This article has been accepted for publication in IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems. This is the author's version which has not been fully edit
content may change prior to final publication. Citation information: DOI 10.1109/TCAD.2025.3586885

SHyPar: A Spectral Coarsening Approach to Hypergraph Partitioning

Hamed Sajadinia⋆, Ali Aghdaei⋆, Zhuo Feng, *Senior Member, IEEE,*

***Abstract*—State-of-the-art hypergraph partitioners utilize a multilevel paradigm to construct progressively coarser hypergraphs across multiple layers, guiding cut refinements at each level of the hierarchy. Traditionally, these partitioners employ heuristic methods for coarsening and do not consider the structural features of hypergraphs. In this work, we introduce a multilevel spectral framework, SHyPar, for partitioning large-scale hypergraphs by leveraging hyperedge effective resistances and flow-based community detection techniques. Inspired by the latest theoretical spectral clustering frameworks, such as HyperEF and HyperSF, SHyPar aims to decompose large hypergraphs into multiple subgraphs with few inter-partition hyperedges (cut size). A key component of SHyPar is a flow-based local clustering scheme for hypergraph coarsening, which incorporates a max-flow-based algorithm to produce clusters with substantially improved conductance. Additionally, SHyPar utilizes an effective resistance-based rating function for merging nodes that are strongly connected (coupled). Compared with existing state-of-the-art hypergraph partitioning methods, our extensive experimental results on real-world VLSI designs demonstrate that SHyPar can more effectively partition hypergraphs, achieving state-of-the-art solution quality.**

***Index Terms*—Hypergraph Partitioning, Effective Resistance, Flow-Based Clustering, Spectral Coarsening.**

I. Introduction

HYPERGRAPHS are more general than simple graphs since they allow modeling higher-order relationships among the entities. Hypergraph partitioning is an increasingly important problem with applications in various areas, including parallel sparse matrix computations [1], computer-aided design (CAD) of integrated circuit systems [2], physical mapping of chromosomes [3], protein-to-protein interactions [4], as well as data mining and machine learning [5].

The problem of hypergraph partitioning involves grouping the nodes of a hypergraph into multiple clusters. This is done by minimizing a given cost function, defined over the hypergraph, under specific constraints. For instance, the cost function can be the hypergraph cut, which is the number (or total weight) of hyperedges that span more than one partition. The constraints may include balance constraints, where the difference in total node (or hyperedge) weight of each cluster should not exceed a given threshold. Due to these balance constraints, the problem of optimally partitioning a hypergraph is classified as NP-hard [6].

⋆ These authors contributed equally to this work.

The authors are with the Electrical and Computer Engineering Department, Stevens Institute of Technology, 1 Castle Point Terrace, Hoboken, NJ 07030, USA (e-mail: hsajadin@stevens.edu; aaghdaei@ucsd.edu; zfeng12@stevens.edu)

State-of-the-art multilevel hypergraph partitioning techniques predominantly rely on relatively simple heuristics for edge coarsening, such as vertex similarity or hyperedge similarity [2], [7]–[10]. For example, hyperedge similarity-based coarsening techniques contract similar large-size hyperedges into smaller ones. While this approach is straightforward to implement, it may adversely affect the original structural properties of the hypergraph. On the other hand, vertex-similarity-based algorithms determine strongly coupled (correlated) node clusters by measuring distances between vertices. This can be facilitated by hypergraph embedding, which maps each vertex to a low-dimensional vector, allowing for the Euclidean distance (coupling) between vertices to be easily computed in constant time. However, these simple metrics often fail to capture the higher-order global (structural) relationships within hypergraphs, potentially leading to suboptimal partitioning solutions.

Spectral methods are increasingly important in various graph and numerical applications [11]. These applications include scientific computing and numerical optimization [12]–[14], graph partitioning and data clustering [15]–[17], data mining and machine learning [18], [19], graph visualization and data analytics [20]–[22], graph signal processing and image segmentation [23]–[25], and integrated circuit (IC) modeling and verification [26]–[31]. Recent theoretical breakthroughs in spectral graph theory have led to the development of nearly-linear time spectral graph sparsification (edge reduction) [31]–[37] and coarsening (node reduction) algorithms [38], [39].

On the other hand, spectral theory for hypergraphs has been less developed due to the more complicated structure of hypergraphs. For example, a mathematically rigorous approach has introduced a nonlinear diffusion process to define the hypergraph Laplacian operator, which measures the flow distribution within each hyperedge [40], [41]; Additionally, the Cheeger's inequality has been validated for hypergraphs under this diffusion-based nonlinear Laplacian operator [40]. However, these theoretical results do not readily translate into practical implementations. Even recent breakthroughs, such as the SpecPart and K-SpecPart algorithms for spectral hypergraph partitioning [42], [43], still depend heavily on the initial solutions provided by existing multilevel hypergraph partitioning methods, which themselves are based entirely on simplistic edge coarsening heuristics [44], [45].

This paper introduces a brand-new multilevel hypergraph partitioning framework that leverages the latest highly scalable spectral hypergraph coarsening algorithms [46], [47]. Unlike traditional methods that depend solely on simple hyperedge

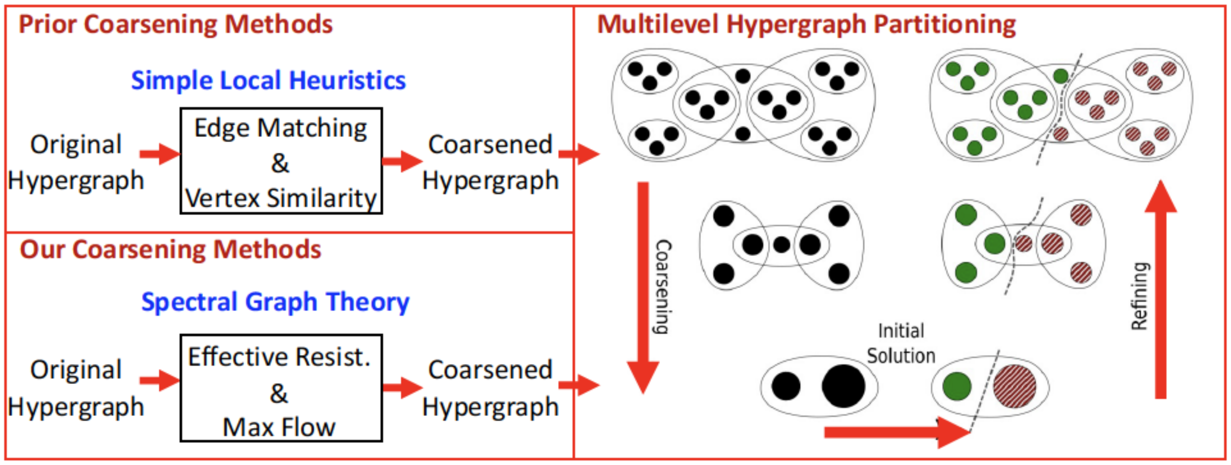


Fig. 1: The proposed multilevel hypergraph partitioning via spectral coarsening.

(node) contraction heuristics and focus on local hypergraph structures, our framework, for the first time, incorporates spectral (global) properties into the multilevel coarsening and partitioning tasks as depicted in Figure 1. To achieve these goals, the paper is organized in a structured sequence of steps:

a) Scalable Spectral Hypergraph Coarsening Algorithms: This paper presents a two-phase scalable algorithmic framework for the spectral coarsening of large-scale hypergraphs, which exploits hyperedge effective resistances and strongly local flow-based methods [46], [47]. The proposed methods facilitate the decomposition of hypergraphs into multiple strongly-connected node clusters with minimal inter-cluster hyperedges, by incorporating the latest diffusion-based nonlinear quadratic operators defined on hypergraphs.

b) Multilevel Hypergraph Partitioning via Spectral Coarsening: This paper develops a brand-new multilevel hypergraph partitioning tool by seamlessly integrating the proposed spectral coarsening methods into the hypergraph partitioning platform. By replacing the traditional simple coarsening heuristics with our theoretically rigorous spectral methods, the multilevel hypergraph partitioning tools developed through this research will potentially offer significantly improved partitioning solutions without compromising runtime efficiency.

c) Validations of Multilevel Hypergraph Partitioning Tools: This paper comprehensively validates the developed hypergraph partitioning tools, focusing on an increasingly important application: integrated circuit (IC) computer-aided design. Both the solution quality and runtime efficiency will be carefully assessed by testing the tools on a wide range of public-domain data sets. Additionally, the developed open-source software packages will be made available for public assessment.

The structure of this paper is organized as follows: Section II provides a foundational overview of the essential concepts and preliminaries in spectral hypergraph theory. Section III introduces the proposed method for hypergraph partitioning via spectral clustering, including resistance-based hypergraph clustering, flow-based clustering, and multilevel hypergraph partitioning. Section IV applies our framework to an established hypergraph partitioning tool, showcasing extensive experimental results on various real-world VLSI design benchmarks. The paper concludes with Section V, which summarizes the findings and implications of this work.

II. Preliminaries and Background

A. Spectral (Hyper)graph Theory

1) Graph Laplacian matrix: In an undirected graph $G = (\mathcal{V}, \mathcal{E}, z)$, the symbol $\mathcal{V}$ represents a set of nodes (vertices), $\mathcal{E}$ represents a set of undirected edges, and z indicates the weights associated with these edges. We denote D as a diagonal matrix where each diagonal element $D(i,i)$ corresponds to the weighted degree of node i. Additionally, A is defined as the adjacency matrix for the undirected graph G, as described below:

$$A(i,j) = \begin{cases} z(i,j) & \text{if } (i,j) \in \mathcal{E} \\ 0 & \text{otherwise} . \end{cases} \tag{1}$$

Subsequently, the Laplacian matrix of the graph G is determined by the formula $L = D - A$. This matrix adheres to several key properties: (1) the sum of the elements in each row or column is zero; (2) all elements outside the main diagonal are non-positive; (3) the graph Laplacian is symmetric and diagonally dominant (SDD), characterized by non-negative eigenvalues.

