C \leq \left(2\sqrt{\ell/2} + 1\right)\sqrt{\gamma_2},$$

where γ_2 is a notion of the eigenvalue of the XOR-CSP Laplacian.[2]

1.3 Technical Overview

In this subsection, we summarize the main ideas that go into our proof of the ability to spectrally sparsify certain classes of CSPs.

1.3.1 Writing as a Quadratic Form

The starting point for our approach is a technique used in the work of Soma and Yoshida [21]. To convey the main idea, we will consider the simplified setting of spectrally sparsifying XOR constraints of arity 4. That is, we consider the predicate $P(y_1, y_2, y_3, y_4) = y_1 \oplus y_2 \oplus y_3 \oplus y_4$, and each constraint $c_i(x) = P(x_{i_1}, x_{i_2}, x_{i_3}, x_{i_4})$. We denote the entire CSP by C, which consists of constraints $c_1, \dots c_m$, and we denote the set of variables by $x_1, \dots x_n$. We will often index the set $[m]$ by a constraint $c \in C$ (as there are m different constraints). Note that we choose arity 4 XORs instead of 3 so that the all 1's assignment is unsatisfying (and thus we can shift any assignment by the all 1's vector WLOG). We later show that arity 3 XORs can be simulated by arity 4 XORs.

Recall then that our goal is to sparsify the set of constraints while still preserving the energy of the CSP to a $(1 \pm \varepsilon)$ factor, where the energy is exactly

$$Q_C(x) = \sum_{c \in C} w_c \cdot \left(\Pr_{\theta \in [0,1]} [c(x^{(\theta)}) = 1] \right)^2 .$$

Now, when we focus on a single constraint, we can actually interpret this energy definition concretely. Since a single constraint is a 4-XOR over variables $x_{i_1}, \dots x_{i_4}$, then with respect to a choice of the rounding parameter θ, the constraint evaluates to 1 if and only if an odd number of the variables $x_{i_1}, \dots x_{i_4}$ are larger than the rounding threshold θ. Thus, if we sort

[2] See the full version for a formal definition.

$x_{i_1}, \dots x_{i_4}$ according to their value (say the sorted order is $x_{i_{o(1)}} \leq x_{i_{o(2)}} \leq x_{i_{o(3)}} \leq x_{i_{o(4)}}$) the constraint evaluates to 1 if and only if $\theta \in (x_{i_{o(1)}}, x_{i_{o(2)}})$ or $(x_{i_{o(3)}}, x_{i_{o(4)}})$. Because θ is chosen uniformly at random from $[0,1]$, we can directly calculate this probability to be

$$\Pr_{\theta \in [0,1]}[c(x^{(\theta)}) = 1] = x_{i_{o(4)}} - x_{i_{o(3)}} + x_{i_{o(2)}} - x_{i_{o(1)}}.$$

Ultimately, the energy is the *square* of this expression, and so it can be written as

$$\left(\Pr_{\theta \in [0,1]}[c(x^{(\theta)}) = 1] \right)^2 = \left(x_{i_{o(4)}} - x_{i_{o(3)}} + x_{i_{o(2)}} - x_{i_{o(1)}} \right)^2.$$

Here is where we use a simple observation: for a *fixed* ordering of the variables π (i.e., such that $x_{\pi(1)} \leq x_{\pi(2)} \leq \dots \leq x_{\pi(n)}$), the energy expression for each constraint evaluates to a fixed quadratic form. We denote this set of vectors that satisfies the ordering π by $[0,1]^\pi$, and call a vector x satisfying $x_{\pi(1)} \leq x_{\pi(2)} \leq \dots \leq x_{\pi(n)}$ π-consistent. Said another way, once we fix the ordering of the variables, then $o(1), o(2), o(3), o(4)$ in the above expression would also all be fixed. For such a permutation π, we call this resulting quadratic form the *induced quadratic form by* π. For example, if we take the ordering π to be the identity permutation (i.e., $x_1 \leq x_2 \leq \dots \leq x_n$), and we suppose that $i_1 < i_2 < i_3 < i_4$, then the formula for the energy of the expression would be exactly $(x_{i_4} - x_{i_3} + x_{i_2} - x_{i_1})^2$. Note that our choice of the words *quadratic form* is intentional, as once the energy expression is a fixed sum of these quadratic terms, we can express the energy as $Q_C(x) = x^T (B^\pi)^T W_C B^\pi x$, where $x \in [0,1]^\pi$, and $B^\pi \in \{-1, 0, 1\}^{m \times n}$ is the matrix such that $((B^\pi)x)_i = \Pr_{\theta \in [0,1]}[c_i(x^{(\theta)}) = 1]$, and W_C is simply the diagonal matrix whose i,ith entry is the weight of the ith constraint in C.

