HyperEF: Spectral Hypergraph Coarsening by Effective-Resistance Clustering

Ali Aghdaei
Stevens Institute of Technology
aaghdae1@stevens.edu

Zhuo Feng
Stevens Institute of Technology
zhuo.feng@stevens.edu

***Abstract*—This paper introduces a scalable algorithmic framework (HyperEF) for spectral coarsening (decomposition) of large-scale hypergraphs by exploiting hyperedge effective resistances. Motivated by the latest theoretical framework for low-resistance-diameter decomposition of simple graphs, HyperEF aims at decomposing large hypergraphs into multiple node clusters with only a few inter-cluster hyperedges. The key component in HyperEF is a nearly-linear time algorithm for estimating hyperedge effective resistances, which allows incorporating the latest diffusion-based non-linear quadratic operators defined on hypergraphs. To achieve good runtime scalability, HyperEF searches within the Krylov subspace (or approximate eigensubspace) for identifying the nearly-optimal vectors for approximating the hyperedge effective resistances. In addition, a node weight propagation scheme for multilevel spectral hypergraph decomposition has been introduced for achieving even greater node coarsening ratios. When compared with state-of-the-art hypergraph partitioning (clustering) methods, extensive experiment results on real-world VLSI designs show that HyperEF can more effectively coarsen (decompose) hypergraphs without losing key structural (spectral) properties of the original hypergraphs, while achieving over $70\times$ runtime speedups over hMetis and $20\times$ speedups over HyperSF.**

***Index Terms*—hypergraph coarsening, effective resistance, spectral graph theory, graph clustering**

I. Introduction

Recent years have witnessed a surge of interest in graph learning techniques for various applications such as vertex (data) classification [11], [25], link prediction (recommendation systems) [28], [37], drug discovery [21], [26], solving partial differential equations (PDEs) [13], [20], and electronic design automation (EDA) [17], [23], [35], [36]. The ever-increasing complexity of modern graphs (networks) inevitably demands the development of efficient techniques to reduce the size of the input datasets while preserving the essential properties. To this end, many research works on graph partitioning, graph embedding, and graph neural networks (GNNs) exploit graph coarsening techniques to improve the algorithm scalability, and accuracy [6], [7], [27], [40].

Hypergraphs are more general than simple graphs since they allow modeling higher-order relationships among the entities [24]. The state-of-the-art hypergraph coarsening techniques are all based on relatively simple heuristics, such as the vertex similarity or hyperedge similarity [3], [8], [16], [29], [34]. For example, the hyperedge similarity based coarsening techniques contract similar hyperedges with large sizes into smaller ones, which can be easily implemented but may impact the original hypergraph structural properties; the vertex-similarity based algorithms rely on checking the distances between vertices for discovering strongly-coupled (correlated) node clusters, which can be achieved leveraging hypergraph embedding that maps each vertex into a low-dimensional vector such that the Euclidean distance (coupling) between the vertices can be easily computed in constant time. However, these **simple metrics often fail to capture higher-order global (structural) relationships in hypergraphs**.

The latest theoretical breakthroughs in spectral graph theory have led to the development of nearly-linear time spectral graph sparsification (edge reduction) [9], [10], [14], [15], [19], [33], [38] and coarsening (node reduction) algorithms [22], [39]. However, **existing spectral methods are only applicable to simple graphs** but not hypergraphs. For example, the effective-resistance based spectral sparsification method [31] exploits an effective-resistance based edge sampling scheme, while the latest practically-efficient sparsification algorithm [10] leverages generalized eigenvectors for recovering the most influential off-tree edges for mitigating the mismatches between the subgraph and the original graph.

On the other hand, spectral theory for hypergraphs has been less developed due to the more complicated structure of hypergraphs. For example, a classical spectral method has been proposed for hypergraphs by converting each hyperedge into undirected edges using star or clique expansions [12]. However, such a naive hyperedge conversion scheme may result in lower performance due to ignoring the multi-way high-order relationship between the entities. A more rigorous approach by Tasuku and Yuichi [30] generalized spectral graph sparsification for hypergraph setting by sampling each hyperedge according to a probability determined based on the ratio of the hyperedge weight over the minimum degree of two vertices inside the hyperedge. Another family of spectral methods for hypergraphs explicitly builds the Laplacian matrix to analyze the spectral properties of hypergraphs: Zhou et al. proposes a method to create the Laplacian matrix of a hypergraph and generalize graph learning algorithms for hypergraph applications [41]. A more mathematically rigorous approach by Chan et al. introduces a nonlinear diffusion process for defining the hypergraph Laplacian operator by measuring the flow distribution within each hyperedge [4], [5]; moreover,

the Cheeger's inequality has been proved for hypergraphs under the diffusion-based nonlinear Laplacian operator [5]. However, these theoretical results do not immediately allow for practically-efficient implementations.

This paper presents a scalable spectral hypergraph coarsening algorithm via effective-resistance clustering to generate much smaller hypergraphs that can still well preserve the original hypergraph structural (global) properties. The key contribution of this work is summarized below:

1) To the best of our knowledge, for the first time, we propose to extend the simple-graph effective resistance formulation to the hypergraph setting by exploiting approximate eigenvectors (Krylov subspace) associated with the hypergraph.
2) Our approach (HyperEF) allows incorporating the recently developed diffusion-based non-linear Laplacian operator defined on hypergraphs [5] for highly-efficient estimation of hyperedge effective resistances.
3) A scalable spectral hyperedge contraction (clustering) scheme and a node weight propagation scheme have been proposed for constructing coarse-level hypergraphs that can well preserve the original spectral (structural) properties.
4) Our extensive experiment results on real-world VLSI designs show that HyperEF is over $70\times$ faster than hMetis, while achieving comparable or better average conductance values for most hypergraph decomposition tasks.

The rest of the paper is organized as follows. In Section II, we provide a background introduction to the basic concepts related to the spectral hypergraph theory. In Section III, we introduce an optimization-based formulation for effective resistance estimation in simple graphs. In section IV, we introduce the technical details of HyperEF and its algorithm flow. In section V, we introduce a multilevel spectral hypergraph decomposition framework. In section VI, we provide the complexity analysis of HyperEF. In Section VII, we demonstrate extensive experimental results to evaluate the performance of HyperEF using a variety of real-word VLSI design benchmarks, which is followed by the conclusion of this work in Section VIII.

II. Background

Classical spectral graph theory shows that the structure of a simple graph is closely related to the graph's spectral properties. Specifically, the Cheeger's inequality shows the close connection between expansion or *conductance* and the first few eigenvalues of graph Laplacians [18]. Moreover, the Laplacian quadratic form computed with the Fiedler vector (the eigenvector corresponding to the smallest nonzero Laplacian eigenvalue) has been exploited to find the minimum boundary size or *cut* for graph partitioning tasks [32]. In recent years, spectral theories for hypergraphs have been extensively studied by theoretical computer scientists [5]. To allow a better understanding of our work, the following important definitions for hypergraphs are introduced.