2) Courant-Fischer Minimax Theorem: The k-th largest eigenvalue of the Laplacian matrix $L \in \mathbb{R}^{|\mathcal{V}| \times |\mathcal{V}|}$ over the subspace U of $\mathbb{R}^{\mathcal{V}}$, can be computed as follows:

$$\lambda_k(L) = \min_{dim(U)=k} \max_{\substack{x \in U \\ x \neq 0}} \frac{x^\top L x}{x^\top x}, \tag{2}$$

This can be leveraged for compute the spectrum of the Laplacian matrix L.

3) Graph conductance: In a graph $G = (\mathcal{V}, \mathcal{E}, z)$ where vertices are partitioned into subsets $(S, \hat{S})$, the conductance of partition S is defined as:

$$\Phi_G(S) := \frac{w(S,\hat{S})}{\min\Big(vol(S), vol(\hat{S})\Big)} = \frac{\sum_{(i,j)\in\mathcal{E}: i\in S, j\notin S} z(i,j)}{\min\Big(vol(S), vol(\hat{S})\Big)}, \tag{3}$$

where the volume of the partition, $vol(S)$, is the sum of the weighted degrees of vertices in S, defined as $vol(S) := \sum_{i\in S} d(i)$. The graph's conductance [48] is defined as:

$$\Phi_G := \min_{\emptyset \subsetneq S \subseteq \mathcal{V}} \Phi(S). \tag{4}$$

4) Cheegers' inequality: Research has demonstrated that the conductance Φ_G of the graph G closely correlates with its spectral properties, as articulated by Cheeger's inequality [48]:

$$\omega_2/2 \leq \Phi_G \leq \sqrt{2\omega_2}, \tag{5}$$

where ω_2 is the second smallest eigenvalue of the normalized Laplacian matrix $\widetilde{L}$, defined as $\widetilde{L} = D^{-1/2} L D^{-1/2}$.

5) Effective resistance distance: Let $G = (\mathcal{V}, \mathcal{E}, z)$ represent a connected, undirected graph with weights $z \in \mathbb{R}^{\mathcal{E}}_{\geq 0}$. let $b_p \in \mathbb{R}^{\mathcal{V}}$ denote the standard basis vector characterized by zero entries except for a one in the p-th position, and let $b_{pq} = b_p - b_q$, The effective resistance between nodes p and q, $(p, q) \in \mathcal{V}$ can be computed by:

$$R_{eff}(p,q) = b_{pq}^\top L_G^\dagger b_{pq} = \sum_{i=2}^{|\mathcal{V}|} \frac{(u_i^\top b_{pq})^2}{\lambda_i} = \max_{x\in\mathbb{R}^{\mathcal{V}}} \frac{(x^\top b_{pq})^2}{x^\top L_G x}, \tag{6}$$

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

Graph conductance is a metric used to evaluate the quality of connectivity within a subset of nodes in a graph. Specifically, a lower conductance value indicates that the subset exhibits a higher number of internal edges relative to the number of edges connecting it to the remainder of the graph. This suggests that the subset forms a strongly-connected cluster with limited external interaction. Alternatively, when a graph is modeled as a resistive electrical network, where nodes represent junctions and edge weights correspond to resistances, the concept of effective resistance between two nodes serves as a measure of their pairwise connectivity. A smaller effective resistance implies the presence of multiple alternative paths between the nodes, indicating stronger connectivity in the network.

6) Spectral methods for hypergraphs: Classical spectral graph theory shows that the structure of a simple graph is closely related to the graph's spectral properties. Specifically, Cheeger's inequality demonstrates the close connection between expansion (or conductance) and the first few eigenvalues of graph Laplacians [16]. 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 [15]. However, there has been very limited progress in developing spectral algorithms for hypergraphs. For instance, a classical spectral method has been proposed for hypergraphs by converting each hyperedge into undirected edges using star or clique expansions [49]. This naive hyperedge conversion scheme may result in lower performance due to ignoring the multi-way high-order relationships between the entities. A more rigorous approach by Tasuku and Yuichi [50] generalized spectral graph sparsification for the hypergraph setting by sampling each hyperedge according to a probability determined based on the ratio of the hyperedge weight to 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. A method has been proposed to create the Laplacian matrix of a hypergraph and generalize graph learning algorithms for hypergraph applications [51]. A more mathematically rigorous approach by Chan et al. introduced a nonlinear diffusion process for defining the hypergraph Laplacian operator by measuring the flow distribution within each hyperedge [40], [41]. Moreover, Cheeger's inequality has been proven for hypergraphs under the diffusion-based nonlinear Laplacian operator [40].

7) Hypergraph conductance: A hypergraph $H = (V, E, w)$ consists of a vertex set V and a set of hyperedges E with weights w. The degree of a vertex d_v is defined as: $d_v := \Sigma_{e\in E: v\in e} w(e)$, where $w(e)$ represents the weight of each hyperedge. The volume of a node set $S \subseteq V$ in the hypergraph is defined as: $vol(S) := \Sigma_{v\in S} d_v$. The conductance of a subset S within the hypergraph is then calculated as:

$$\Phi(S) := \frac{cut(S,\hat{S})}{min\{vol(S), vol(\hat{S})\}}, \tag{7}$$

where $cut(S, \hat{S})$ quantifies the number of hyperedges that cross between S and $\hat{S}$. This computation uses an "all or nothing" splitting function that uniformly penalizes the splitting of hyperedges. The hypergraph's overall conductance is defined as:

$$\Phi_H := \min_{\emptyset \subsetneq S \subseteq V} \Phi(S), \tag{8}$$

B. Hypergraph Partitioning Methods

The previous hypergraph partitioners leverage a multilevel paradigm to construct a hierarchy of coarser hypergraphs using local clustering methods. Computing a sequence of coarser hypergraphs that preserve the structural properties of the original hypergraph is a key step in every partitioning method. The coarsening algorithm in existing partitioning methods either computes matching or clustering at each level by utilizing a rating function to cluster strongly correlated vertices. Hyperedge matching and vertex similarity methods are used in the coarsening phase to cluster the nodes and contract the hyperedges. Existing well-known hypergraph partitioners, such as hMETIS [2], KaHyPar [45], PaToH [9], and Zoltan [7], all use heuristic clustering methods to compute the sequence of coarser hypergraphs.

1) Hypergraph coarsening: Multi-level coarsening techniques typically employ either matchings or clusterings on each level of the coarsening hierarchy. These algorithms utilize various rating functions to decide whether vertices should be matched or grouped together, using the contracted vertices to form the vertex set of the coarser hypergraph at the subsequent level. In contrast, n-level partitioning algorithms, such as the graph partitioner KaSPar [52], establish a hierarchy of (nearly) n levels by contracting just one vertex pair between two levels. This approach eliminates the need for matching or clustering algorithms during the graph reduction process. KaSPar utilizes a priority queue to determine the next vertex pair to be contracted. After each contraction, it updates the priority of every neighboring vertex of the contracted vertex to maintain consistency in priorities. However, in hypergraphs, this method faces significant speed limitations because the size of the neighborhood can greatly expand due to a single large hyperedge. To address this limitation, KaHyPar [45] adopts the heavy-edge rating function. This strategy involves initially selecting a random vertex p and contracting it with the best neighboring node that has the highest rating. The rating function specifically selects a vertex pair (p, q) that is involved in a large number of hyperedges with relatively small sizes, optimizing the coarsening process based on the most significant connections between vertices:

$$r(p,q) = \sum_{e=(p,q)\in E} \frac{w(e)}{|e|-1}, \tag{9}$$

where $r(p, q)$ is the rating of the vertex pair, $w(e)$ is the weight of hyperedge e, and $|e|$ is the hyperedge cardinality.

2) Community Detection: The coarsening phase aims to generate progressively smaller yet structurally consistent approximations of the input hypergraph. However, certain scenarios may arise where the inherent structure becomes obscured. For example, tie-breaking decisions may be necessary when multiple neighbors of a vertex share the same rating. Consequently, to improve coarsening schemes, existing partitioners like KaHyPar utilize a preprocessing step involving community detection to guide the coarsening phase. In this approach, the hypergraph is divided into several communities and then the coarsening phase is applied to each community separately. Existing community detection algorithms, such as the Louvain algorithm, partition hypergraph vertices into communities characterized by dense internal connections and sparse external ones. This method reformulates the problem into a task of modularity maximization in graphs.

3) Partitioning objectives: Hypergraph partitioning extends the concept of graph partitioning. Its objective is to distribute the vertex set into multiple disjoint subsets while minimizing a specified cut metric and adhering to certain imbalance constraints. The process of dividing into two subsets is known as *bipartitioning*, whereas dividing into multiple subsets, typically referred to as *k-way partitioning*, involves partitioning into k parts. More formally, consider a hypergraph $H = (V, E, w)$, where k is a positive integer (with $k \geq 2$) and ϵ is a positive real number (where $\epsilon \leq \frac{1}{k}$). The objective of k-way balanced hypergraph partitioning is to divide V into k disjoint subsets $S = \{V_0, V_1, \ldots, V_{k-1}\}$ such that:

- $(\frac{1}{k} - \epsilon)W \leq \sum_{v\in V_i} w_v \leq (\frac{1}{k} + \epsilon)W$, for $0 \leq i \leq k-1$
- $cutsize_H(S) = \sum_{\{e|e\not\subseteq V_i \text{ for any } i\}} w_e$ is minimized

Here, k represents the number of partitions, W is the hypergraph total weight ($W = \sum_{v\in V} w_v$), ϵ denotes the allowable imbalance among the partitions, and each V_i is a block of the partition. We denote S as an ϵ-balanced partitioning solution.

III. SHyPar: Hypergraph Partitioning via Spectral Coarsening

To address the limitations of existing hypergraph coarsening methods that rely on simple heuristics, we propose a theoretically sound and practically efficient framework for hypergraph coarsening. Specifically, we propose a two-phase spectral hypergraph coarsening scheme based on the recent research on spectral hypergraph clustering [46], [47]. **Phase A** utilizes spectral hypergraph coarsening (HyperEF) to decompose a given hypergraph into smaller node partitions with bounded effective-resistance diameters [47]. This is followed by **Phase B**, which guides the coarsening stage using a flow-based community detection method (HyperSF) aimed at minimizing the ratio cut [46]. Next, we exploit the proposed two-phase spectral hypergraph coarsening method for multilevel hypergraph partitioning: the prior heuristic hypergraph coarsening schemes will be replaced by the proposed spectral coarsening methods to create a hierarchy of coarser hypergraphs that can preserve the key structural properties of the original hypergraph.

