ComPart: Community-Guided Post-Coarsening for High-Quality Hypergraph Partitioning

Yugao Zhu, Zhicheng Guo, Yuchao Wu, Mengming Li, Jing Wang, Zhiyao Xie*
Hong Kong University of Science and Technology
{yzhuel, zguobx, ywu092, mengming.li, jwangjw}@connect.ust.hk, eezhiyao@ust.hk

Abstract

Hypergraph partitioning is a critical step in the design of complex embedded systems, essential for optimizing task mapping on heterogeneous MPSoCs and enabling multi-FPGA prototyping. Many existing methods rely on community detection to identify modules with dense internal and sparse external connections, typically utilizing them to constrain the coarsening phase—a widely adopted paradigm. In this work, we propose ComPart, a generalized framework that integrates diverse community detection methods to uncover high-quality clusterings throughout the post-coarsening stages (i.e., initial partitioning and uncoarsening). These discovered clusterings serve as distinct structural guides, enabling the refinement process to identify superior partitioning solutions. Our framework offers two key advantages: (1) it establishes a new paradigm that leverages community structures detected during uncoarsening to escape local optima and explore globally meaningful solution subspaces, transcending the limitations of standard local refinements; and (2) it flexibly accommodates both existing and future community detection methods. Furthermore, we theoretically generalize locally-dense decomposition—originally from graphs—to the hypergraph domain. We provide the formal extension and necessary proofs to apply this technique to hypergraphs, marking its first application in hypergraph partitioning. Specifically, we utilize this rigorously derived decomposition to guide the initial partitioning phase toward superior starting points. Experimental results on standard benchmarks demonstrate that our method consistently outperforms state-of-the-art methods in solution quality.

ACM Reference Format:
Yugao Zhu, Zhicheng Guo, Yuchao Wu, Mengming Li, Jing Wang, Zhiyao Xie. 2026. ComPart: Community-Guided Post-Coarsening for High-Quality Hypergraph Partitioning. In *63rd ACM/IEEE Design Automation Conference (DAC '26), July 26–29, 2026, Long Beach, CA, USA.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3770743.3804131

1 Introduction

Hypergraph partitioning serves as a fundamental algorithmic kernel in the design and verification of modern embedded systems. It is essential not only for task mapping on heterogeneous MPSoCs during hardware/software co-design[12, 27], but also for multi-FPGA prototyping [19, 26] in logic emulation flows. In both scenarios, modeling system components and dependencies as hypergraphs allows for minimizing inter-device communication and satisfying strict resource constraints. Over the past decades, a wide range of methods have been developed to address these combinatorial challenges, including spectral approaches [2, 6, 7], max-flow/min-cut formulations [14], deep learning-based methods [9, 20], and multi-level frameworks.

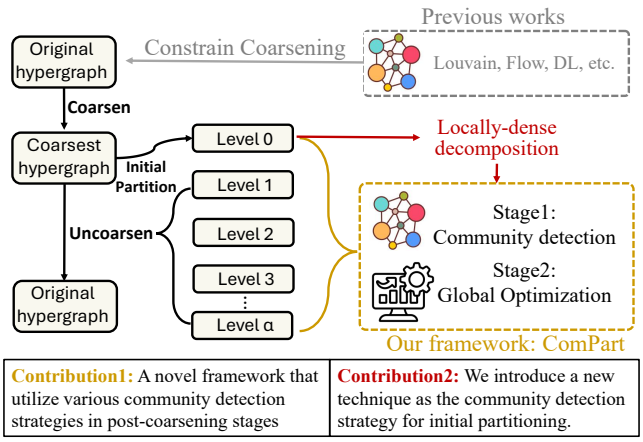

Figure 1: Framework Comparison between our method ComPart and prior works.

Among these methodologies, the multi-level framework has established itself as the dominant paradigm for high-performance hypergraph partitioning, underpinning leading solvers like PaToH [8], hMetis [17], and KaHyPar [25]. This framework typically consists of three key phases: (1) **Coarsening**: densely connected vertices are iteratively merged to construct a hierarchy of progressively smaller hypergraphs until the problem size allows for direct partitioning; (2) **Initial partitioning**: a initial solution is computed on the coarsest hypergraph; (3) **Uncoarsening and refinement**: the solution is projected back to finer levels, where local heuristics such as Fiduccia-Mattheyses (FM) [13] or Kernighan-Lin (KL) [18] are applied to iteratively improve solution quality.

Multi-level frameworks have undergone continuous enhancement, with one of the most prominent directions being the integration of *community detection* strategies. These methods typically introduce community detection **for coarsening**, where the coarsening process is restricted within each detected community. For example, KaHyPar adopts the Louvain method [4] to guide the coarsening process, while SHyPar [24] employs a flow-based community detection approach to optimize hypergraph local conductance (HLS). Both the V-cycle [17] and recombination methods [1, 3, 22] treat candidate partitions as distinct communities, restricting coarsening within their boundaries. Similarly, GenPart [9] utilizes a graph convolutional network (GCN)-based generative model to produce initial solutions before executing the V-cycle.

As illustrated in Fig. 1, while almost all prior works concentrate on utilizing community detection to guide the coarsening process, they largely neglect its potential in other stages. In this work, we unprecedentedly target these neglected stages. To this end,

*corresponding author

DAC '26, Long Beach, CA, USA

ACM ISBN 979-8-4007-2254-7/2026/07
https://doi.org/10.1145/3770743.3804131

we propose **ComPart**, a novel framework that leverages various community detection methods throughout the **post-coarsening phases** (i.e., initial partitioning and uncoarsening). This design enables the integration of a wide range of existing community-based techniques—spectral, modularity-based, deep learning, and beyond—thereby generalizing previous coarsening-centric approaches and introducing a new, flexible paradigm centered on the post-coarsening process.

