Noise-Aware Circuit Clustering based on Analytical Placement Evolution

Zhiyuan Chen, Chung-Kuan Cheng, Zhan Song*, Yucheng Wang
University of California, San Diego
La Jolla, California, USA
{zhc065,ckcheng,zhs017,yuw132}@ucsd.edu

Abstract

Clustering is a critical step in VLSI design. Good clusters can aid in partitioning, floorplanning, and netlist reduction. In this work, we adopt a data mining technique to extract modular clusters and networks between clusters from placement evolution. We observe that state-of-the-art analytical placers propagate standard cells as Modular Clusters during placement evolution. And other noisy cells, which cannot be assigned to a specific cluster, form a network structure called Inter-Cluster Networks (ICNs). We propose Cascade HDBSCAN to extract Modular Clusters at different granularity as well as Inter-Cluster Networks in placement evolution. And we reveal Modular Clusters and Inter-Cluster Networks adhere to different Rent's rules. Furthermore, we provide a detailed analysis on the clustering structure and demonstrate VLSI netlists are small-world networks. Modular Clusters are localized groupings connected by Inter-Cluster Networks that function as shortcuts in the networks.

CCS Concepts: • **Hardware** → **Partitioning and floorplanning**.

Keywords: Clustering, Data Mining, Placement, Rent's Rule, Small-World Networks, Floorplanning, Partitioning

1 Introduction

Clustering, a classic unsupervised machine learning method, has been widely applied to various tasks in VLSI physical design, including sampling layout pattern [14], automating standard cell design [7], and partitioning netlists [8]. Cutting-edge circuit designs can particularly benefit from clustering due to their increasing complexity. Clusters can be used for floorplanning, placement, and complexity reduction, and numerous approaches have been explored in recent years. Lu *et al.* [13] demonstrates a novel framework using Graph Neural Networks (GNN) to iteratively generate clusters that leads to better power, performance, and area (PPA) results when used to guide placement. Kahng *et al.* [10] proposed a clustering method to perform macro placement with the clustered netlist. However, these methods enforce that each cell must belong to one of the clusters, overlooking the fact that some cells may not logically fit within any cluster, such as those in clock trees, scan chains, and buses.

In this work, we propose a data mining technique for VLSI designs, *Cascade HDBSCAN*, which is based on Hierarchical Density-Based Spatial Clustering (HDBSCAN) [3]. From n snapshots in the evolution of 2D analytical placement, we have $2n$ dimensional data that describe the coordinates of standard cells throughout the process. We observe that standard cells that belong to the same logical modules tend to move together and propagate as clusters throughout the evolution of analytical placement, which motivates us to extract these modular structures for further analysis. We also identify a set of noisy cells that do not belong to any clusters during placement evolution. *Cascade HDBSCAN* can capture modular structures at different scales using the dimensional data. We refer to these clusters as *Modular Clusters*. And the noisy cells detected by *Cascade HDBSCAN* are referred to as as *Inter-Cluster Networks*.

Figure 1 illustrates the clustering phenomenon we discovered during DREAMPlace [11] evolution in the global placement stage. The colored groups represent naturally-formed *Modular Clusters*, while the black dots represent the *networks* between these clusters, referred to as *Inter-Cluster Networks*. *Cascade HDBSCAN* identifies *Inter-Cluster Networks*, which consist of standard cells considered as "noise" due to their sparse nature and inability to attach to any *Modular Clusters*. Our main contributions are summarized as follows:

- We identify naturally-formed *Modular Clusters* during placement evolution and enhance the existing HDBSCAN algorithm to iteratively refine *Modular Clusters* and *Inter-Cluster Networks*, enabling the detection of patterns at varying levels of granularity.
- We provide a detailed analysis on *Modular Clusters* and *Inter-Cluster Networks*, exploring their different parameterizations of Rent's rule.
- We demonstrate that VLSI netlists exhibit small-world network properties and show that *Modular Clusters*

* Corresponding Author.

SLIP '24, October 31, 2024, Newark, NJ, USA
© 2024 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-1400-9/24/10
https://doi.org/10.1145/3708358.3709348