A. Resistance-Based Hypergraph Coarsening (Phase A)

We leverage the effective resistance parameter to coarsen the hyperedges successively by contracting the hyperedges with small effective resistance. Effective resistance in simple graphs has been used to detect the critical edges for the global structure and it shows how different graph components are connected with each other.

a) Limitations of Existing Coarsening Methods: The existing coarsening algorithms contract vertices at each level of the hierarchy. The primary method involves contracting each vertex with the best neighboring node. This is commonly done by using rating functions to identify and contract highly connected vertices. However, these determinations often rely solely on the weights and sizes of the hyperedges. Rating functions, such as those described in Eq. (9), based on hyperedge size are limited to the local structural properties of the hypergraph and do not account for its global structure. In contrast, the effective-resistance diameter provides a more comprehensive criterion, which considers the global structure. To illustrate this limitation, consider a scenario where the hypergraph contains a bridge with few nodes (small size hyperedge), as illustrated in Figure 2. Algorithms that use hyperedge size tend to inappropriately contract bridge nodes (node 4 and node 7). This contraction can lead to the collapse of the overall structure of the hypergraph. Since the effective resistance of a bridge is high, algorithms based on effective resistance do not contract these nodes and preserve the integrity of the hypergraph structure.

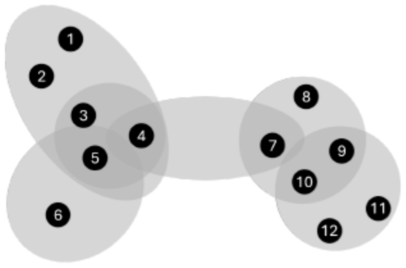


Fig. 2: The example of contraction in a simple hypergraph.

In [35], the authors introduced a spectral algorithm based on effective resistance to sparsify hypergraphs. This method achieves nearly-linear-sized sparsifiers by sampling hyperedges according to their effective resistances [35]. Despite its theoretical appeal, the technique involves a non-trivial procedure for the estimation of hyperedge effective resistances, which could hinder practical efficiency. The need to convert hypergraphs into graphs through clique expansion and the iterative updating of edge weights significantly adds to the complexity of the algorithm [35].

1) HyperEF: Hypergraph Coarsening via effective resistance Clustering: A spectral hypergraph coarsening technique, HyperEF, clusters nodes within each hyperedge when those nodes have a low effective-resistance diameter, as illustrated in Figure 3. This approach significantly reduces the hypergraph size while preserving the original structural characteristics. A crucial part of HyperEF is an efficient procedure for estimating hyperedge effective resistances, which adapts the optimization-based method from Eq. (6) to hypergraphs. Specifically, the effective resistance of a hyperedge is determined by finding an optimal vector χ^* via the following optimization:

$$R_e(\chi^*) = \max_{\chi \in \mathbb{R}^{\mathcal{V}}} \frac{(\chi^\top b_{pq})^2}{Q_H(\chi)}, \quad p, q \in e \tag{10}$$

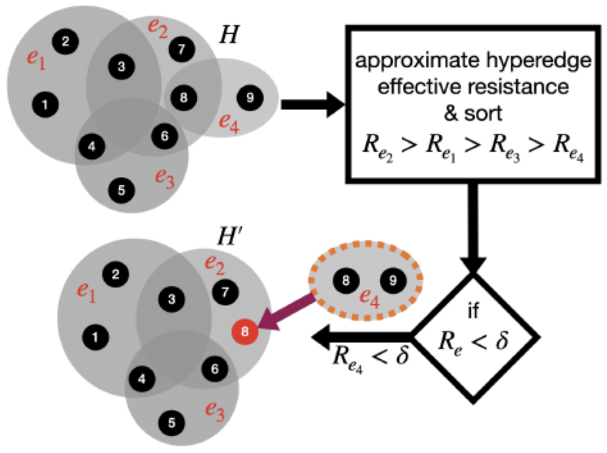

H
e_1
e_2
e_3
e_4
1
2
3
4
5
6
7
8
9
approximate hyperedge effective resistance & sort
$R_{e_2} > R_{e_1} > R_{e_3} > R_{e_4}$
if
$R_e < \delta$
$R_{e_4} < \delta$
H'

Fig. 3: Overview of the HyperEF method.

where the original quadratic form $x^\top L_G x$ in Eq. (6) is replaced by the nonlinear quadratic form $Q_H(\chi)$ [40]:

$$Q_H(\chi) := \sum_{e \in E} w_e \max_{u,v \in e} (\chi_u - \chi_v)^2. \tag{11}$$

As shown in Figure 3, HyperEF generates a significantly smaller hypergraph $H' = (V', E', w')$ from the original hypergraph $H = (V, E, w)$ utilizing effective resistances of the hyperedge, achieving reductions in the number of vertices, edges and weights ($|V'| < |V|$, $|E'| < |E|$ and $|w'| < |w|$).

2) Low-Resistance-Diameter Decomposition: Consider a weighted undirected graph $G = (\mathcal{V}, \mathcal{E}, z)$ with weights $z \in \mathbb{R}^{\mathcal{E}} > 0$ and sufficiently large $\gamma > 1$. The effective-resistance diameter is defined as $\max_{u,v \in \mathcal{V}} R_{eff}(u, v)$. Recent studies demonstrate that it is possible to partition a graph G into multiple node clusters $G[\mathcal{V}_i]$ with low effective-resistance diameters by removing only a small fraction of edges [53]:

$$\max_{u,v \in \mathcal{V}_i} R_{eff_{G[\mathcal{V}_i]}}(u, v) \lesssim \gamma^3 \frac{|\mathcal{V}|}{z(\mathcal{E})}. \tag{12}$$

Additionally, let Φ_G represent the conductance of G. Cheeger's inequality provides a way to establish a relationship between the effective-resistance diameter of the graph and its conductance [53]:

$$\max_{u,v \in \mathcal{V}} R_{eff}(u, v) \lesssim \frac{1}{\Phi_G^2}. \tag{13}$$

Leveraging recent spectral hypergraph theory [40], [41], HyperEF extends the above theorems to hypergraphs. Inequality (12) implies that it is possible to decompose a hypergraph into multiple (hyperedge) clusters that have small effective-resistance diameters by removing only a few inter-cluster hyperedges, while (13) implies that contracting the hyperedges (node clusters) with small effective-resistance diameters will not significantly impact the original hypergraph conductance. Based on these theoretical foundations, HyperEF consists of the following three main steps:

- Constructing the Krylov subspace to approximate the eigensubspace related to the original hypergraph;
- Estimating the effective resistance of each hyperedge by applying the proposed optimization-based method;
- Constructing the coarsened hypergraph by aggregating node clusters with low effective-resistance diameters.

3) Fast Estimation of Hyperedge Effective Resistances: To achieve high efficiency, the search for χ^* in Eq. (10) is restricted to an eigensubspace represented by a few select Laplacian eigenvectors from the simplified graph derived from the original hypergraph. Consider $G_b = (\mathcal{V}_b, \mathcal{E}_b, z_b)$ as the bipartite graph equivalent to the hypergraph $H = (V, E, w)$, where $|\mathcal{V}_b| = |V| + |E|$, $|\mathcal{E}_b| = \Sigma_{e \in E}|e|$, and z_b denotes the scaled edge weights: $z(e, p) = \frac{w(e)}{d(e)}$.

According to Eq.(6), hyperedge effective resistances can be approximated by identifying a set of orthogonal eigenvectors that maximize Eq. (10). To avoid the computational complexity of determining eigenvalues and eigenvectors, a scalable

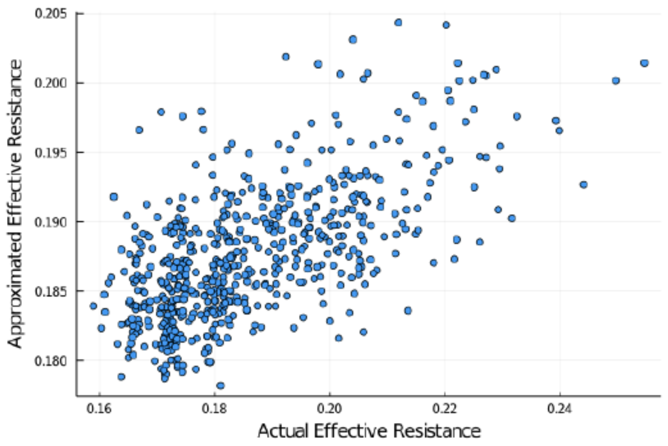


Fig. 4: The approximated effective resistances for a simple graph.

algorithm is used in HyperEF to approximate the eigenvectors by leveraging the Krylov subspace, defined as follows:

For a non-singular matrix $A_{n\times n}$ and a non-zero vector $\chi \neq 0 \in \mathbb{R}^n$, the order-$(\rho+1)$ Krylov subspace generated by A from x is:

$$\kappa_\rho(A,x) := span(x, Ax, A^2x, ..., A^\rho x), \tag{14}$$

where x is a random vector and A is the normalized adjacency matrix of the simple graph obtained from the hypergraph via star expansion.