Definition II.1. *Let $H = (V, E, w)$ denote a weighted hypergraph, where $w : E \to \mathbb{R}_+$ is a weight function over hyperedges, $d_u := \Sigma_{u\in e, e\in E} w_e$, and $vol(S) := \Sigma_{S\in V: u\in S} d_u$. The conductance of a given node set $S \in V$ is defined as*

$$\mathcal{C}_H(S) := \frac{cut(S, \bar{S})}{min\{vol(S), vol(\bar{S})\}}, \tag{1}$$

where the cut is the sum of the weights of the hyperedges containing the nodes from both S and $\bar{S}$. The hypergraph conductance is defined as $\mathcal{C}_H := \min_{\emptyset \not\subseteq S\subseteq V} \mathcal{C}_H(S)$.

Definition II.2. *The non-linear quadratic form of a hypergraph $H = (V, E, w)$ for any input vector $\chi \in \mathbb{R}^V$ is defined as [5]*

$$Q_H(\chi) := \sum_{e\in E} w_e \max_{u,v\in e} (\chi_u - \chi_v)^2. \tag{2}$$

III. Effective Resistances in Simple Graphs

Definition III.1. *Let $G = (\mathcal{V}, \mathcal{E}, z)$ denote a weighted and connected undirected graph with weights $z \in \mathbb{R}^{\mathcal{V}}_{\geq 0}$, $b_p \in \mathbb{R}^{\mathcal{V}}$ denote the standard basis vector with all zero entries except for the p-th entry being 1, and $b_{pq} = b_p - b_q$, respectively. The effective resistance between nodes $(p, q) \in \mathcal{V}$ is defined as*

$$R_{eff}(p, q) = b_{pq}^\top L_G^\dagger b_{pq} = \sum_{i=2}^{|\mathcal{V}|} \frac{(u_i^\top b_{pq})^2}{\lambda_i} = \sum_{i=2}^{|\mathcal{V}|} \frac{(u_i^\top b_{pq})^2}{u_i^\top L_G u_i}, \tag{3}$$

where $L_G^\dagger$ denotes the Moore-Penrose pseudo-inverse of the graph Laplacian matrix L_G, and $u_i \in \mathbb{R}^{\mathcal{V}}$ for $i = 1, ..., |\mathcal{V}|$ denote the unit-length, mutually-orthogonal eigenvectors corresponding to Laplacian eigenvalues λ_i for $i = 1, ..., |\mathcal{V}|$.

Theorem 1. *The effective resistance between p and q can be computed by solving the following optimization problem:*

$$R_{eff}(p, q) = \max_{x\in\mathbb{R}^{\mathcal{V}}} \frac{(x^\top b_{pq})^2}{x^\top L_G x}. \tag{4}$$

Proof. The original problem in (4) can be alternatively solved by tackling the following convex optimization problem:

$$\begin{aligned} &\min_{x\in\mathbb{R}^{\mathcal{V}}} x^\top L_G x \\ &s.t. \quad x^\top b_{pq} = 1. \end{aligned} \tag{5}$$

After introducing a Lagrangian multiplier λ, the following Lagrangian function needs to be minimized

$$F(x, \lambda) = x^\top L_G x - \lambda(x^\top b_{pq} - 1), \tag{6}$$

which will lead to the following optimal solution:

$$\lambda^* = \frac{1}{R_{eff}(p, q)}, \quad x^* = \frac{L_G^\dagger b_{pq}}{R_{eff}(p, q)}. \tag{7}$$

Substituting x^* into (4) will complete the proof. □

IV. Spectral Coarsening via Resistance Clustering

A. Overview of HyperEF

A recent theoretical work proves that it is possible to decompose a simple graph into multiple node clusters with effective-resistance diameter at most the inverse of average node degree (up to constant losses) by removing only a

Table I: Symbol descriptions

symbols	descriptions	symbols	descriptions	symbols	descriptions
$H=(V,E,w)$	weighted hypergraph	$G=(\mathcal{V},\mathcal{E},z)$	weighted simple graph	$H'=(V',E',w')$	coarsened hypergraph
E	hyperedge set	$\mathcal{E}$	simple graph edge set	V'	node set in coarsened hypergraph
w	hyperedge weight set	z	simple graph edge weight set	E'	hyperedge set in coarsened hypergraph
$\mathcal{C}_H(S)$	conductance of set S in H	$R_{eff}(p,q)$	effective resistance between nodes p and q in G	w'	hyperedge weight set in coarsened hypergraph
$\mathcal{C}$	average conductance of clusters in H	r_e^i	hyperedge ratio of e associated with $\chi^{(i)}$	ρ	the order of Krylov subspace
$Q_H(\chi)$	non-linear quadratic form in H	R_e	effective resistance of hyperedge e	χ	node embedding vector in H
$G_b=(\mathcal{V}_b,\mathcal{E}_b,z_b)$	bipartite graph of H	x	node embedding vector in G	d_u	number of hyperedges belong to node u
$\mathcal{V}_b$	node set in bipartite graph	$\mathcal{C}_G$	conductance of G	L	number of levels coarsening the hypergraph
$\mathcal{E}_b$	edge set in bipartite graph	m	number of resistance ratios	η	node weights in hypergraph
z_b	edge weight set in bipartite graph	R	vector of hyperedge effective resistances	δ	effective resistance threshold

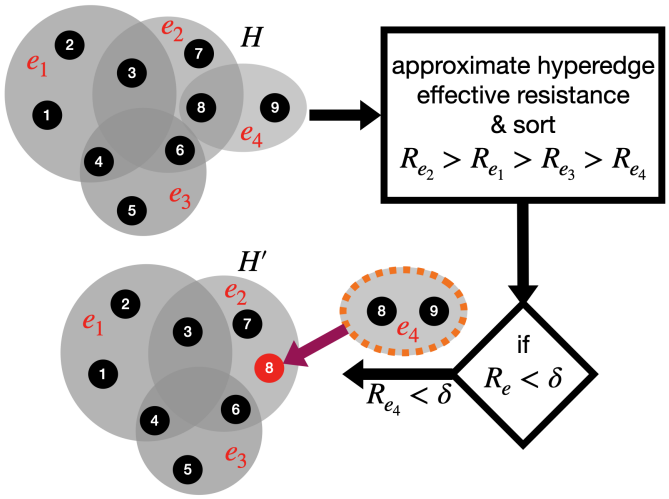

Figure 1: Overview of the proposed HyperEF framework.

constant fraction of edges [2]. This result also leads to the development of algorithms that use effective-resistance metric for spectral graph clustering (decomposition) and spectral graph sparsification [2].

In [15], an effective-resistance based spectral algorithm has been proposed for sparsification of hypergraphs. Specifically, by sampling each hyperedge according to its effective resistance, nearly-linear-sized sparsifiers can be obtained for hypergraphs [15]. However, such a method involves a non-trivial procedure for estimating hyperedge effective resistances and may not allow for practically efficient implementations since the required hypergraph-to-graph conversion using clique expansion and the iterative edge weight updating scheme will drastically increase the algorithm complexity [15].