Specifically, this paradigm allows us to transcend the limitations of traditional refinement. Instead of relying solely on vertex-wise moves like FM refinement—which often yields only marginal improvements and becomes trapped in local optima—we introduce a *structure-aware* optimization strategy. We **periodically** identify community structures within the post-coarsening phases and formulate their reassignment as an integer linear programming (ILP) problem [16]. This allows us to optimize the movement of entire community fragments simultaneously. By shifting from a local, node-centric view to a global, community-centric perspective, our approach facilitates large-scale adjustments, effectively escaping the local optima that limit existing frameworks.

In addition, we introduce a theoretically grounded innovation at the coarsest level by generalizing *locally-dense decomposition* [11, 28]—originally a graph-theoretic technique—to the hypergraph domain. We provide the formal extension and necessary proofs to adapt this concept to hypergraphs, marking its first application in hypergraph partitioning. Crucially, unlike heuristic clustering methods, this decomposition is formulated as a **convex optimization problem**, guaranteeing convergence to a theoretically optimal solution that reveals hierarchically nested dense substructures. Leveraging these mathematically rigorous structural insights, we guide the initial partitioning toward superior starting points.

The main contributions of this work are summarized as follows:

(1) We propose a *generalized framework*, ComPart, that integrates diverse community detection methods into the post-coarsening stages to uncover high-quality clustering structures. Unlike traditional approaches that rely solely on local refinements, our framework leverages these identified communities to guide global solution adjustments, allowing the algorithm to escape local optima. This establishes a novel paradigm that transcends the limitations of local optimization, providing new insights into the effective utilization of community detection in hypergraph partitioning.
(2) We generalize the *locally-dense decomposition* technique from the graph to the hypergraph domain. We provide the theoretical extension required to adapt this rigorous method to hyperedges, marking its first application in hypergraph partitioning. By identifying hierarchically nested dense structures within the coarsest hypergraph, this decomposition serves as a principled guide for generating superior initial solutions.
(3) We conduct extensive experiments on standard benchmark datasets, comparing our method against both mainstream solvers and recent state-of-the-art approaches. The results demonstrate that our algorithm consistently achieves superior solution quality. Furthermore, comprehensive ablation studies validate the efficacy of each component in our framework. Additionally, we perform a comparison of different community detection strategies to evaluate their specific impact on partitioning performance.

2 Problem Formulation

Given a hypergraph $H(V, E)$, where V and E denote the sets of vertices and hyperedges, respectively, each vertex $v \in V$ and hyperedge $e \in E$ is associated with a positive weight w_v and w_e. The hypergraph partitioning problem[1] aims to determine a partition scheme S that divides V into k disjoint blocks $V_1, V_2, \ldots, V_k$, satisfying $V_i \cap V_j = \emptyset$ for all $i \neq j$ and $\bigcup_{i=0}^{k-1} V_i = V$. Each block V_i must satisfy the balance constraint

$$W_{V_i} \leq (1 + \epsilon) \cdot \left\lceil \frac{W_V}{k} \right\rceil,$$

where $W_V = \sum_{v \in V} w_v$, $W_{V_i} = \sum_{v \in V_i} w_v$, and ϵ is the imbalance factor. Under this constraint, the objective is to minimize the cut size of the partition:

$$\text{cutsize}_H(S) = \sum_{e \not\subseteq V_i \text{ for all } V_i \in S} w_e$$

3 Methodology

In this section, we first present our new framework, ComPart, in which multiple community detection algorithms are integrated into the post-coarsening phases, as detailed in Section 3.1. Subsequently, in Section 3.2, we formally generalize *locally-dense decomposition* from graphs to hypergraphs, extending both its theoretical properties and algorithmic formulation. We leverage this decomposition as a specialized community detection strategy to significantly improve the quality of initial partitioning.

3.1 Our Framework: ComPart

While prior community-aware frameworks predominantly utilize community structure to restrict coarsening, they significantly underutilize its potential in post-coarsening stages. Consequently, ComPart introduces a fundamental paradigm shift: instead of guiding coarsening, we unprecedentedly target the post-coarsening phases—specifically, initial partitioning and uncoarsening. The overall architecture of ComPart, highlighting this novel approach, is depicted in Figure 2.

Our framework is built upon the n-level architecture of `KaHyPar`. Here, n denotes the node count of the original hypergraph: each coarsening step merges one pair of nodes, and each uncoarsening step restores one pair while performing local refinement. During the uncoarsening phase, we perform a two-stage operation at α distinct levels. These levels are triggered whenever the number of nodes reaches a value defined by the set:

$$\left\{ n_c^{1-1/\alpha} n^{1/\alpha},\ n_c^{1-2/\alpha} n^{2/\alpha},\ \ldots,\ n_c^{1/\alpha} n^{1-1/\alpha},\ n \right\},$$

where n_c is the node count of the coarsest hypergraph. Intuitively, as the node count increases proportionally, the topological details evolve smoothly across levels, which aligns with the "Kronecker product" principle [15]; thus, applying community detection at these intervals yields more effective results. The two stages are:

(1) Applying specified community detection algorithms to the hypergraph at this level;
(2) Considering potential moves for *sub-communities*, i.e., the portions of a community created by cuts.

[1] Our formulation follows that of established tools such as KaHyPar, PaToH, and the recent hMETIS release (v2.0-pre1).