Let $x^{(1)}, x^{(2)}, ..., x^{(\rho)} \in \mathbb{R}^{\mathcal{V}_b}$ be the ρ mutually-orthogonal vectors based on the order-$(\rho+1)$ Krylov subspace $\kappa_\rho(A,x)$. The effective resistance estimation in Eq. (6) is extended in HyperEF by incorporating the non-linear quadratic operator of hypergraphs from Eq. (11) to include the spectral properties of the hypergraph. 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. Each node in the hypergraph can then be embedded into a ρ-dimensional space. The resistance ratio (r_e) associated with a vector $\chi^{(i)} \in \chi^{(1)}, ..., \chi^{(\rho)}$ for each hyperedge e is computed as:

$$r_e(\chi^{(i)}) = \frac{(\chi^{(i)\top} b_{pq})^2}{Q_H(\chi^{(i)})}, \quad p,q \in e \tag{15}$$

where p and q are the two maximally-separated nodes in the ρ-dimensional embedding space. Multiple resistance ratios $r_e^{(1)}, ..., r_e^{(\rho)}$ are returned by Eq. (10), corresponding to $\chi^{(1)}, ..., \chi^{(\rho)}$. After sorting resistance ratios in descending order, we have:

$$r_e^1 > r_e^2 > ... > r_e^\rho. \tag{16}$$

In HyperEF, the top m resistance ratios are selected to estimate the effective resistance of each hyperedge. The hyperedge effective resistance (R_e) is specifically approximated by:

$$R_e = \sum_{i=1}^{m} r_e^i, \quad e \in E. \tag{17}$$

Note that for each hypergraph, the Krylov subspace vectors in Eq. (14) 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. (15). As shown in Figure 4, the approximate effective resistances of simple graphs obtained using a few Krylov vectors correlate well with the ground-truth values, with a correlation coefficient of about 0.76. Algorithm 1 provides the detailed flow of the proposed hyperedge effective resistance estimation method.

Algorithm 1 The effective resistance estimation algorithm flow

```
Input: Hypergraph $H = (V, E, w)$, $\rho$.
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 $\rho$ resistance ratios using (15).
5: Obtain all hyperedge effective resistances $R$ based on (17).
6: Return $R$.
```

4) Hierarchical Effective Resistance Propagation : We preserve the hypergraph structure by employing a technique that iteratively computes a vector of effective resistance R and contracts hyperedges exhibiting low effective resistances $(R_e < \delta)$, where δ is the effective resistance threshold. Throughout this process, node clusters are contracted by merging the nodes within each cluster and replacing that cluster with a new *supernode* in the next level. By assigning a weight to each supernode—equal to the hyperedge's effective resistance evaluated at the previous level—we propagate essential structural information through all levels of coarsening. As an example, in Figure 3, when hyperedge e_4 in H is contracted, the node 8 in H' takes on a weight that is equal to the effective resistance of e_4. This ensures that the structural information from the original hyperedge remains available in subsequent coarsening steps.

Let $H^{(l)} = (V^{(l)}, E^{(l)}, w^{(l)})$ represent the hypergraph at the l-th level. The vector of effective resistance R is updated in each coarsening level according to the following equation:

$$R_e^{(l)} \leftarrow \sum_{v\in e} \eta(v) + R_e^{(l)}, \tag{18}$$

where $\eta(v)$ represents the weight of the nodes $v \in e$ corresponding to a contracted hyperedge from the previous level, initially set to all zeros for the original hypergraph. Consequently, the effective resistance of a hyperedge at a coarser level depends not only on $R_e^{(l)}$ (computed at the current level), but also on the effective-resistance data transferred from all previous levels. This allows the algorithm to preserve global structural information through the multilevel coarsening process.

The experimental results indicate that using Eq. (18) for effective resistances estimation yields more balanced hypergraph clustering outcomes compared to approaches that ignore the previous clustering information. The complete workflow for the effective resistance clustering algorithm, HyperEF, used for coarsening a hypergraph H across L levels, is detailed in Algorithm 2.

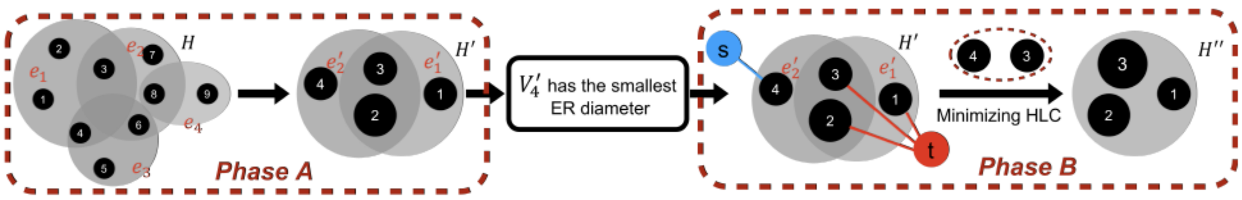


Fig. 5: Overview of the HyperSF method.

Algorithm 2 The HyperEF algorithm for hypergraph clustering flow

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

1: Initialize $H' \leftarrow H$
2: for $l \leftarrow 1$ to $L$ do
3:     Call Algorithm 1 to compute a vector of effective resistance
       $R$ with the size $|E'|$ for given hypergraph $H'$.
4:     Update the effective resistance vector $R$ by including the
       supernode weights using Eq. (18)
5:     Sort the hyperedges with ascending $R$ values.
6:     Starting with the hyperedges that have the lowest effective
       resistances, contract (cluster) the hyperedge (nodes) if $R_e < \delta$.
7:     Construct a coarsened hypergraph $H'$ accordingly.
8: end for
9: Return $H'$.
```

B. Flow-Based Community Detection (Phase B)

To improve coarsening schemes for hypergraph partitioning, we utilize a community structure that integrates global hypergraph information into the coarsening process. This community structure directs the coarsening phase by permitting contractions solely within clusters.

In the flow-based community detection, we employ multilevel clustering through HyperEF, which enhances the spectral hypergraph clustering method by integrating a multilevel coarsening approach. Let $H = (V, E, w)$, the hypergraph local conductance (**HLC**) with respective to a node-set S is defined as follows [54]:

$$\mathbf{HLC}_{\mathcal{C}}(S) = \frac{cut(S, \hat{S})}{vol(S \cap \mathcal{C}) - \beta vol(S \cap \hat{\mathcal{C}})}, \tag{19}$$

where $\mathcal{C} \subseteq V$ is reference node set, and β is a locality parameter that modulates the penalty for incorporating nearby nodes outside set $\mathcal{C}$.

A spectral hypergraph coarsening algorithm (HyperSF) is proposed by minimizing the **HLC**, which has achieved promising results in hypergraph coarsening and partitioning in realistic VLSI designs.

1) Overview of Coarsening Refinement (HyperSF): Figure 5 shows an overview of the HyperSF method. In this work, HyperSF is leveraged for only refining the most imbalanced node clusters (initially identified by HyperEF) with significantly smaller resistance diameters compared to the rest. Utilizing the flow-based clustering method that is proposed in [54], HyperSF aggregates strongly-coupled node clusters via minimizing Eq. (19): for each selected node set with large imbalance, HyperSF repeatedly solves a max s-t flow, min s-t cut problem to detect a set of neighboring node clusters that minimizes the local conductance **HLC** in Eq. (19). To this end, the following key steps are applied (as shown in Figure 6): **(Step 1)** an auxiliary hypergraph is constructed by introducing a source vertex s and sink vertex t; **(Step 2)** each hyperedge is replaced with a directed graph; **(Step 3)** each seed node-set is iteratively updated by including new nodes into the set to minimize the **HLC** by repeatedly solving the max s-t flow, min s-t cut problem:

$$cut^{s-t}(S) = cut_H(S) + vol_H(\hat{S} \cap \mathcal{C}) + \beta vol_H(S \cap \hat{\mathcal{C}}). \tag{20}$$

(Step 4) The node sets obtained from flow-based methods that minimize the local conductance are exploited to produce a smaller hypergraph with fewer nodes while preserving the key structural properties of the original hypergraph.

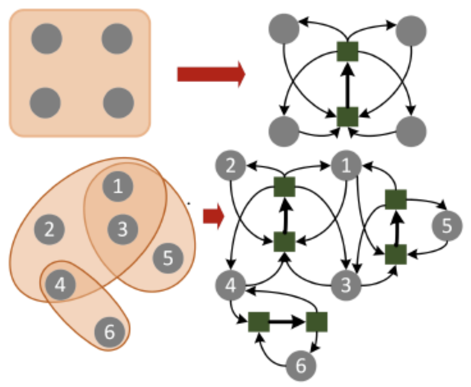
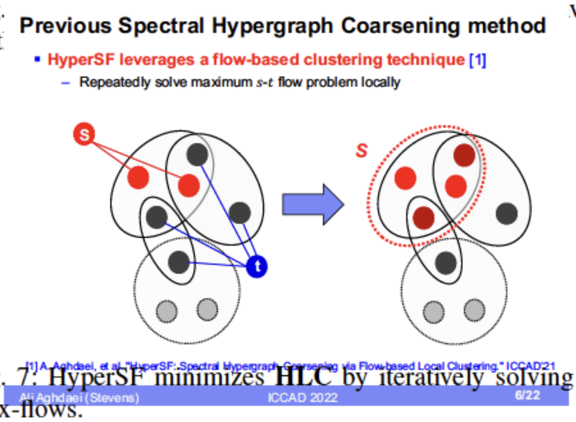


Fig. verted
to t


Previous Spectral Hypergraph Coarsening method

- **HyperSF leverages a flow-based clustering technique [1]**
  - Repeatedly solve maximum s-t flow problem locally

[1] A. Aghdaei, et al. "HyperSF: Spectral Hypergraph Coarsening via Flow-based Local Clustering." ICCAD'21

Ali Aghdaei (Stevens) ICCAD 2022 6/22

Fig. 7: HyperSF minimizes **HLC** by iteratively solving local max-flows.

2) Local clustering algorithms: The proposed flow-based clustering algorithm in [54] is strongly local by expanding the network around the seed nodes $\mathcal{C}$, which benefits the coarsening framework in two ways: (1) applying the max s-t flow, min s-t cut problem on the local neighborhood of the seed nodes restricts node-aggregation locally and keeps the global hypergraph structure intact; (2) such a local clustering scheme significantly improves the algorithm efficiency due to the small-scale input dataset.

3) Flow-based Local Clustering in HyperSF: First, we apply HyperEF (Algorithm 2) to the hypergraph $H = (V, E, w)$ to compute a coarsened hypergraph $H' = (V', E', w')$ and identify isolated (unclustered) nodes, denoted by $\mathcal{C}$. These isolated nodes are simply those that remain unclustered after HyperEF's multilevel coarsening step. Leveraging the introduced flow-based clustering in [54], HyperSF constructs a sub-hypergraph H'_L by iteratively expanding the hypergraph around the seed node set $\mathcal{C}$ and then repeatedly solve the hypergraph cut problem to minimize **HLC** until no significant changes in local conductance are observed. Define $E'(S) = \cup_{v' \in V', v' \in \{e'\}_{e' \in E'}} E'(v)$ for any set $S \subseteq V'$ and let $H'_\varsigma = (V' \cup \{s,t\}, E' \cup E'^{st})$, where E'^{st} denotes the terminal edge set. HyperSF aims to construct a sub-hypergraph H'_L to replace H'_ς that minimizes **HLC** by repeatedly solving a local version of (20). To this end, HyperSF sets up an oracle to discover a set of best neighborhood vertices for a given vertex v':

$$\kappa(v') = \{u' \in V'\}_{(u',v') \in e', e' \in E'}. \tag{21}$$

HyperSF lets the oracle accept a set of seed nodes $\mathcal{C}$ and return $\kappa(\mathcal{C}) = \cup_{v' \in \mathcal{C}} \kappa(v')$. By utilizing the best neighborhood of the seed nodes $\kappa(\mathcal{C})$, HyperSF builds a local hypergraph $H'_L = (V'_L \cup \{s,t\}, E'_L \cup E'^{st}_L)$, where $V'_L = \mathcal{C} \cup \kappa(\mathcal{C})$ and $E'_L = \{e' \in E' \mid V'_L \in e'\}$. As shown in Figure 7, HyperSF creates the local auxiliary hypergraph of H'_L by introducing the source node s and the sink node t, so that $E'^{st}_L \subseteq E'^{st}$ and repeatedly solves the max s-t flow, min s-t cut problem to minimize **HLC**. The algorithm continuously expands H'_L and includes more vertices and hyperedges from H'_ς by solving (20) for the local hypergraph H'_L.

Algorithm 3 Flow-based Hypergraph Clustering

