An Information-theoretic Perspective of Hierarchical Clustering on Graphs

Yicheng Pan[1] **Bingchen Fan**[1] **Pengyu Long**[1] **Feng Zheng**[1]

[1]State Key Laboratory of Complex & Critical Software Environment, Beihang University, Beijing, China

Abstract

The seminal work of [Dasgupta, 2016] has introduced a combinatorial cost function for hierarchical graph clustering that has inspired numerous follow-up studies adopting similar combinatorial approaches. In this paper, we investigate this problem from the *information-theoretic* perspective. We formulate a new cost function that is fully explainable and establish the relationship between combinatorial and information-theoretic perspectives. We present two algorithms for expander-like and well-clustered cardinality weighted graphs, respectively, and show that both of them achieve $O(1)$-approximation for our new cost function. Addressing practical needs, we consider non-binary hierarchical clustering problem, and propose a hyperparameter-free framework HCSE that recursively stratifies cluster trees through sparsity-aware partitioning, automatically determining the optimal hierarchy depth via an interpretable mechanism. Extensive experimental results demonstrate the superiority of our cost function and algorithms in binary clustering performance, hierarchy level identification, and reconstruction accuracy compared to existing approaches.

1 INTRODUCTION

Hierarchical clustering on graphs plays an important role in the structural analysis of a given data set. Understanding hierarchical structures on the levels of multi-granularity is fundamental in various disciplines including artificial intelligence, physics, biology, sociology, etc [Brown et al., 1992, Eisen et al., 1998, Gorban et al., 2008, Culotta et al., 2007]. Hierarchical clustering requires a cluster tree that represents a recursive partitioning of a graph into smaller clusters as the tree nodes get deeper. A leaf represents a graph node while a non-leaf node represents a cluster containing its descendant leaves. The root is the largest one containing all leaves. In this paper, we investigate the hierarchical clustering problem from the perspective of information theory. Our study is based on Li and Pan's structural information theory [Li and Pan, 2016] whose core concept named structural entropy measures the complexity of hierarchical networks. We propose from the information-theoretical perspective a new and fully explainable cost function that has advantages over the combinatorial ones, and also establish the relationship to them. In the remaining parts of this section, we introduce the combinatorial perspective first and then our contributions.

1.1 THE COMBINATORIAL PERSPECTIVE OF HIERARCHICAL CLUSTERING

Clustering is typically formulated as an optimization problem for some cost function. For hierarchical clustering, [Dasgupta, 2016] introduced his celebrated cost function as a combinatorial explanation for cluster trees. In this definition, similarity or dissimilarity between data points is represented by weighted edges. Taking the similarity-based metrics as an example, a cluster is a set of nodes with relatively denser intra-links compared with its inter-links, and in a good cluster tree, heavier edges tend to connect leaves whose lowest common ancestor (LCA) is assigned as deep as possible. This intuition leads to Dasgupta's cost function that is a bilinear combination of edge weights and the sizes of corresponding LCAs (refer to Section 2 for the formal definition).

Motivated by Dasgupta's cost function, [Cohen-Addad et al., 2019] proposed admissible cost functions. In their definition, the size of each LCA in Dasgupta's objective is generalized to be a function of the sizes of its left and right children. For all similarity-based graphs generated from a minimal ultrametric (see Section 2.2, [Cohen-Addad et al., 2019]), a cluster tree achieves the minimum cost if and only if it is a generating tree that is a "natural" ground truth tree in an axiomatic sense therein. A necessary condition of admissi-

bility of a cost function is that it achieves the same value for every cluster tree for a uniformly weighted clique that has no significant clustering structure in common sense. However, any slight deviation of edge weights would generally separate the two end-points of a light edge on a high level of its optimal (similarity-based) cluster tree. Thus, it seems that admissible cost functions, which take Dasgupta's cost function as a specific form, ought to be an unchallenged criterion in evaluating cluster trees since they are formulated by an axiomatic approach.

However, an admissible cost function seems imperfect in practice. The arbitrariness of optima of cluster trees for cliques indicates that the division of each internal node on an optimal cluster tree totally neglects the *balance* of its two children. Edge weight therein is the unique factor that decides the structure of optimal trees. But a balanced tree is commonly considered as an ideal candidate in hierarchical clustering compared to an unbalanced one. Balance of clustering has many applications in practice, for example, load balancing for cloud computing and management, network design, and avoidance of outlier cluster formation, etc. Balance has also been widely considered in partitioning and clustering problems, which has motivated some classic optimization problems such as balanced cut and balanced clustering (e.g., balanced k-means). Even clustering for cliques, a balanced partition should be preferable for each internal node. At least, an optimal cluster tree whose height is logarithm of graph size n is intuitively more reasonable than a caterpillar shaped cluster tree whose height is $n-1$. Moreover, a simple proof would imply that the optimal cluster tree for any connected graphs is binary. This property is not always useful in practice since a real system usually has its inherent number of hierarchies and a natural partition for each internal cluster. For instance, the natural levels of administrative division in a country is usually intrinsic, and it is not suitable to differentiate hierarchies for parallel cities in the same state. This structure cannot be obtained by simply minimizing admissible cost functions, and new non-binary clustering algorithms are worth study.

1.2 RELATED WORK

Along with the line of study on Dasgupta's cost function, several alternative objectives have been presented. All of them are bilinear functions of edge weights and some function of the corresponding LCAs. For Dasgupta's cost function and the worst case study, Dasgupta showed that by applying Arora's seminal recursive bipartition algorithm for the sparsest cut problem [Arora et al., 2009], we have $O(\log^{1.5} n)$-approximation. This guarantee was improved by [Roy and Pokutta, 2017] and [Charikar and Chatziafratis, 2017, Cohen-Addad et al., 2019] to $O(\log n)$ and $\sqrt{\log n}$, respectively. It is NP-hard to optimize the cluster tree [Dasgupta, 2016] and even a $O(1)$-approximation is impossible under the Small Set Expansion hypothesis [Roy and Pokutta, 2017, Charikar and Chatziafratis, 2017]. Beyond the worst case, Cohen-Addad et al. [Cohen-Addad et al., 2019] showed that a SVD-based algorithm achieves a $O(1+o(1))$-approximation for the stochastic block model with high probability. Manghiuc and Sun [Manghiuc and Sun, 2021] presented a $O(1)$-appromation algorithm for more generalized well-clustered graphs. The outline of their method is to utilize a flat clustering algorithm [Gharan and Trevisan, 2014] to obtain the underlying clusters first, and then some relatively easy heuristics for clustering in and out of these clusters are enough for the guarantee. Our proof follows this route also.

For other lines of this study, Moseley and Wang [Moseley and Wang, 2017] studied the dual of Dasgupta's cost function and showed that the average-linkage algorithm achieves a $(1/3)$-approximation. This factor has been improved by a series of works to 0.336 [Charikar et al., 2019], 0.4246 [Chatziafratis et al., 2020] and 0.585 [Alon et al., 2020], respectively. Cohen-Addad et al. [Cohen-Addad et al., 2019] considered maximization of Dasgupta's cost function for the dissimilarity-based metrics. They proved that the average-link and random partitioning algorithms achieve a $(2/3)$-approximation, which has been improved to 0.667 [Charikar et al., 2019], 0.716 [Rahgoshay and Salavatipour, 2021] and 0.74 [Naumov et al., 2021], respectively.

For non-binary cluster tree construction, the most popular and representative algorithm is LOUVAIN [Blondel et al., 2008]. More recently, a hierarchical label propagation based algorithm HLP has been presented [Rossi et al., 2020]. Both of these two algorithms construct a non-binary cluster tree with the same efficient framework, that is, the hierarchies are formed from bottom to top one by one. In each round, they invoke different flat clustering heuristics, Modularity and Label Propagation, respectively.

For other metrics of cluster trees, Charpentier and Bonald [Charpentier and Bonald, 2019] proposed an information-theoretic metric named tree sampling divergence (TSD) for hierarchical clustering. Two distributions of non-leaf nodes on cluster tree T of graph G are defined, one by sampling a random edge of G with probability proportional to edge weights and taking the LCA of its two endpoints, and the other, considered as a null model, by sampling two random leaves and taking their LCA. TSD is defined as the Kullback-Leibler divergence of these two distributions that measures the distance of them. Intuitively, if G is well clustered by T, then the mass of the former distribution will aggregate on LCAs that are far down from the root, which makes it far from the null model, and thus TSD is large. Otherwise, TSD is small. Note that this definition is quite different from our cost function since they measure the uncertainties of different distributions.

1.3 OUR CONTRIBUTIONS

Our study is able to address above issues to some extent. We summarize our contributions as follows.

(1) We formulate **a new and explainable cost function for cluster trees from the information-theoretic perspective**, which bridges combinatorial and information-theoretic perspectives of hierarchical clustering. For this cost function, the balance of cluster trees will be involved naturally as a factor just like we design optimal codes (cf. Huffman codes), for which the balance of probability over objects is fundamental in constructing an efficient coding tree. Our theoretical result on complete graphs (Proposition 2) and experimental results demonstrate the advantages of our cost function.

(2) For our new cost function, we present **two polynomial-time approximation algorithms** respectively for two cases of the conductance $\Phi(G)$ of a cardinality weighted graph G. Our first result shows that *any* cluster tree of G has an approximation factor $O(\Phi(G)^{-1})$ (Theorem 1). So any cluster tree achieves $O(1)$-approximation when $\Phi(G)$ is a constant. The second result is a $O(1)$-approximation algorithm for G that can be well clustered into a constant number of expander-like clusters (Theorem 2). The main idea of this algorithm is inspired by Manghiuc and Sun's work [Manghiuc and Sun, 2021], and our approximation factors for our new objective also match their results in these two cases. An expander-like graph has high conductance, and a well-clustered graph is its opposite. So, our results has covered two representative scenarios in the field of clustering study.