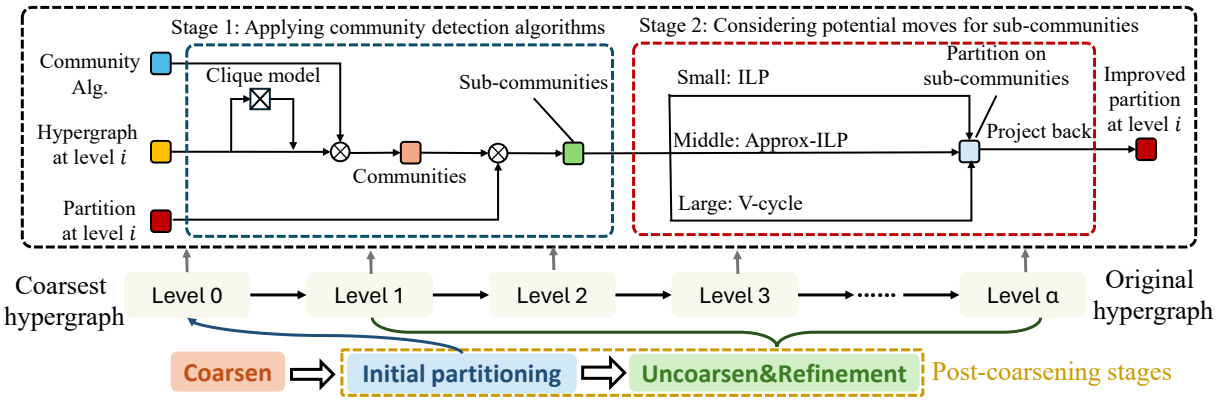

Figure 2: Overview of the proposed ComPart framework. Locally-dense decomposition is used for community detection in the coarsest hypergraph (i.e., level 0).

Stage ❶: Applying community detection algorithms. While some community detection algorithms operate directly on hypergraphs, many are restricted to standard graphs. To leverage these, we convert the hypergraph into a graph using the **clique model**.

In this model, each hyperedge e is selectively transformed into a clique by connecting its vertices pairwise with edge weights of $w_e/(|e|-1)$. Specifically, hyperedges with size $n < S_1$ are expanded directly, while those within the range $S_1 \le n < S_2$ are placed into a temporary set Q, and larger hyperedges ($n \ge S_2$) are ignored (with $S_1 = 100$ and $S_2 = 200$ in our experiments). If the resulting graph is not connected[2], we iteratively apply clique expansion to the hyperedges stored in the set Q, prioritizing those with the smallest number of vertices. We continue this process until the graph becomes fully connected. If expanding the hyperedges in Q yields no further reduction in the number of connected components, we introduce edges with minimal weight (e.g., 10^{-6}) to bridge nodes from different components until the entire graph is connected. This approach allows us to preserve the structural information of the original hypergraph as much as possible, while ensuring the conversion is completed within a reasonable time complexity.

We apply community detection to the hypergraph or its graph representation using four `igraph` methods: Label Propagation [23], Walktrap [21], Louvain [4], and Fast Greedy [10]. The resulting clusters are categorized as *intra-partition communities* (contained within a single partition) or *edge communities* (spanning multiple partitions). Edge communities are intersected with partition boundaries to form smaller segments. These segments, together with the intra-partition communities, are treated as candidate *sub-communities* for potential moves in stage 2.

Stage ❷: Considering potential moves for sub-communities. In this stage, we refine the partition assignment of these sub-communities. Several methods can be employed, such as Integer Linear Programming (ILP), V-cycle, or local heuristic algorithms (e.g., the FM algorithm)[3].

The ILP formulation of the hypergraph partitioning problem can be expressed as follows. We define binary decision variables $z_{u,i} \in \{0,1\}$ for each vertex u and partition block P_i ($i \in \{0,\dots,k-1\}$), and $h_{e,i} \in \{0,1\}$ for each hyperedge e and partition block P_i. Here, $z_{u,i} = 1$ indicates that vertex u is assigned to block P_i, while $h_{e,i} = 1$ denotes that all vertices of hyperedge e are fully contained within block P_i. The constraints are formulated as:

- $\sum_{i=0}^{k-1} z_{u,i} = 1, \quad$ for all $u \in V$;
- $h_{e,i} \le z_{u,i}, \quad$ for all $e \in E$, and for every $u \in e$;
- $\sum_{u \in V} w_u z_{u,i} \le (1+\epsilon)\left\lceil \frac{W_V}{k} \right\rceil, \quad$ for all $i \in \{0,\dots,k-1\}$;

The optimization objective is to maximize the total weight of uncut hyperedges:

$$\text{maximize} \quad \sum_{e \in E} \sum_{i=0}^{k-1} w_e \, h_{e,i}.$$

Given that ILP inherently exhibits high computational complexity, we adopt a trade-off strategy between runtime and solution quality based on the size of the clustered hypergraph. Specifically, for *small* hypergraphs, we directly solve the ILP using the current partitioning scheme as an initial solution to accelerate convergence; for *medium* hypergraphs, we similarly initialize with the current partitioning but terminate early once the optimality gap drops below 1%; and for *large* hypergraphs, we substitute the ILP with the V-cycle in KaHyPar to minimize the cut size efficiently.

3.2 Locally-dense decomposition in initial partitioning

We introduce the first application of *locally-dense decomposition* to hypergraphs. This adaptation yields a hierarchy of nested, provably denser substructures. Distinct from heuristic approaches, our decomposition is formulated as a **convex optimization problem with convergence guarantees**, offering a mathematically rigorous foundation for initialization.

To apply this technique to hypergraph partitioning, we generalize the existing formulation [11, 28] in two key aspects: (1) extending the decomposition from *graphs* to *hypergraphs*, and (2) generalizing node weights from *unit weights* to *positive real weights*. We provide rigorous proofs for both generalizations to ensure the theoretical properties are preserved. Given a hypergraph $\mathcal{H} = (\mathcal{V}, \mathcal{E}, \mathcal{W}_E, \mathcal{W}_V)$, we begin with the following problem definition:

Problem 1 (Locally-dense Decomposition Problem on Hypergraph). *Given an hypergraph $\mathcal{H} = (\mathcal{V}, \mathcal{E}, \mathcal{W}_E, \mathcal{W}_V)$, find its locally-dense decomposition $\emptyset = B_0 \subsetneq B_1 \subsetneq \dots \subsetneq B_\gamma = \mathcal{V}$, such that*

[2]Disconnected graphs are incompatible with some methods.
[3]We employ the first two methods to pursue higher solution quality.