Inspired by the aforementioned research, we propose a spectral hypergraph coarsening method (HyperEF) that aggressively clusters the nodes within each hyperedge with low effective-resistance diameters (as shown in Figure 1), such that much smaller hypergraphs can be constructed without impacting the key structural properties of the original hypergraph. A key technical component in HyperEF is a highly-efficient scheme for estimating hyperedge effective resistances, which is achieved by extending the optimization-based effective-resistance estimation method in (4) to the hypergraph setting: hyperedge effective resistance can be obtained by searching for an optimal vector χ^* leveraging the following optimization procedure:

$$R_e = \max_{\chi\in\mathbb{R}^{\mathcal{V}}} \frac{(\chi^\top b_{pq})^2}{\sum_{e\in E} w_e \max_{u,v\in e}(\chi_u-\chi_v)^2}, \tag{8}$$

where the original quadratic form $x^\top L_G x$ in (4) is replaced by the nonlinear quadratic form $Q_H(\chi)$ in (2) [5].

B. Key Phases in HyperEF

As illustrated in Figure 1, HyperEF computes a much smaller hypergraph $H'=(V',E',w')$ given the original hypergraph $H=(V,E,w)$ by exploiting hyperedge effective resistances, where $|V'|<|V|$, $|E'|<|E|$, and $|w'|<|w|$. Specifically, HyperEF consists of the following three phases: **Phase (A)** constructs the Krylov subspace to approximate the eigensubspace related to the original hypergraph; **Phase (B)** estimates the effective resistance of each hyperedge by applying the proposed optimization-based method; **Phase (C)** constructs the coarsened hypergraph by aggregating node clusters with low effective-resistance diameters. For the sake of simplicity, key symbols used in this paper are summarized in Table I.

C. Low-Resistance-Diameter Decomposition

Lemma 2. *Let $G=(\mathcal{V},\mathcal{E},z)$ be a weighted undirected graph with weights $z\in\mathbb{R}^{\mathcal{V}}_{>0}$, sufficiently large $\gamma>1$, and the effective-resistance diameter be defined as $\max_{u,v\in\mathcal{V}} R_{eff}(u,v)$. It is always possible to decompose a simple graph G into multiple node clusters $G[\mathcal{V}_i]$ with low effective-resistance diameters by removing only a constant fraction of edges [2]:*

$$\max_{u,v\in\mathcal{V}_i} R_{eff_{G[\mathcal{V}_i]}}(u,v) \lesssim \gamma^3 \frac{|\mathcal{V}|}{z(\mathcal{E})}. \tag{9}$$

Lemma 3. *Let $G=(\mathcal{V},\mathcal{E},z)$ be a weighted undirected graph with weights $z\in\mathbb{R}^{\mathcal{V}}_{>0}$ and $\mathcal{C}_G$ be the conductance of G. The Cheeger's inequality allows obtaining the following relationship between the effective-resistance diameter and the graph conductance [2]:*

$$\max_{u,v\in\mathcal{V}} R_{eff}(u,v) \lesssim \frac{1}{\mathcal{C}_G^2}. \tag{10}$$

The proposed HyperEF algorithm is based on extending the above theorems to hypergraph settings: Lemma 2 implies that we can decompose a hypergraph into multiple (hyperedge) clusters that have small effective-resistance diameters by removing only a few inter-cluster hyperedges, while Lemma 3 implies that contracting the hyperedges (node clusters) with small effective-resistance diameters will not significantly impact the original hypergraph conductance.

D. *Phase (A): Spectral Approximation via Krylov Subspace*

Based on Eq. (2) and the optimization task in Eq. (4), we propose to estimate effective resistance of a hyperedge e by searching for a vector χ^* that maximizes the following ratio:

$$R_e(\chi^*) = \max_{\chi \in \mathbb{R}^{\mathcal{V}}} \frac{(\chi^\top b_{pq})^2}{Q_H(\chi)}, \quad p, q \in e \tag{11}$$

To achieve high efficiency, our search for $\chi*$ will be limited to the eigensubspace spanned by a few Laplacian eigenvectors of the simple graph converted from the original hypergraph. Let $G_b = (\mathcal{V}_b, \mathcal{E}_b, z_b)$ be the bipartite graph corresponding to the hypergraph $H = (V, E, w)$, where $|\mathcal{V}_b| = |V| + |E|$, $|\mathcal{E}_b| = \Sigma_{e \in E}|e|$, and z_b is the scaled edge weights: $z(e, p) = \frac{w(e)}{d(e)}$.

Eq. (3) implies that the approximate effective resistance of each hyperedge can be obtained by finding a few orthogonal eigenvectors that maximize Eq. (11). To avoid high complexity of computing eigenvalues/eigenvectors, we will leverage a scalable algorithm for approximating the eigenvectors by exploiting Krylov subspace defined as follows.

Definition IV.1. *Given a nonsingular matrix $A_{n \times n}$, a vector $\chi \neq 0 \in \mathbb{R}^n$, the order-$(\rho + 1)$ Krylov subspace generated by A from x is*

$$\kappa_\rho(A, x) := span(x, Ax, A^2x, ..., A^\rho x), \tag{12}$$

where x denotes a random vector, and A denotes the normalized adjacency matrix. In our algorithm, A is obtained based on the simple graph converted from the hypergraph using star expansion.

E. *Phase (B): Effective Resistance Estimation*

Assume that $x^{(1)}, x^{(2)}, ..., x^{(\rho)} \in \mathbb{R}^{\mathcal{V}_b}$ are the ρ mutually-orthogonal vectors based on the order-$(\rho+1)$ Krylov subspace $\kappa_\rho(A, x)$ constructed in **Phase(A)**. Our algorithm extends the effective resistance estimation method in (4) by incorporating the non-linear quadratic operator of hypergraphs (2) to include the hypergraph spectral properties. By excluding the node embedding values associated with the star nodes in $x^{(i)}$, we generate a new set of vectors $\chi^{(i)}$ that are all mutually orthogonal. Next, each node in the hypergraph can be embedded into a ρ-dimensional space. Then, for each hyperedge e its resistance ratio (r_e) associated with a vector $\chi^{(i)} \in \chi^{(1)}, ..., \chi^{(\rho)}$ can be computed by

$$r_e(\chi^{(i)}) = \frac{(\chi^{(i)\top} b_{pq})^2}{Q_H(\chi^{(i)})}, \quad p, q \in e \tag{13}$$

where p and q are the two maximally-separated nodes in the ρ-dimensional embedding space. Eq. (11) returns multiple resistance ratios $r_e^{(1)}, ..., r_e^{(\rho)}$ corresponding to $\chi^{(1)}, ..., \chi^{(\rho)}$. After sorting resistance ratios in a descending order, we have

$$r_e^1 > r_e^2 > ... > r_e^\rho. \tag{14}$$

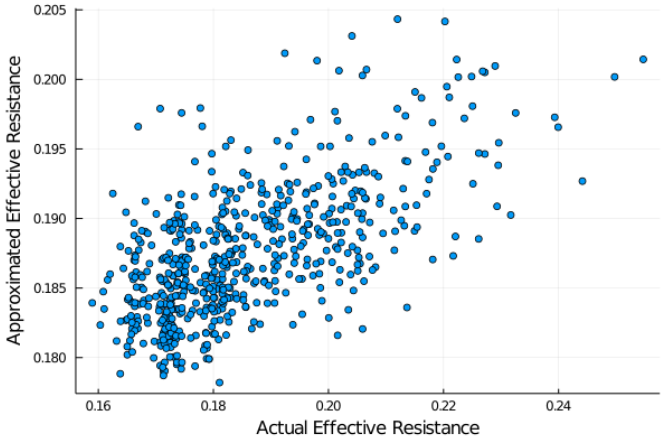

Figure 2: Actual effective resistance vs approximated effective resistance using our method for a simple graph.