(3) For practical scenarios, we develop **a new interpretable framework for natural hierarchical clustering** that outputs a non-binary cluster tree. The idea of our framework is essentially different from the traditional recursive division or agglomeration ones. In our framework, the *sparsest level* of the cluster tree is stratified recursively. This coincides with the intuition that when we differentiate the hierarchies of a complex system, the clearest level should be stratified first, rather than in a rigid divisive or agglomerative fashion. Therefore, this framework has much better interpretability than the traditional ones.

(4) We develop **a new non-binary clustering algorithm** (HCSE) under the new clustering framework. To find the sparsest level in each iteration, we formulate two basic operations called *stretch* and *compress*, respectively. Guided by our new cost function, HCSE constructs a binary cluster tree in the stretch step, and then the sparsest level is stratified after compress. HCSE terminates when a specific criterion that intuitively coincides with the natural hierarchies is met, and *no* hyperparameter is needed. Our experimental results demonstrate that HCSE has a great advantage in both finding the intrinsic number of hierarchies and hierarchy reconstructions.

2 A COST FUNCTION FROM INFORMATION-THEORETIC PERSPECTIVE

In this section, we first introduce the structural information theory of [Li and Pan, 2016] as well as the combinatorial cost functions of [Dasgupta, 2016] and [Cohen-Addad et al., 2019]. Then we propose a new cost function that is developed from structural information theory and establish the relationship between the information-theoretic and combinatorial perspectives.

Notations. Let $G = (V, E, w)$ be an undirected weighted graph with a set of vertices V, a set of edges E and a weight function $w : E \to \mathbb{R}^+$, where $\mathbb{R}^+$ denotes the set of all positive real numbers. An unweighted multigraph can be viewed as a cardinality weighted one whose edge weight is the number of parallel edges. For each vertex $u \in V$, denote by $d_u = \sum_{(u,v)\in E} w(u, v)$ the weighted degree of u. For a subset of vertices $S \subseteq V$, define the volume of S to be the sum of degrees of vertices. We denote it by $\mathrm{vol}(S) = \sum_{u\in S} d_u$. The conductance of S, denoted by $\Phi_G(S)$, is defined as $w(E(S, \overline{S}))/\mathrm{vol}(S)$ where $w(E(S, \overline{S}))$ is the total weight of edges crossing S and $\overline{S}$, and the conductance of graph G is the minimum conductance over all subset S whose volume is at most $\mathrm{vol}(V)/2$. We denote by $G[S]$ the subgraph induced by S. A cluster tree T for graph G is a rooted tree with $|V|$ leaves, each of which is labeled by a distinct vertex $v \in V$. Each non-leaf node on T is labeled by a subset S of V that consists of all the leaves treating S as an ancestor. For each node α on T, denote by α^- the parent of α, and by $|\alpha|$ its size. For each pair of leaves u and v, denote by $u \vee v$ the LCA of them on T.

Structural entropy of graphs. We first give the formal definition of structural entropy, and then introduce the idea behind it. Given a weighted graph $G = (V, E, w)$ and a cluster tree T for G, the *structural entropy of G on T* is defined as

$$\mathcal{H}^T(G) = -\sum_{\alpha\in T} \frac{g_\alpha}{\mathrm{vol}(V)} \log \frac{\mathrm{vol}(\alpha)}{\mathrm{vol}(\alpha^-)},^{1} \qquad (1)$$

where g_α denotes the sum of weights of edges in G with exactly one end-point in the set of vertices corresponding to α. The *structural entropy of G* is defined as the minimum one among all cluster trees, denoted by $\mathcal{H}(G) = \min_T\{\mathcal{H}^T(G)\}$.

The main idea of structural information is based on the random walk that has been well-known as an effective tool

[1]For notational convenience, for the root λ of T, set $\lambda^- = \lambda$. So the term for λ in the summation is 0. In this paper, the default base of logarithm is always 2.

in revealing clustering structures. In this theory, the random walk is encoded with a certain rule by using a high-dimensional encoding system for a graph G. In each step of the random walk, a neighbor is randomly chosen with probability proportional to edge weights, and so it has a stationary distribution on vertices that is proportional to vertex degree. For connected graphs, this stationary distribution is unique, but not for disconnected ones. Here, we consider this canonical one for all graphs. So, to position a random walk under its stationary distribution, the amount of information needed is typically the Shannon's entropy, denoted by $\mathcal{H}^{(1)}(G) = -\sum_{v\in V} \frac{d_v}{\text{vol}(V)} \log \frac{d_v}{\text{vol}(V)}$. By Shannon's noiseless coding theorem, $\mathcal{H}^{(1)}(G)$ is the tight lower bound of average code length generated from the *memoryless* source for one step of the random walk. However, dependence of locations may shorten the code length.

The core concept of structural information is structural entropy. To demonstrate the underlying idea, consider flat (rather than hierarchical) clustering as a simple case example. Structural entropy measures the uncertainty (information) of locations under the stationary distribution of the random walk. This uncertainty consists of two parts, one from that of clusters, and the other from that of vertices in its cluster. These two parts can also be viewed as two parts of an encoding for each vertex. By the additivity of Shannon's entropy, the sum of these two parts of uncertainty should be equal to the original one of locations for memoryless sampling under the stationary distribution. The key idea is as follows. For one step of the random walk from a vertex u to v, to measure the uncertainty of v, if u and v are in the same cluster, the uncertainty of clusters may not be involved. For a good clustering structure, this kind of in-cluster movements will happen with high probability, and thus the uncertainty of clusters will decrease sharply due to the dependence of steps of random walks. The resulting measurement of uncertainty under the stationary distribution is exactly the structural entropy of flat clustering.

The above idea can be generalized to hierarchical clustering. For each level of a cluster tree, the uncertainty of locations during the random walk is measured by the entropy of the stationary distribution on the clusters of this level. Consider an encoding for every cluster, including the leaves. Each non-root node α is labeled by its order among the children of its parent α^-. So, the amount of self-information of α within this local parent-children substructure is $-\log(\text{vol}(\alpha)/\text{vol}(\alpha^-))$, which is also roughly the length of Shannon code for α and its siblings. The codeword of α consists of the sequential labels of nodes along the unique path from the root (excluded) to itself (included). For one step of the random walk from u to v in G, to indicate v, we omit from v's codeword the longest common prefix of u and v that is exactly the codeword of $u \vee v$. This means that the random walk takes this step in the cluster $u \vee v$ (and also in $u \vee v$'s ancestors) and the uncertainty at this level may not be involved. Therefore, intuitively, a quality similarity-based cluster tree would trap the random walk with high frequency in the deep clusters that are far from the root, and the long codeword of $u \vee v$ would be omitted. This shortens the average code length of the random walk. Note that we ignore the uniqueness of decoding since a practical design of codewords is not our purpose. We utilize this scheme to evaluate hierarchical structures.

Then we formulate the above scheme and measure the average code length as follows. Given a weighted graph $G = (V, E, w)$ and a cluster tree T for G, note that under the stationary distribution, the random walk takes one step out of a cluster α on T with probability $g_\alpha/\text{vol}(V)$. Therefore, the aforementioned uncertainty measured by the average code length is exactly $\mathcal{H}^T(G)$ that has been defined as the structural entropy of G on T (Eq. (1)). To minimize this uncertainty, the structural entropy $\mathcal{H}(G)$ of G has been defined as the minimum one among all cluster trees. Note that the structural entropy of G on the trivial 1-level cluster tree is consistent with the previously defined $\mathcal{H}^{(1)}(G)$. It doesn't have any non-trivial cluster.

Combinatorial explanation of structural entropy. Dasgupta's cost function [Dasgupta, 2016] of a cluster tree T for graph $G = (V, E)$ is defined to be $c^T(G) = \sum_{(u,v)\in E} w(u, v)|u \vee v|$. The admissible cost function introduced by [Cohen-Addad et al., 2019] generalizes the term $|u \vee v|$ in the definition of $c^T(G)$ to be a general function $g(|L|, |R|)$ for a binary cluster tree, where L and R are the two children of $u \vee v$, respectively. Thus, Dasgupta had defined $g(x, y) = x + y$. For both definitions, the optimal hierarchical clustering of G is in correspondence with a cluster tree of minimum cost in the combinatorial sense that heavy edges are cut as far down the tree as possible.

The following proposition establishes the relationship between structural entropy and this kind of combinatorial form of cost functions.

Proposition 1. *For a weighted graph $G = (V, E, w)$, minimizing $\mathcal{H}^T(G)$ (over T) is equivalent to minimizing the cost function*

$$cost^T(G) = \sum_{(u,v)\in E} w(u, v) \log vol(u \vee v). \qquad (2)$$

We defer the proof of Proposition 1 to Appendix A. We denote by *cost(SE)* the cost function in Proposition 1 from now on. Proposition 1 indicates that when we view g as a function of the LCA rather than that of its size and define $g(u, v) = \log \text{vol}(u \vee v)$, the "admissible" function becomes equivalent to structural entropy in evaluating cluster trees, although it is not admissible any more.

So what is the difference between these two cost functions? As stated by [Cohen-Addad et al., 2019], an important axiomatic hypothesis for admissible function, thus also for

Dasgupta's cost function, is that the cost for every binary cluster tree of an unweighted clique is identical. So any binary tree for clustering on cliques is reasonable, which coincides with the common sense that structureless datasets can be organized hierarchically free. However, for structural entropy, the following theorem indicates that balanced organization is of importance even though for structureless datasets.

Proposition 2. *For any positive integer n, let K_n be the clique of n vertices with identical weight on every edge. Then a cluster tree T of K_n achieves minimum structural entropy if and only if T is a balanced binary tree, that is, the two children clusters of each non-leaf node of T have difference in size at most* 1.

The proof of Proposition 2 is a bit technical, and we defer it to Appendix B. The intuition behind Proposition 2 is that balanced codes are the most efficient encoding scheme for unrelated data, cf. Huffman codes (recall the main idea behind the structural entropy). So the codewords of the random walk that jumps freely among clusters on each level of a cluster tree have the minimum average length if all the clusters on this level are in balance.