```
Input: Hypergraph $H = (V, E, w)$, and $\xi$ (Convergence parameter)
Output: A set of vertices $S$ that minimizes HLC(S);

 1: Apply HyperEF to $H$ to compute coarsened hypergraph $H' = (V', E', w')$, and isolated nodes $\mathcal{C} \subseteq V$;
 2: Assign isolated nodes as seed nodes $S \leftarrow \mathcal{C}$;
 3: $\Delta_{\mathbf{HLC}} \leftarrow \infty$;
 4: while $\Delta_{\mathbf{HLC}} > \xi$ do
 5:     Identify the best neighborhood of seed nodes $\kappa(S)$;
 6:     Update $S$ according to $\kappa(S)$ to construct $H'_L$;
 7:     Add a source node $s$ and sink node $t$ to $H'_L$;
 8:     Repeatedly solve the max $s$-$t$ flow, min $s$-$t$ cut problem by minimizing (20) for $H'_L$;
 9:     $\Delta_{\mathbf{HLC}} \leftarrow \mathbf{HLC(S)}^j - \mathbf{HLC(S)}^{j-1}$;
10: end while
11: Return $S$.
```

The algorithm 3 presents the details of the flow-based local clustering technique. It accepts the original hypergraph $H = (V, E, w)$, and the convergence parameter ξ, which will output a set of strongly connected vertices S that minimizes **HLC**.

C. Algorithm Complexity for Spectral Coarsening

In HyperEF, the complexity 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 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 $O(\rho|E| + |\mathcal{E}_b|)$. In HyperSF, the runtime complexity of the proposed strongly local flow-based algorithm is $O\left(k^3 vol_H(\mathcal{C})^3 (1+\epsilon^{-1})^3\right)$, where k is the maximum hyperedge cardinality. Since each phase in the proposed spectral hypergraph coarsening method has a nearly linear-time complexity, the entire two-phase spectral coarsening procedure will be highly scalable for handling large-scale hypergraphs.

D. Hypergraph Partitioning with Spectral Coarsening and Flow-Based Clustering

Building on previous hypergraph partitioning methods, we replace the coarsening algorithm with our proposed resistance-based approach and incorporate flow-based community detection to further enhance partitioning quality.

1) Multilevel Spectral Coarsening with HyperEF: While multilevel hypergraph partitioning frameworks like KaHyPar use hyperedge size metrics in their coarsening phase to determine which vertex pairs should be contracted, we introduce a resistance-based rating function that preserves structural properties when generating a hierarchy of smaller hypergraphs. For a given vertex pair (p, q), we define the rating function as:

$$r(p,q) = \sum_{\substack{e \in E \\ \{p,q\} \subseteq e}} \frac{w(e)}{R_e - 1} \tag{22}$$

where $w(e)$ represents the weight of hyperedge e, and R_e denotes its effective resistance. The effective resistance of each hyperedge, R_e for $e \in E$, is computed using Eq. (17) in nearly-linear time. The effective resistance information is further propagated throughout the multilevel coarsening scheme by using a node weight propagation technique, as described in Eq. (18). This approach enables us to obtain a bound on the maximum distance between any pair of nodes within a hyperedge, thereby reducing the hypergraph size while maintaining the key spectral properties of the original hypergraph.

2) Flow-Based clustering with HyperSF: To further improve the quality of partitioning, we incorporate flow-based community detection using HyperSF as described in Algorithm 3, as a preprocessing step. HyperSF creates high-quality clusters by analyzing the hypergraph flow structure.

The key insight of our approach is to first identify communities using HyperSF and then apply the multilevel spectral coarsening algorithm to each community separately. This community-aware coarsening strategy produces more balanced partitions that better preserve the hypergraph's structural properties.

HyperSF sorts clusters by their resistance diameter and prioritizes those with the lowest values. It then applies flow-based techniques to merge nearby clusters with low resistance diameters within the same neighborhood. The resulting clusters exhibit low hypergraph local conductance (**HLC**), which helps preserve the spectral properties of the original hypergraph during the coarsening process.

3) SHyPar Algorithm Flow: In Algorithm 4, we present the detailed procedure of SHyPar, which leverages resistance-based spectral coarsening and flow-based clustering techniques for hypergraph partitioning.

The input is a weighted hypergraph $H = (V, E, w)$, and the output is a partitioned hypergraph where each node is assigned a partition index. **Line 1** represents the one-time computation of effective resistance using the HyperEF algorithm, which is employed during the coarsening phase. **Line 2** performs a one-time computation of flow-based clusters via the HyperSF algorithm, aiding in community detection. **Lines 3–5** outline the multilevel hypergraph partitioning pipeline, including coarsening, initial cut computation, and iterative refinement. **Line 6** produces the final partitioning solution of the hypergraph.

Algorithm 4 SHyPar Algorithm

```
Input: Hypergraph $H = (V, E, w)$
Output: Partitioned Hypergraph;