1.3.2 Sufficient Conditions for Preserving Quadratic Forms

Now, as observed in [21], in order to sparsify the CSP C, it suffices to choose a re-weighted subset of indices $\hat{S} \subseteq [m]$, such that *simultaneously for every ordering* π, the induced quadratic form by π has its energy preserved to a $(1 \pm \varepsilon)$ factor. Because of our above simplifications, we can re-write this another way: we want to find a new, sparser set of weights on our constraints (which we denote by $\widehat{C}$), such that *for every* ordering π, and *for every* vector $x \in [0,1]^\pi$, it is the case that

$$Q_{\hat{C}}(x) = x^T (B^\pi)^T W_{\hat{C}} B^\pi x \in (1 \pm \varepsilon) x^T (B^\pi)^T W_C B^\pi x = (1 \pm \varepsilon) Q_C(x).$$

We can re-write this condition as saying that

$$x^T \left((B^\pi)^T W_{\hat{C}} B^\pi - (1 - \varepsilon)(B^\pi)^T W_C B^\pi \right) x \geq 0,$$

and

$$x^T \left((1 + \varepsilon)(B^\pi)^T W_C B^\pi - (B^\pi)^T W_{\hat{C}} B^\pi \right) x \geq 0.$$

If this condition holds, then we will have indeed created a $(1 \pm \varepsilon)$ spectral sparsifier. Recall that we want to use *as few* re-weighted constraints as possible, and so we want this new set of re-weighted constraints $\hat{C}$ to be as sparse as possible.

Next, observe that the expression above is a type of *positive definiteness* condition, where we focus our attention on vectors in $[0,1]^\pi$, and want to ensure that the quadratic form of some matrix is non-negative. In particular, the work of [21] studied similar matrices, and gave an exact characterization for when such a claim holds:

▶ **Lemma 9** (Lemma 3.3 in [21]). *Let $A \in \mathbb{R}^{n \times n}$ be a symmetric matrix, such that $A \cdot \mathbf{1} = 0$, and let π be a permutation on $[n]$. Then, $x^T Ax \geq 0$ for all $x \in [0,1]^\pi$ if and only if A is of the form $P_\pi^T J^T KJP_\pi$ for a matrix $K \in \mathbb{R}^{(n-1)\times(n-1)}$ such that $\forall y \in \mathbb{R}_+^n, y^T Ky \geq 0$, and where $P_\pi \in \mathbb{R}^{n \times n}$ is the matrix such that $P_\pi(i,j) = 1$ if $j = \pi_i$, and otherwise is 0 and $J \in \mathbb{R}^{(n-1)\times n}$, where $J(i,i) = 1$, $J(i,i+1) = -1$, and otherwise is 0.*

This lemma gives us a roadmap for how to proceed. Recall that there are two matrices that we wish to show satisfy the positivity condition, namely $\left((B^\pi)^T W_{\hat{C}} B^\pi - (1-\varepsilon)(B^\pi)^T W_C B^\pi\right)$ and $\left((1+\varepsilon)(B^\pi)^T W_C B^\pi - (B^\pi)^T W_{\hat{C}} B^\pi\right)$, and these will each be the matrix A in the above lemma (though to simplify discussion, we will simply focus on the first, as the analysis will be identical). Our goal is to show that we can re-write the matrix as

$$\left((B^\pi)^T W_{\hat{C}} B^\pi - (1-\varepsilon)(B^\pi)^T W_C B^\pi\right) = P_\pi^T J^T KJP_\pi,$$

where the matrix K is *co-positive* (meaning for all positive vectors, the quadratic form is non-negative). To do this, we will ultimately define a new, specific matrix B^π_{cross} with the intention that $B^\pi = B^\pi_{\text{cross}} JP_\pi$. While [21] also define a crossing matrix, ours is necessarily different as it models a different structure. It is here where our analysis begins to diverge from theirs.