In HyperEF, we chooses the top m resistance ratios to estimate the effective resistance of each hyperedge. Specifically, we approximate the hyperedge effective resistance (R_e) by

$$R_e = \sum_{i=1}^{m} r_e^i, \quad e \in E. \tag{15}$$

Note that for each hypergraph, the Krylov subspace vectors in Eq. (12) only need to be computed once, which can be achieved in nearly linear time using only sparse matrix-vector operations. The effective resistance of each hyperedge can be estimated in constant time by identifying a few (m) Krylov subspace vectors that maximize the resistance ratio in Eq. (13).

Since it is not clear how to efficiently compute the accurate hyperedge effective resistances for hypergraphs, we only evaluate the proposed method for a real-world simple graph. As shown in Figure 2, the approximate edge effective resistances obtained using our method can well correlate with the ground truths. Algorithm 1 provides the detailed flow of the proposed hyperedge effective resistance estimation method.

Algorithm 1 The effective resistance estimation algorithm flow

Input: Hypergraph $H = (V, E, w)$, ρ.
Output: A vector of effective resistance R with the size $|E|$.

1: Construct the bipartite graph G_b corresponding to H.
2: Construct the order-$(\rho + 1)$ Krylov subspace.
3: Use Gram–Schmidt method to obtain the orthogonal vectors.
4: For each hyperedge compute its ρ resistance ratios using (13).
5: Obtain all hyperedge effective resistances R based on (15).
6: Return R.

F. *Phase (C): Effective-Resistance Clustering*

Lemma 2 implies that by removing only a few edges it is possible to decompose a simple graph into multiple node clusters with respect to the effective-resistance diameter of each cluster. For hypergraphs, HyperEF adopts the similar idea for identifying node clusters.

Specifically, for a given hypergraph $H = (V, E, w)$, assume that $R \in \mathbb{R}^E_{>0}$ is a vector including all hyperedge effective resistances computed in **Phase(B)**. Then, HyperEF will first exploit the effective resistance vector R to decompose a hypergraph into multiple node clusters with relatively low effective-resistance diameters. Specifically, HyperEF will produce node clusters such that the effective-resistance diameter of each cluster is strictly lower than a given threshold $\delta \in \mathbb{R}_{>0}$. Subsequently, by treating each node cluster as a new node, HyperEF will create a much smaller hypergraph $H' = (V', E', w')$ that can well preserve the spectral (structural) properties of the original hypergraph H. Since the nodes within each cluster are strongly coupled with each other, contracting such clusters (or hyperedges) will not significantly alter the structural (spectral) properties of the original hypergraph.

In our implementation, HyperEF will contract the entire hyperedge e with low effective resistance (below a given threshold δ) if all the nodes within e have not been processed (clustered) before. Otherwise, HyperEF will only contract the unprocessed nodes within the hyperedge.

V. Multilevel Hypergraph Coarsening

In this section, we extend the proposed spectral hypergraph coarsening method to a multilevel coarsening framework that iteratively computes a vector of effective resistance R and contract the hyperedges with low effective resistances ($R_e < \delta$). Our algorithm iteratively contracts the node clusters by merging the nodes within each cluster and replacing that cluster with a new node that we call *supernode* in the next level. We transfer the hypergraph structural information through the levels by assigning a weight (that is equal to the hyperedge effective resistance evaluated at the previous level) to the supernode corresponding to each cluster.

Definition V.1. *Let $H^{(l)} = (V^{(l)}, E^{(l)}, w^{(l)})$ be the hypergraph at the l-th level, $S \in V^{(l)}$ be a cluster of nodes that produces a supernode $\vartheta \in V^{(l+1)}$ at the level $l+1$. We define the vector of node weights to be $\eta^{(l)} \in \mathbb{R}^{V^{(l)}}_{\geq 0}$ and initialize it with all zeros for the original hypergraph. We update η at level l by*

$$\eta_\vartheta := \sum_{j=1}^{|S|} \eta(v_j^{(l)}). \quad v \in V^{(l)} : v \in S \tag{16}$$

Node Weight Propagation (NWP). The vector of effective resistance R will be updated according to the node weights η to pass the clustering information obtained from previous levels to the current level:

$$R_e^{(l)} = \sum_{k=1}^{|e|} \eta(v_k^{(l)}) + R_e^{(l)}. \tag{17}$$

As the result, the hyperedge effective resistance at a coarse level not only depends on the evaluated effective resistance ($R_e^{(l)}$) computed at the current level but also the results transferred from all the previous (finer) levels.

As observed in our extensive experiment results, using (17) for estimating effective resistances leads to more balanced hypergraph clustering results when compared with the implementation that ignores the previous clustering information. The complete HyperEF algorithm flow has been shown in Algorithm 2 for coarsening a given hypergraph H within L levels.

Algorithm 2 HyperEF algorithm flow

Input: Hypergraph $H = (V, E, w)$, δ, L, η.
Output: A coarsened hypergraph $H' = (V', E', w')$ that $|V'| \ll |V|$.

```
Initialize H' ← H
for l ← 1 to L do
    Call Algorithm 1 to compute a vector of effective resistance
    R with the size |E'| for given hypergraph H'.
    Compute the node weights using (16).
    Update the effective resistance vector R by applying (17).
    Sort the hyperedges with ascending R values.
    Starting with the hyperedges that have the lowest effective
    resistances, contract (cluster) the hyperedge (nodes) if R_e < δ.
    Construct a coarsened hypergraph H' accordingly.
end for
Return H'.
```

VI. Complexity Analysis

The algorithm complexity of **Phase(A)** for constructing the Krylov subspace for the bipartite graph $G_b = (\mathcal{V}_b, \mathcal{E}_b, z_b)$ corresponding to the original hypergraph $H = (V, E, w)$ is $O(|\mathcal{E}_b|)$; the complexity of the hyperedge effective resistance estimation and hyperedge clustering corresponding to **Phase(B)** and **Phase(C)** is $O(\rho|E|)$; the complexity of computing the node weights through the multilevel framework is $O(|E|)$ that leads to the overall nearly-linear algorithm complexity of $O(\rho|E| + |\mathcal{E}_b|)$.

VII. Experimental Results

This section presents the results of a variety of experiments for evaluating the performance and efficiency of the proposed spectral hypergraph coarsening algorithm (HyperEF). The real-world VLSI design benchmarks "ibm01", "ibm02", ..., "ibm18" with $13,000$ to $210,000$ cells have been adopted [1]. All experiments have been evaluated on a laptop with 8 GB of RAM and a 2.2 GHz Quad-Core Intel Core i7 processor.