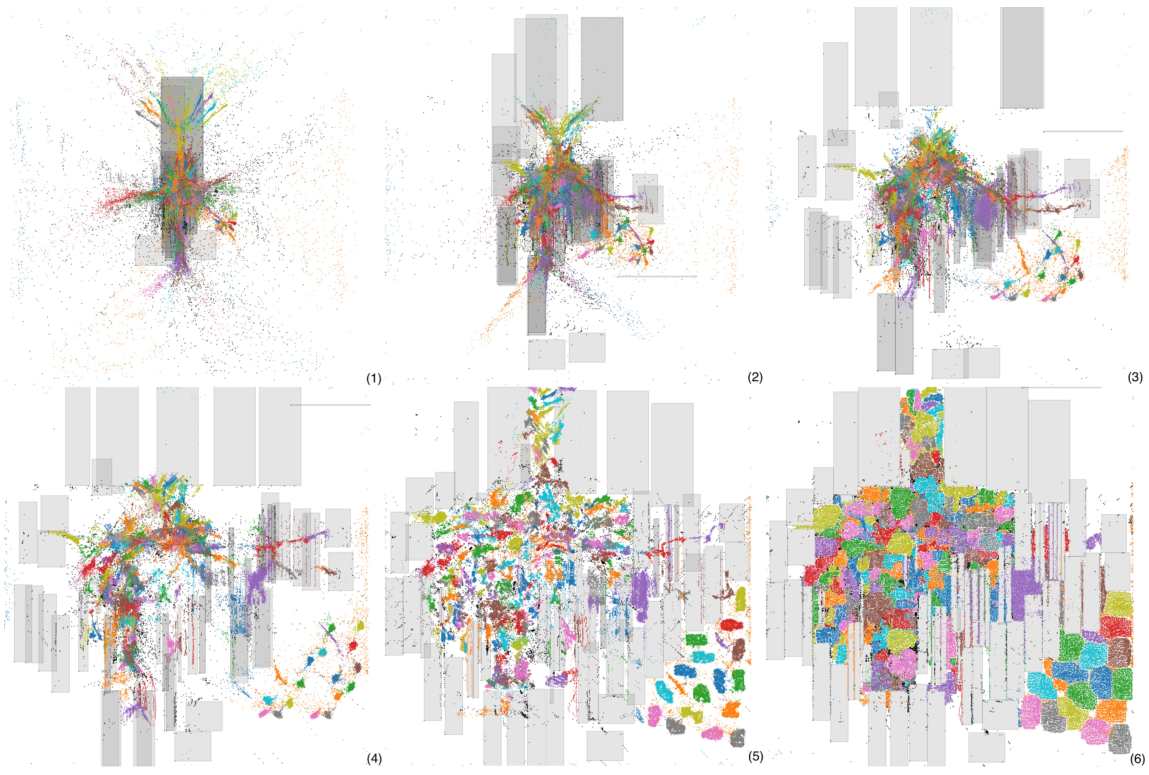

Figure 1. Clustering phenomenon in the evolution of placement solutions. Snapshot (1)-(6) are taken chronologically when running mixed-mode DREAMPlace on `adaptec1` in ISPD 2005 benchmarks [15]. We perform cluster analysis on standard cells throughout placement evolution using *Cascade HDBSCAN*, where colored cell groups represent *Modular Clusters* and scattered black cells represent the *Inter-Cluster Networks* in standard cells. Gray boxes are macros.

and *Inter-Cluster Networks* serve distinct roles in the network.

The rest of the paper is structured as follows: Section 2 provides background information and motivations; Section 3 details the formulation behind *Cascade HDBSCAN*; Section 4 and 5 provide the formulations of Rent's rule and small-world networks respectively, which are used in our analysis; Section 6 validates the meaningfulness of *Modular Clusters* and *Inter-Cluster Networks* using Rent's rule parameterization and small-world network properties; Section 7 concludes the paper and highlights future work. Our code is available on GitHub [1].

2 Analytical Placement Evolution

We use the evolution of state-of-the-art analytical placement based on ePlace[12] for clustering, which provides high-quality placement solutions and wins various benchmarks. In ePlace and its derivative DREAMPlace, a novel density function is developed that draws an analogy between the placement instance and an electrostatic system. Leveraging electric force, each standard cell smoothly follows charge movement and converges toward a global equilibrium state. By incorporating techniques such as Fast Fourier Transform and GPU acceleration, ePlace, and its derivative DREAMPlace ensure scalable analytical placement, enabling efficient placement of netlists containing millions of cells in seconds.

Here, we define the placement formulation. Each cell (or macro) i in the netlist is analogous to a positively charged particle, which emits an electric force denoted as F_i. Formally, we can define this force as:

$$F_i = q_i \xi_i, \tag{1}$$

where ξ_i is the local electric field and q_i is the electric quantity on cell i. Its potential energy N_i can be computed by:

$$N_i = q_i \psi_i, \tag{2}$$

where ψ_i is the electric potential.

To speed up the placement process, ePlace employs Poisson's equation to efficiently capture the distribution of the electric potential:

$$\nabla \cdot \nabla \psi(x) = -\rho(x), \tag{3}$$

where $\rho(x)$ is the charge density.

The objective function of DREAMPlace is formulated as:

$$\min_X (\textstyle\sum_{e \in \mathcal{E}} Wl(e; X_e) + \gamma \mathcal{D}(X)), \tag{4}$$

where X_e are the component-wise coordinates within net e; $\mathcal{E}$ denotes a set of given nets; $Wl(\cdot;\cdot)$ is a function that computes the cumulative wirelength of a net instance e; $\mathcal{D}(\cdot)$ is a function that computes the bin-wise density; and γ is the density penalty weight. The $Wl(\cdot;\cdot)$ term can be regarded as an attraction force among the components, while $\mathcal{D}(\cdot)$ term can be regarded as a repulsive force to balance out the attraction force. By minimizing this objective function, DREAMPlace is able to generate a compact and even placement for the standard cells and macros.

Throughout the DREAMPlace iterations, the density penalty weight γ gradually increases, and the repulsive force also increases over time. This increased repulsive force drives the cells to spread across the entire placement region. During this placement evolution, the movement of the tangible clusters is continuous and consistent. The smooth and continuous placement evolution is a common characteristic of the state-of-the-art analytical placement algorithms [4, 5, 11, 12].

3 Clustering based on Data Mining Techniques

Data mining techniques, such as clustering algorithms, can be used to analyze high-dimensional datasets with noise and find inherent patterns in the data. Hierarchical clustering creates a dendrogram, a tree-like diagram revealing the arrangement of potential clusters. Such an approach can handle clusters of different shapes and sizes.

Density-based clustering is another technique that identifies dense regions with many data points and separates them from sparse or empty regions, making the clusters more resistant to noise in the data. Hence, Density-based clustering can effectively distinguish the meaningful clusters from the noisy data points.

HDBSCAN algorithm combines the strengths of hierarchical and density-based clustering, allowing it to find consistent cluster patterns in high-dimensional datasets while being robust to noise. This makes HDBSCAN well-suited for our application, as we are analyzing the placement evolution data, which can be large and noisy. Due to the complexity of placement tasks, cells can travel long distances before reaching their final destinations in a placement evolution. Therefore, identifying groups of cells that stay together consistently as the placement evolves over time is crucial in our flow. The following subsections describe the flow of our proposed data mining technique, *Cascade HDBSCAN*, in the context of standard cell placement.