B_i is the maximal[4] *subgraph strictly containing B_{i-1} that satisfies:* $B_i = \arg\max_{S \supsetneq B_{i-1}} \frac{\mathcal{W}_E(S) - \mathcal{W}_E(B_{i-1})}{\mathcal{W}_V(S \setminus B_{i-1})}$

Here, $\mathcal{W}_E(S)$ and $\mathcal{W}_E(B_{i-1})$ represent the sum of weights of hyperedges whose endpoints are all contained within the set S and B_{i-1}, respectively. $S \setminus B_{i-1}$ denotes the set of vertices in S that are not in B_{i-1}, and $\mathcal{W}_V(S \setminus B_{i-1})$ represents the sum of weights of these vertices. We introduce a quadratic programming formulation associated with the studied problem. Theorems 1 and 2 characterize their key properties, serving as a basis for Theorem 3, which demonstrates the equivalence.

$$\begin{aligned} \text{minimize} \quad & \sum_{v \in \mathcal{V}} w_v \ell_v^2 \\ \text{subject to} \quad & \sum_{v \in e} f_e(v) = w_e \qquad \forall e \in \mathcal{E} \\ & \ell_v = \frac{\sum_{e \ni v} f_e(v)}{w_v} \qquad \forall v \in \mathcal{V} \\ & f_e(v) \geq 0 \qquad \forall e \in \mathcal{E}, \forall v \in e \end{aligned} \tag{1}$$

Theorem 1. *Given an hypergraph $\mathcal{H} = (\mathcal{V}, \mathcal{E}, \mathcal{W}_E, \mathcal{W}_V)$, its locally-dense decomposition is unique.*

The uniqueness is proved via mathematical induction. Specifically, the union of the smallest differing sets B_i^1 and B_i^2 from two decompositions satisfies the argmax condition, which contradicts maximality. The detailed derivation is omitted for brevity.

Theorem 2. *For an optimal solution $(\boldsymbol{f}^*, \boldsymbol{\ell}^*)$ and an hyperedge. For any node $u \in e$, if there is a node $v \in e$ satisfying ${\ell_u}^* > {\ell_v}^*$, then ${f_e}^*(u) = 0$.*

Proof of Theorem 2. Define $f^*(u) := \sum_{e \ni u} f_e(u)$ and $\ell_u^* = f^*(u)/w_u$. Assume $\ell_u^* > \ell_v^*$ and $f_e^*(u) > 0$. This implies $w_v f^*(u) > w_u f^*(v)$. Consider shifting $d \in (0, f_e^*(u)]$ from $f_e^*(u)$ to $f_e^*(v)$. The new total flows are $f'(u) = f^*(u) - d$ and $f'(v) = f^*(v) + d$. We show this shift reduces the objective, which is a contradiction:

$$\begin{aligned} & w_u(\ell_u')^2 + w_v(\ell_v')^2 < w_u(\ell_u^*)^2 + w_v(\ell_v^*)^2 \\ \Leftrightarrow\ & \frac{(f^*(u) - d)^2}{w_u} + \frac{(f^*(v) + d)^2}{w_v} < \frac{f^*(u)^2}{w_u} + \frac{f^*(v)^2}{w_v} \\ \Leftrightarrow\ & \frac{-2df^*(u) + d^2}{w_u} + \frac{2df^*(v) + d^2}{w_v} < 0 \\ \Leftrightarrow\ & d^2(w_u + w_v) + 2d(w_u f^*(v) - w_v f^*(u)) < 0 \\ \Leftrightarrow\ & d(w_u + w_v) < 2(w_v f^*(u) - w_u f^*(v)) \end{aligned} \tag{2}$$

Since $w_v f^*(u) > w_u f^*(v)$, the right-hand side (RHS) is a positive constant. As $d \to 0^+$, the left-hand side (LHS) approaches 0. Thus, a sufficiently small $d > 0$ always exists that satisfies the inequality. This contradicts the optimality. □

Theorem 3. *For any node $u \in B_i \setminus B_{i-1}$, let*

$$\lambda_u = \lambda_i \coloneqq \frac{\mathcal{W}_E(B_i) - \mathcal{W}_E(B_{i-1})}{\mathcal{W}_V(B_i \setminus B_{i-1})},$$

the optimal solution $\boldsymbol{\ell}^*$ is unique and is given by ${\ell_v}^* = \lambda_v$ for each node v. And $\lambda_1 > \lambda_2 > ... > \lambda_\gamma$.

[4]Maximality means that no other set strictly containing B_i satisfies the argmax condition.

Algorithm 1 BCD for Load Balancing via Water-Filling

Input: Hypergraph $G = (\mathcal{V}, \mathcal{E})$, vertex weights $\{w_v\}_{v \in \mathcal{V}}$, hyperedge weights $\{w_e\}_{e \in \mathcal{E}}$
Output: The final allocation $f = \{f_e(v) \mid e \in \mathcal{E}, v \in e\}$

Initialize: Create a feasible allocation f
for each hyperedge $e \in \mathcal{E}$ **do**
  $\mathcal{W}_V(e) \leftarrow \sum_{v \in e} w_v$ ▷ Total vertex weight in e
  for each vertex $v \in e$ **do**
    $f_e(v) \leftarrow w_e \cdot (w_v / \mathcal{W}_V(e))$ ▷ Proportional allocation

repeat
  $f_{\text{prev}} \leftarrow f$ ▷ Store allocation from previous iteration
  for each hyperedge $e \in \mathcal{E}$ (in a fixed order) **do**
    ▷ Locally optimize the allocation for hyperedge e
    $\{f_e(v)\}_{v \in e} \leftarrow \text{WaterFilling}(e, w_e, \{w_v\}_{v \in e}, \{f_{e'}\}_{e' \neq e})$
  until convergence ▷ e.g., $f = f_{\text{prev}}$ or $||f - f_{\text{prev}}|| < \epsilon$
return f