A. Experiment Setup

The HyperEF algorithm has been implemented as follows: (1) construct the bipartite graph G_b corresponding to the hypergraph H (2) generate a random vector x orthogonal to the all-one vector $\mathbf{1}$ satisfying $1^\top x = 0$; (3) construct a ρ-order Krylov subspace using x where $\rho = 200$; (4) select 10 vectors from the Krylov subspace for embedding each node into a 10-dimensional space; (5) compute 10 hyperedge resistance ratios; (6) the estimated effective resistance of each hyperedge is based on the top resistance ratio (m = 1); (7) δ is set to be the maximum hyperedge effective

[1] https://vlsicad.ucsd.edu/UCLAWeb/cheese/ispd98.html

Table II: Impact of the node weight propagation (NWP) technique on estimating effective resistances.

	level 1			level 2	
	R_1	R_2	R_3	R_1	R_2
without NWP	2.5	3.3	1.8	3.1	2.8
with NWP	2.6	4.2	1.96	5.1	6.1

resistance at each level. The input arguments of hMetis are set as default values and UBfactor $= 5$ through all the experiments. An implementation of our algorithm and the code for reproducing our experimental results are available online at https://github.com/Feng-Research/HyperEF.

B. Node Weight Propagation (NWP)

The proposed node weight propagation technique preserves the spectral features by passing the effective resistance values along the levels. It allows us to exploit the previous cluster information for computing hyperedge effective resistances of the current level. To illustrate the benefit of NWP, we conduct an experiment for a small hypergraph H illustrated in Figure 1, to compare the estimated effective resistances with and without employing the NWP scheme. We apply HyperEF for $L = 2$ levels and contract one hyperedge at each level to create a coarsened hypergraph with 6 nodes and 2 hyperedges. Table II summarizes the estimated effective resistance of every hyperedge with and without using the NWP method at different levels. We observe that the employing NWP technique leads to more accurate estimation results in both levels.

C. Spectral Hypergraph Decomposition

In this section, we provide comprehensive experimental results to evaluate the performance of the proposed spectral hypergraph coarsening algorithm by comparing HyperEF with both spectral and non-spectral coarsening methods. We leverage HyperEF to decompose the hypergraph into multiple clusters by repeatedly clustering the nodes and aggregating the vertices within each cluster. The Cheeger's inequality implies that the quality of a cluster S is higher if the conductance of S is smaller. Accordingly, we utilize the following average conductance of clusters as a measure to analyze the performance of each method

$$\mathcal{C} = \frac{1}{N}\sum_{i=1}^{N}\mathcal{C}(S_i), \tag{18}$$

where N is the number of the clusters and $\mathcal{C}(S_i)$ is the conductance of a node cluster S_i. Additionally, our algorithm can accept a set of node weights proportional to the size of the cells that is beneficial for VLSI applications. In this case, our objective denominator in II.1 will be modified by adding the node weights to the node degrees.

We compare HyperEF with the SoTA hypergraph partitioner, hMetis, in terms of performance and runtime. HyperEf is compared with the hMetis partitioning results by considering the hypergraph conductance metric (II.1). It should be noted that a direct comparison of our proposed coarsening algorithm with the coarsening method used in hMetis is impossible since the code is unavailable.

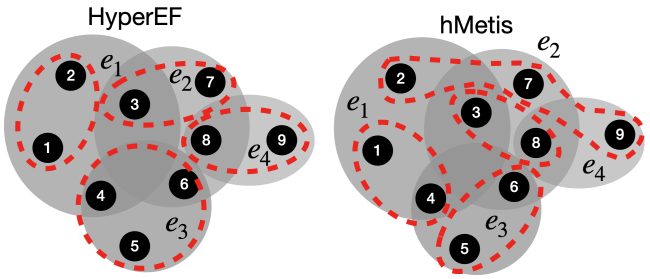

Figure 3: Hypergraph clustering results: HyperEF vs hMetis.

Figure 3 demonstrates the node clustering results obtained using HyperEF and hMetis. Both methods partition the hypergraph into 4 clusters, and the average conductance of clusters has been computed to evaluate the performance of each method. HyperEF computes the effective resistance of hyperedges, and sort them accordingly: $R_{e_4} < R_{e_3} < R_{e_2} < R_{e_1}$. In this example, e_4 has the smallest effective resistance resulting in clustering nodes 8 and 9. The next hyperedge with the smallest effective resistance is e_3, forming another cluster including nodes 4, 5, and 6. Then, e_2 has been discovered, and nodes 3 and 7 are clustered together since nodes 6 and 8 have already been processed. Finally, HyperEF explores e_1 and produces another node cluster containing nodes 1 and 2, since nodes 3 and 4 have been previously clustered. The results show that HyperEF outperforms hMetis by creating clusters with a significantly lower average conductance.

Table III, Table IV, Table V, and Table VI show the average conductance of clusters $\mathcal{C}$ computed with both HyperEF and hMetis by decomposing the hypergraph into the same number of node clusters with the same node reduction ratios (NRs) for "ibm01", ..., "ibm18" datasets. HyperEF achieves NR = 52%[1] with L=1, NR = 75% with L=2, NR = 87% with L=3, and NR = 94% with L=4. All the NRs are chosen unbiased to evaluate the performance of HyperEF for different coarsening ratios. Our extensive results show that HyperEF always produces better results (lower average conductance) for all the test cases with NRs from 52% to 87%, and similar results with NR = 94%. We also report the runtime $\mathcal{T}$ (seconds) of both methods for decomposing the hypergraphs. Figure 4 shows that HyperEF achieves up to $72\times$ speedup over hMetis.

To further investigate the performance of HyperEF, we designed various experiments to decompose the hypergraph using spectral and non-spectral methods. For spectral hypergraph coarsening benchmarks, traditional simple graph spectral coarsening techniques are leveraged to decompose hypergraphs [40] after converting them into simple graphs using star and clique expansions. Additionally, we compare HyperEF with HyperSF, which is a recently developed spectral hypergraph

[1]If an original hypergraph has 100 nodes, the coarsened hypergraph will have 48 nodes.

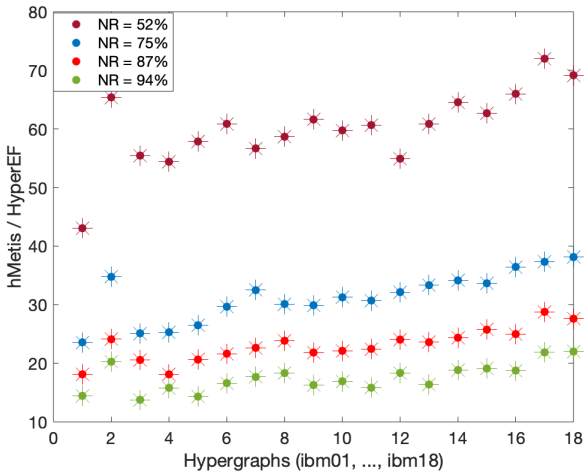

Figure 4: Comparison of runtime performance.