Why is Proposition 2 able to explain that cost(SE) has taken balance into account in hierarchical clustering? We remark that two factors impact the quality of hierarchical clustering and should be considered in a cost function, one is edge weight, and the other is balance. Edge weight means structure, which should be the main factor in clustering. If one needs to show that balance is also a factor, it is better to exclude the influence of structure. Complete graph is the most ideal candidate, because it is unstructured. We utilize complete graph since it is a counterpart to Dasgupta's equal-cost property of unstructured graphs with his cost function.

Note that for regular graphs (e.g. cliques), replacing $\mathrm{vol}(u \vee v)$ by $|u \vee v|$ is equivalent for optimization. [Dasgupta, 2016] claimed that for the clique K_4 of four vertices, a balanced tree is preferable when replace $|u \vee v|$ by $g(|u \vee v|)$ for any strictly increasing concave function g with $g(0) = 0$. However, it is worth noting that this does not hold for all those concave functions. For example, it is easy to check that for $g(x) = 1 - e^{-x}$, the cluster tree of K_6 that achieves minimum cost partitions K_6 into K_2 and K_4 on the first level, rather than K_3 and K_3. In contrast, Proposition 2 shows that for all cliques, balanced trees are preferable when concave g is a logarithmic function.

It is worth noting that the admissible function introduced by [Cohen-Addad et al., 2019] is defined from the viewpoint of generating trees. A generating tree T of a similarity-based graph G is generated from a minimal ultrametric and achieves the minimum cost. In this definition, the monotonicity of edge weights between sibling clusters from bottom to top on T, which is given by [Cohen-Addad et al., 2019] as a property of a "natural" ground-truth hierarchical clustering, is the unique factor when evaluating T. However, as discussed earlier, Proposition 2 implies that for cost(SE), besides edge weights, the balance factor of cluster trees is implicitly involved as another factor. Moreover, for cliques, the minimum cost should be achieved on every subtree, which makes an optimal cluster tree balanced everywhere. This unique optimal clustering for cliques is also robust in the sense that a slight perturbation to the minimal ultrametric, which can be considered as slight variations to the weights of a batch of edges, will not change the optimal cluster tree wildly due to the holdback force of balance.

3 APPROXIMATION ALGORITHMS FOR COST(SE)

In this section, we present approximation algorithms for expander-like and well-clustered graphs, respectively. These approximation factors work for cardinality edge weights whose value is at least one (e.g. the multiplicity of edges).

Why cardinality weights? In general, the term $\log \mathrm{vol}(u \vee v)$ in Eq. 2 and cost(SE) can become negative as the volume of $u \vee v$ varies, causing the approximation analysis to fail. The cardinality weight function w is at least one, which makes cost(SE) non-negative. Although the dependence of cost(SE) on the scale of edge weights violates the scale-invariance principle, we emphasize that $\mathcal{H}^T(G)$ is scale-invariant and Proposition 1 holds for any scale variation. In this paper, we present approximation algorithms for cost(SE) in well-defined settings, and our approximation guarantees hold for all graphs with edge weights at least one.

Theorem 1. *For any cardinality weighted graph $G = (V, E, w)$ with conductance $\Phi(G)$, it holds that for any cluster tree, $cost^T(G) = O(\Phi(G)^{-1}) \cdot OPT$, where OPT is the minimum cost(SE) of G.*

We defer the proof of Theorem 1 to Appendix C. When $\Phi(G)$ is a constant, Theorem 1 implies that any cluster tree achieves $O(1)$-approximation for expander-like graphs. We remark that expander-like graphs do not have good clustering structures, and so the cost of any cluster tree is constant times near to the universal upper bound (as shown in the proof). This result is in fact a lower bound of cost(SE) for graphs without clustering structure, which demonstrates the rationality of our objective. Considering balance as an important factor, we present a Huffman-merging heuristic (Algorithm 1). It will serve as a subroutine within expander-like clusters in the algorithm for well-clustered graphs.

Next, we consider well-clustered graphs that are composed by a collection of densely-connected components with high inner conductance and weakly interconnections. Our settings for well-clustered graphs is the same as those in [Manghiuc and Sun, 2021]. We start from the following (Φ_{in}, Φ_{out})-decomposition presented by Gharan and Tre-

Algorithm 1 HuffmanMerge

Input: a graph $G = (V, E, w)$.
Output: a cluster tree T of G.
Create n singleton trees.
while there are at least two trees **do**
  Select the two trees T_1 and T_2 with the least volumes.
  Construct a new tree T_0 with T_1 and T_2 as two subtrees of the root.
return the resulting binary tree T_0.

Algorithm 2 Balanced Binary Merge (BBM)

Input: a graph $G = (V, E, w)$, an integer $k \geq 2$ such that $\lambda_k > 0$.
Output: a cluster tree T of G.
Apply the partitioning algorithm in Lemma 1 on input (G, k) to obtain $\{P_i\}_{i=1}^{l}$ for some $l < k$.
Sort $P_1, ..., P_l$ be such that $\text{vol}_G(P_i) \leq \text{vol}_G(P_{i+1})$, for all $1 \leq i < l$.
Let $T_i = \text{HuffmanMerge}(G[P_i])$.
Initialize $T = T_1$.
for $i = 2, ..., l$ **do**
  Let T be the tree with T and T_i as its two children.
return T.

visan [Gharan and Trevisan, 2014]. Let λ_k be the k-th smallest eigenvalue of the normalized Laplacian matrix of G and $\Phi_G(S)$ be the conductance of a vertex set S in G.

Lemma 1. *([Gharan and Trevisan, 2014], Theorem 1.5) Let $G = (V, E, w)$ be a graph such that $\lambda_k > 0$, for some $k \geq 1$. Then, there is a local search algorithm that finds a l-partition $\{P_i\}_{i=1}^{l}$ of V, for some $l < k$, such that for every $1 \leq i \leq l$, $\Phi_G(P_i) = \mathcal{O}(k^6\sqrt{\lambda_{k-1}})$ and $\Phi(G[P_i]) = \Omega(\lambda_k^2/k^4)$.*

Lemma 1 implies that, when G exhibits a clear clustering structure, there is a partition $\{P_i\}_{i=1}^{l}$ of V such that for each P_i both the outer and inner conductance can be bounded. This is a crucial insight that we can use $\{P_i\}_{i=1}^{l}$ directly to construct a cluster tree.

Our algorithm consists of two phases: Partition and Merge. In the Partition phase, it invokes the algorithm in Lemma 1 to partion V into subsets $\{P_i\}_{i=1}^{l}$. In the Merge phase, it combines the trees in a "caterpillar style" according to an increasing order of their volumes. This algorithm is described as Algorithm 2. Note that Algorithm 2 degenerates to Algorithm 1 when $k = 2$. For the approximation guarantee, we have the following theorem.

Theorem 2. *Let $G = (V, E, w)$ be a cardinality weighted graph such that $\lambda_k > 0$ for some $k \geq 1$. Then Algorithm 2 constructs in polynomial time a cluster tree T of G that achieves $O\left(\frac{1}{(1-\alpha)\beta}\log\frac{k}{1-\alpha}\right)$-approximation for cost$^T(G)$, where $\alpha = O(k^6\sqrt{\lambda_{k-1}})$, $\beta = \Omega(\lambda_k^2/k^4)$. Consequently, when $\lambda_k = \Omega(1/poly(k))$ and $\lambda_{k-1} = O(1/k^{12})$ such that $\alpha < 1 - \rho$ for some constant $\rho \in (0, 1)$, Algorithm 2 achieves $O(poly(k))$-approximation. In addition, when k is a constant, Algorithm 2 achieves $O(1)$-approximation.*

The proof of Theorem 2 is given in Appendix D. It follows the intuition that the cost contributed by the inner edges of clusters is upper bounded due to the cluster volumes, and that contributed by the outer edges is upper bounded due to their few number. Our proof is more natural than that in [Manghiuc and Sun, 2021] since it circumvents the complicated decomposition into critical nodes.

4 NON-BINARY HIERARCHICAL CLUSTERING ALGORITHM HCSE

In this section, we develop a non-binary hierarchical clustering algorithm. At present, most algorithms for hierarchical clustering can be categorized into two frameworks: top-down division and bottom-up agglomeration [Cohen-Addad et al., 2019]. The top-down division approach usually yields a binary tree by recursively dividing a cluster into two parts by a cut-based subroutine. But a binary clustering tree is far from practical application. For practical use, bottom-up agglomeration that is also known as hierarchical agglomerative clustering (HAC) is commonly preferable. It constructs a cluster tree from leaves to the root recursively, during each round of which the newly generated clusters shrink into single vertices. We remark that there is no proper cost function for non-binary hierarchical clustering yet, and all existing algorithms for this purpose cannot be evaluated by any cost. We provide a new one by using cost(SE) in its subroutines. However, it is unclear and worth study whether cost(SE) and cost(Das) fit to evaluate non-binary cluster trees.

Our algorithm jumps out of these two frameworks. We establish a new one that stratifies the *sparsest* level of a cluster tree recursively rather than in a sequential order. In general, guided by cost(SE), we construct a $(k+1)$-level cluster tree from the previous k-level one, during which we find the level whose stratification makes the average cost in a local reduced subgraph decrease most, and then differentiate it into two levels. The process of stratification consists of two basic operations: *stretch* and *compression*. In stretch step, given an internal node of a cluster tree, a local binary subtree is constructed, while in compression step, the paths that are overlong from the root to leaves on the binary tree are compressed by shrinking tree edges that make the cost reduce most. The intuition behind the "stretch-and-compress" scheme is as follows. First, we run a fast and simple, but probably rough clustering algorithm to obtain a binary cluster subtree. So, after stretch, we have unfolded all the potential hierarchies such that the sparsest level can probably be seen. Second, we compress every overlong path that is supposed to get through each level of this subtree,

during which, the tree edge on the sparsest level whose compression makes too many graph edges amplify the sizes of their LCAs greatly will be retained. This edge is expected in the backbone of the final non-binary cluster tree. We remark that this framework can be collocated with any cost function and any binary cluster tree algorithm. For computational efficiency, we will adopt in our experiments an HAC construction of binary cluster trees with no hyper-parameter in stretch steps.