Proof of Theorem 3. Suppose $\boldsymbol{\ell}^*$ is one optimal solution, we sort them in descending order by $\ell_{u_1}^* \geq \ell_{u_2}^* \geq ... \geq \ell_{u_{|\mathcal{V}|}}^*$.Suppose the first inequality is the t-th inequality, then

$$\ell_{u_1}^* = \ell_{u_2}^* = ... = \ell_{u_t}^* \Rightarrow \frac{f^*(u_1)}{w_{u_1}} = \frac{f^*(u_2)}{w_{u_2}} = ... = \frac{f^*(u_t)}{w_{u_t}}$$

Set $S := \{u_1, u_2, ..., u_t\}$, we claim $S = B_1$. Since $B_0 = \emptyset$,

$$\begin{aligned} & \frac{\mathcal{W}_E(S) - \mathcal{W}_E(B_0)}{\mathcal{W}_V(S) - \mathcal{W}_V(B_0)} = \frac{\mathcal{W}_E(S)}{\mathcal{W}_V(S)} \\ & \leq \frac{f^*(u_1) + f^*(u_2) + ... + f^*(u_t)}{w_{u_1} + w_{u_2} + ... + w_{u_t}} = \ell_{u_1}^* = \ell_{u_2}^* = ... = \ell_{u_t}^* \end{aligned} \tag{3}$$

The first inequality holds because, on one hand, $\mathcal{W}_V(S) = w_{u_1} + w_{u_2} + \cdots + w_{u_t}$, and on the other hand, the weights of all hyperedges with endpoints entirely in S are totally distributed to $f^*(u_1), \ldots, f^*(u_t)$, which implies $\mathcal{W}_E(S) \leq f^*(u_1) + f^*(u_2) + \cdots + f^*(u_t)$. The inequality is for the general case, but due to Theorem 2, equality holds here. This is because any hyperedge e with endpoints not all in S does not assign a non-zero value to $f^*(u_1), \ldots, f^*(u_t)$, i.e., $f_e^*(u) = 0$ for any $u \in \{u_1, \ldots, u_t\}$.

Therefore, we assert that $S = B_1$. This is because, by the inequality, no other subgraph (that is not a proper subset of S) has a ratio of the sum of edge weights to the sum of vertex weights exceeding

$$\frac{f^*(u_1) + f^*(u_2) + \cdots + f^*(u_t)}{w_{u_1} + w_{u_2} + \cdots + w_{u_t}}.$$

On the other hand, the equality

$$\frac{\mathcal{W}_E(S)}{\mathcal{W}_V(S)} = \frac{f^*(u_1) + f^*(u_2) + \cdots + f^*(u_t)}{w_{u_1} + w_{u_2} + \cdots + w_{u_t}}$$

holds, so $S = B_1$. Furthermore, by Theorem 1, B_1 is unique, so this set $\{u_1, \ldots, u_t\}$ is unique and corresponds exactly one-to-one with the nodes of B_1. For the nodes in $B_2 \setminus B_1, \ldots, B_\gamma \setminus B_{\gamma-1}$, we can use mathematical induction to prove the result. Therefore, Theorem 3 holds. □

Theorems 1 and 3 establish the uniqueness and quadratic formulation of the decomposition. Crucially, Theorem 2 implies that optimal allocation occurs only when nodes receiving weight have

the minimum load. This motivates our Block-Coordinate Descent (BCD) method (Algorithm 1), which employs the **water-filling** algorithm [5] to drive each hyperedge to its local optimum. Initialized with proportional allocations, the algorithm iteratively updates hyperedge weights via water-filling until convergence. Convergence to the global optimum of (1) is guaranteed by two key properties:

(1) The objective function,

$$\sum_{v\in\mathcal{V}} w_v \ell_v^2 = \sum_{v\in\mathcal{V}} w_v \left(\frac{\sum_{e\ni v} f_e(v)}{w_v}\right)^2 = \sum_{v\in\mathcal{V}} \frac{1}{w_v}\left(\sum_{e\ni v} f_e(v)\right)^2$$

is continuously differentiable, as it is a quadratic polynomial of the variables $f_e(v)$.

(2) The objective is **blockwise strictly convex**. When optimizing for a single hyperedge e (a block), the sub-problem's objective (ignoring constants) is:

$$\sum_{v\in e} \frac{1}{w_v}(f_e(v) + C_v)^2 \quad \text{where } C_v = \sum_{e'\ni v, e'\neq e} f_{e'}(v) \text{ is constant.}$$

This function is strictly convex, as its Hessian matrix H is positive definite:

$$H = \text{diag}(2/w_{v_1}, 2/w_{v_2}, \ldots, 2/w_{v_k})$$

As mentioned earlier, the key advantage of this technique lies in its ability to accurately characterize the density hierarchy of the hypergraph, such that subgraphs with different density levels are distinctly separated (assigned to different density layers B_i and B_j). Since the coarsening process typically merges locally dense structures into single nodes, the locally-dense decomposition provides a global perspective and complementary guidance for the coarsest hypergraph in the initial partitioning phase. We regard Algorithm 1 as a community detection method, and employ the two-stage procedure in Section 3.1 to improve our initial solution.

4 experiments

The structure of this chapter is organized as follows. In Section 4.1, we introduce the experimental setup. In Section 4.2, we compare our algorithm with the leading methods, including *hMetis*, *KaHyPar*, *SpecPart*, and *SHyPar*, on both the Titan23 and ISPD98 benchmarks.

4.1 Experimental setup

(1) In Sections 4.2.1 and 4.2.2, we evaluate ComPart on the Titan23 and ISPD98 benchmarks against leading algorithms, including *hMetis*, *KaHyPar*, *SpecPart*, and *SHyPar*. We employ four iterations ($\alpha = 4$) of the `Fast Greedy` algorithm for community detection. To ensure fair comparison, ComPart, *hMetis*, and *KaHyPar* were restricted to **the same total execution time**, reporting the best result found. Results for *SpecPart* and *SHyPar* ($k = 2$) are cited from their respective papers. For *SHyPar* with $k > 2$, we executed the source code but limited the comparison to ISPD98 due to excessive runtime on Titan23 (over 20 hours for medium-sized datasets). All experiments enforce a maximum imbalance of 2% per partition (i.e., $\epsilon = 0.02k$).