1: Setup: Hyperedge Effective resistance computation using HyperEF algorithm 1
2: Setup: Flow-based community detection using HyperSF algorithm 3
3: Multilevel spectral coarsening using Eq.22
4: Initial partitioning based on KaHyPar algorithm
5: Solution refinement based on KaHyPar algorithm
6: Return the partitioned hypergraph
```

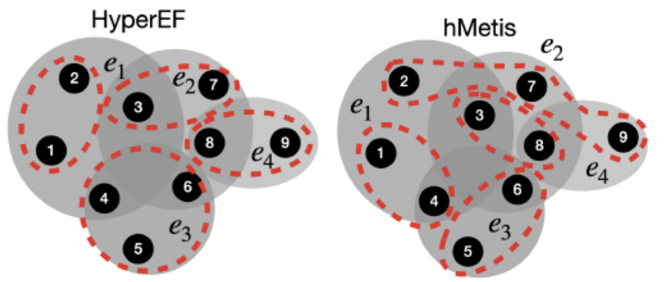
IV. Experimental Validation

There are many applications related to hypergraph partitioning. This paper focuses on comprehensively evaluating the performance of the proposed hypergraph partitioning framework for increasingly important applications related to integrated circuits computer-aided design. Both the solution quality and runtime efficiency will be carefully assessed by testing them on a wide range of public-domain data sets. The implementation of the proposed algorithm, along with the code for reproducing the experimental results, is publicly available at https://github.com/Feng-Research/SHyPar.

To assess the performance of the proposed multilevel hypergraph partitioning tools in applications related to VLSI designs, We apply the multilevel hypergraph partitioning tool developed to partition public-domain VLSI design benchmarks. For example, the ISPD98 benchmarks that include "IBM01", "IBM02", ..., "IBM18" hypergraph models with $13,000$ to $210,000$ nodes be adopted [55]. The performance metrics for multilevel hypergraph partitioning, including the total hyperedge cut, imbalance factors, and runtime efficiency, will be considered for comparisons with state-of-the-art hypergraph partitioning tools, such as hMETIS [2], and KaHyPar [45].

A. Spectral coarsening with HyperEF and HyperSF

In this section, we compare the preliminary implementations of resistance-based hypergraph clustering (HyperEF) and flow-based clustering (HyperSF) with the well-known hypergraph partitioning tool, hMETIS [2]. Real-world VLSI design (hypergraph) benchmarks have been tested [55]. All experiments have been evaluated on a computing platform with 8 GB of RAM and a 2.2 GHz Quad-Core Intel Core i7 processor.



Average conductance = 0.48 *Average conductance* = 0.77

Fig. 8: Resistance-based vs hMETIS Clustering Results.

1) HyperEF vs hMETIS for Hypergraph Coarsening: HyperEF is compared to hMETIS for hypergraph coarsening considering both solution quality and runtime efficiency. The following average conductance of the node clusters is used to analyze the performance of each method.

$$\Phi_{\text{avg}} = \frac{1}{|S|}\sum_{i=1}^{|S|}\Phi(S_i) \tag{23}$$

Where $\Phi(S_i)$ denotes the conductance of node cluster S_i. Figure 8 demonstrates the node clustering results for a small hypergraph obtained using HyperEF and hMETIS. Both methods partition the hypergraph into four clusters, and the average conductance of node clusters has been computed to evaluate the performance of each method. The results show that HyperEF outperforms hMETIS by creating node clusters with a significantly lower average conductance. In addition, Table I shows the average conductance of node clusters Φ_{avg} computed with both HyperEF and hMETIS by decomposing the hypergraph with the same node reduction ratios (NRs). With an NR = 75% (3× node reduction) HyperEF outperforms hMETIS in average conductance while achieving $24 - 38\times$ speedups over hMETIS.

2) HyperSF vs hMETIS for Hypergraph Coarsening: In this section, we evaluate the performance of HyperSF against the hMETIS hypergraph partitioning tool. We measure the average local conductance $\mathbf{HLC}_{\text{avg}}$ of the node clusters generated by each method, calculated as follows:

$$\mathbf{HLC}_{\text{avg}} = \frac{1}{|S|}\sum_{i=1}^{|S|}\mathbf{HLC}(S^i). \tag{24}$$

Table II presents the average local conductance $\mathbf{HLC}_{\text{avg}}$ for various methods under the same hypergraph reduction ratio (RR), where we reduce the number of nodes in each original hypergraph by 75%. The experimental data illustrate that HyperSF significantly enhances the average local conductance compared to the hMETIS method in all test scenarios.

TABLE I: HyperEF vs hMETIS coductance (NR=75%)

Benchmark	Φ_{avg} HyperEF	Φ_{avg} hMETIS	$\mathcal{T}$ (seconds) HyperEF	$\mathcal{T}$ (seconds) hMETIS
IBM01	**0.62**	0.65	1.23	29 (**24×**)
IBM02	**0.62**	0.67	1.41	49 (**35×**)
IBM03	**0.63**	0.66	2.11	53 (**25×**)
IBM04	**0.64**	0.66	2.37	60 (**25×**)
IBM05	**0.59**	0.63	2.34	62 (**26×**)
IBM06	**0.64**	0.66	2.63	78 (**30×**)
IBM07	**0.63**	0.67	3.54	115 (**32×**)
IBM08	**0.61**	0.67	4.15	125 (**30×**)
IBM09	**0.64**	0.66	4.38	131 (**30×**)
IBM10	**0.63**	0.67	5.79	181 (**31×**)
IBM11	**0.64**	0.67	5.73	176 (**31×**)
IBM12	**0.65**	0.7	5.94	191 (**32×**)
IBM13	**0.65**	0.68	6.87	229 (**33×**)
IBM14	**0.62**	0.66	11.51	393 (**34×**)
IBM15	**0.66**	0.69	14.44	486 (**34×**)
IBM16	**0.63**	0.67	14.62	533 (**36×**)
IBM17	**0.66**	0.7	15.22	568 (**37×**)
IBM18	**0.6**	0.67	15.79	602 (**38×**)

TABLE II: HyperSF vs hMETIS local coductance (NR=75%)

Benchmark	HLC_{avg} HyperSF	HLC_{avg} hMETIS	$\mathcal{T}$ (seconds) HyperSF	$\mathcal{T}$ (seconds) hMETIS
IBM01	**0.44**	0.65	9.4	29 (3×)
IBM02	**0.52**	0.69	22.6	49 (2×)
IBM03	**0.48**	0.67	14.1	53 (4×)
IBM04	**0.47**	0.68	15	60 (4×)
IBM05	**0.55**	0.65	29.2	62 (2×)
IBM06	**0.51**	0.68	30.1	78 (3×)
IBM07	**0.48**	0.68	26.4	115 (4×)
IBM08	**0.48**	0.68	43.3	125 (3×)
IBM09	**0.47**	0.69	24.5	131 (5×)
IBM10	**0.48**	0.68	45.2	181 (4×)
IBM11	**0.46**	0.69	30.1	176 (6×)
IBM12	**0.50**	0.71	42.4	191 (5×)
IBM13	**0.48**	0.69	50.4	229 (5×)
IBM14	**0.48**	0.67	85.7	393 (5×)
IBM15	**0.47**	0.71	96	486 (5×)
IBM16	**0.50**	0.70	116.8	533 (5×)
IBM17	**0.51**	0.73	141.3	568 (4×)
IBM18	**0.46**	0.68	129	602 (5×)

B. Hypergraph Partitioning with Spectral Coarsening

We compared SHyPar with leading hypergraph partitioners hMETIS [2], SpecPart [43], KaHyPar [45], and MedPart [56] using two sets of publicly available benchmarks: the ISPD98 VLSI Circuit Benchmark Suite [55] and the Titan23 Suite [57]. The details of these benchmarks are outlined in Table III and Table IV. All tests were conducted on a server equipped with Intel(R) Xeon(R) Gold 6244 processors with 1546GB of memory.

1) Experimental Setup: To implement SHyPar, we have developed new hypergraph partitioning tools based on the existing open-source multilevel hypergraph partitioner, KaHyPar. We are utilizing the proposed two-phase spectral hypergraph coarsening method. Specifically, the heuristic coarsening scheme has been replaced by our novel spectral coarsening algorithm to create a hierarchy of coarser hypergraphs that preserve the key structural properties of the original hypergraph. Accordingly, we have substituted the existing coarsening method in KaHyPar with our proposed method, incorporating a new rating function. Additionally, the existing algorithm for community detection, Louvain, has been replaced with our proposed flow-based community detection method (Phase 2).

2) SHyPar Performance on ISPD98 Benchmarks: Table III presents a comparison of the cut sizes achieved by SHyPar on the ISPD98 VLSI circuit benchmark against those obtained from hMETIS, SpecPart, KaHyPar, and MedPart. The results for SHyPar show an average improvement of approximately 0.54% for $\epsilon = 2\%$ and 0.4% for $\epsilon = 10\%$, affirming its superiority over the best-published results. In several instances, SHyPar outperforms the best-published results by up to 5%; these instances are specifically underlined for emphasis. Figure 9 depicts the cut sizes obtained by SHyPar, KaHyPar, and hMETIS, normalized against the KaHyPar results. It is evident that SHyPar significantly enhances performance over both KaHyPar and hMETIS across many tests. Moreover, when SHyPar was applied to four partitions with $\epsilon = 1\%$, the improvements were consistent, as demonstrated in Figure 10, which compares the cut sizes with those from KaHyPar and hMETIS, also normalized by KaHyPar results.

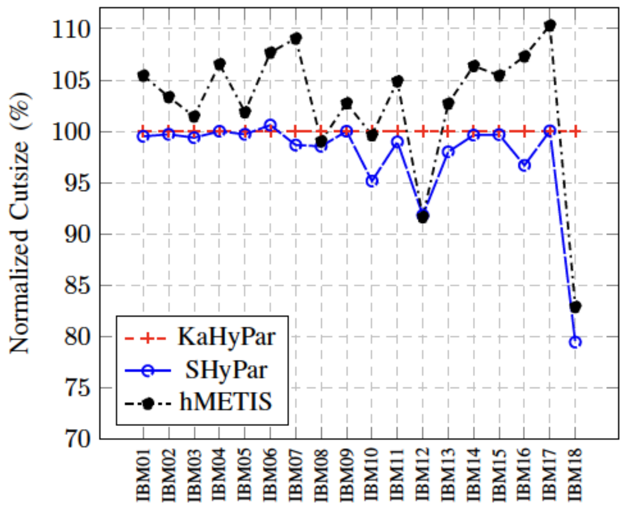


Fig. 9: ISPD98 benchmarks with unit weights $\epsilon = 2\%$ k = 2.

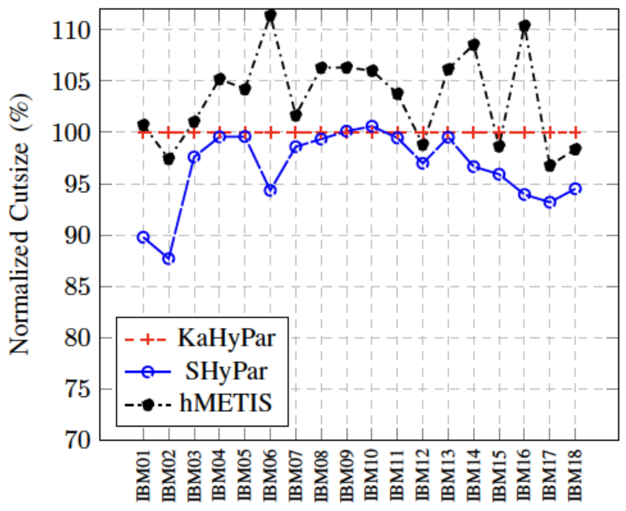


Fig. 10: ISPD98 benchmarks with unit weights $\epsilon = 1\%$ k = 4.

TABLE III: Statistics of ISPD98 VLSI circuit benchmark suite and cut sizes obtained by different approaches. The best results among all methods are highlighted in red.

Benchmark	Statistics		$\epsilon = 2\%$					$\epsilon = 10\%$				
	$\|V\|$	$\|E\|$	SpecPart	hMETIS	KaHyPar	MedPart	SHyPar	SpecPart	hMETIS	KaHyPar	MedPart	SHyPar
IBM01	12,752	14,111	202	213	202	202	201	171	190	173	166	166
IBM02	19,601	19,584	336	339	328	352	327	262	262	262	264	262
IBM03	23,136	27,401	959	972	958	955	952	952	960	950	955	950
IBM04	27,507	31,970	593	617	579	583	579	388	388	388	389	388
IBM05	29,347	28,446	1720	1744	1712	1748	1707	1688	1733	1645	1675	1645
IBM06	32,498	34,826	963	1037	963	1000	969	733	760	735	788	733
IBM07	45,926	48,117	935	975	894	913	882	760	796	760	773	760
IBM08	51,309	50,513	1146	1146	1157	1158	1140	1140	1145	1120	1131	1120
IBM09	53,395	60,902	620	637	620	625	620	519	535	519	520	519
IBM10	69,429	75,196	1318	1313	1318	1327	1254	1261	1284	1250	1259	1244