1.3.3 Building the Crossing Matrix

However, before defining this crossing matrix B^π_{cross}, we first require some definitions. First is the notion of an *active region*, which is essentially the intervals where the contribution to the energy comes from (i.e., the choices of θ for which a constraint is satisfied):

▶ **Definition 10.** *For a fixed ordering π on $[n]$, and a 4-XOR constraint c operating on elements $x_{i_1} \leq x_{i_2} \leq x_{i_3} \leq x_{i_4}$, we say the active regions are*

$$[x_{i_1}, x_{i_2}] \cup [x_{i_3}, x_{i_4}]$$

For an index $i \in [n-1]$, let us consider the bisector between x_{π_i} and $x_{\pi_{i+1}}$ (i.e., between the ith and $i+1$st smallest elements in the ordering π). This bisector takes on value $(x_{\pi_i} + x_{\pi_{i+1}})/2$. We use this to define *crossing indices*:

▶ **Definition 11.** *For $i \in [n-1]$, we say that i is a crossing index with respect to the ordering π and constraint c if the bisector $(x_{\pi_i} + x_{\pi_{i+1}})/2$ is in an active region of the constraint c with ordering π.*

Likewise, we generalize this definition to a pair of indices $i, j \in [n-1]$:

▶ **Definition 12.** *We say that pairs of indices $i, j \in \binom{[n-1]}{2}$ are crossing indices with respect to the constraint c, ordering π if both i, j individually are crossing indices with c, π.*

For instance, let us consider what happens when the permutation $\pi = \text{Id}_n$, and suppose we are looking at a constraint $c_i(x)$ which is the XOR of x_2, x_3, x_6, x_9. Then the crossing indices are exactly $2, 6, 7$ and 8. More intuitively, the crossing indices correspond to the intervals $[x_i, x_{i+1}]$ which contribute to the overall energy (i.e., for which choices of θ does the constraint evaluate to 1). For instance, because $[x_6, x_9]$ is an active region the energy contributed from these points is $x_9 - x_6$ and so every sub-interval $[x_6, x_7], [x_7, x_8], [x_8, x_9]$ contributes to the energy. This is because if $\theta \in [x_7, x_8]$ for instance, then in the rounding $x_2 = x_3 = x_6 = 1$, and $x_9 = 0$, and hence the constraint is satisfied.

With these definitions established, we define the crossing matrix $B^\pi_{\text{cross}} \in \mathbb{R}^{m \times n-1}$ such that $B^\pi_{\text{cross}}(c, i) = 1$ if and only if i is a crossing index for the constraint c under permutation π (and otherwise, the c, ith entry is 0). In particular, this specific definition allows us to show that

$$B^\pi = B^\pi_{\text{cross}} J P_\pi.$$

We do not discuss this equality exactly here, as it requires an extensive case analysis; we defer this to the technical sections below.

However, using this, we can re-write our original expression:

$$\left((B^\pi)^T W_{\hat{C}} B^\pi - (1-\varepsilon)(B^\pi)^T W_C B^\pi\right) = P_\pi^T J^T ((B^\pi_{\text{cross}})^T W_{\hat{C}} B^\pi_{\text{cross}} - (1-\varepsilon)(B^\pi_{\text{cross}})^T W_C B^\pi_{\text{cross}}) J P_\pi.$$

Thus, if we revisit Lemma 9, the matrix K is exactly this interior portion of the expression $K = ((B^\pi_{\text{cross}})^T W_{\hat{C}} B^\pi_{\text{cross}} - (1-\varepsilon)(B^\pi_{\text{cross}})^T W_C B^\pi_{\text{cross}})$, and thus our goal becomes to show that $y^T K y \geq 0 \ \forall y \in \mathbb{R}^{n-1}_+$ (while allowing for sparsity in the selected constraints of course).

