# Spectral Hypergraph Partitioning Revisited (Extended Abstract)

Bodhisatta Pramanik
bodhi91@iastate.edu

Ioannis Koutis
ikoutis@njit.edu

**Abstract**

Most state-of-the-art hypergraph partitioning algorithms follow a multilevel approach that constructs a hierarchy of coarser hypergraphs that in turn is used to drive partition refinements. These partitioners are widely accepted as the current standard, as they have proven to be quite effective. On the other hand, spectral partitioners are considered to be less effective in cut quality, and too slow to be used in industrial applications. In this work, we revisit spectral hypergraph partitioning and we demonstrate that the use of appropriate solvers eliminates the running time deficiency; in fact, spectral algorithms can compute competing solutions in a fraction of the time needed by standard partitioning algorithms, especially on larger designs. We also introduce several novel modifications in the common spectral partitioning workflow, that enhance significantly the quality of the computed solutions. We run our partitioner on FPGA benchmarks generated by an industry leader, generating solutions that are directly competitive both in runtime and quality [1].

## 1 Introduction.

A *hypergraph* $H = (V, E, w, c)$ is defined as a set of vertices $V$ and hyperedges $E$ where each hyperedge is a subset of the vertex set $V$. $w$ represents the weight of the vertices where $w\colon V \rightarrow \mathbb{R}_{>0}$ and $c$ represents the weight of the hyperedges where $c\colon E \rightarrow \mathbb{R}_{>0}$. A *two-way partition* of $H$ is defined as a partition of the vertex set $V$ into two disjoint nonempty subsets $(S, V-S)$ such that an objective function is minimized.

## 2 Preliminaries.

In our spectral framework we approximate the hypergraph with a *proxy* graph, $G = (V', E')$, where each hyperedge is approximated by a weighted clique on its nodes. We term $G$ the *attraction graph*. A cut on the attraction graph is denoted by $cut_G(S, V-S)$, and serves as a proxy of the cut in the hypergraph. The weight on the vertices $w$ is expressed as the complete graph $J$ where $J(i,j) = w_i w_j \ \forall \ i \in V, j \in V$. We term graph $J$ the *repulsion graph*. A cut on the repulsion graph denoted by $cut_J(S, V-S) = w(S)w(V-S)$ quantifies the balance of the partition.

The objective function our framework minimizes is:

$$\Phi(G,J) = min_{S \subseteq V} \frac{cut_G(S, V-S)}{cut_J(S, V-S)} \tag{2.1}$$

Minimizing (2.1) implies minimizing the cut on the attraction graph while maximizing the cut on the repulsion graph. Generally, this favors balanced cuts. We approximately optimize $\Phi(G,J)$ by solving a non-standard generalized eigenvalue problem using the *Laplacians* on $G$ and $J$:

$$Gx = \lambda Jx \tag{2.2}$$

where $\lambda$ is the eigenvalue and $x$ is the equivalent eigenvector. The coordinates of $x$ represents a linear ordering of the vertex set $V$ which is then partitioned using some heuristic. This procedure is widely regarded as *spectral partitioning*.

## 3 Related Work.

Hypergraph partitioning is a well-studied problem with much research going into it over the last decade. The most widely used heuristic is the multilevel paradigm where a sequence of coarser hypergraphs is constructed, thus forming a hierarchy. hMetis[2], KaHyPar[1] and PaToH[3] are the most widely used partitioners in the industry. All these partitioners follow the multilevel paradigm.

Hypergraph spectral partitioning heuristics have also been studied in the literature. Several heuristics have been proposed, that in general fail to generate high quality partitions compared to the likes of hMetis, PaToH, KaHyPar. In general spectral partitioning is considered to be runtime expensive and of inferior partitioning quality. Our work aims to break this notion and point towards a potential paradigm shift in hypergraph partitioning.

[1] **In this work we use benchmarks and baseline software provided by an industry leader, subject to final permission that will be communicated to the ACDA program committee in a timely manner.**

## 4 Proposed Spectral Framework.

Our spectral framework is derived from classical spectral methods but adds substantial new ideas. The complete algorithm is shown in Algorithm 1. A core component is the computation of the smallest eigenvector of $G_L x = \lambda B_L x$, where $G_L, B_L$ are the Laplacians of the graphs defined in Section 2. The Laplacians $G_L, B_L$ are dense, but they are never explicitly calculated. Instead, we are only computing matrix-vector multiplications with them, in time linear in the size of the hypergraph.