IBM11	70,558	81,454	1062	1114	1062	1069	1051	764	782	769	774	763
IBM12	71,076	77,240	1920	1982	2163	1955	1986	1842	1940	1841	1914	1841
IBM13	84,199	99,666	848	871	848	850	831	693	721	693	697	655
IBM14	147,605	152,772	1859	1967	1849	1876	1842	1768	1665	1534	1639	1534
IBM15	161,570	186,608	2741	2886	2737	2896	2728	2235	2262	2135	2169	2135
IBM16	183,484	190,048	1915	2095	1952	1972	1887	1619	1708	1619	1645	1619
IBM17	185,495	189,581	2354	2520	2284	2336	2285	1989	2300	1989	2024	1989
IBM18	210,613	201,920	1535	1587	1915	1955	1521	1537	1550	1915	1829	1520
Average Improvement over hMETIS (%)			3.64	0	2.19	1.11	4.92	2.91	0	2.77	1.65	4.77

TABLE IV: Statistics of the Titan23 benchmark suite and cut sizes obtained by different approaches. The best results among all methods are highlighted in red.

Benchmark	Statistics		$\epsilon = 2\%$					$\epsilon = 20\%$				
	$\|V\|$	$\|E\|$	SpecPart	hMETIS	KaHyPar	MedPart	SHyPar	SpecPart	hMETIS	KaHyPar	MedPart	SHyPar
sparcT1_core	91,976	92,827	1012	1066	974	1067	974	903	1290	873	624	631
neuron	92,290	125,305	252	260	244	262	243	206	270	244	270	244
stereo_vision	94,050	127,085	180	180	169	176	169	91	143	91	93	91
des90	111,221	139,557	402	402	380	372	379	358	441	380	349	345
SLAM_spheric	113,115	142,408	1061	1061	1061	1061	1061	1061	1061	1061	1061	1061
cholesky_mc	113,250	144,948	285	285	283	283	283	345	667	591	281	479
segmemtation	138,295	179,051	126	136	107	114	107	78	141	78	78	78
bitonic_mesh	192,064	235,328	585	614	593	594	586	483	590	592	493	506
dart	202,354	223,301	807	844	924	805	784	540	603	594	549	539
openCV	217,453	284,108	510	511	560	635	499	518	554	501	554	473
stap_qrd	240,240	290,123	399	399	371	386	371	295	295	275	287	275
minres	261,359	320,540	215	215	207	215	207	189	189	199	181	191
cholesky_bdti	266,422	342,688	1156	1157	1156	1161	1156	947	1024	1120	1024	848
denoise	275,638	356,848	416	722	416	516	416	224	478	244	224	220
sparcT2_core	300,109	302,663	1244	1273	1186	1319	1183	1245	1972	1186	1081	918
gsm_switch	493,260	507,821	1827	5974	1759	1714	1621	1407	5352	1719	1503	1407
mes_noc	547,544	577,664	634	699	649	699	651	617	633	755	633	617
LU230	574,372	669,477	3273	4070	4012	3452	3602	2677	3276	3751	2720	2923
LU_Network	635,456	726,999	525	550	524	550	524	524	528	524	528	524
sparcT1_chip2	820,886	821,274	899	1524	874	1129	873	783	1029	856	877	757
directrf	931,275	1,374,742	574	646	646	646	632	295	379	295	317	295
bitcoin_miner	1,089,284	1,448,151	1297	1570	1576	1562	1514	1225	1255	1287	1255	1282
Average Improvement over hMETIS (%)			10.99	0	9.95	6.82	12.16	21.73	0	14.14	20.94	22.53

3) SHyPar Performance on Titan23 Benchmarks: Building on the findings from the ISPD98 benchmarks, our research extended to the Titan23 benchmarks, notable for their high-degree hyperedges. Table IV details the results on the Titan23 benchmarks, where SHyPar dramatically outperforms hMETIS, achieving a 12% improvement for $\epsilon = 2\%$ and an impressive 22.5% for $\epsilon = 20\%$. Also SHyPar outperforms SpecPart, achieving a 3% improvement for $\epsilon = 2\%$ and 1% for $\epsilon = 20\%$. In cases such as sparcT2_core, SHyPar even exceeds the best published results by up to 15%; these notable achievements are highlighted by underlining.

4) SHyPar Runtime Comparison with KaHypar: Table V presents a runtime comparison between SHyPar and KaHyPar on the Titan23 benchmark for $\epsilon = 2\%$. The setup time corresponds to the one-time computation of the effective resistance (algorithm 1) and the flow-based clustering method (algorithm 3). The SHyPar/KaHyPar runtime ratio compares the runtime of SHyPar with the KaHyPar method, excluding the one-time setup time of effective resistance computation and flow-based clustering in SHyPar. To ensure a fair comparison, the runtime of Louvain clustering is also excluded from KaHyPar. The results demonstrate that our method achieves up to 8% improvement in runtime over KaHyPar.

5) Ablation Study: To evaluate the impact of spectral coarsening and flow-based community detection, we consider a new configuration that employs only spectral coarsening within KaHyPar. Table VI presents the cut size results for some benchmarks for $\epsilon = 2\%$. While replacing KaHyPar's default

TABLE V: SHyPar runtime comparison with KaHyPar

Benchmark	KaHyPar (s)	Setup Time (s)	SHyPar / KaHyPar
sparcT1_core	52.99	10.19	0.94
neuron	41.06	10.71	0.93
stereo_vision	3.19	10.63	0.99
des90	49.90	13.29	0.94
SLAM_spheric	10.12	12.37	0.99
cholesky_mc	7.18	12.15	0.99
segmemtation	8.79	13.92	0.99
bitonic_mesh	43.52	18.04	0.93
dart	47.19	16.80	0.94
openCV	47.23	18.07	0.94
stap_qrd	22.52	19.28	0.93
minres	23.65	20.25	0.93
cholesky_bdti	35.40	20.86	0.92
denoise	22.66	23.57	0.93
sparcT2_core	49.21	22.05	0.94
gsm_switch	139.76	32.38	0.98
mes_noc	74.12	36.82	0.96
LU230	1071.02	38.26	1.00
LU_Network	78.99	43.66	0.96
sparcT1_chip2	167.14	50.63	0.98
directrf	80.89	68.07	0.96
bitcoin_miner	471.46	71.27	0.99

coarsening with spectral coarsening (SC) already outperforms the baseline, adding flow-based community detection (Flow-Based CD) on top yields even better solutions.

TABLE VI: Cut size improvement with Spectral Coarsening

Benchmark	KaHyPar	KaHyPar with SC	KaHyPar with SC and Flow-Based CD
dart	924	803	784
OpenCV	560	549	499
sparcT2_core	1186	1184	1183
gsm_switch	1759	1732	1621
LU230	4012	3610	3602
IBM05	1712	1709	1707
IBM07	894	892	882
IBM08	1157	1146	1140
IBM10	1318	1272	1254
IBM18	1915	1551	1521

V. Conclusion

In this study, we introduced SHyPar, a multilevel hypergraph partitioning framework that enhances partitioning solutions and surpasses earlier studies in performance. We developed an innovative algorithm that incorporates spectral hypergraph coarsening techniques, leverages hyperedge effective resistances and flow-based community detection. Our comprehensive experimental analysis, conducted on real-world VLSI test cases, demonstrates that SHyPar consistently achieves a significant reduction in hypergraph partitioning cut size, improving results up to 15 percent compared to state-of-the-art methods.

VI. Acknowledgments

This work is supported in part by the National Science Foundation under Grants CCF-2417619, CCF-2021309, CCF-2011412, CCF-2212370, and CCF-2205572.

References

[1] U. V. Catalyurek and C. Aykanat, "Hypergraph-partitioning-based decomposition for parallel sparse-matrix vector multiplication," *IEEE Transactions on parallel and distributed systems*, vol. 10, no. 7, pp. 673–693, 1999.

[2] 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*, vol. 7, no. 1, pp. 69–79, 1999.

[3] D. Kucar, S. Areibi, and A. Vannelli, "Hypergraph partitioning techniques," *DYNAMICS OF CONTINUOUS DISCRETE AND IMPULSIVE SYSTEMS SERIES A*, vol. 11, pp. 339–368, 2004.

[4] K. A. Murgas, E. Saucan, and R. Sandhu, "Hypergraph geometry reflects higher-order dynamics in protein interaction networks," *Scientific reports*, vol. 12, no. 1, pp. 1–12, 2022.

[5] Ü. V. Çatalyürek, K. D. Devine, M. F. Faraj, L. Gottesbüren, T. Heuer, H. Meyerhenke, P. Sanders, S. Schlag, C. Schulz, D. Seemaier, *et al.*, "More recent advances in (hyper) graph partitioning," *ACM Computing Surveys*, 2022.

[6] M. R. Garey and D. S. Johnson, "Computers and intractability," *A Guide to the*, 1979.

[7] 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*, pp. 10–pp, IEEE, 2006.

[8] B. Vastenhouw and R. H. Bisseling, "A two-dimensional data distribution method for parallel sparse matrix-vector multiplication," *SIAM review*, vol. 47, no. 1, pp. 67–95, 2005.

[9] Ü. V. Çatalyürek and C. Aykanat, "Patoh (partitioning tool for hypergraphs)," in *Encyclopedia of Parallel Computing*, pp. 1479–1487, Springer, 2011.

[10] R. Shaydulin, J. Chen, and I. Safro, "Relaxation-based coarsening for multilevel hypergraph partitioning," *Multiscale Modeling and Simulation*, vol. 17, pp. 482–506, Jan 2019.

[11] S.-H. Teng, "Scalable algorithms for data and network analysis," *Foundations and Trends® in Theoretical Computer Science*, vol. 12, no. 1–2, pp. 1–274, 2016.

[12] D. Spielman and S. Teng, "Nearly linear time algorithms for preconditioning and solving symmetric, diagonally dominant linear systems," *SIAM Journal on Matrix Analysis and Applications*, vol. 35, no. 3, pp. 835–885, 2014.

[13] J. A. Kelner, Y. T. Lee, L. Orecchia, and A. Sidford, "An almost-linear-time algorithm for approximate max flow in undirected graphs, and its multicommodity generalizations," in *Proceedings of the twenty-fifth annual ACM-SIAM symposium on Discrete algorithms*, pp. 217–226, SIAM, 2014.

[14] P. Christiano, J. Kelner, A. Madry, D. Spielman, and S. Teng, "Electrical flows, laplacian systems, and faster approximation of maximum flow in undirected graphs," in *Proc. ACM STOC*, pp. 273–282, 2011.

[15] 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*, pp. 96–105, IEEE, 1996.

[16] J. R. Lee, S. O. Gharan, and L. Trevisan, "Multiway spectral partitioning and higher-order cheeger inequalities," *Journal of the ACM (JACM)*, vol. 61, no. 6, p. 37, 2014.

[17] R. Peng, H. Sun, and L. Zanetti, "Partitioning well-clustered graphs: Spectral clustering works," in *Proceedings of The 28th Conference on Learning Theory (COLT)*, pp. 1423–1455, 2015.