3.1 Data Preparation

To prepare the data for our clustering analysis, we concatenate the standard cell locations across a series of snapshots from a DREAMPlace mixed-mode placement evolution. We denote the initial placement of standard cells as X_{cell}^0. We then take a series of snapshots from the placement evolution, resulting in a set of placements $X_{cell} = \{X_{cell}^0, X_{cell}^1, \ldots, X_{cell}^{T-1}, X_{cell}^T\}$. Here, T represents the total iteration count.

3.2 Graph Construction

HDBSCAN is a density-based clustering method that operates on a mutual reachability graph. This mutual reachability graph can be constructed using standard cells as nodes. First, we construct a weighted graph $G = (V, E)$, where $v_i \in V$ is a node that represents a standard cell i. The weight w_{ij} on the edge $e_{ij} \in E$ is based on the mutual reachability distance $dmr(\cdot,\cdot)$ between the corresponding standard cells.

We compute the mutual reachability distance dmr using the collected set of placements X_{cell}. Denoting the coordinate of a standard cell i at time t as x_i^t, the placement evolution for such a standard cell is represented as $x_i^{\{0,\ldots,T\}}$. To determine the pair-wise distance between cells i and j, we use the function $dt(\cdot,\cdot)$, which computes the average Manhattan Distance over time:

$$dt(x_i^{\{0,\ldots,T\}}, x_j^{\{0,\ldots,T\}}) = \frac{1}{T+1}\sum_{t=0}^{T} |x_i^t - x_j^t|, \tag{5}$$

We define the core distance $\kappa_i = dt(x_i^{\{0,\ldots,T\}}, x_k^{\{0,\ldots,T\}})$, where $x_k^{\{0,\ldots,T\}}$ for cell i. It is the location over time for the k-th nearest neighbor cell to cell i.[1] We compute the mutual reachability distance $dmr(\cdot,\cdot)$ between cell i and cell j as:

$$dmr(x_i^{\{0,\ldots,T\}}, x_j^{\{0,\ldots,T\}}) = \max(\kappa_i, \kappa_j, dt(x_i^{\{0,\ldots,T\}}, x_j^{\{0,\ldots,T\}})). \tag{6}$$

While Euclidean distance assumes a uniform spherical region, the mutual reachability distance $dmr(\cdot,\cdot)$ used for density-based clustering considers different neighborhood regions for each standard cell. As depicted in Figure 1, the potential clusters emerging from the intermediate placements can have arbitrary shapes. Therefore, to accommodate these varying cluster shapes, the $dmr(\cdot,\cdot)$ metric is more suitable than Euclidean distance. Once the mutual reachability distance $dmr(\cdot,\cdot)$ is computed between each pair of cells, we perform hierarchical clustering using the weighted graph G constructed from these distances.

3.3 Single-linkage Hierarchical Clustering

Hierarchical clustering is a form of unsupervised clustering that, different from k-means clustering, does not require a predefined number of clusters. For a typical placement problem, which involves millions of components, this is particularly advantageous as determining the number of clusters beforehand can be challenging.

[1]The default value for k is the same as θ, which is the minimum cluster size. The value of k can be adjusted to make the clustering more or less conservative. HDBSCAN with a lower value of k will identify fewer standard cells as noisy.

To identify clusters, we first construct a Minimum Spanning Tree (MST), $T = (V', E')$ using G we obtained previously. In an MST, the removal of any single edge creates multiple connected graphs that can be regarded as cluster candidates. So an edge can be viewed as a *single linkage*, and we define the inverse of the weight of an edge as the λ value for the corresponding removal action.

We then sort each edge of the tree in descending order of $dmr(\cdot, \cdot)$, and remove *single linkages* to form clusters. By cutting horizontally at any depth of this cluster tree, the resulting clusters share a similar $dmr(\cdot, \cdot)$. However, the granularity and density of each cluster may vary significantly across placement solutions to different designs. Therefore, we need to take additional steps to obtain individual clusters at different tree depths.

3.4 Condensed Tree

Our clustering method considers not only clusters but also noises (as *networks*). Not all leaves inside the cluster tree will fall into a cluster. To distinguish *Modular Clusters* from *Inter-Cluster Networks*, we define *stability* for potential clusters [2].

Stability, denoted as $\mathrm{S}(\cdot)$, can be defined as the lifetime of a cluster within the tree before it is reduced to a size smaller than a user-defined size threshold θ (minimum number of standard cells in a *Modular Clusters*) at a deeper tree depth. For each cluster c, we define its $\mathrm{S}(\cdot)$ as:

$$\mathrm{S}(c) = \sum_{cell \in c} \left(\lambda_p - \lambda_{\text{birth}}\right). \tag{7}$$

This captures the stability of each cluster by summing the differences between the λ value at which each point p in the cluster fell out of the cluster (denoted as λ_p) and the λ value at which the cluster was born (denoted as λ_{birth}).

In the context of placement evolution, this metric evaluates if standard cells of a cluster stay together consistently throughout all the intermediate placements. If a cell does not hold a membership to any clusters before splitting up, it will be considered as a cell in *Inter-Cluster Networks* (or noisy points). We summarize our flow in Algorithm 1.