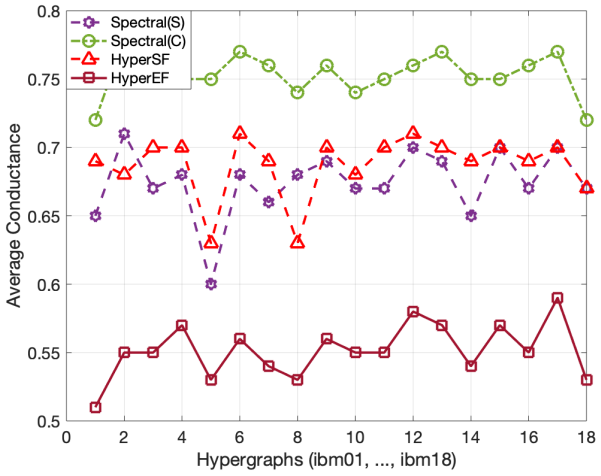

Figure 5: HyperEF vs spectral methods.

coarsening method [1]. Note that in [1], the average local conductance values are reported, whereas, in our experiments, the average global conductance of clusters is reported. For all the above experiments, we decompose hypergraphs into the same number of node clusters with NR = 83%. Figure 5 shows that HyperEF beats all other spectral coarsening methods by achieving the lowest conductance for all the test cases. For non-spectral methods, we use Metis to partition the simple graph corresponding to the hypergraph (achieved by applying star and clique expansions). In Figure 6, we compare the performance of Metis (star and clique expansions), hMetis and HyperEF. We observe that HyperEF outperforms all non-spectral methods by returning the lowest average conductance when NR = 83%.

D. Cut (Conductance) Preservation in Coarsened Hypergraphs

To further evaluate the performance of HyperEF, we incorporate HyperEF with hMetis to partition the hypergraphs and compare the cut and conductance values before and

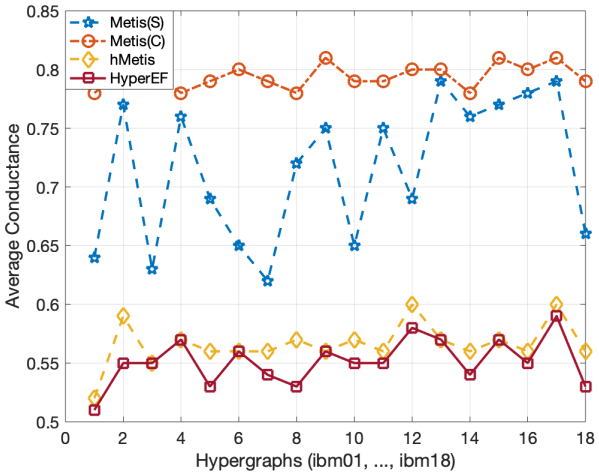

Figure 6: HyperEF vs non-spectral methods.

Table III: NR = 52%, and L=1 (HyperEF only).

ibm	$N_{cluster}$	$\mathcal{C}$(HyperEF)	$\mathcal{C}$(hMetis)	$\mathcal{T}$(HyperEF)	$\mathcal{T}$(hMetis)
1	6,183	**0.75**	0.81	**0.72**	31 (**43**×)
2	8,746	**0.74**	0.8	**0.78**	51 (**65**×)
3	10,755	**0.76**	0.8	**1.01**	56 (**55**×)
4	12,713	**0.76**	0.81	**1.25**	68 (**54**×)
5	12,334	**0.69**	0.75	**1.21**	70 (**58**×)
6	14,935	**0.77**	0.81	**1.38**	84 (**61**×)
7	21,727	**0.77**	0.81	**2.17**	123 (**57**×)
8	23,285	**0.75**	0.81	**2.25**	132 (**59**×)
9	23,888	**0.76**	0.81	**2.24**	138 (**62**×)
10	32,427	**0.76**	0.8	**3.23**	193 (**60**×)
11	31,443	**0.77**	0.81	**3.05**	185 (**61**×)
12	32,530	**0.78**	0.82	**3.66**	201 (**55**×)
13	39,473	**0.78**	0.82	**3.96**	241 (**61**×)
14	68,607	**0.75**	0.8	**6.32**	408 (**65**×)
15	72192	**0.78**	0.82	**8.01**	502 (**63**×)
16	81,431	**0.76**	0.81	**8.44**	557 (**66**×)
17	81,789	**0.78**	0.83	**8.46**	609 (**72**×)
18	93,000	**0.74**	0.81	**9.34**	646 (**69**×)

Table IV: NR = 75%, and L=2 (HyperEF only).

ibm	$N_{cluster}$	$\mathcal{C}$(HyperEF)	$\mathcal{C}$(hMetis)	$\mathcal{T}$(HyperEF)	$\mathcal{T}$(hMetis)
1	3,160	**0.62**	0.65	**1.23**	29 (**24**×)
2	4,314	**0.62**	0.67	**1.41**	49 (**35**×)
3	5,329	**0.63**	0.66	**2.11**	53 (**25**×)
4	6,300	**0.64**	0.66	**2.37**	60 (**25**×)
5	6,140	**0.59**	0.63	**2.34**	62 (**26**×)
6	7,353	**0.64**	0.66	**2.63**	78 (**30**×)
7	10,900	**0.63**	0.67	**3.54**	115 (**32**×)
8	11,800	**0.61**	0.67	**4.15**	125 (**30**×)
9	11,362	**0.64**	0.66	**4.38**	131 (**30**×)
10	16,052	**0.63**	0.67	**5.79**	181 (**31**×)
11	16,070	**0.64**	0.67	**5.73**	176 (**31**×)
12	16,207	**0.65**	0.7	**5.94**	191 (**32**×)
13	19,617	**0.65**	0.68	**6.87**	229 (**33**×)
14	34,136	**0.62**	0.66	**11.51**	393 (**34**×)
15	35,524	**0.66**	0.69	**14.44**	486 (**34**×)
16	39,645	**0.63**	0.67	**14.62**	533 (**36**×)
17	39,951	**0.66**	0.7	**15.22**	568 (**37**×)
18	47,702	**0.6**	0.67	**15.79**	602 (**38**×)

Table V: NR = 87%, and L=3 (HyperEF only).

ibm	$N_{cluster}$	$\mathcal{C}$(HyperEF)	$\mathcal{C}$(hMetis)	$\mathcal{T}$(HyperEF)	$\mathcal{T}$(hMetis)
1	1,642	**0.51**	0.52	**1.49**	27 (**18**×)
2	2,342	**0.55**	0.59	**1.95**	47 (**24**×)
3	2,712	**0.55**	**0.55**	**2.43**	50 (**21**×)
4	3,311	**0.57**	**0.57**	**2.98**	54 (**18**×)
5	3,128	**0.53**	0.56	**2.81**	58 (**21**×)
6	3,713	**0.56**	**0.56**	**3.42**	74 (**22**×)
7	5,501	**0.54**	0.56	**4.68**	106 (**23**×)
8	6,369	**0.53**	0.57	**5.03**	120 (**24**×)
9	5,798	**0.56**	**0.56**	**5.63**	123 (**22**×)
10	8,400	**0.55**	0.57	**7.23**	160 (**22**×)
11	8,371	**0.55**	0.56	**7.39**	166 (**22**×)
12	8,458	**0.58**	0.6	**7.57**	182 (**24**×)
13	10,026	**0.57**	**0.57**	**9.15**	216 (**24**×)
14	17,509	**0.54**	0.56	**14.77**	360 (**24**×)
15	18,070	**0.57**	**0.57**	**17.79**	458 (**26**×)
16	19,992	**0.55**	0.56	**19.61**	490 (**25**×)
17	20,551	**0.59**	0.6	**19.46**	560 (**29**×)
18	25,315	**0.53**	0.56	**21.04**	581 (**28**×)