(2) In Section 4.2.3, we perform an ablation study on the two components of our method: community detection in uncoarsening and locally-dense decomposition, and compare the results with the performance of KaHyPar and ShYPar.

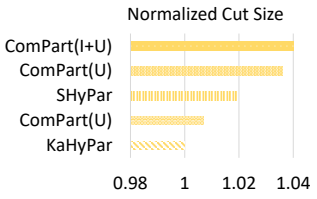

Figure 3: Ablation study.

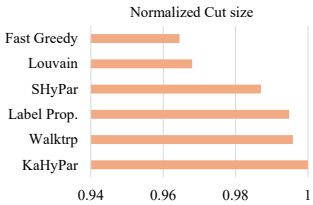

Figure 4: Different Community Detection Strategies.

(3) In Section 4.2.4, we compare the performance of different community detection strategies applied during the uncoarsening phase, and also compare our results with KaHyPar and ShYPar.

Our implementation is based on `KaHyPar`, using the configuration file *cut_kKaHyPar_sea20.ini*. All experiments were executed on a single core of an `Intel(R) Xeon(R) Gold 6438Y+ CPU @ 2.00GHz` system running `Ubuntu 22.04.5 LTS` with `512GB RAM`. The implementation was compiled from source using `g++ 11.4.0` under the `C++17` standard.

4.2 Comprehensive performance on standard benchmarks

4.2.1 ComPart against Baselines on Titan23 Benchmark. Table 1 compares our method, ComPart, with the baselines on the Titan23 benchmark, indicating that our algorithm outperforms all existing algorithms on all setups. The Titan23 benchmark features many large hyperedges (in terms of the number of nodes). Overall, the best baseline is SHyPar. When $k = 2$, we improve by 3.62% over the base partitioner KaHyPar and 0.87% over the best baseline (SHyPar). When $k = 3$, we improve by 5.74% over KaHyPar. When $k = 4$, we improve by 4.50% over KaHyPar. In general, our lead is larger for $k = 3$ and $k = 4$ than for $k = 2$. This suggests that existing works are generally stronger in the bipartition domain ($k = 2$) but weaker for $k > 2$. Furthermore, our algorithm achieves a greater overall improvement on Titan23 than on ISPD98. We attribute this to our community detection strategy, which can effectively handle cases with large hyperedge sizes better than these baselines.

4.2.2 ComPart against Baselines on ISPD98 Benchmark. Table 2 presents the comparison results of our method, ComPart, against the baselines on the ISPD98 benchmark. The hyperedge sizes in the ISPD98 benchmark are typically smaller. Overall, the best baseline is still SHyPar. Our lead is greater for $k = 3$ and $k = 4$ compared to $k = 2$. When $k = 2$, we achieve a 2.10% lead over the base partitioner KaHyPar and a 0.17% lead relative to SHyPar. When $k = 3$, we achieve a 4.11% lead over KaHyPar and a 2.68% lead over SHyPar. When $k = 4$, we achieve a 1.14% lead over KaHyPar and a 2.70% lead over SHyPar.

4.2.3 Ablation Study. Fig. 3 exhibits the ablation study to evaluate the contributions of two key components of `ComPart`: Locally-Dense Decomposition in Initial Partitioning (I) and Community Detection in Uncoarsening (U). We also compare our results with `KaHyPar` and `SHyPar`. The results show that 'U' is the primary contributor to the performance gain. Specifically, the 'U' component alone achieves a 3.6% improvement over `KaHyPar`, while the 'I' component alone provides a 0.7% improvement.

Table 1: Experimental results on Titan23 benchmark.

Design	$k=2$					$k=3$				$k=4$			
	hMETIS	**K-Spec**	**KaHyPar**	**SHyPar**	**Ours**	**hMETIS**	**K-Spec**	**KaHyPar**	**Ours**	**hMETIS**	**K-Spec**	**KaHyPar**	**Ours**
sparcT1_core	985	977	**974**	**974**	**974**	2112	1889	1765	**1682**	2617	2492	2591	**2111**
neuron	284	244	245	**243**	244	514	**396**	407	400	566	431	415	**384**
stereo_vision	173	**169**	**169**	**169**	**169**	342	336	352	**322**	435	475	392	**386**
des90	380	**374**	450	379	376	514	535	504	**503**	747	747	691	**660**
SLAM_spheric	**1061**	**1061**	**1061**	**1061**	**1061**	2854	2720	**2674**	2686	3351	3241	3206	**3184**
cholesky_mc	287	**282**	286	283	283	918	**864**	879	877	980	984	975	**975**
segmentation	**107**	120	**107**	**107**	**107**	476	453	440	**411**	564	490	478	**477**
bitonic_mesh	586	584	589	586	**583**	927	895	893	**890**	1149	1311	1102	**1097**
dart	807	805	798	**784**	**784**	1258	1243	1166	**1053**	1499	1401	1280	**1264**
openCV	465	**434**	686	499	**434**	522	525	677	**489**	581	522	693	**520**
stap_qrd	383	464	**371**	**371**	**371**	533	497	496	**468**	760	674	614	**611**
minres	211	**207**	**207**	**207**	**207**	343	**309**	**309**	**309**	415	**407**	443	410
cholesky_bdti	1172	**1136**	1238	1156	1156	1929	1755	1716	**1652**	1881	**1865**	1875	1874
denoise	456	418	**416**	**416**	**416**	925	915	782	**721**	1015	1001	890	**862**
sparcT2_core	1214	1188	**1183**	**1183**	**1183**	3172	2249	2057	**2032**	3088	3558	**2741**	2752
gsm_switch	4881	1833	1651	**1621**	1645	5213	3694	**2432**	2439	5379	4404	**2855**	2864
mes_noc	671	**633**	649	651	640	1158	1125	**1110**	1123	1459	1346	**1307**	**1307**
LU230	3523	**3363**	3555	3602	3529	4677	4548	5564	**5331**	**5748**	6310	5915	5958
LU_Network	524	524	524	524	**523**	967	882	**784**	**784**	1507	1417	1278	**1248**
sparcT1_chip2	908	876	874	**873**	874	1490	1404	1256	**1177**	2151	1601	1595	**1431**
directrf	538	**515**	631	632	630	730	762	1013	**710**	1071	1092	1069	**974**
bitcoin_miner	1574	1562	1541	1514	**1489**	2119	1917	1886	**1867**	2800	2737	1894	**1834**
Norm Avg.	1.035	0.975	1	0.972	**0.964**	1.102	1.016	1	**0.943**	1.131	1.080	1	**0.955**