Algorithm 1 Single-linkage Hierarchical Clustering

```
Require: Set of standard cells $\{x_i\}_{i=1}^N$, mutual reachability
    distances $dmr(\cdot,\cdot)$
Ensure: Modular Clusters and Inter-Cluster Networks
    Compute pairwise distance metric using Equation 6
    Construct a graph $G = (V, E)$ on $\{x_i\}_{i=1}^N$ using $dmr(\cdot,\cdot)$ as
    weight
    Construct a Minimum Spanning Tree $T = (V', E')$ from $G$

    Sort edges in $E'$ of $T$ by distance in descending order
    Initialize modular clusters $\mathbb{C} \leftarrow \{\{x_i\}_{i=1}^N\}$, dendrogram
    $\mathbb{D} \leftarrow \emptyset$
    for each edge $e \in E'$ do
        Remove the current cluster containing $e$ from $\mathbb{C}$
        Add the two new clusters from single linkage to $\mathbb{D}$
    end for
    for each cluster $c$ in the dendrogram $\mathbb{D}$ do
        Calculate $\lambda$ value for $c$
        Compute the stability $\mathrm{S}(c)$ using Equation 7
    end for
    Construct the condensed tree retaining only clusters with
    significant stability
    Add clusters with the highest stability scores from the
    condensed tree to $\mathbb{C}$
    for each standard cell $x_i$ do
        Assign $x_i$ to the most stable cluster it belongs to in $\mathbb{C}$
    end for
    Initialize inter-cluster networks $ICN \leftarrow \emptyset$
    for each standard cell $x_i$ do
        if $x_i$ does not belong to any stable cluster then
            Add $x_i$ to $ICN$
        end if
    end for
    return $\mathbb{C}$ and $ICN$
```

3.5 Cascade HDBSCAN

To uncover as many sizable *Modular Clusters* as possible without overlooking smaller clusters, we design *Cascade HDBSCAN* shown in Algorithm 2, which executes the HDBSCAN algorithm multiple times. We begin with a large minimum cluster size θ to identify the prominent modules in the original netlist. Subsequently, we decrease θ and continue to extract smaller *Modular Clusters* from the *Inter-Cluster Networks* in the previous HDBSCAN results. These smaller clusters exhibit good modularity, but they were previously categorized as "noise" since their size did not exceed θ.

Cascade HDBSCAN allows us to leverage the strengths of the HDBSCAN algorithm to hierarchically discover modular structures at different scales within the VLSI design. By adjusting the minimum cluster size parameter, we can progressively uncover both the large and dominant modules as well as the small yet meaningful modular components that may have been overlooked in a single-pass clustering.

4 Rent's Rule Parametrization

Rent's rule is an empirical observation that relates the number of external signal pins to the number of internal gates in a circuit [6]. For a reasonable clustering outcome, clusters (or sub-circuits) should reflect a natural hierarchical structure of the original circuit while sharing a similar complexity. Given a $Cluster$ with number of external pins $Cluster_{pin}$ and number of internal gates $Cluster_{gate}$, Rent's rule states that:

$$Cluster_{pin} = K \cdot Cluster_{gate}^{p}. \tag{8}$$

where K is the Rent's coefficient and p is the Rent's exponent.

We use Rent's rule to demonstrate that *Cascade HDBSCAN* is an effective clustering algorithm and that *Modular Clusters* exhibit different properties from *Inter-Cluster Networks*. The

Algorithm 2 Cascade HDBSCAN

```
Require: Set of standard cells $\{x_i\}_{i=1}^N$, mutual reachability
    distances $dmr(\cdot,\cdot)$, $\theta_{1,2,3}$
Ensure: All Modular Clusters and Inter-Cluster Networks
    Initialize Clusters ← ∅
    Initialize Remaining Cells ← $\{x_i\}_{i=1}^N$
    // Iteratively run HDBSCASN three times
    for ($i = 0; i < 3; i++$) do
        Set Minimum Cluster Size ← $\theta_i$
        Run Algorithm 1 on Remaining Cells
        Add all clusters in $\mathbb{C}$ to Clusters
        Update Remaining Cells ← ICN
    end for
    return Clusters and Remaining Cells
```

traditional way of obtaining Rent's rule data is through a recursive partitioning process [6]. The choice of partitioning algorithm influences the Rent's exponent p. Our *Cascade HDBSCAN* can also be considered as a partitioning algorithm, with *Modular Clusters* as its results. Consequently, Rent's rule can be evaluated on these *Modular Clusters*. In Section 6, we will analyze the difference between the Rent exponents of *Modular Clusters* and those of *Inter-Cluster Networks*.

5 Small-World Networks

In many technological, biological, and social systems, networks often exhibit structures that are neither fully regular nor entirely random [16]. These so-called small-world networks are characterized by a high degree of local clustering combined with "shortcuts" between distant nodes, a phenomenon known as the "small-world effect" [17]. This network structure, composed of tightly clustered nodes connected predominantly by short-range links, with a few long-range connections, enables efficient communication across the network. The small-world effect is also evident in electronic circuits [9]. Inspired by this concept, Ogras *et al.* [16] introduced long-range links into standard mesh networks to act as shortcuts, reducing circuit latency.

The structural properties of networks are often quantified by the characteristic path length L and the clustering coefficient C. L characterizes the distance between vertices in the graph and is defined as the average number of edges in the shortest paths between all pairs of vertices. C quantifies the clustering phenomenon within the neighborhood of single vertices. For a vertex v having k neighbors, the maximum number of edges between them is $k(k-1)/2$. C_v is the ratio of the actual number of edges in the neighborhood to this maximum. C is defined as the average of C_v over all vertices.

Using the Watts–Strogatz model [17], a random graph with the same number of nodes and edges as the original small-world graph is generated. A small-world network has the property that $L_{\text{small-world}}$ is slightly greater than $L_{\text{random graph}}$, while $C_{\text{small-world}}$ is significantly larger than $C_{\text{random graph}}$.

In Section 6, we will demonstrate that modern VLSI netlists are also small-world networks, with *Modular Clusters* and *Inter-Cluster Networks* serving different functions. Specifically, *Modular Clusters* create localized groupings, while *Inter-Cluster Networks* function as shortcuts, connecting distant clusters.