Table VI: NR = 94%, and L=4 (HyperEF only).

ibm	$N_{cluster}$	$\mathcal{C}$(HyperEF)	$\mathcal{C}$(hMetis)	$\mathcal{T}$(HyperEF)	$\mathcal{T}$(hMetis)
1	862	0.45	**0.41**	**1.73**	25 (**14**×)
2	1,350	0.52	**0.53**	**2.17**	44 (**20**×)
3	1,395	0.49	**0.45**	**3.34**	46 (**14**×)
4	1,735	0.52	**0.46**	**3.35**	53 (**16**×)
5	1,619	**0.49**	**0.49**	**3.77**	54 (**14**×)
6	1,888	0.5	**0.46**	**3.97**	66 (**17**×)
7	2,836	0.48	**0.46**	**5.49**	97 (**18**×)
8	3,574	**0.49**	**0.49**	**5.89**	108 (**18**×)
9	3,017	0.49	**0.45**	**6.87**	112 (**16**×)
10	4,481	0.49	**0.48**	**9.45**	160 (**17**×)
11	4,391	0.5	**0.46**	**9.79**	155 (**16**×)
12	4,528	0.54	**0.52**	**9.04**	166 (**18**×)
13	5,174	0.52	**0.47**	**12.39**	203 (**16**×)
14	9,055	0.48	**0.47**	**17.93**	338 (**19**×)
15	9,277	0.51	**0.47**	**22.03**	421 (**19**×)
16	10,308	0.49	**0.47**	**23.07**	433 (**19**×)
17	10,782	0.54	**0.51**	**23.69**	518 (**22**×)
18	13,864	**0.48**	**0.48**	**24.72**	544 (**22**×)

after spectral hypergraph coarsening. Initially, we coarsen the hypergraph using HyperEF to generate a smaller hypergraph and then use hMetis to bipartite the coarsened hypergraph. This experiment compares the following two results: **Part 1:** bisect the original hypergraph using hMetis and compute the cut and conductance; **Part 2:** coarsen the original hypergraph using HyperEF, and subsequently use hMetis to bisect the coarsened hypergraph. For **Part 2**, all the node partitions associated with the coarsened hypergraphs are directly mapped to the original hypergraph nodes before the cut and conductance values are computed. Table VII shows that the coarsened hypergraphs can reasonably retain the key spectral properties of the original hypergraph by well preserving the cut and conductance values. In Table VII, H is the original hypergraph (**Part 1**), and H' is the coarsened hypergraph (**Part 2**). NR and ER are the node reduction ratios and hyperedge reduction ratios, respectively. These results imply that HyperEF only clusters the strongly-coupled nodes and will not significantly impact the cut and conductance after coarsening, leaving the global hypergraph structure intact.

Table VII: The results of HyperEF for preserving the spectral hypergraph properties.

ibm	NR(%)	ER(%)	cut(H)	cut(H')	$\mathcal{C}$(H)	$\mathcal{C}$(H')
01	31	25	180	185	0.0082	0.0085
02	30	28	264	278	0.0069	0.0073
03	31	28	954	996	0.022	0.022
04	35	25	537	544	0.0108	0.0111
05	32	29	1708	1700	0.0293	0.03
06	34	28	899	906	0.0158	0.0162
07	31	27	859	887	0.0109	0.0113
08	29	28	1147	1165	0.0116	0.0121
09	28	23	633	678	0.0057	0.0061
10	34	28	1286	1379	0.01	0.0094
11	31	23	962	1070	0.0073	0.0079
12	31	26	1893	1975	0.0132	0.0139
13	32	24	840	921	0.0049	0.0052
14	28	26	1858	1959	0.0072	0.0076
15	26	21	2670	2869	0.0085	0.0086
16	31	27	1746	1865	0.0049	0.0052
17	28	25	2222	2380	0.0055	0.0059
18	27	26	1881	1947	0.0048	0.0049

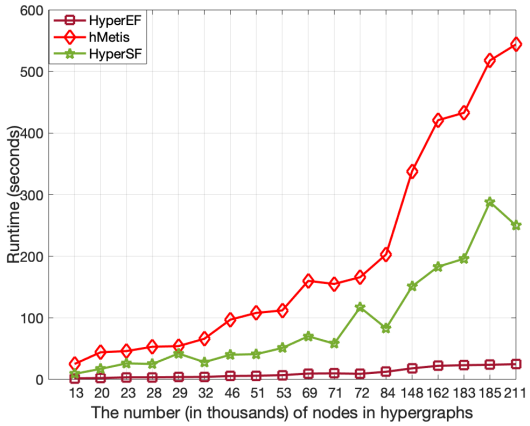

Figure 7: Runtime scalability comparisons.

E. Runtime Scalability

Figure 7 shows that HyperEF has much better runtime scalability when compared with HyperSF and hMetis, achieving about 70× speedups over hMetis and 20× speedups over HyperSF.

VIII. Conclusion

This work presents a highly-efficient spectral hypergraph coarsening algorithm (HyperEF) that allows creating a much smaller hypergraph while preserving the key spectral (structural) properties of the original hypergraph. HyperEF is built upon a novel effective resistance estimation method for decomposing a hypergraph into multiple strongly-coupled node clusters. To achieve more effective decomposition, a node weight propagation scheme is introduced to allow extending HyperEF to a multilevel hypergraph coarsening framework. When compared to state-of-the-art methods, our extensive experiment results on real-world VLSI test cases show that HyperEF can significantly improve the hypergraph clustering (partitioning) quality while achieving up to 70× runtime speedups.

IX. Acknowledgments

This work is supported in part by the National Science Foundation under Grants CCF-2021309, CCF-2011412, CCF-2212370, and CCF-2205572.

References

[1] A. Aghdaei, Z. Zhao, and Z. Feng. Hypersf: Spectral hypergraph coarsening via flow-based local clustering. *CoRR*, abs/2108.07901, 2021.

[2] V. L. Alev, N. Anari, L. C. Lau, and S. Oveis Gharan. Graph clustering using effective resistance. In *9th Innovations in Theoretical Computer Science Conference (ITCS 2018)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2018.

[3] Ü. V. Çatalyürek and C. Aykanat. Patoh (partitioning tool for hypergraphs). In *Encyclopedia of Parallel Computing*, pages 1479–1487. Springer, 2011.

[4] T.-H. H. Chan and Z. Liang. Generalizing the hypergraph laplacian via a diffusion process with mediators. *Theoretical Computer Science*, 806:416–428, 2020.

[5] T.-H. H. Chan, A. Louis, Z. G. Tang, and C. Zhang. Spectral properties of hypergraph laplacian and approximation algorithms. *Journal of the ACM (JACM)*, 65(3):15, 2018.

[6] J. Chen, Y. Saad, and Z. Zhang. Graph coarsening: from scientific computing to machine learning. *SeMA Journal*, 79(1):187–223, 2022.