⋆ The Normalized Average (Norm. Avg.) for each method is calculated as the geometric mean relative to KaHyPar.

Table 2: Experimental results on ISPD98 benchmark.

Design	$k=2$					$k=3$					$k=4$				
	hMETIS	**K-Spec**	**KaHyPar**	**SHyPar**	**Ours**	**hMETIS**	**K-Spec**	**KaHyPar**	**SHyPar**	**Ours**	**hMETIS**	**K-Spec**	**KaHyPar**	**SHyPar**	**Ours**
ibm01	205	203	202	**201**	202	352	352	**335**	338	**335**	485	522	463	466	**457**
ibm02	345	333	329	**327**	**327**	340	**339**	340	340	**339**	609	706	642	**587**	602
ibm03	961	957	957	**952**	**952**	1483	1480	1627	1549	**1434**	1673	1690	**1650**	1718	1655
ibm04	581	580	580	**579**	580	1176	1212	1160	1146	**1135**	1638	1626	1590	1683	**1589**
ibm05	1728	1716	**1706**	1707	**1706**	2666	2635	2749	2764	**2619**	2950	**2946**	2956	3097	2956
ibm06	987	976	980	969	**963**	1293	1305	1297	1323	**1280**	1493	1476	**1470**	1522	**1470**
ibm07	926	935	900	**882**	888	1852	1846	1774	1722	**1697**	2206	2154	2066	2076	**2062**
ibm08	1147	**1140**	**1140**	**1140**	**1140**	2000	2037	2050	1944	**1899**	2332	2328	2291	2340	**2246**
ibm09	628	**620**	**620**	**620**	**620**	1395	1384	1427	1357	**1322**	1700	1676	**1666**	1701	1667
ibm10	1335	1257	1313	**1254**	**1254**	1924	1880	1873	1913	**1872**	2444	2400	2190	2234	**2187**
ibm11	1062	**1051**	1062	**1051**	1062	1810	1843	1777	1784	**1739**	2432	2452	2383	2472	**2346**
ibm12	1952	1937	**1920**	1986	**1920**	2874	2791	2778	2778	**2718**	3893	3844	**3774**	3802	3776
ibm13	833	832	848	**831**	**831**	1390	1335	1386	1312	**1272**	1865	1904	**1715**	1736	**1715**
ibm14	1892	1850	1849	**1842**	1849	2737	2710	2606	2691	**2603**	3283	3475	**3192**	3230	3195
ibm15	2751	2741	**2728**	**2728**	**2728**	4121	4333	4132	4140	**4008**	4825	4720	4724	4809	**4584**
ibm16	1960	1921	1929	1887	**1855**	3120	3062	3477	3043	**2890**	4182	4060	3805	3802	**3757**
ibm17	2336	2307	2294	**2285**	**2285**	4352	4248	**4053**	4095	4054	5708	5583	5494	5386	**5213**
ibm18	1827	1523	1915	**1521**	**1521**	2706	2401	**2331**	2361	2334	3262	2918	**2822**	2993	**2822**
Norm Avg.	1.009	0.987	1	0.981	**0.979**	1.010	0.994	1	0.985	**0.959**	1.041	1.058	1	1.016	**0.989**

4.2.4 Comparison of Different Community Detection Strategies in the Uncoarsening Phase. Figure 4 presents the performance comparison of Fast Greedy, Louvain, Label Propagation, and Walktrap ($\alpha = 4$ is used for each community detection strategy) within the uncoarsening phase. We include KaHyPar and SHyPar as strong baselines for reference. The results demonstrate that Fast Greedy and Louvain yield superior quality gains, whereas the contributions from Label Propagation and Walktrap are relatively modest.

5 Conclusion

In this paper, we propose ComPart, a novel framework that integrates community detection into the uncoarsening phase to guide global solution exploration. Furthermore, theoretically generalize the *locally-dense decomposition* technique from graphs to hypergraphs, providing a rigorous foundation for improving the initial partitioning. Experimental results demonstrate that our method consistently achieves state-of-the-art performance on standard benchmarks compared to existing solvers.

6 Acknowledgement

This work is supported by Hong Kong Research Grants Council (RGC) CRF-YCRG C6003-24Y, GRF 16216825. It was partially conducted by ACCESS – AI Chip Center for Emerging Smart Systems, supported by the InnoHK initiative of the Innovation and Technology Commission of the Hong Kong Special Administrative Region Government.

References

[1] Utku Umur Acikalin and Bugra Caskurlu. 2022. Multilevel memetic hypergraph partitioning with greedy recombination. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*. 168–171.

[2] Ali Aghdaei and Zhuo Feng. 2022. HyperEF: Spectral hypergraph coarsening by effective-resistance clustering. In *Proceedings of the 41st IEEE/ACM International Conference on Computer-Aided Design*. 1–9.

[3] Robin Andre, Sebastian Schlag, and Christian Schulz. 2018. Memetic multilevel hypergraph partitioning. In *Proceedings of the Genetic and Evolutionary Computation Conference*. 347–354.