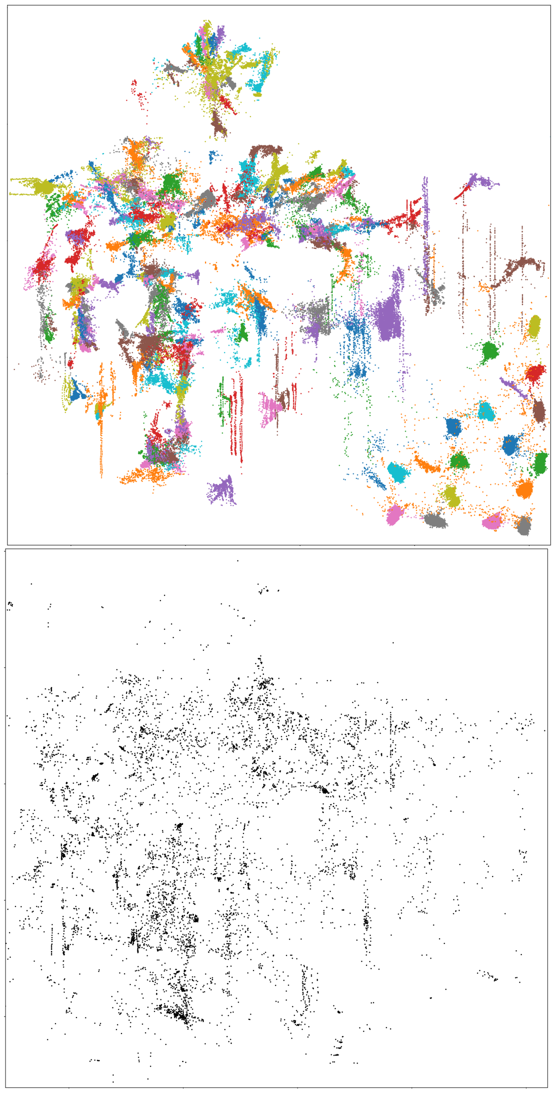

Figure 2. Cascade HDBSCAN clustering results for `adaptec1` in a placement snapshot. Macros are excluded for clarity. Top plot shows *Modular Cluster* cells. Bottom plot shows *Inter-Cluster Networks* cells. *Modular Cluster* cells are significantly denser than *Inter-Cluster Networks* cells.

6 Experimental Results

In this section, we provide a comprehensive analysis of the generated *Modular Clusters* and *Inter-Cluster Networks*. We hypothesize that the generated *Modular Clusters* and *Inter-Cluster Networks* follow different Rent's rules, and VLSI designs exhibit the small-world effect. The ISPD 2005 contest benchmarks [15] were used in our experiments.

6.1 Cascade HDBSCAN Clustering on DREAMPlace Snapshots

The cell coordinates are extracted from the DREAMPlace global placement evolution every 50 iterations, starting from iteration 200. These coordinates are concatenated to form the data for clustering. The first 200 iterations are excluded because the cell coordinates do not change significantly during this period. After extracting the data, we perform three iterative rounds of HDBSCAN for clustering. The hyperparameter θ mentioned in Section 3 is set to 400, 200, and 100 for each round. In rounds 2 and 3, HDBSCAN is applied to the noisy points identified in the previous round to iteratively uncover smaller clusters and reduce the number of *Inter-Cluster Networks* cells.

Table 1 presents the clustering statistics. On average, 5.16% of the standard cells are identified as *Inter-Cluster Networks* cells, and the number of *Modular Clusters* is approximately one-thousandth of the total number of cells. Figure 2 illustrates the clustering results for `adaptec1` in one placement snapshot, with separate plots of *Modular Cluster* and *Inter-Cluster Networks*. Each *Modular Cluster* exhibits high cell density with a well-defined clustering structure. In contrast, the *Inter-Cluster Networks* are much sparser and primarily distributed between clusters. This density contrast affirms the validity of the *Modular Clusters*. However, some smaller clusters are also observed within the *Inter-Cluster Networks*, which is due to the algorithm's granularity. These smaller clusters could be further captured by additional HDBSCAN iterations, though this may lead to some *Inter-Cluster Networks* being misclassified as clusters.

6.2 Rent's Rule Analysis

Our analysis of Rent's rule is conducted on four different types of groups. Each group, except *Modular Clusters*, is formed by dividing the final placement region into $n \times n$ windows, where $n \in \{4, 8, 16, 32\}$. The relevant cell types within each window are then treated as individual groups.

- *Whole Circuit*: Traditionally, Rent's rule is evaluated on the whole circuit, without differentiating between *Modular Clusters* and *Inter-Cluster Networks*. To measure Rent's rule of the whole circuit, we extract the final placement coordinates of standard cells, take all the cells inside a window as a group, and evaluate the number of cells in each group (grid) and the number of pins connected to the rest of the circuit.
- *Modular Clusters*: We also evaluate Rent's rule for the clusters we find. Each cluster is treated as a group, and its external connections to the rest of the circuit are counted.
- *Inter-Cluster Networks*: Rent's rule is obtained for *Inter-Cluster Networks* only. Inside each window, we take all the standard cells belonging to the *Inter-Cluster Networks* as a group. The external connections of these groups are then counted.
- *Intra-Cluster Networks*: To characterize the internal connectivity of *Modular Clusters*, we obtain Rent's rule for each *Modular Cluster*. Inside each window, we take all the cells belonging to the target *Modular Cluster* as a group, and count the number of external connections.

After obtaining the number of nodes and pins (external connections) for each group, we use linear regression on the logarithmic scale to derive Rent's exponent p and Rent's coefficient K.

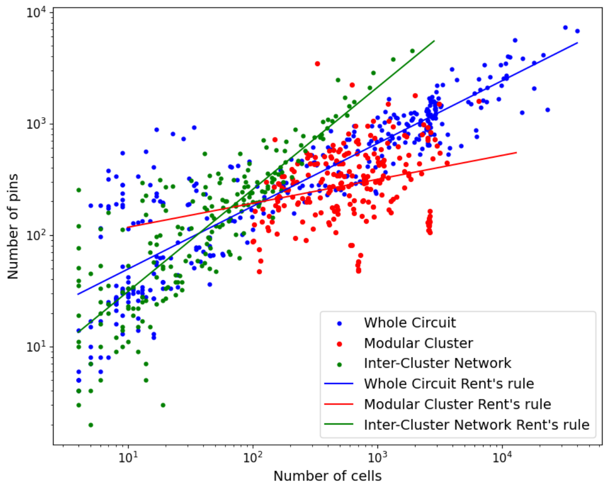

Figure 3. Rent's rule of three different types of groups on `adaptec1` benchmark: *Whole Circuit*, *Modular Clusters*, and *Inter-Cluster Networks*. Each point represents a specific group of cells. The lines represent the linear regression of Rent's rule. *Intra-Cluster Networks* are excluded here, as each cluster has its own *Intra-Cluster Networks* Rent's rule, and these results are shown in Figure 4.