Stretch and compress. Given a cluster tree T for graph $G = (V, E)$, let u be an internal node on T and $v_1, v_2, \ldots, v_\ell$ be its children. We call this 1-level local parent-children structure rooted at u to be a *u-triangle* of T, denoted by T_u. These two operations are defined on u-triangles. Note that each child v_i of u represents a cluster in G. We reduce G by shrinking each v_i to be a single vertex v_i' while maintaining each inter-link and ignoring each internal edge of v_i. This reduction captures the connections of clusters at this level in the parent cluster u. The stretch operation constructs a binary tree for u-triangle with an HAC process. Initially, view each v_i' as a cluster and then recursively merge two clusters into a new one such that cost(SE) drops most. This yields a binary subtree T_u' rooted at u which has $v_1, v_2, \ldots, v_\ell$ as leaves. Then the compression operation is proposed to reduce the height of T_u' to be 2. Let $\hat{E}(T')$ be the set of edges on T', each of which appears on a path of length more than 2 from the root of T' to some leaf. Denote by $\Delta(e)$ the amount of structural entropy enhanced by shrinking edge e. We pick from $\hat{E}(T_u')$ the edge e with least $\Delta(e)$. Note that compressing a tree edge makes the grandchildren of some internal node to be children, which must amplify the cost. The compression operation picks the least amplification. The processes of stretch and compress are illustrated in Figure 3 and stated as Algorithms 3 and 4, respectively.

Algorithm 3 Stretch(T_u)

Input: a u-triangle T_u.
Output: a binary tree rooted at u.
Let $\{v_1, v_2, \ldots, v_\ell\}$ be the set of leaves of T_u.
Compute $\eta(a, b)$ which is the cost reduced by merging siblings a, b into a single cluster.
for $t \in [\ell - 1]$ **do**
  $(\alpha, \beta) \leftarrow \arg\max_{(a,b) \text{ are siblings}} \{\eta(a, b)\}$.
  Add a new node γ.
  $\gamma.parent \leftarrow \alpha.parent$.
  $\alpha.parent = \gamma$.
  $\beta.parent = \gamma$.
return T_u.

Sparsest level. Let U_j be the set of j-level nodes on cluster tree T, that is, U_j is the set of nodes each of which has distance j from T's root. Suppose that the height of T is k, then $U_0, U_1, \ldots, U_{k-1}$ is a partition for all internal nodes of T. For each internal node u, define

Algorithm 4 Compress(T)

Input: a binary tree T.
while T's height is more than 2 **do**
  $e \leftarrow \arg\min_{e' \in \hat{E}(T)} \{\Delta(e')\}$
  Denote $e = (u, v)$ where u is the parent of v
  for $w \in v.children$ **do**
    $w.parent \leftarrow u$
  Delete v from T

$\mathcal{H}(u) = -\sum_{v:v^-=u} \frac{g_v}{\text{vol}(V)} \log \frac{\text{vol}(v)}{\text{vol}(u)}$. Note that $\mathcal{H}(u)$ is the partial sum contributed by u in $\mathcal{H}^T(G)$. After a "stretch-and-compress" round on u-triangle, denote by $\Delta\mathcal{H}(u)$ the structural entropy by which the new cluster tree reduces. Since the reconstruction of u-triangle stratifies cluster u, $\Delta\mathcal{H}(u)$ is always non-negative. Define the sparsity of u to be $\text{Spar}(u) = \Delta\mathcal{H}(u)/\mathcal{H}(u)$, which is the relative variation of structural entropy in cluster u. From the information-theoretic perspective, this means that the uncertainty of random walk can be measured locally in any internal cluster, which reflects the quality of clustering in this local area. At last, we define the *sparsest level* of T to be the j-th level such that the average sparsity of triangles rooted at nodes in U_j is maximum, that is $\arg\max_j \{\overline{\text{Spar}}_j(T)\}$, where $\overline{\text{Spar}}_j(T) = \sum_{u \in U_j} \text{Spar(u)}/|U_j|$. Then stratification works for the sparsest level of T. This process is illustrated in Figure 4 in Appendix G.

For a given positive integer k, to construct a cluster tree of height k, we start from the trivial 1-level cluster tree that involves all vertices of G as leaves. Then we do not stop stratifying at the sparsest level recursively until a k-level cluster tree is obtained. The pseudocode is described as Algorithm 5 in Appendix G.

To determine the height of the cluster tree automatically, we derive the natural clustering from the variation of sparsity on each level. Intuitively, a natural hierarchical cluster tree T should have not only sparse boundary on clusters, but also low sparsity for triangles of T, which means that further stratification within the reduced subgraphs corresponding to such triangles makes little sense. For this reason, we consider the inflection points of the sequence $\{\delta_t(\mathcal{H})\}_{t=1,2,\ldots}$, where $\delta_t(\mathcal{H})$ is the structural entropy by which the t-th round of stratification reduces. Formally, denote $\Delta_t\delta = \delta_{t-1}(\mathcal{H}) - \delta_t(\mathcal{H})$ for each $t \geq 2$. We say that $\Delta_t\delta$ is an inflection point if both $\Delta_t\delta \geq \Delta_{t-1}\delta$ and $\Delta_t\delta \geq \Delta_{t+1}\delta$ hold. Our algorithm finds the least t such that $\Delta_t\delta$ is an inflection point and fix the height of the cluster tree to be t (Note that after $t-1$ rounds of stratification, the number of levels is t). The pseudocode is described as Algorithm 6 in Appendix G.

Time complexity. The running time of HCSE on graph $G = (V, E)$ for which $|V| = n$ and $|E| = m$ depends mainly on the iterations of stratification for the sparsest

level. For each round of t-HCSE in Algorithm 6, since the change of structure entropy can be calculated incrementally and locally when merge siblings, the time complexity for the Stretch process is $O((m+n)\log n)$. Note that the LCA of each edge of G can be recorded during Stretch. Since at most n times of shrinking operations on tree edges will happen, and $\Delta(e)$ can be calculated locally, the time complexity for the Compress process is $O((m+n)\log n)$. Combining these two, the time complexity of HCSE (and also k-HCSE) is $O(k(m+n)\log n$. Practically, k is usually very small, for example $O(\log n)$. In this case, the time complexity is merely $O((m+n)\log^2 n)$.

5 EXPERIMENTS

In this section, we evaluate experimentally our binary clustering algorithm BBM and the non-binary hierarchical clustering algorithm HCSE mainly on synthetic networks generated from the stochastic block model (SBM) [Ana and Jain, 2003] and the hierarchical stochastic block model (HSBM) [Lyzinski et al., 2017], respectively. Due to the lack of ground truth for hierarchical clustering on real datasets, we conduct our experiments on the Amazon network that is the only one we suppose to have overlapping, possibly hierarchical, ground-truth clusters (see Appendix F.5). All algorithms were implemented in python 3.8 and the experiments were performed using an Intel(R) Core(TM) i5-12400 CPU @ 2.50GHz processor, with 16 GB RAM. For the source codes, please refer to `https://github.com/Hardict/HCSE`.

Binary clustering: cost and balancedness on SBM graphs. We evaluate our binary clustering algorithm BBM in two aspects: cost and balancedness. We denote Dasgupta's cost function by cost(Das). The baselines include four linkage-based methods [Cohen-Addad et al., 2019]: average-linkage (AL), single-linkage (SL), complete-linkage (CL) and Linkage++ (L++, previous state-of-the-art for cost(Das)). PruneMerge proposed by [Manghiuc and Sun, 2021] always has complicated operations on critical nodes and sometimes is hard to terminate in short time. So we do not include it as a baseline.

The datasets we use are random graphs generated from SBM. We remark that a good clustering algorithm should always treat clustering structure as the most significant impact, and when cluster outlines become vague, the balance factor makes an effect. Therefore, we produce a series of SBM graphs, each of which has a significant variation in cluster sizes. Two strategies for generating clusters of different sizes are used, one is a totally biased way and the other is random. More details are introduced in Appendix E.

For balance evaluation on a binary tree, we define three indices, for each of which, the smaller, the better. The first is *size balance index*, denoted by B_{size}. For each node α on T, if α is a non-leaf node, let α_ℓ and α_r be its two children, and α's size balance index is defined as a normalized size difference between the children, that is $\mathrm{B}_{\text{size}}(\alpha) = ||\alpha_\ell| - |\alpha_r||/|\alpha|$. If α is a leaf, $\mathrm{B}_{\text{size}}(\alpha) = 0$. The size balance index of T is a weighted sum of $\mathrm{B}_{\text{size}}(\alpha)$, that is $\mathrm{B}_{\text{size}}(T) = \sum_{\alpha\in T}\rho_\alpha \cdot \mathrm{B}_{\text{size}}(\alpha)$, where $\rho_\alpha = |\alpha|/\sum_{\alpha'\in T}|\alpha'|$. So, after simplicity, we have

$$\mathrm{B}_{\text{size}}(T) = \sum_{\alpha\in T}\frac{||\alpha_\ell| - |\alpha_r||}{\sum_{\alpha'\in T}|\alpha'|}.$$

The second index is the counterpart of B_{size} that uses volume, named *volume balance index* and denoted by B_{vol}. Formally,

$$\mathrm{B}_{\text{vol}}(T) = \sum_{\alpha\in T}\frac{|\mathrm{vol}(\alpha_\ell) - \mathrm{vol}(\alpha_r)|}{\sum_{\alpha'\in T}\mathrm{vol}(\alpha')}.$$

The last index is *depth balance index*, denoted by B_{dep}. $\mathrm{B}_{\text{dep}}(T)$ is simply the standard deviation of the depths of all leaves on T.