[4] Vincent D Blondel, Jean-Loup Guillaume, Renaud Lambiotte, and Etienne Lefebvre. 2008. Fast unfolding of communities in large networks. *Journal of statistical mechanics: theory and experiment* 2008, 10 (2008), P10008.

[5] Stephen Boyd and Lieven Vandenberghe. 2004. *Convex optimization*. Cambridge university press.

[6] Ismail Bustany, Andrew B Kahng, Ioannis Koutis, Bodhisatta Pramanik, and Zhiang Wang. 2022. SpecPart: A supervised spectral framework for hypergraph partitioning solution improvement. In *Proceedings of the 41st IEEE/ACM International Conference on Computer-Aided Design*. 1–9.

[7] Ismail Bustany, Andrew B Kahng, Ioannis Koutis, Bodhisatta Pramanik, and Zhiang Wang. 2023. K-SpecPart: Supervised embedding algorithms and cut overlay for improved hypergraph partitioning. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 43, 4 (2023), 1232–1245.

[8] Ümit V Çatalyürek and Cevdet Aykanat. 2011. PaToH (Partitioning Tool for Hypergraphs).

[9] Magi Chen and Ting-Chi Wang. 2024. A Hypergraph Partitioner Utilizing a Novel Graph Generative Model. In *Proceedings of the 43rd IEEE/ACM International Conference on Computer-Aided Design*. 1–9.

[10] Aaron Clauset, Mark EJ Newman, and Cristopher Moore. 2004. Finding community structure in very large networks. *Physical Review E—Statistical, Nonlinear, and Soft Matter Physics* 70, 6 (2004), 066111.

[11] Maximilien Danisch, T-H Hubert Chan, and Mauro Sozio. 2017. Large scale density-friendly graph decomposition via convex programming. In *Proceedings of the 26th International Conference on World Wide Web*. 233–242.

[12] Robert P Dick, David L Rhodes, and Wayne Wolf. 1998. TGFF: task graphs for free. In *Proceedings of the Sixth International Workshop on Hardware/Software Codesign.(CODES/CASHE'98)*. IEEE, 97–101.

[13] Charles M Fiduccia and Robert M Mattheyses. 1988. A linear-time heuristic for improving network partitions. In *Papers on Twenty-five years of electronic design automation*. 241–247.

[14] Lars Gottesbüren, Michael Hamann, and Dorothea Wagner. 2019. Evaluation of a flow-based hypergraph bipartitioning algorithm. *arXiv preprint arXiv:1907.02053* (2019).

[15] Alexander Graham. 2018. *Kronecker products and matrix calculus with applications*. Courier Dover Publications.

[16] Tobias Heuer. 2015. *Engineering initial partitioning algorithms for direct k-way hypergraph partitioning*. Ph. D. Dissertation. Karlsruher Institut für Technologie (KIT).

[17] George Karypis, Rajat Aggarwal, Vipin Kumar, and Shashi Shekhar. 1997. Multilevel hypergraph partitioning: Application in VLSI domain. In *Proceedings of the 34th annual Design Automation Conference*. 526–529.

[18] Brian W Kernighan and Shen Lin. 1970. An efficient heuristic procedure for partitioning graphs. *The Bell system technical journal* 49, 2 (1970), 291–307.

[19] Benzheng Li, Shunyang Bi, Hailong You, Zhongdong Qi, Guangxin Guo, Richard Sun, and Yuming Zhang. 2024. MaPart: An Efficient Multi-FPGA System-Aware Hypergraph Partitioning Framework. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 43, 10 (2024), 3212–3225.

[20] Rongjian Liang, Anthony Agnesina, and Haoxing Ren. 2024. Medpart: A multilevel evolutionary differentiable hypergraph partitioner. In *Proceedings of the 2024 International Symposium on Physical Design*. 3–11.

[21] Pascal Pons and Matthieu Latapy. 2005. Computing communities in large networks using random walks. In *International symposium on computer and information sciences*. Springer, 284–293.

[22] Merten Popp, Sebastian Schlag, Christian Schulz, and Daniel Seemaier. 2021. Multilevel Acyclic Hypergraph Partitioning. In *2021 Proceedings of the Workshop on Algorithm Engineering and Experiments (ALENEX)*. SIAM, 1–15.

[23] Usha Nandini Raghavan, Réka Albert, and Soundar Kumara. 2007. Near linear time algorithm to detect community structures in large-scale networks. *Physical Review E—Statistical, Nonlinear, and Soft Matter Physics* 76, 3 (2007), 036106.

[24] Hamed Sajadinia, Ali Aghdaei, and Zhuo Feng. 2025. SHyPar: A Spectral Coarsening Approach to Hypergraph Partitioning. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* (2025).

[25] Sebastian Schlag, Tobias Heuer, Lars Gottesbüren, Yaroslav Akhremtsev, Christian Schulz, and Peter Sanders. 2023. High-quality hypergraph partitioning. *ACM Journal of Experimental Algorithmics* 27 (2023), 1–39.

[26] Shengbo Tong, Haoyuan Li, Jiahao Xu, Chunyan Pei, Wenjian Yu, Shengjun Liu, and Jian Shen. 2024. EasyPart: An effective and comprehensive hypergraph partitioner for FPGA-based emulation. In *Proceedings of the 43rd IEEE/ACM International Conference on Computer-Aided Design*. 1–9.

[27] Wayne Wolf, Ahmed Amine Jerraya, and Grant Martin. 2008. Multiprocessor system-on-chip (MPSoC) technology. *IEEE transactions on computer-aided design of integrated circuits and systems* 27, 10 (2008), 1701–1713.

[28] Yugao Zhu, Shenghua Liu, Wenjie Feng, and Xueqi Cheng. 2023. Fast Searching The Densest Subgraph And Decomposition With Local Optimality. *arXiv preprint arXiv:2307.15969* (2023).