Figure 3 shows the Rent's rule results on `adaptec1` benchmark for the first three types of groups: *Whole Circuit*, *Modular Clusters*, and *Inter-Cluster Networks* (*ICNs*). It demonstrates that the first three types of groups have different Rent's rule. The data points from *Inter-Cluster Networks* are at the top, the *Whole Circuit* data points are mainly in the middle, and the data points of *Modular Clusters* are generally at the bottom. Due to the relatively small size of each cluster compared to the total number of cells, Region II of Rent's rule is not observed in our experiments.

Table 2 presents the numerical results for Rent's rule. Across all ISPD 2005 benchmarks, the Rent's exponent p value for *Modular Clusters* is, on average, 33.3% lower than that of the *Whole Circuit*. This indicates that the external connectivity of *Modular Clusters* is weaker than that of the partitions derived from the traditional Rent's rule process. This suggests that *Cascade HDBSCAN* is an effective clustering algorithm, as it produces clusters with fewer external connections for the same number of cells.

Table 1. Cascade HDBSCAN clustering result on ISPD 2005 benchmarks.

Design	#Cells	#nets	#ICN cells	Total ICN%	#Clusters	DREAMPlace Runtime (s)	HDBSCAN Runtime (s)
adaptec1	211K	221K	7,353	3.48%	223	25	21
adaptec2	255K	266K	10,800	4.23%	315	33	18
adaptec3	452K	467K	20,914	4.63%	520	54	79
adaptec4	496K	516K	30,143	6.08%	549	61	83
bigblue1	278K	284K	19,676	7.07%	256	34	45
bigblue2	558K	577K	20,733	3.72%	483	72	53
bigblue3	1,097K	1,123K	69,941	6.38%	1,236	115	284
bigblue4	2,177K	2,230K	123,391	5.67%	2,354	2556	721

Table 2. Rent's exponent p and Rent's coefficient K for the ISPD 2005 benchmarks.

Design	Whole Circuit		Modular Cluster		ICN		Intra-Cluster Network	
	p	K	p	K	p	K	p	K
adaptec1	0.56	13.48	0.21	71.44	0.92	3.76	0.89	3.00
adaptec2	0.55	15.32	0.41	17.27	1.09	1.17	0.76	3.76
adaptec3	0.57	13.59	0.51	10.36	0.89	4.02	0.87	2.50
adaptec4	0.52	18.56	0.41	16.41	0.80	7.36	0.96	1.59
bigblue1	0.69	5.00	0.45	22.78	0.98	2.70	0.86	3.39
bigblue2	0.66	4.09	0.42	14.04	0.96	2.20	0.91	2.40
bigblue3	0.65	5.81	0.41	15.98	0.95	2.98	0.90	2.33
bigblue4	0.57	13.50	0.41	19.95	0.91	4.52	0.99	1.27
Average	0.60	11.17	0.40	23.53	0.94	3.59	0.89	2.53

The p for *Inter-Cluster Networks* is on average 56.7% higher than that of the *Whole Circuit*. Since Rent's rule expects a consistent level of connectivity across the netlist, this higher-than-average p suggests that the *Inter-Cluster Networks* identified by our algorithm have connectivity properties that differ from those of other cells in the netlist, which are *Modular Clusters* cells.

However, in Figure 3, some *Modular Cluster* data points are above the *Whole Circuit* Rent's rule, exhibiting a strong external connection. To justify these clusters, we obtain the Rent's rule of *Intra-Cluster Networks* for each *Modular Cluster*. Experimental results in Table 2 show that their internal connection is even stronger, thus forming *Modular Clusters*.

Figure 4 shows the Rent's exponent p and Rent's coefficient K of the *Intra-Cluster Networks* for each *Modular Cluster* on adaptec1 benchmark. The Rent's rule data of the *Intra-Cluster Networks* are finally aggregated and shown in Table 2. The results show that *Modular Clusters* have a Rent's exponent p that is 48.3% higher than of the *Whole Circuit* and 122.5% higher than the *Inter-Cluster Networks* on average. This indicates that the *Modular Clusters* have stronger internal connectivity than external connectivity, which is the reason for their formation.

6.3 Small-World Network Analysis

To verify that if the netlists from the ISPD 2005 benchmarks exhibit small-world properties, each netlist is compared to a random graph. C_{netlist} represents the clustering coefficient for the netlist, while $C_{\text{random graph}}$ denotes that of the random graph. They are calculated based on the definitions in

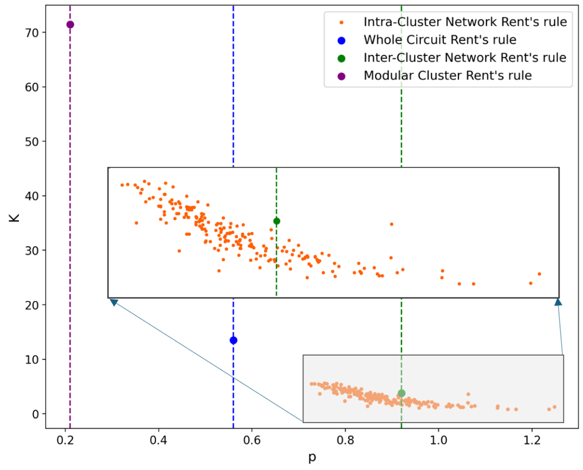

Figure 4. Intra-cluster Rent's rule on adaptec1 benchmark. Each data point represents Rent's rule for a *Modular Cluster*. The x-axis is the Rent's exponent p, and the y-axis is the Rent's coefficient K. All *Modular Clusters* have a Rent's exponent p larger than both the *Whole Circuit* and the *Inter-Cluster Networks*.