We show the cost and balancedness results on SBM graphs in Table 1. On both datasets with two different generating strategies, our algorithm BBM achieves the best cost(SE), and also competitive cost(Das) with Linkage++. Meanwhile, BBM achieves the best balances factors for all three indices, which means that small cost(SE) corresponds to good balance of cluster trees in the case that the underlying ground truth clusters can be well constructed. More comprehensive experimental results are provided in Appendix F.1.

Non-binary clustering: NMIs on HSBM graphs. To evaluate the effective of HCSE, we adopt two popular and representative non-binary clustering methods as baselines, LOUVAIN [Blondel et al., 2008] and HLP [Rossi et al., 2020], that follow the traditional bottom-up agglomeration framework. LOUVAIN admits a sequential input of vertices. To avert the worst-case trap, vertices come randomly, and the resulting cluster tree depends on their order. HLP invokes the common Label Propagation algorithm recursively, and so it cannot be guaranteed to avoid under-fitting in each round. This can be seen in our experiments on synthetic datasets, for which these two algorithms sometimes miss ground-truth levels. To evaluate the effectiveness of our cost function, we replace the stretch step of HCSE by linkage-based methods AL, SL and CL, each of which also depends on no hyper-parameter. Our experiments evaluate our frame work incorporated with these methods in the Stretch step, for which we denote by HC_{ave}, HC_{sin}, HC_{com} these three algorithms, respectively.

We utilize the k-level HSBM (introduced in Appendix E.2) for the task to reconstruct the ground-truth HSBM cluster trees. Table 4 in Appendix 4 demonstrates the results on two groups of random graphs generated from HSBM whose heights are $k = 4, 5$, respectively. We compare the Normalized Mutual Information (NMI) at each level of the ground-truth cluster tree. Due to the randomness in LOUVAIN and

	(Biased)					(Random)				
Method	B_{size}	B_{vol}	B_{dep}	cost(Das)	cost(SE)	B_{size}	B_{vol}	B_{dep}	cost(Das)	cost(SE)
AL	0.619	0.666	6.79	1.07E7	7.16E5	0.411	0.438	2.83	4.68E5	6.20E4
SL	0.232	0.241	1.41	1.62E7	7.43E5	0.263	0.271	1.51	7.45E5	6.72E4
CL	0.977	0.978	64.7	1.43E7	7.50E5	0.933	0.935	22.2	7.08E5	6.75E4
L++	0.897	0.921	30.2	**9.45E6**	7.12E5	0.721	0.735	7.31	**3.82E5**	6.09E4
BBM	**0.0148**	**0.0681**	**0.181**	9.48E6	**7.03E5**	**0.0981**	**0.0980**	**0.645**	3.94E5	**6.06E4**

Table 1: Cost and balancedness on SBM graphs. "Biased" and "Random" denote the two strategies for generating cluster sizes for SBM. Five clusters are generated by Biased strategy, and their sizes are set to be $[32, 32, 64, 128, 256]$ with ± 5 random perturbation on each figure. The sizes of five clusters generated by Random strategy are uniformly and randomly drawn from integers in the interval $[2^4, 2^6)$. The probability of presence for intra-cluster edges is 0.9 and that for inter-cluster edges is 0.1. All values are averaged over 100 trials.

convergence of HLP, we choose the most effective strategy and pick the best results in five runs for both of these two baselines. In contrast, our algorithm HCSE yields deterministic results. We need to emphasize that since there is no proper cost function for non-binary clustering yet, we do not evaluate HCSE with any cost.

The main results are summarized as follows. HCSE is always able to find the correct height k, while HLP and LOUVAIN are not. Although LOUVAIN is comparable with HCSE in NMI values, it cannot find the correct level number in any group. HLP and the three linkage-based methods cannot achieve the best NMI for any probability vector, even if we calculate the NMI between each ground-truth level and the one that achieves the maximum NMI in the resulting tree when the height of cluster tree is incorrect. The reason why HCSE performs not so well as LOUVAIN on deep levels is as follows. The accuracy of HCSE for deep levels depends on that for the intermediate levels that are probably stratified earlier. The early errors will accumulate later to the bottom. Comparatively, LOUVAIN starts with the deepest level in a bottom-up fashion. So it has better NMIs on deep levels than HCSE, but due to error accumulation also, it has worse NMIs on high levels, even miss the top level. For the three linkage-based methods, the structure of resulting non-binary tree is quite different from the ground truth. This indicates that the binary tree after Stretch are not accurate enough in the sense that the real sparse levels have not been properly unfolded by the internal nodes of the binary tree, and so they cannot be stratified properly. This also demonstrates the advantage of our cost function in binary clustering. More ablation experiments are provided in Appendix F.3.

6 CONCLUSIONS AND DISCUSSIONS

In this paper, we investigate the hierarchical clustering problem on graphs from an information-theoretic perspective and propose a new cost function that relates to the combinatorial objective raised by Dasgupta [Dasgupta, 2016]. We present two $O(1)$-approximation algorithms for it on two canonical kinds of graphs, i.e., expander-like and well-clustered cardinality weighted ones, respectively. For non-binary hierarchical clustering, we propose a new interpretable framework that stratifies the sparsest level of the cluster tree recursively, which can be collocated with any binary clustering algorithm. We also present an interpretable strategy to find the intrinsic number of levels without any hyper-parameter. The experimental results have verified the effectiveness of our cost function and algorithms.

There are several directions that are worth further study. The first problem is about the complexity of minimizing cost(SE). It is unknown whether computing $\min$ cost(SE) is NP-hard. For approximation guarantee, note that there is a trivial factor $\log \mathrm{vol}(V)$ for the cardinality weighted graphs simply due to the definition of cost(SE). However, there is no non-trivial factor in the worst case yet. The second problem is about the relationship between the concavity of g of the cost function and the balance of the optimal cluster tree. It can be checked that for cliques, being concave is not a sufficient condition for total balance. Whether is it a necessary condition? Moreover, is there any explicit necessary and sufficient condition for total balance of the optimal cluster tree for cliques? The third one is about more precise characterizations for "natural" hierarchical clustering whose depth is limited. Since any reasonable choice of g makes the cost function achieve optimum on some binary tree, a blind pursuit of minimization of cost functions seems not to be a rational approach. More criteria in this scenario are worth study.

Acknowledgements

The corresponding author is Yicheng Pan. This work is partly supported by National Key R&D Program of China (2021YFB3500700), and partly supported by State Key Laboratory of Complex & Critical Software Environment (CCSE-2024ZX-20).

References

Noga Alon, Yossi Azar, and Danny Vainstein. Hierarchical clustering: A 0.585 revenue approximation. In Jacob D. Abernethy and Shivani Agarwal, editors, *Conference on Learning Theory, COLT 2020, 9-12 July 2020, Virtual Event [Graz, Austria]*, volume 125 of *Proceedings of Machine Learning Research*, pages 153–162. PMLR, 2020. URL http://proceedings.mlr.press/v125/alon20b.html.

L.N.F. Ana and A.K. Jain. Robust data clustering. In *2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2003. Proceedings.*, volume 2, pages II–II, 2003. doi: 10.1109/CVPR.2003.1211462.

Sanjeev Arora, Satish Rao, and Umesh V. Vazirani. Expander flows, geometric embeddings and graph partitioning. *J. ACM*, 56(2):5:1–5:37, 2009. doi: 10.1145/1502793.1502794. URL https://doi.org/10.1145/1502793.1502794.

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

Peter F Brown, Vincent J Della Pietra, Peter V Desouza, Jennifer C Lai, and Robert L Mercer. Class-based n-gram models of natural language. *Computational linguistics*, 18(4):467–480, 1992.

Moses Charikar and Vaggos Chatziafratis. Approximate hierarchical clustering via sparsest cut and spreading metrics. In Philip N. Klein, editor, *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2017, Barcelona, Spain, Hotel Porta Fira, January 16-19*, pages 841–854. SIAM, 2017. doi: 10.1137/1.9781611974782.53. URL https://doi.org/10.1137/1.9781611974782.53.

Moses Charikar, Vaggos Chatziafratis, and Rad Niazadeh. Hierarchical clustering better than average-linkage. In Timothy M. Chan, editor, *Proceedings of the Thirtieth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2019, San Diego, California, USA, January 6-9, 2019*, pages 2291–2304. SIAM, 2019. doi: 10.1137/1.9781611975482.139. URL https://doi.org/10.1137/1.9781611975482.139.

Bertrand Charpentier and Thomas Bonald. Tree sampling divergence: An information-theoretic metric for hierarchical graph clustering. In *IJCAI*, pages 2067–2073. ijcai.org, 2019.

Vaggos Chatziafratis, Grigory Yaroslavtsev, Euiwoong Lee, Konstantin Makarychev, Sara Ahmadian, Alessandro Epasto, and Mohammad Mahdian. Bisect and conquer: Hierarchical clustering via max-uncut bisection. In Silvia Chiappa and Roberto Calandra, editors, *The 23rd International Conference on Artificial Intelligence and Statistics, AISTATS 2020, 26-28 August 2020, Online [Palermo, Sicily, Italy]*, volume 108 of *Proceedings of Machine Learning Research*, pages 3121–3132. PMLR, 2020. URL http://proceedings.mlr.press/v108/chatziafratis20a.html.

Vincent Cohen-Addad, Varun Kanade, Frederik Mallmann-Trenn, and Claire Mathieu. Hierarchical clustering: Objective functions and algorithms. *Journal of the ACM (JACM)*, 66(4):1–42, 2019.

Aron Culotta, Pallika Kanani, Robert Hall, Michael Wick, and Andrew McCallum. Author disambiguation using error-driven machine learning with a ranking loss function. In *Sixth International Workshop on Information Integration on the Web (IIWeb-07), Vancouver, Canada*, 2007.

Sanjoy Dasgupta. A cost function for similarity-based hierarchical clustering. In Daniel Wichs and Yishay Mansour, editors, *Proceedings of the 48th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2016, Cambridge, MA, USA, June 18-21, 2016*, pages 118–127. ACM, 2016. doi: 10.1145/2897518.2897527. URL https://doi.org/10.1145/2897518.2897527.