[7] C. Deng, Z. Zhao, Y. Wang, Z. Zhang, and Z. Feng. GraphZoom: A Multi-level Spectral Approach for Accurate and Scalable Graph Embedding. In *International Conference on Learning Representations*, 2019.

[8] K. D. Devine, E. G. Boman, R. T. Heaphy, R. H. Bisseling, and U. V. Catalyurek. Parallel hypergraph partitioning for scientific computing. In *Proceedings 20th IEEE International Parallel & Distributed Processing Symposium*, pages 10–pp. IEEE, 2006.

[9] Z. Feng. Similarity-aware spectral sparsification by edge filtering. In *Design Automation Conference (DAC)*. IEEE, 2018.

[10] Z. Feng. Grass: Graph spectral sparsification leveraging scalable spectral perturbation analysis. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 39(12):4944–4957, 2020.

[11] A. Grover and J. Leskovec. node2vec: Scalable Feature Learning for Networks. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 855–864. ACM, 2016.

[12] L. Hagen and A. Kahng. New spectral methods for ratio cut partitioning and clustering. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 11(9):1074–1085, 1992.

[13] V. Iakovlev, M. Heinonen, and H. Lähdesmäki. Learning continuous-time pdes from sparse data with graph neural networks. In *International Conference on Learning Representations*, 2020.

[14] M. Kapralov, R. Krauthgamer, J. Tardos, and Y. Yoshida. Towards tight bounds for spectral sparsification of hypergraphs. In *Proceedings of the 53rd Annual ACM SIGACT Symposium on Theory of Computing*, pages 598–611, 2021.

[15] M. Kapralov, R. Krauthgamer, J. Tardos, and Y. Yoshida. Spectral hypergraph sparsifiers of nearly linear size. In *2021 IEEE 62nd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 1159–1170. IEEE, 2022.

[16] G. Karypis, R. Aggarwal, V. Kumar, and S. Shekhar. Multilevel hypergraph partitioning: Applications in vlsi domain. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 7(1):69–79, 1999.

[17] K. Kunal, T. Dhar, M. Madhusudan, J. Poojary, A. Sharma, W. Xu, S. M. Burns, J. Hu, R. Harjani, and S. S. Sapatnekar. Gana: Graph convolutional network based automated netlist annotation for analog circuits. In *2020 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pages 55–60. IEEE, 2020.

[18] J. R. Lee, S. O. Gharan, and L. Trevisan. Multiway spectral partitioning and higher-order cheeger inequalities. *Journal of the ACM (JACM)*, 61(6):1–30, 2014.

[19] Y. T. Lee and H. Sun. An SDP-based Algorithm for Linear-sized Spectral Sparsification. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing*, STOC 2017, pages 678–687, New York, NY, USA, 2017. ACM.

[20] Z. Li, N. Kovachki, K. Azizzadenesheli, B. Liu, A. Stuart, K. Bhattacharya, and A. Anandkumar. Multipole graph neural operator for parametric partial differential equations. *Advances in Neural Information Processing Systems*, 33, 2020.

[21] J. Lim, S. Ryu, K. Park, Y. J. Choe, J. Ham, and W. Y. Kim. Predicting drug–target interaction using a novel graph neural network with 3d structure-embedded graph representation. *Journal of chemical information and modeling*, 59(9):3981–3988, 2019.

[22] A. Loukas and P. Vandergheynst. Spectrally approximating large graphs with smaller graphs. In *International Conference on Machine Learning*, pages 3237–3246. PMLR, 2018.

[23] A. Mirhoseini, A. Goldie, M. Yazgan, J. W. Jiang, E. Songhori, S. Wang, Y.-J. Lee, E. Johnson, O. Pathak, and A. Nazi. A graph placement methodology for fast chip design. *Nature*, 594(7862):207–212, 2021.

[24] X. Ouvrard. Hypergraphs: an introduction and review. *arXiv preprint arXiv:2002.05014*, 2020.

[25] B. Perozzi, R. Al-Rfou, and S. Skiena. Deepwalk: Online Learning of Social Representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 701–710. ACM, 2014.

[26] P. C. Rathi, R. F. Ludlow, and M. L. Verdonk. Practical high-quality electrostatic potential surfaces for drug discovery using a graph-convolutional deep neural network. *Journal of Medicinal Chemistry*, 2019.

[27] I. Safro, P. Sanders, and C. Schulz. Advanced coarsening schemes for graph partitioning. *Journal of Experimental Algorithmics (JEA)*, 19:1–24, 2015.

[28] M. Schlichtkrull, T. N. Kipf, P. Bloem, R. Van Den Berg, I. Titov, and M. Welling. Modeling relational data with graph convolutional networks. In *European Semantic Web Conference*, pages 593–607. Springer, 2018.

[29] R. Shaydulin, J. Chen, and I. Safro. Relaxation-based coarsening for multilevel hypergraph partitioning. *Multiscale Modeling and Simulation*, 17(1):482Ð506, Jan 2019.

[30] T. Soma and Y. Yoshida. Spectral sparsification of hypergraphs. In *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 2570–2581. SIAM, 2019.

[31] D. Spielman and N. Srivastava. Graph Sparsification by Effective Resistances. *SIAM Journal on Computing*, 40(6):1913–1926, 2011.

[32] D. Spielman and S. Teng. Spectral partitioning works: Planar graphs and finite element meshes. In *Foundations of Computer Science (FOCS), 1996. Proceedings., 37th Annual Symposium on*, pages 96–105. IEEE, 1996.

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

[34] B. Vastenhouw and R. H. Bisseling. A two-dimensional data distribution method for parallel sparse matrix-vector multiplication. *SIAM review*, 47(1):67–95, 2005.

[35] H. Wang, K. Wang, J. Yang, L. Shen, N. Sun, H.-S. Lee, and S. Han. Gcn-rl circuit designer: Transferable transistor sizing with graph neural networks and reinforcement learning. In *2020 57th ACM/IEEE Design Automation Conference (DAC)*, pages 1–6. IEEE, 2020.

[36] G. Zhang, H. He, and D. Katabi. Circuit-gnn: Graph neural networks for distributed circuit design. In *International Conference on Machine Learning*, pages 7364–7373. PMLR, 2019.

[37] M. Zhang and Y. Chen. Link prediction based on graph neural networks. In *Advances in Neural Information Processing Systems*, pages 5165–5175, 2018.

[38] Y. Zhang, Z. Zhao, and Z. Feng. Sf-grass: Solver-free graph spectral sparsification. In *2020 IEEE/ACM International Conference On Computer Aided Design (ICCAD)*, pages 1–8. IEEE, 2020.

[39] Z. Zhao and Z. Feng. Effective-resistance preserving spectral reduction of graphs. In *Proceedings of the 56th Annual Design Automation Conference 2019*, DAC '19, pages 109:1–109:6, New York, NY, USA, 2019. ACM.

[40] Z. Zhao, Y. Zhang, and Z. Feng. Towards scalable spectral embedding and data visualization via spectral coarsening. In *Proceedings of the 14th ACM International Conference on Web Search and Data Mining*, pages 869–877, 2021.

[41] D. Zhou, J. Huang, and B. Schölkopf. Learning with hypergraphs: Clustering, classification, and embedding. *Advances in neural information processing systems*, 19:1601–1608, 2006.