Section 5. Since calculating L_{netlist} and $L_{\text{random graph}}$ for all possible pairs of cells is computationally expensive, their values are estimated by randomly sampling 10,000 pairs. The results, shown in Table 3, indicate that L_{netlist} is 4.51% greater than $L_{\text{random graph}}$, while C_{netlist} is 8,845 times larger than $C_{\text{random graph}}$ on average. This is consistent with the two small-world properties discussed in Section 5 and confirms that the netlists are indeed small-world networks.

Next, we aim to demonstrate the distinct roles of *Modular Clusters* and *Inter-Cluster Networks* in small-world networks. We measure L_{ICN} as the average distance between two randomly picked *Inter-Cluster Networks* cells, and L_{cluster} as the average distance for pairs from *Modular Clusters*. Similarly, C_{ICN} and C_{cluster} are measured separately for these two types of cells. The results are presented in Table 3. In the table, C_{ICN} and C_{cluster} show marginal difference. This can be attributed to the fact that the clustering coefficient C characterizes individual vertices (cells), and its granularity is too small to represent the global clustering properties of our *Modular Clusters*. However, L_{ICN} is smaller than L_{cluster} in the first six benchmarks, supporting our hypothesis that *Modular Clusters* form localized groupings, while *Inter-Cluster Networks*

Table 3. Comparison of characteristic Path Length L and Clustering Coefficient C for netlists from the ISPD 2005 benchmarks against random graphs. Additionally, L_{cluster} and C_{cluster} are measured for *Modular Clusters*, and L_{ICN} and C_{ICN} are measured for *Inter-Cluster Networks*.

Design	Characteristic Path Length L				Clustering Coefficient C			
	L_{netlist}	$L_{\text{random graph}}$	L_{cluster}	L_{ICN}	C_{netlist}	$C_{\text{random graph}}$	C_{cluster}	C_{ICN}
adaptec1	9.85	9.68	9.83	9.59	0.0631	13.0e-6	0.0627	0.0651
adaptec2	11.82	9.84	11.88	11.76	0.0303	6.7e-6	0.0301	0.0255
adaptec3	13.13	11.31	13.06	12.67	0.0599	8.0e-6	0.0587	0.0549
adaptec4	15.14	13.37	15.06	14.93	0.1033	9.0e-6	0.1017	0.1202
bigblue1	7.39	7.42	7.42	7.15	0.0399	10.0e-6	0.0401	0.0397
bigblue2	7.58	7.84	7.59	6.87	0.0487	9.3e-6	0.0486	0.0482
bigblue3	7.85	8.22	7.80	8.34	0.0924	2.3e-6	0.0923	0.0941
bigblue4	7.03	8.65	6.97	7.31	0.1145	4.3e-6	0.1147	0.1055
Average	9.97	9.54	9.95	9.83	0.0690	7.8e-6	0.0484	0.0692

Table 4. Proportion of *Inter-Cluster Network* cells on shortest paths versus their overall proportion in the netlist.

Design	Total ICN%	Shortest Path ICN%
adaptec1	3.48%	15.93%
adaptec2	4.23%	17.58%
adaptec3	4.63%	19.78%
adaptec4	6.08%	19.93%
bigblue1	7.07%	13.15%
bigblue2	3.72%	18.94%
bigblue3	6.38%	11.41%
bigblue4	5.67%	21.40%
Average	5.16%	17.26%

Table 5. Comparison of path length L with and without *Inter-Cluster Network* cells. Removing ICNs increases L by an average of 21.5% over the control.

Design	L_{netlist}	$L_{\text{remove ICN}}$	$L_{\text{remove control}}$
adaptec1	9.85	10.89	9.99
adaptec2	11.82	13.32	11.96
adaptec3	13.13	15.26	13.47
adaptec4	15.14	17.79	16.63
bigblue1	7.39	8.35	7.66
bigblue2	7.58	14.74	7.72
bigblue3	7.85	9.32	8.14
bigblue4	7.03	10.97	7.24
Average	9.97	12.58	10.35

act as shortcuts connecting distant clusters. In bigblue3 and bigblue4, L_{ICN} is larger than L_{cluster}. This is due to a significant number of *Inter-Cluster Networks* located near the placement edge. *Inter-Cluster Networks* are primarily composed of IO-related cells and scan chains, which have low connectivity to the rest of the graph.

To further validate our hypothesis, we conduct two experiments. The first experiment evaluates the proportion of *Inter-Cluster Network* cells on the shortest paths compared to their overall proportion in the netlist. In this experiment, we randomly select 10,000 pairs of cells and calculate the number of *Inter-Cluster Network* cells on these paths, then divide it by the total number of cells in the paths. Results are shown in Table 4. Since *Inter-Cluster Networks* act as shortcuts within the whole circuit, they appear more frequently along the shortest paths. As a result, *Inter-Cluster Networks* make up a higher proportion of the shortest paths compared to their scale in the whole circuit.

The first experiment shows that the proportion of *Inter-Cluster Network* cells is significantly higher on the shortest paths. Despite their low overall proportion of 5.2% on average in the entire netlist, their presence on the shortest paths is notably large, averaging 17.3%, more than three times greater.

In the second experiment, we examine how L changes when all *Inter-Cluster Network* cells are removed, denoted as $L_{\text{remove ICN}}$. For comparison, we conduct a control experiment where the same number of randomly selected cells are removed from the graph, denoted the result as $L_{\text{remove control}}$. Results are shown in Table 5.