In order to solve the eigenvalue problem we use LOBPCG[5] with CMG[6] as a preconditioner. To construct the preconditioner we work with a spectral sparsifier of $G$. The low stretch spanning tree (LST) is computed using [8]. For refinement purpose we implement our own single shot refinement where we identify nodes on the cut which can close a hyperedge while not violating the area constraints and move them as a group to the other partition. This technique seems to work well with our spectral partitioner. In addition we use FM heuristics [7] to improve the cut even further.

**Algorithm 1:** Modified Spectral Partitioning

**Input:** Hypergraph $H = (V, E, w, c)$, Area Constraints, $\tau$

**Output:** Bipartition P = (S, V-S)

1 Find the smallest eigenvector $\mu$ of .
2 $G'$ : Reweigh edge each edge $(u, v)$ in $G$ with $|\mu_i - \mu_j|$.
3 Compute Low Stretch Spanning Tree (LST) of reweighed graph G'.
4 Sweep LST for best cut.
5 Refine best cut using local refinement heuristics.
6 Remove cut hyperedges and all $E_j \geq \tau$ from H.
7 Contract connected components in H and build coarse hypergraph $H_c = (V_c, E_c, w_c, c_c)$.
8 Run steps 1-8 on $H_c$ and obtain partition $P_c = (S_c, V_c - S_c)$.
9 Project $P_c$ to H: $P = (S, V - S)$.
10 Refine $P$ using local refinement heuristics.

## 5 Results and Conclusion.

All code has been written in Julia. We run our partitioner on an Intel i7-9700, 2.67 GHz, 6 cores processor. B-Cut and B-Time are obtained with a hMetis-like industrial partitioner.

B-Cut and B-Time in the table denotes the baseline partitioner cut and runtime respectively. S-Cut and S-Time denotes the Spectral partitioner cut and runtime. The B-Cut and S-Cut solutions have similar balance.

| Benchmark | B-Cut | B-Time(s) | S-Cut | S-Time(s) |
|---|---|---|---|---|
| B1 | 1312 | 5.76 | 1274 | 14.28 |
| B2 | 23794 | 122.6 | 26870 | 51.75 |
| B3 | 9866 | 23.27 | 9132 | 27.65 |
| B4 | 8973 | 43.51 | 12549 | 24.99 |
| B5 | 2552 | 4.75 | 2591 | 12.23 |
| B6 | 2721 | 5.45 | 2573 | 13.01 |
| B7 | 782 | 19.48 | 764 | 18.28 |

Table 1: Benchmark Results

We believe we can attain more faster speeds with optimized C++ and use parallelism, which is already maximized in the baseline partitioner.

## References


[1] Tobias Heuer and Nikolai Maas and Sebastian Schlag, *Multilevel Hypergraph Partitioning with Vertex Weights Revisited*, 2021, 2102.01378, arXiv.

[2] George Karypis and Vipin Kumar *Multilevel Hypergraph Partitioning: Applications in VLSI Domain*, VLSI Design, Vol. 11, No. 3, pp. 285-300, 2000.

[3] Çatalyürek Ü., Aykanat C. (2011), *PaToH (Partitioning Tool for Hypergraphs)*. In: Padua D. (eds) Encyclopedia of Parallel Computing. Springer, Boston, MA.

[4] Jeff Bezanson, Alan Edelman, Stefan Karpinski, and Viral B. Shah, *Julia: A Fresh Approach to Numerical Computing*, SIAM Rev., 59(1), 65–98. (34 pages).

[5] Andrew V. Knyazev, *Toward the Optimal Preconditioned Eigensolver: Locally Optimal Block Preconditioned Conjugate Gradient Method*, SIAM J. Sci. Comput., 23(2), 517–541. (25 pages).

[6] I Koutis, GL Miller, D Tolliver, *Combinatorial preconditioners and multilevel solvers for problems in computer vision and image processing* , Computer Vision and Image Understanding, Vol 115, Issue 12, 2011, Pages 1638-1646, ISSN 1077-3142.

[7] C.M. Fiduccia and R. M. Mathheyses. *A linear time heuristic for improving network partitions*. In Proc. 19th IEEE Design Automation Conf, pages 438–446, May 1984.

[8] Noga Alon, Richard M. Karp, David Peleg and Douglas West, *A Graph-Theoretic Game and Its Application to the k-Server Problem*, Society for Industrial and Applied Mathematics, Vol 24, Feb 1995.