1.3.4 Understanding Crossing Indices Through Codes

Now, we make a simple observation: a sufficient (but not necessary) condition for K to satisfy $y^T K y \geq 0 \ \forall y \in \mathbb{R}^{n-1}_+$ is for *every entry* in K to be non-negative. Thus, it remains to understand exactly what the entries in K look like: we start by looking at a simpler expression, namely $(B^\pi_{\text{cross}})^T W_C B^\pi_{\text{cross}}$. Indeed, in this matrix the i, jth entry can be written as

$$\left((B^\pi_{\text{cross}})^T W_C B^\pi_{\text{cross}}\right)_{i,j} = \sum_{c \in C} w_c \cdot \mathbf{1}[i \text{ crossing } c, \pi \wedge j \text{ crossing } c, \pi] = d_{C,\pi}(i, j),$$

where we simply use $d_{C,\pi}(i, j)$ to denote the total weight of constraints that have both i and j as crossing indices under the permutation π.

Here comes the final, key technical lemma: we show that there is a matrix $G \in_2^{m \times n^2}$ such that *for every* choice of i, j and permutation π, there is a vector $z_{\pi,i,j} \in_2^{n^2}$ such that $(G z_{\pi,i,j})_c$ (i.e., the cth coordinate of the vector $(G z_{\pi,i,j})$) is exactly the indicator of whether or not i, j are crossing indices for the constraint c under permutation π. Thus, this value $d_{C,\pi}(i, j) = \left((B^\pi_{\text{cross}})^T W_C B^\pi_{\text{cross}}\right)_{i,j}$ can be written as the *weighted hamming weight* of the vector $(G z_{\pi,i,j})$.

To construct this matrix G, we start with the generating matrix of our original XOR CSP C: this is the matrix $F \in_2^{m \times n}$, where for the constraint c operating on $x_{u_1}, x_{u_2}, x_{u_3}, x_{u_4}$, there is a single row in the matrix F corresponding to c, with 1's exactly in columns u_1, u_2, u_3, u_4. Now, the matrix G is a type of *tensor-product code* that is generated by $F \oplus F$. Explicitly, when we consider the linear space generated by F (denoted $\text{Im}(F)$), we want

$$\text{Im}(G) = \text{Im}(F \oplus F) = \{z_1 \cdot z_2 : z_1, z_2 \in \text{Im}(F)\},$$

where $z_1 \cdot z_2$ refers to the *entry-wise* multiplication of the two vectors. Note that G will be expressible as a space of dimension $\leq O(n^2)$ as it essentially corresponds to a space of degree 2 polynomials over n variables (though see the full version for a more thorough discussion).

Now, recall that our goal is to show that for a fixed permutation π, as well as indices i, j, that we can find a vector in $\text{Im}(G)$ whose ℓth coordinate is exactly the indicator of whether i, j are crossing indices for the ℓth constraint under permutation π. For simplicity, let us suppose that the permutation π is just the identity permutation; under this permutation, recall that an index i is considered to be a crossing index for a constraint c if and only if i is in an active region of c. If we denote the variables in the constraint c by $x_{u_1}, \dots x_{u_4}$, then i is a crossing index under π *if and only if* x_i is greater than or equal to an odd number of $x_{u_1}, x_{u_2}, x_{u_3}, x_{u_4}$: this is because the active regions are $[x_{u_1}, x_{u_2}]$ and $[x_{u_3}, x_{u_4}]$, and so x_i must either be between $[x_{u_1}, x_{u_2}]$, or between $[x_{u_3}, x_{u_4}]$.