[18] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," *arXiv preprint arXiv:1609.02907*, 2016.

[19] M. Defferrard, X. Bresson, and P. Vandergheynst, "Convolutional neural networks on graphs with fast localized spectral filtering," in *Advances in Neural Information Processing Systems*, pp. 3844–3852, 2016.

[20] Y. Koren, "On spectral graph drawing," in *International Computing and Combinatorics Conference*, pp. 496–508, Springer, 2003.

[21] X. Hu, A. Lu, and X. Wu, "Spectrum-based network visualization for topology analysis," *IEEE Computer Graphics and Applications*, vol. 33, no. 1, pp. 58–68, 2013.

[22] P. Eades, Q. Nguyen, and S.-H. Hong, "Drawing big graphs using spectral sparsification," in *International Symposium on Graph Drawing and Network Visualization*, pp. 272–286, Springer, 2017.

[23] D. I. Shuman, S. K. Narang, P. Frossard, A. Ortega, and P. Vandergheynst, "The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular

domains," *IEEE Signal Processing Magazine*, vol. 30, no. 3, pp. 83–98, 2013.

[24] F. Galasso, M. Keuper, T. Brox, and B. Schiele, "Spectral graph reduction for efficient image and streaming video segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 49–56, 2014.

[25] A. Ortega, P. Frossard, J. Kovačević, J. M. Moura, and P. Vandergheynst, "Graph signal processing: Overview, challenges, and applications," *Proceedings of the IEEE*, vol. 106, no. 5, pp. 808–828, 2018.

[26] X. Zhao, Z. Feng, and C. Zhuo, "An efficient spectral graph sparsification approach to scalable reduction of large flip-chip power grids," in *Proc. of IEEE/ACM ICCAD*, pp. 218–223, 2014.

[27] L. Han, X. Zhao, and Z. Feng, "An Adaptive Graph Sparsification Approach to Scalable Harmonic Balance Analysis of Strongly Nonlinear Post-Layout RF Circuits," *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, vol. 34, no. 2, pp. 173–185, 2015.

[28] Z. Feng, "Spectral graph sparsification in nearly-linear time leveraging efficient spectral perturbation analysis," in *Design Automation Conference (DAC), 2016 53nd ACM/EDAC/IEEE*, pp. 1–6, IEEE, 2016.

[29] Z. Zhao and Z. Feng, "A spectral graph sparsification approach to scalable vectorless power grid integrity verification," in *Proceedings of the 54th Annual Design Automation Conference 2017*, p. 68, ACM, 2017.

[30] Z. Zhao, Y. Wang, and Z. Feng, "SAMG: Sparsified Graph Theoretic Algebraic Multigrid for Solving Large Symmetric Diagonally Dominant (SDD) Matrices," in *Proceedings of the 36th International Conference on Computer-Aided Design (ICCAD)*, ACM, 2017.

[31] Z. Feng, "Similarity-aware spectral sparsification by edge filtering," in *Design Automation Conference (DAC), 2018 55nd ACM/EDAC/IEEE*, IEEE, 2018.

[32] D. Spielman and S. Teng, "Spectral sparsification of graphs," *SIAM Journal on Computing*, vol. 40, no. 4, pp. 981–1025, 2011.

[33] Z. Feng, "Grass: Graph spectral sparsification leveraging scalable spectral perturbation analysis," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 39, no. 12, pp. 4944–4957, 2020.

[34] 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, (New York, NY, USA), pp. 678–687, ACM, 2017.

[35] 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)*, pp. 1159–1170, IEEE, 2022.

[36] 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*, pp. 598–611, 2021.

[37] 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)*, pp. 1–8, IEEE, 2020.

[38] A. Loukas and P. Vandergheynst, "Spectrally approximating large graphs with smaller graphs," in *International Conference on Machine Learning*, pp. 3243–3252, 2018.

[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, (New York, NY, USA), pp. 109:1–109:6, ACM, 2019.

[40] 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)*, vol. 65, no. 3, pp. 1–48, 2018.

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

[42] I. Bustany, A. B. Kahng, I. Koutis, B. Pramanik, and Z. Wang, "K-specpart: Supervised embedding algorithms and cut overlay for improved hypergraph partitioning," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2023.

[43] I. Bustany, A. B. Kahng, I. Koutis, B. Pramanik, and Z. Wang, "Specpart: A supervised spectral framework for hypergraph partitioning solution improvement," in *Proceedings of the 41st IEEE/ACM International Conference on Computer-Aided Design*, pp. 1–9, 2022.

[44] G. Karypis and V. Kumar, "Multilevel k-way hypergraph partitioning," *VLSI design*, vol. 11, no. 3, pp. 285–300, 2000.

[45] S. Schlag, T. Heuer, L. Gottesbüren, Y. Akhremtsev, C. Schulz, and P. Sanders, "High-quality hypergraph partitioning," *ACM Journal of Experimental Algorithmics*, vol. 27, pp. 1–39, 2023.

[46] A. Aghdaei, Z. Zhao, and Z. Feng, "Hypersf: Spectral hypergraph coarsening via flow-based local clustering," in *2021 IEEE/ACM International Conference On Computer Aided Design (ICCAD)*, pp. 1–8, ACM, 2021.

[47] A. Aghdaei and Z. Feng, "Hyperef: Spectral hypergraph coarsening by effective-resistance clustering," in *2022 IEEE/ACM International Conference On Computer Aided Design (ICCAD)*, pp. 1–9, ACM, 2022.

[48] F. R. Chung and F. C. Graham, *Spectral graph theory*. No. 92, American Mathematical Soc., 1997.

[49] 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*, vol. 11, no. 9, pp. 1074–1085, 1992.

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

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

[52] V. Osipov and P. Sanders, "n-level graph partitioning," in *Algorithms–ESA 2010: 18th Annual European Symposium, Liverpool, UK, September 6-8, 2010. Proceedings, Part I 18*, pp. 278–289, Springer, 2010.

[53] 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.

[54] N. Veldt, A. R. Benson, and J. Kleinberg, "Minimizing localized ratio cut objectives in hypergraphs," in *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 1708–1718, 2020.

[55] C. J. Alpert, "The ispd98 circuit benchmark suite," in *Proceedings of the 1998 international symposium on Physical design*, pp. 80–85, 1998.

[56] R. Liang, A. Agnesina, and H. Ren, "Medpart: A multi-level evolutionary differentiable hypergraph partitioner," in *Proceedings of the 2024 International Symposium on Physical Design*, pp. 3–11, 2024.

[57] K. E. Murray, S. Whitty, S. Liu, J. Luu, and V. Betz, "Titan: Enabling large and complex benchmarks in academic cad," in *2013 23rd International Conference on Field programmable Logic and Applications*, pp. 1–8, IEEE, 2013.

Hamed Sajadinia received the M.Sc. degree in telecommunication engineering from Iran University of Science and Technology, Tehran, Iran, in 2016. He is currently pursuing the Ph.D. degree in Electrical and Computer Engineering with Stevens Institute of Technology, Hoboken, USA.

His research interests include VLSI Design, Machine Learning, and Graph-related problems.

Ali Aghdaei received the M.Sc. degree in electrical and computer engineering from Michigan Technological University, Houghton, MI, in 2016, and the Ph.D. degree in electrical and computer engineering from Stevens Institute of Technology, Hoboken, NJ, in 2023. He is currently a postdoctoral scholar at the University of California, San Diego, where he is a member of the ABKGroup research team. His research interests include electronic design automation (EDA), very large-scale integration (VLSI) design, and spectral (hyper)graph algorithms.

He received the Best Ph.D. Dissertation Award from the Department of Computer Engineering at Stevens Institute of Technology in 2023. He has also served as a Technical Program Committee (TPC) member for the ACM/IEEE Great Lakes Symposium on VLSI (GLSVLSI) 2025 conference.

Zhuo Feng (S'03-M'10-SM'13) received the B.Eng. degree in information engineering from Xi'an Jiaotong University, Xi'an, China, in 2003, the M.Eng. degree in electrical engineering from National University of Singapore, Singapore, in 2005, and the Ph.D. degree in electrical and computer engineering from Texas A&M University, College Station, TX, in 2009. He is currently an associate professor at Stevens Institute of Technology. His research interests include high-performance spectral methods, very large scale integration (VLSI) and computer-aided design (CAD), scalable hardware and software systems, as well as heterogeneous parallel computing.

He received a Faculty Early Career Development (CAREER) Award from the National Science Foundation (NSF) in 2014, a Best Paper Award from ACM/IEEE Design Automation Conference (DAC) in 2013, and two Best Paper Award Nominations from IEEE/ACM International Conference on Computer-Aided Design (ICCAD) in 2006 and 2008. He was the principle investigator of the CUDA Research Center named by Nvidia Corporation. He has served on the technical program committees of major international conferences related to electronic design automation (EDA), including DAC, ASP-DAC, ISQED, and VLSI-DAT, and has been a technical referee for many leading IEEE/ACM journals in VLSI and parallel computing. In 2016, he became a co-founder of LeapLinear Solutions to provide highly scalable software solutions for solving sparse matrices and analyzing graphs (networks) with billions of elements, based on the latest breakthroughs in spectral graph theory.