The second experiment demonstrates that $L_{\text{remove ICN}}$ is, on average, 21.5% greater than $L_{\text{remove control}}$. Results from both experiments further validate the critical role *Inter-Cluster Networks* play in the connectivity of the netlist, acting as shortcuts that efficiently link distant clusters.

7 Conclusion

In this work, we discover the clustering phenomenon during placement evolution. To effectively extract these *Modular Clusters* and the *Inter-Cluster Networks*, we utilize *Cascade HDBSCAN* for noise-aware clustering across intermediate placements. We verify that *Modular Clusters* and *Inter-Cluster Networks* adhere to different Rent's rules, with *Inter-Cluster Networks* exhibiting stronger external connections than *Modular Clusters.* Additionally, we confirm that VLSI netlists exhibit small-world properties, where *Modular Clusters* form localized groupings and *Inter-Cluster Networks* serve as shortcuts connecting distant clusters. Our future studies include combining *Cascade HDBSCAN* with placement to study its impact on power, performance, area, and potential design complexity reduction.

8 Acknowledgement

This work is supported by NSF under Grant CCF-2110419.

References

[1] [n. d.]. Noise Aware Circuit Clustering. https://github.com/Hikipeko/noise-aware-circuit-clustering.

[2] Magnus Bakke Botnan and Michael Lesnick. 2023. An Introduction to Multiparameter Persistence. arXiv:2203.14289 [math.AT]

[3] Ricardo JGB Campello, Davoud Moulavi, and Jörg Sander. 2013. Density-based clustering based on hierarchical density estimates. In *Pacific-Asia conference on knowledge discovery and data mining*. Springer, 160–172.

[4] Yifan Chen, Zaiwen Wen, Yun Liang, and Yibo Lin. 2023. Stronger Mixed-Size Placement Backbone Considering Second-Order Information. In *2023 IEEE/ACM International Conference on Computer Aided Design (ICCAD)*. 1–9. https://doi.org/10.1109/ICCAD57390.2023.10323700

[5] Chung-Kuan Cheng, Andrew B. Kahng, Ilgweon Kang, and Lutong Wang. 2019. RePlAce: Advancing Solution Quality and Routability Validation in Global Placement. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 38, 9 (2019), 1717–1730. https://doi.org/10.1109/TCAD.2018.2859220

[6] Phillip Christie and Dirk Stroobandt. 2000. The interpretation and application of Rent’s rule. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 8, 6 (2000), 639–648.

[7] Chia-Tung Ho, Ajay Chandna, David Guan, Alvin Ho, Minsoo Kim, Yaguang Li, and Haoxing Ren. 2024. Novel Transformer Model Based Clustering Method for Standard Cell Design Automation. In *Proceedings of the 2024 International Symposium on Physical Design*. 195–203. https://doi.org/10.1145/3626184.3633314

[8] D.J.-H. Huang and A.B. Kahng. 1995. When clusters meet partitions: new density-based methods for circuit decomposition. In *Proceedings the European Design and Test Conference*. 60–64. https://doi.org/10.1109/EDTC.1995.470419

[9] Ramon Ferrer i Cancho, Christiaan Janssen, and Ricard V Solé. 2001. Topology of technology graphs: Small world patterns in electronic circuits. *Physical Review E* 64, 4 (2001), 046119.

[10] Andrew B Kahng, Ravi Varadarajan, and Zhiang Wang. 2022. RTL-MP: toward practical, human-quality chip planning and macro placement. In *Proceedings of the 2022 International Symposium on Physical Design*. 3–11.

[11] Yibo Lin, Zixuan Jiang, Jiaqi Gu, Wuxi Li, Shounak Dhar, Haoxing Ren, Brucek Khailany, and David Z. Pan. 2021. DREAMPlace: Deep Learning Toolkit-Enabled GPU Acceleration for Modern VLSI Placement. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 40, 4 (2021), 748–761. https://doi.org/10.1109/TCAD.2020.3003843

[12] Jingwei Lu, Hao Zhuang, Pengwen Chen, Hongliang Chang, Chin-Chih Chang, Yiu-Chung Wong, Lu Sha, Dennis Huang, Yufeng Luo, Chin-Chi Teng, and Chung-Kuan Cheng. 2015. ePlace-MS: Electrostatics-Based Placement for Mixed-Size Circuits. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 34, 5 (2015), 685–698. https://doi.org/10.1109/TCAD.2015.2391263

[13] Yi-Chen Lu, Tian Yang, Sung Kyu Lim, and Haoxing Ren. 2022. Placement Optimization via PPA-Directed Graph Clustering. *2022 ACM/IEEE 4th Workshop on Machine Learning for CAD (MLCAD)* (2022), 1–6.

[14] Tetsuaki Matsunawa, Bei Yu, and David Z. Pan. 2016. Laplacian eigenmaps and bayesian clustering based layout pattern sampling and its applications to hotspot detection and OPC. In *2016 21st Asia and South Pacific Design Automation Conference (ASP-DAC)*. 679–684. https://doi.org/10.1109/ASPDAC.2016.7428090

[15] Gi-Joon Nam, Charles J. Alpert, Paul Villarrubia, Bruce Winter, and Mehmet Yildiz. 2005. The ISPD2005 placement contest and benchmark suite. In *Proceedings of the 2005 International Symposium on Physical Design*. 216–220. https://doi.org/10.1145/1055137.1055182

[16] Umit Y Ogras and Radu Marculescu. 2006. " It’s a small world after all": NoC performance optimization via long-range link insertion. *IEEE Transactions on very large scale integration (VLSI) systems* 14, 7 (2006), 693–706.

[17] Duncan J Watts and Steven H Strogatz. 1998. Collective dynamics of ‘small-world’networks. *nature* 393, 6684 (1998), 440–442.