A priori, it may seem that this condition is completely arbitrary. However, it can be *exactly* captured by our matrix F: indeed, let us consider the vector $e_{\leq i} \in_2^n$ which is 1 in the first i entries, and 0 in the others. The ℓth constraint then performs an XOR on the variables $x_{u_1}, x_{u_2}, x_{u_3}, x_{u_4}$. Plugging this vector in, we obtain that

$$(Fe_{\leq i})_\ell = (e_{\leq i})_{u_1} \oplus (e_{\leq i})_{u_2} \oplus (e_{\leq i})_{u_3} \oplus (e_{\leq i})_{u_4},$$

which is *exactly* the indicator of whether x_i is greater than or equal to an odd number of $x_{u_1}, x_{u_2}, x_{u_3}, x_{u_4}$ (since we are using the identity ordering of $x_1 \leq x_2 \leq \cdots \leq x_n$).

Thus, the matrix F alone captures when single indices are crossing a constraint. To generalize to *pairs* of crossing indices, we then simply take the coordinate-wise product of whether indices i, j are *both* crossing a constraint c under a permutation π. This is exactly what is captured by the code $\mathrm{Im}(G)$.

1.3.5 Creating Sparsifiers through Code Sparsification

Because of this correspondence above, this means that $d_{C,\pi}(i,j)$ can be exactly written as the *weighted hamming weight* of the *codeword* $Gz_{\pi,i,j}$, where $z_{\pi,i,j}$ is the particular vector we use to enforce

$$(Gz_{\pi,i,j})_c = \mathbf{1}[i \text{ crossing } c, \pi \wedge j \text{ crossing } c, \pi] = d_{C,\pi}(i,j).$$

To conclude then, we recall the work of Khanna, Putterman, and Sudan [15], who showed that every code admits a sparsifier which preserves only a set of coordinates of size nearly-linear in the dimension of the code. In our case, the dimension is $O(n^2)$, so this means that there exists a re-weighted subset of coordinates $\hat{C}$, of size $\widetilde{O}(n^2/\varepsilon^2)$ which preserves the weight of every codeword to a $(1 \pm \varepsilon)$ factor. In particular, this means for the CSP defined on the same re-weighted subset of constraints, it must be the case that

$$d_{\hat{C},\pi}(i,j) \in (1 \pm \varepsilon) d_{C,\pi}(i,j).$$

By sparsifying this auxiliary code, we obtain a re-weighted subset of the constraints which satisfies that the parameter $d_{C,\pi}(i,j)$ is approximately preserved. When we revisit the matrices we created: $K = ((B^\pi_{\text{cross}})^T W_{\hat{C}} B^\pi_{\text{cross}} - (1-\varepsilon)(B^\pi_{\text{cross}})^T W_C B^\pi_{\text{cross}})$, because the entries in $(B^\pi_{\text{cross}})^T W_{\hat{C}} B^\pi_{\text{cross}}$ are exactly $d_{\hat{C},\pi}(i,j)$, and the entries in $(B^\pi_{\text{cross}})^T W_C B^\pi_{\text{cross}}$ are exactly $d_{C,\pi}(i,j)$, we see that the constraints $\hat{C}$ computed by the code sparsifier do indeed lead to a matrix with all non-negative entries as

$$d_{\hat{C},\pi}(i,j) - (1-\varepsilon) d_{C,\pi}(i,j) \geq 0,$$

and thus this matrix is non-negative on all non-negative vectors (and so too constitutes a spectral sparsifier by our previous discussions).

Likewise, observe that because the matrix G we defined above is *the same* for all permutations π (we only need to change the *vector* $e_{\leq i}$ above that we multiply the matrix by), this implies that the *same* set of constraints will be a sparsifier across all choices of permutations, and thus we have a set of $\widetilde{O}(n^2/\varepsilon^2)$ constraints which is a spectral CSP sparsifier of our original instance. By generalizing this argument (and using the efficient constructions of code sparsifiers [16]), we obtain Theorem 7.