Michael B Eisen, Paul T Spellman, Patrick O Brown, and David Botstein. Cluster analysis and display of genome-wide expression patterns. *Proceedings of the National Academy of Sciences*, 95(25):14863–14868, 1998.

Shayan Oveis Gharan and Luca Trevisan. Partitioning into expanders. In Chandra Chekuri, editor, *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2014, Portland, Oregon, USA, January 5-7, 2014*, pages 1256–1266. SIAM, 2014. doi: 10.1137/1.9781611973402.93. URL https://doi.org/10.1137/1.9781611973402.93.

Alexander N Gorban, Balázs Kégl, Donald C Wunsch, Andrei Y Zinovyev, et al. *Principal manifolds for data visualization and dimension reduction*, volume 58. Springer, 2008.

Angsheng Li and Yicheng Pan. Structural information and dynamical complexity of networks. *IEEE Trans. Inf. Theory*, 62(6):3290–3339, 2016. doi: 10.1109/TIT.2016.2555904. URL https://doi.org/10.1109/TIT.2016.2555904.

Vince Lyzinski, Minh Tang, Avanti Athreya, Youngser Park, and Carey E. Priebe. Community detection and classification in hierarchical stochastic blockmodels. *IEEE Transactions on Network Science and Engineering*, 4(1):13–26, 2017. doi: 10.1109/TNSE.2016.2634322.

Bogdan-Adrian Manghiuc and He Sun. Hierarchical clustering: O(1)-approximation for well-clustered graphs. In Marc'Aurelio Ranzato, Alina Beygelzimer, Yann N. Dauphin, Percy Liang, and Jennifer Wortman Vaughan, editors, *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pages 9278–9289, 2021. URL https://proceedings.neurips.cc/paper/2021/hash/4d68e143defa221fead61c84de7527a3-Abstract.html.

Benjamin Moseley and Joshua R. Wang. Approximation bounds for hierarchical clustering: Average linkage, bisecting k-means, and local search. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 3094–3103, 2017. URL https://proceedings.neurips.cc/paper/2017/hash/d8d31bd778da8bdd536187c36e48892b-Abstract.html.

Stanislav Naumov, Grigory Yaroslavtsev, and Dmitrii Avdiukhin. Objective-based hierarchical clustering of deep embedding vectors. In *Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Thirty-Third Conference on Innovative Applications of Artificial Intelligence, IAAI 2021, The Eleventh Symposium on Educational Advances in Artificial Intelligence, EAAI 2021, Virtual Event, February 2-9, 2021*, pages 9055–9063. AAAI Press, 2021. URL https://ojs.aaai.org/index.php/AAAI/article/view/17094.

Mirmahdi Rahgoshay and Mohammad R. Salavatipour. Hierarchical clustering: New bounds and objective. *CoRR*, abs/2111.06863, 2021. URL https://arxiv.org/abs/2111.06863.

Ryan A Rossi, Nesreen K Ahmed, Eunyee Koh, and Sungchul Kim. Fast hierarchical graph clustering in linear-time. In *Companion Proceedings of the Web Conference 2020*, pages 10–12, 2020.

Aurko Roy and Sebastian Pokutta. Hierarchical clustering via spreading metrics. *J. Mach. Learn. Res.*, 18:88:1–88:35, 2017. URL http://jmlr.org/papers/v18/17-081.html.

An Information-theoretic Perspective of Hierarchical Clustering on Graphs (Supplementary Material)

Yicheng Pan[1] **Bingchen Fan**[1] **Pengyu Long**[1] **Feng Zheng**[1]

[1]State Key Laboratory of Complex & Critical Software Environment, Beihang University, Beijing, China

A PROOF OF PROPOSITION 1

Proof. For each internal node α on T, denote by $\partial(\alpha)$ the sets of edges in G with exactly one end-point in the set of vertices corresponding to α. So $g_\alpha = \sum_{e\in\partial(\alpha)} w(e)$. Note that

$$\begin{aligned}\mathcal{H}^T(G) &= -\sum_{\alpha\in T}\frac{g_\alpha}{\mathrm{vol}(V)}\log\frac{\mathrm{vol}(\alpha)}{\mathrm{vol}(\alpha^-)}\\ &= -\sum_{\alpha\in T}\sum_{(u,v)\in\partial(\alpha)}\frac{w(u,v)}{\mathrm{vol}(V)}\log\frac{\mathrm{vol}(\alpha)}{\mathrm{vol}(\alpha^-)}\\ &= -\sum_{(u,v)\in E}\left(\frac{w(u,v)}{\mathrm{vol}(V)}\sum_{\alpha:(u,v)\in g_\alpha}\log\frac{\mathrm{vol}(\alpha)}{\mathrm{vol}(\alpha^-)}\right).\end{aligned}$$

For a single edge $(u,v)\in E$, all the terms $\log(\mathrm{vol}(\alpha)/\mathrm{vol}(\alpha^-))$ for leaf u satisfying $(u,v)\in g_\alpha$ sum (over α) up to $\log(d_u/\mathrm{vol}(u\vee v))$ along the unique path from u to $u\vee v$. It is symmetric for v. Therefore, considering ordered pair $(u,v)\in E$,

$$\begin{aligned}\mathcal{H}^T(G) &= -\sum_{\text{ordered }(u,v)\in E}\frac{w(u,v)}{\mathrm{vol}(V)}\log\frac{d_u}{\mathrm{vol}(u\vee v)}\\ &= \frac{1}{\mathrm{vol}(V)}\left(-\sum_{u\in V}d_u\log d_u+\sum_{\text{ordered }(u,v)\in E}w(u,v)\log\mathrm{vol}(u\vee v)\right)\\ &= \frac{1}{\mathrm{vol}(V)}\left(-\sum_{u\in V}d_u\log d_u+2\cdot\sum_{(u,v)\in E}w(u,v)\log\mathrm{vol}(u\vee v)\right).\end{aligned}$$

The second equality follows from the fact that for a fixed u, $d_u=\sum_{v:(u,v)\in E}w(u,v)$, and the last equality from the symmetry of (u,v). Since the first summation is independent of T, Proposition 1 follows. □

B PROOF OF PROPOSITION 2

Proof. Note that a balanced binary tree (BBT for abbreviation) means the tree is balanced on every internal node. Formally, for an internal node of cluster size k, its two sub-trees are of cluster sizes $\lfloor k/2\rfloor$ and $\lceil k/2\rceil$, respectively.

For cliques, since the weights of each edge are identical, we assume it safely to be 1. By Proposition 2.1, minimizing the structural entropy is equivalent to minimizing the cost function (over T)

$$\begin{aligned}\text{cost}^T(G) &= \sum_{(u,v)\in E} \log \text{vol}(u \vee v) \\ &= \sum_{(u,v)\in E} \log\left((n-1)|u \vee v|\right) \\ &= \sum_{(u,v)\in E} \log(n-1) + \sum_{(u,v)\in E} \log|u \vee v|\end{aligned}$$

Since the first term in the last equation is independent of T, the optimization turns to minimizing the last term, which we denote by $\Gamma(T)$. Grouping all edges in E by LCA of two end-points, the cost $\Gamma(T)$ can be written as the sum of the cost γ at every internal node N of T. Formally, for every internal node N, let $A, B \subseteq V$ be the leaves of the sub-trees rooted at the left and right child of N, respectively. We have

$$\begin{aligned}\Gamma(T) &= \sum_N \gamma(N) \\ \gamma(N) &= \left(\sum_{x\in A, y\in B} 1\right) \cdot \log\left(|A| + |B|\right) \\ &= |A| \cdot |B| \cdot \log(|A| + |B|)\end{aligned}$$

Now we only have to show the following lemma.

Lemma 2. *For any positive integer n, a cluster tree T of K_n achieves minimum cost $\Gamma(T)$ if and only if T is a BBT.*

Proof. Lemma 2 is proved by induction on $|V|$. The key technique of tree swapping we use here is inspired by Cohen-Addad et al [4]. The basis step holds since for $|V| = 2$ or 3, the cluster tree is balanced and unique. It certainly achieves the minimum cost exclusively.

Now, consider a clique $G = (V, E)$ with $n = |V| \geq 4$. Let T_1 be an arbitrary unbalanced cluster tree and λ be its root. We need to prove that the cost $\Gamma(T_1)$ does not achieve the minimum. Without loss of generality, we can safely assume the root node is unbalanced, since otherwise, we set T_1 to be the sub-tree that is rooted at an unbalanced node. Let T_2 be a tree with root λ whose left and right sub-trees are BBTs such that they have the same sizes with the left and right sub-trees of T_1, respectively. Let V_{ll}, V_{lr}, V_{rl} and V_{rr} be the sets of nodes on the four sub-trees at the second level of T_2 (V_{ll} means the left child of left child, V_{lr} the right child of left child, and so on), and n_{ll}, n_{lr}, n_{rl} and n_{rr} denote their sizes, respectively. Our proof is also available when some of them are empty. We always assume $n_{ll} \leq n_{lr}$ and $n_{rl} \geq n_{rr}$. Next, we construct T_3 by swapping (transplanting) V_{lr} and V_{rl} with each other. Finally, let T_4 be a tree with root λ whose left and right sub-trees are BBTs after balancing the left and right sub-trees of T_3. So T_4 is a BBT. Then we only have to prove that $\Gamma(T_1) > \Gamma(T_4)$. Note that the strict "$>$" is necessary since we need to negate all unbalanced cluster trees.

Then we show that the transformation process that consists of the above three steps makes the cost decrease step by step. Formally,

(a) T_1 to T_2. The sub-trees of T_1 become BBTs in T_2. Since the number of edges whose end-points treat the root as LCA is the same, by induction we have $\Gamma(T_1) \geq \Gamma(T_2)$.

(b) T_2 to T_3. We will show that $\Gamma(T_2) > \Gamma(T_3)$ in Lemma 3.

(c) T_3 to T_4. The sub-trees of T_3 become BBTs in T_4. For the same reason as (a), we have $\Gamma(T_3) \geq \Gamma(T_4)$.