1.3.6 Discussion

We end the technical overview with some high-level remarks:

1. Although we build on the framework of [21], the hypergraph sparsifiers in [21] were of size $O(n^3/\varepsilon^2)$. Our refinement of their framework (specifically, the explicit connection with codes), allows for an improved sparsifier size of $\widetilde{O}(n^2/\varepsilon^2)$.
2. Likewise, the framework from [21] is particular suited for sparsifying hypergraphs. Our framework reveals a much more general connection between spectral sparsification and sparsifying a type of *tensor product* of the underlying object which holds *across the entire CSP landscape.*
3. The method presented in this paper avoids any complex continuous machinery like chaining or matrix-chernoff, which have appeared in prior works on spectral sparsification [13, 12, 9]. We view it is an interesting open question whether those techniques can be combined with ours to create nearly-linear size spectral CSP sparsifiers.

References

1 Noga Alon. Eigenvalues and expanders. *Comb.*, 6(2):83–96, 1986. doi:10.1007/BF02579166.

2 Noga Alon and V. D. Milman. lambda$_1$, isoperimetric inequalities for graphs, and superconcentrators. *J. Comb. Theory B*, 38(1):73–88, 1985. doi:10.1016/0095-8956(85)90092-9.

3 Nikhil Bansal, Ola Svensson, and Luca Trevisan. New notions and constructions of sparsification for graphs and hypergraphs. In David Zuckerman, editor, *60th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2019, Baltimore, Maryland, USA, November 9-12, 2019*, pages 910–928. IEEE Computer Society, 2019. doi:10.1109/FOCS.2019.00059.

4 Joshua D. Batson, Daniel A. Spielman, and Nikhil Srivastava. Twice-Ramanujan sparsifiers. In Michael Mitzenmacher, editor, *Proceedings of the 41st Annual ACM Symposium on Theory of Computing, STOC 2009, Bethesda, MD, USA, May 31 - June 2, 2009*, pages 255–262. ACM, 2009. doi:10.1145/1536414.1536451.

5 András A. Benczúr and David R. Karger. Approximating s-t minimum cuts in $\tilde{O}(n^2)$ time. In Gary L. Miller, editor, *Proceedings of the Twenty-Eighth Annual ACM Symposium on the Theory of Computing, Philadelphia, Pennsylvania, USA, May 22-24, 1996*, pages 47–55. ACM, 1996. doi:10.1145/237814.237827.

6 Joshua Brakensiek and Venkatesan Guruswami. Redundancy is all you need. *arXiv preprint arXiv:2411.03451*, 2024. doi:10.48550/arXiv.2411.03451.

7 T.-H. Hubert Chan, Anand Louis, Zhihao Gavin Tang, and Chenzi Zhang. Spectral properties of hypergraph laplacian and approximation algorithms. *J. ACM*, 65(3):15:1–15:48, 2018. doi:10.1145/3178123.

8 Yu Chen, Sanjeev Khanna, and Ansh Nagda. Near-linear size hypergraph cut sparsifiers. In Sandy Irani, editor, *61st IEEE Annual Symposium on Foundations of Computer Science, FOCS 2020, Durham, NC, USA, November 16-19, 2020*, pages 61–72. IEEE, 2020. doi:10.1109/FOCS46700.2020.00015.

9 Arun Jambulapati, James R. Lee, Yang P. Liu, and Aaron Sidford. Sparsifying sums of norms. *CoRR*, abs/2305.09049, 2023. doi:10.48550/arXiv.2305.09049.

10 Arun Jambulapati, James R. Lee, Yang P. Liu, and Aaron Sidford. Sparsifying generalized linear models. In Bojan Mohar, Igor Shinkar, and Ryan O'Donnell, editors, *Proceedings of the 56th Annual ACM Symposium on Theory of Computing, STOC 2024, Vancouver, BC, Canada, June 24-28, 2024*, pages 1665–1675. ACM, 2024. doi:10.1145/3618260.3649684.

11 Arun Jambulapati, Yang P. Liu, and Aaron Sidford. Chaining, group leverage score overestimates, and fast spectral hypergraph sparsification. In Barna Saha and Rocco A. Servedio, editors, *Proceedings of the 55th Annual ACM Symposium on Theory of Computing, STOC 2023, Orlando, FL, USA, June 20-23, 2023*, pages 196–206. ACM, 2023. doi:10.1145/3564246.3585136.