Putting them together, we get $\Gamma(T_1) > \Gamma(T_4)$ and Lemma 2 follows.

□

Lemma 3. *After swapping V_{lr} and V_{rl}, we obtain T_3 from T_2, for which $\Gamma(T_2) > \Gamma(T_3)$.*

Proof. We only need to consider the changes in cost of three nodes: root and its left and right children, since the cost contributed by each of the remaining nodes does not change after swapping. Ignoring the unchanged costs, define

$$\begin{aligned}\text{cost}(T_2) &= n_l n_r \log n + n_{ll} n_{lr} \log n_l + n_{rl} n_{rr} \log n_r \\ &= n_l n_r \log n + \left\lfloor \frac{n_l}{2} \right\rfloor \left\lceil \frac{n_l}{2} \right\rceil \log n_l + \left\lceil \frac{n_r}{2} \right\rceil \left\lfloor \frac{n_r}{2} \right\rfloor \log n_r,\end{aligned}$$

where $n_l = n_{ll} + n_{lr}$, $n_r = n_{rl} + n_{rr}$. Both of them are at least 1. Similarly, define

$$\begin{aligned}\text{cost}(T_3) &= (n_{ll} + n_{rl})(n_{lr} + n_{rr}) \log n + n_{ll} n_{rl} \log (n_{ll} + n_{rl}) + n_{lr} n_{rr} \log (n_{lr} + n_{rr}) \\ &= \left\lfloor \frac{n}{2} \right\rfloor \left\lceil \frac{n}{2} \right\rceil \log n + \left\lfloor \frac{n_l}{2} \right\rfloor \left\lceil \frac{n_r}{2} \right\rceil \log \left(\left\lfloor \frac{n_l}{2} \right\rfloor + \left\lceil \frac{n_r}{2} \right\rceil \right) + \left\lceil \frac{n_l}{2} \right\rceil \left\lfloor \frac{n_r}{2} \right\rfloor \log \left(\left\lceil \frac{n_l}{2} \right\rceil + \left\lfloor \frac{n_r}{2} \right\rfloor \right)\end{aligned}$$

Denote

$$\begin{aligned}\Delta &= \Gamma(T_2) - \Gamma(T_3) \\ &= \text{cost}(T_2) - \text{cost}(T_3) \\ &= \left\lfloor \frac{n_l}{2} \right\rfloor \left\lceil \frac{n_l}{2} \right\rceil \log \left(\frac{n_l}{n} \right) + \left\lceil \frac{n_r}{2} \right\rceil \left\lfloor \frac{n_r}{2} \right\rfloor \log \left(\frac{n_r}{n} \right) \\ &\quad - \left\lfloor \frac{n_l}{2} \right\rfloor \left\lceil \frac{n_r}{2} \right\rceil \log \left(\frac{\left\lfloor \frac{n_l}{2} \right\rfloor + \left\lceil \frac{n_r}{2} \right\rceil}{n} \right) - \left\lceil \frac{n_l}{2} \right\rceil \left\lfloor \frac{n_r}{2} \right\rfloor \log \left(\frac{\left\lceil \frac{n_l}{2} \right\rceil + \left\lfloor \frac{n_r}{2} \right\rfloor}{n} \right) \qquad (3)\end{aligned}$$

So we only have to show that $\Delta > 0$. We consider the following three cases according to the odevity of n_l and n_r.

Case 1: n_l and n_r are even.

Case 2: n_l and n_r are odd.

Case 3: n_l is odd while n_r is even.

The case that n_l is even while n_r is odd is symmetric to **Case** 3.

For **Case** 1, if both n_l and n_r are even, then notations of rounding in Eq. (3) can be removed and Δ can be simplified as

$$\Delta = \frac{n_l^2}{4} \log \left(\frac{n_l}{n} \right) + \frac{n_r^2}{4} \log \left(\frac{n_r}{n} \right) + \frac{n_l n_r}{2}.$$

Let $p = n_l/n, q = n_r/n$, and so $p + q = 1$. Recall that T_1 is unbalanced on the root λ, so is T_2. Thus $p \neq q$. Multiplying by $\frac{4}{n^2}$ on both sides, we only have to prove that

$$p^2 \log p + q^2 \log q + 2pq > 0.$$

That is,

$$\frac{p}{q} \log p + \frac{q}{p} \log q + 2 > 0.$$

Let $g(x) = \frac{x}{1-x} \log x$. Then we only need to show that $g(p) + g(q) + 2 > 0$ when $p \neq q$. Since

$$\begin{aligned}g'(x) &= \frac{(1-x) + \ln x}{\ln 2 \cdot (1-x)^2}, \\ g''(x) &= -\frac{x^2 - 2x \ln x - 1}{\ln 2 \cdot x(1-x)^3}.\end{aligned}$$

It is easy to check that $g''(x) > 0$ when $0 < x < 1$. So $g(x)$ is strictly convex in the interval $(0, 1)$. Since $p \neq q$,

$$g(p) + g(q) > 2g \left(\frac{p+q}{2} \right) = -2.$$

Thus $\Delta > 0$ holds.

For **Case** 2, if both n_l and n_r are odd, then Δ can be split into two parts $\Delta = \Delta_1 + \Delta_2$, in which

$$\begin{aligned}\Delta_1 &= \frac{n_l^2}{4}\log\left(\frac{n_l}{n}\right) + \frac{n_r^2}{4}\log\left(\frac{n_r}{n}\right) + \frac{n_l n_r}{2}\\ \Delta_2 &= -\frac{1}{4}\log\left(\frac{n_l}{n}\right) - \frac{1}{4}\log\left(\frac{n_r}{n}\right) - \frac{1}{2}\end{aligned}$$

Since we have shown that $\Delta_1 > 0$, if we can prove $\Delta_2 \geq 0$, then the lemma will hold for **Case** 2. Due to the convexity of logarithmic function, this holds clearly since

$$2\log\left(\frac{n}{2}\right) \geq \log n_l + \log n_r.$$

For **Case** 3, if n_l is odd while n_r is even,

$$\Delta = \frac{n_l^2 - 1}{4}\log\left(\frac{n_l}{n}\right) + \frac{n_r^2}{4}\log\left(\frac{n_r}{n}\right) - \left[\frac{(n_l-1)n_r}{4}\log\left(\frac{n-1}{2n}\right) + \frac{(n_l+1)n_r}{4}\log\left(\frac{n+1}{2n}\right)\right].$$

Multiplying the above equation by $4\ln 2$, without changing its sign, yields

$$(4\ln 2)\Delta = (n_l^2-1)\ln\left(\frac{n_l}{n}\right) + n_r^2\ln\left(\frac{n_r}{n}\right) - \left[(n_l-1)n_r\ln\left(\frac{n-1}{2n}\right) + (n_l+1)n_r\ln\left(\frac{n+1}{2n}\right)\right]$$

Splitting the right hand side into two parts,

$$\begin{aligned}A &= n_l^2\ln\left(\frac{n_l}{n}\right) + n_r^2\ln\left(\frac{n_r}{n}\right) + 2n_l n_r\ln 2\\ B &= -\ln\left(\frac{n_l}{n}\right) - (n_l+1)n_r\ln\left(1+\frac{1}{n}\right) - (n_l-1)n_r\ln\left(1-\frac{1}{n}\right)\end{aligned}$$

Since n is odd and the root λ of T_2 is unbalanced, we only need to consider the case that $n_l = (n-i)/2$, $n_r = (n+i)/2$ (Note that n_l and n_r are symmetric. So if $(n-i)/2$ is even, exchange n_l and n_r), where both n and i are odd satisfying $n > i \geq 3$. Next we show that in this case, $A \geq \ln(1/5) + 4^2\ln(4/5) + 2\cdot 4\ln 2$ and $B > \ln 2 - 3/4 - (2/3)\cdot(1/5^2)$. By calculation, $\Delta = A + B > 0$ for **Case** 3.

Claim 1. $A \geq \ln(1/5) + 4^2\ln(4/5) + 2\cdot 4\ln 2$ *for odd integers* $n > i \geq 3$.

Proof. Substituting $n_l = (n-i)/2$, $n_r = (n+i)/2$ into the A yields

$$A = C(n,i) \triangleq \left(\frac{n-i}{2}\right)^2\ln\left(\frac{n-i}{2n}\right) + \left(\frac{n+i}{2}\right)^2\ln\left(\frac{n+i}{2n}\right) + 2\cdot\frac{n-i}{2}\cdot\frac{n+i}{2}\ln 2.$$

Treat n as a continuous variable, we have

$$\frac{\partial C(n,i)}{\partial n} = \frac{1}{2}\left[(n+i)\ln\left(1+\frac{i}{n}\right) + (n-i)\ln\left(1-\frac{i}{n}\right) - \frac{i^2}{n}\right]$$

Multiplying the above equation by $2/n$ and setting $x = i/n$ yields

$$\begin{aligned}f(x) &\triangleq (1+x)\ln(1+x) + (1-x)\ln(1-x) - x^2,\\ f'(x) &= \ln(1+x) - \ln(1-x) - 2x,\\ f''(x) &= \frac{2x^2}{1-x^2}.\end{aligned}$$

It is easy to check that $f(0) = 0$ and $f'(0) = 0$. When $0 < x < 1$, $f''(x) > 0$. Thus $f'(x) > 0$ and $f(x) > 0$. This means that $\partial C(n,i)/\partial n > 0$ for all $n > 0$. So $C(n,i) \geq C(i+2,i)$ for $n \geq i+2$ (When i is fixed, the minimum value of n can be taken to $i+2$, which makes $n_l = (n-i)/2$ and $n_r = (n+i)/2$ integral). The curves of $C(n,i)$ for varying i are plotted in Figure 1.