12 Michael Kapralov, Robert Krauthgamer, Jakab Tardos, and Yuichi Yoshida. Spectral hypergraph sparsifiers of nearly linear size. In *62nd IEEE Annual Symposium on Foundations of Computer Science, FOCS 2021, Denver, CO, USA, February 7-10, 2022*, pages 1159–1170. IEEE, 2021. doi:10.1109/FOCS52979.2021.00114.

13 Michael Kapralov, Robert Krauthgamer, Jakab Tardos, and Yuichi Yoshida. Towards tight bounds for spectral sparsification of hypergraphs. In Samir Khuller and Virginia Vassilevska Williams, editors, *STOC '21: 53rd Annual ACM SIGACT Symposium on Theory of Computing, Virtual Event, Italy, June 21-25, 2021*, pages 598–611. ACM, 2021. doi:10.1145/3406325.3451061.

14 David R. Karger. Global min-cuts in rnc, and other ramifications of a simple min-cut algorithm. In Vijaya Ramachandran, editor, *Proceedings of the Fourth Annual ACM/SIGACT-SIAM Symposium on Discrete Algorithms, 25-27 January 1993, Austin, Texas, USA*, pages 21–30. ACM/SIAM, 1993. URL: http://dl.acm.org/citation.cfm?id=313559.313605.

15 Sanjeev Khanna, Aaron Putterman, and Madhu Sudan. Code sparsification and its applications. In *Proceedings of the 2024 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 5145–5168. SIAM, 2024.

16 Sanjeev Khanna, Aaron L Putterman, and Madhu Sudan. Characterizations of sparsifiability for affine CSPs and symmetric CSPs. *arXiv preprint arXiv:2404.06327*, 2024.

17 Dmitry Kogan and Robert Krauthgamer. Sketching cuts in graphs and hypergraphs. In Tim Roughgarden, editor, *Proceedings of the 2015 Conference on Innovations in Theoretical Computer Science, ITCS 2015, Rehovot, Israel, January 11-13, 2015*, pages 367–376. ACM, 2015. doi:10.1145/2688073.2688093.

18 James R. Lee. Spectral hypergraph sparsification via chaining. In Barna Saha and Rocco A. Servedio, editors, *Proceedings of the 55th Annual ACM Symposium on Theory of Computing, STOC 2023, Orlando, FL, USA, June 20-23, 2023*, pages 207–218. ACM, 2023. doi:10.1145/3564246.3585165.

19 Anand Louis. Hypergraph markov operators, eigenvalues and approximation algorithms. In Rocco A. Servedio and Ronitt Rubinfeld, editors, *Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing, STOC 2015, Portland, OR, USA, June 14-17, 2015*, pages 713–722. ACM, 2015. doi:10.1145/2746539.2746555.

20 Anand Louis and Yury Makarychev. Approximation algorithms for hypergraph small set expansion and small set vertex expansion. In Klaus Jansen, José D. P. Rolim, Nikhil R. Devanur, and Cristopher Moore, editors, *Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, APPROX/RANDOM 2014, September 4-6, 2014, Barcelona, Spain*, volume 28 of *LIPIcs*, pages 339–355. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2014. doi:10.4230/LIPICS.APPROX-RANDOM.2014.339.

21 Tasuku Soma and Yuichi Yoshida. Spectral sparsification of hypergraphs. In Timothy M. Chan, editor, *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2019, San Diego, California, USA, January 6-9, 2019*, pages 2570–2581. SIAM, 2019. doi:10.1137/1.9781611975482.159.

22 Daniel A. Spielman. Spectral graph theory and its applications. In *48th Annual IEEE Symposium on Foundations of Computer Science (FOCS'07)*, pages 29–38, 2007. doi:10.1109/FOCS.2007.56.

23 Daniel A. Spielman and Shang-Hua Teng. Spectral sparsification of graphs. *SIAM J. Comput.*, 40(4):981–1025, 2011. doi:10.1137/08074489X.