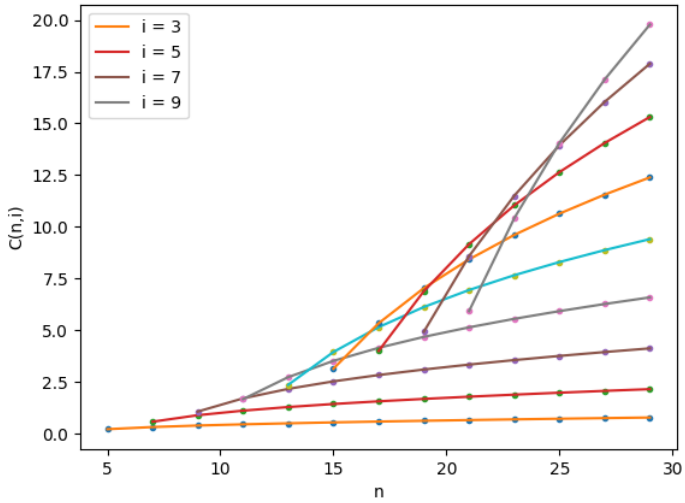


Figure 1: Functions $C(n,i)$

When $n = i+2$, we get $n_l = (n-i)/2 = 1$ and $n_r = (n+i)/2 = n-1$. Substituting them into A yields

$$\begin{aligned} D(n) &\triangleq \ln\left(\frac{1}{n}\right) + (n-1)^2 \ln\left(1-\frac{1}{n}\right) + 2(n-1)\ln 2, \\ \frac{dD}{dn} &= 1 - \frac{2}{n} + 2\ln 2 + 2(n-1)\ln\left(1-\frac{1}{n}\right). \end{aligned}$$

When $n > 2$, it is easy to check that $dD/dn > 0$. So the minimum value of $d(n)$, which is also the minimum value of $C(i+2,i)$, is achieved at $n = i+2 = 5$. So $A = C(n,i) \geq C(i+2,i) \geq C(5,3) = \ln(1/5) + 4^2\ln(4/5) + 2\cdot 4\ln 2$. □

Claim 2. $B > \ln 2 - 3/4 - (2/3)\cdot(1/5^2)$.

Proof. Due to the facts that

$$\begin{aligned} \ln\left(1+\frac{1}{n}\right) &< \frac{1}{n} - \frac{1}{2n^2} + \frac{1}{3n^3}, \\ \ln\left(1-\frac{1}{n}\right) &< -\frac{1}{n} - \frac{1}{2n^2} - \frac{1}{3n^3}, \end{aligned}$$

we have

$$\begin{aligned} B &= -\ln\left(\frac{n_l}{n}\right) - (n_l+1)n_r \ln\left(1+\frac{1}{n}\right) - (n_l-1)n_r\ln\left(1-\frac{1}{n}\right) \\ &> -\ln\left(\frac{n_l}{n}\right) + \frac{n_l n_r}{n^2} - \frac{2n_r}{n} - \frac{2n_r}{3n^3} \\ &> -\ln\left(\frac{n_l}{n}\right) + \frac{n_l n_r}{n^2} - \frac{2n_r}{n} - \frac{2}{3n^2}. \end{aligned}$$

Let $\alpha = n_l/n$, then

$$\begin{aligned} B &> -\ln\alpha + \alpha(1-\alpha) - 2(1-\alpha) - \frac{2}{3n^2} \\ &\geq \ln 2 - \frac{3}{4} - \frac{2}{3n^2}. \end{aligned}$$

When $n \geq 5$, $B > \ln 2 - 3/4 - (2/3)\cdot(1/5^2)$. □

Combining Claims 1 and 2, Lemma 3 follows. □

This completes the proof of Theorem 1.

□

C PROOF OF THEOREM 1

Proof. Note that $\text{cost}^T(G)$ for any cluster tree T has a trivial upper bound. That is,

$$\text{cost}^T(G) = \sum_{(u,v)\in E} w(u,v)\cdot\log(\text{vol}(u\vee v)) \leq \sum_{(u,v)\in E} w(u,v)\cdot\log(\text{vol}(V)) \leq \frac{\text{vol}(V)\cdot\log(\text{vol}(V))}{2}.$$

Let T^* be the optimal cluster tree that achieves the minimum cost, we present here a lower bound for $\text{cost}^{T^*}(G)$. Referring to the dense branch technique [Dasgupta, 2016, Manghiuc and Sun, 2021], we start with the root node A_0 and walk along T^* recursively as follows: at every internal node A_i, walk down to the node A_{i+1} of higher volume between its two children. This process stops when we reach node A_k such that $\text{vol}_G(A_k) \leq \frac{2\text{vol}(V)}{3}$. Denote $A = A_k$ as well as $B = V \setminus A_k$. By construction, it holds that $\text{vol}_G(A) > \frac{\text{vol}(V)}{3}$ and $\text{vol}_G(B) \geq \frac{\text{vol}(V)}{3}$. Moreover, $\text{vol}(A_i) > \frac{2\text{vol}(V)}{3}$ for every $0 \leq i < k$. The basic idea behind the dense branch is that cut (A, B) has a significant contribution to $cost^{T^*}(G)$. Denote by $E(A, B)$ the set of edges between cut (A, B).

$$\begin{aligned}
cost^{T^*}(G) &= \sum_{(u,v)\in E} w(u,v)\cdot\log(\text{vol}(u\vee v)) \\
&\geq \sum_{(u,v)\in E(A,B)} w(u,v)\cdot\log(\text{vol}(u\vee v)) \\
&\geq w(A,B)\cdot\log\left(\frac{2\text{vol}(V)}{3}\right). \\
&\geq \Phi(G)\cdot\frac{\text{vol}(V)}{3}\cdot\log\left(\frac{2\text{vol}(V)}{3}\right).
\end{aligned}$$

Let T be an arbitrary cluster tree, and T^* be an optimal tree. We have

$$\frac{cost^T(G)}{cost^{T^*}(G)} \leq \frac{3}{2\Phi(G)}\cdot\frac{\log(\text{vol}(V))}{\log\left(\frac{2\text{vol}(V)}{3}\right)} = O(\Phi(G)^{-1}).$$

□

D PROOF OF THEOREM 2

Proof. To prove Theorem 2, we only have to prove the following lemma. Then the theorem follows from a simplification of the approximation factor.

Lemma 4. *Let* $\alpha = \max_i\{\Phi_G(P_i)\}$ *and* $\beta = \min_i\{\Phi(G[P_i])\}$. *Algorithm 2 achieves*

$$\left(\left(\left(\log\left(\frac{1}{1-\alpha}\right)+1\right)+\frac{2\alpha}{1-\alpha}\left(1+\log\frac{k}{1-\alpha}\right)\right)\cdot\frac{3}{2\beta\log\left(\frac{4}{3}\right)}\right)$$

approximation.

Proof. We group the edges of G into two categories: let E_1 be the set of edges in the induced subgraphs $G[P_i]$ for all $1 \leq i \leq l$, i.e.,

$$E_1 \triangleq \cup_{i=1}^{l} E[G[P_i]],$$

and E_2 be the remaining crossing edges. Then we have

$$\text{cost}^T(G) = \sum_{e \in E_1} \text{cost}^T(e) + \sum_{e \in E_2} \text{cost}^T(e).$$

We denote by $\text{vol}(G[P_i])$ the volume of the induced graph $G[P_i]$, by $\text{vol}_G(P_i)$ the volume of P_i in G, and by $\text{parent}^T(P_i)$ the parent of P_i on T. By the construction of T, for every P_i,

$$\text{vol}_G(\text{parent}^T(P_i)) = \sum_{j=1}^{i} \text{vol}_G(P_j) \leq i \cdot \text{vol}_G(P_i) \leq k \cdot \text{vol}_G(P_i).$$

Note that

$$w(P_i, V \backslash P_i) = \text{vol}_G(P_i) - \text{vol}(G[P_i]) \leq \alpha \cdot \text{vol}_G(P_i),$$

$$(1-\alpha) \cdot \text{vol}_G(P_i) \leq \text{vol}(G[P_i]),$$

and thus

$$\text{vol}_G(\text{parent}^T(P_i)) \leq k \cdot \text{vol}_G(P_i) \leq \frac{k}{1-\alpha} \text{vol}(G[P_i]).$$

Combining the above, we have that

$$\begin{aligned}
\sum_{e \in E_1} \text{cost}^T(e) &\leq \sum_{e \in E_1} w_e \cdot \log(\text{vol}_G(P_i)) \\
&\leq \sum_{e \in E_1} w_e \cdot \log\left(\frac{1}{1-\alpha} \text{vol}(G[P_i])\right) \\
&= \sum_{e \in E_1} \left(w_e \cdot \log \frac{1}{1-\alpha} + w_e \cdot \log(\text{vol}(G[P_i]))\right) \\
&\leq \left(\log \frac{1}{1-\alpha} + 1\right) \cdot \sum_{i=1}^{k} \frac{\text{vol}(G[P_i]) \cdot \log(\text{vol}(G[P_i]))}{2},
\end{aligned}$$

and

$$\begin{aligned}
\sum_{e \in E_2} \text{cost}^T(e) &\leq \sum_{i=1}^{k} w(P_i, V \backslash P_i) \cdot \log(\text{vol}_G(\text{parent}^T(P_i))) \\
&\leq \sum_{i=1}^{k} \frac{\alpha}{1-\alpha} \text{vol}(G[P_i]) \log\left(\frac{k}{1-\alpha} \text{vol}(G[P_i])\right) \\
&\leq \sum_{i=1}^{k} \frac{\alpha}{1-\alpha} \left(1 + \log \frac{k}{1-\alpha}\right) \text{vol}(G[P_i]) \log(\text{vol}(G[P_i])) \\
&= \frac{2\alpha}{1-\alpha} \left(1 + \log \frac{k}{1-\alpha}\right) \cdot \sum_{i=1}^{k} \frac{\text{vol}(G[P_i]) \cdot \log(\text{vol}(G[P_i]))}{2}.
\end{aligned}$$

Let T^* be the optimal cluster tree of G, and OPT_G be the optimal value. We have

$$OPT_G = \text{cost}_G(T^*) \geq \sum_{i=1}^{l} \sum_{e \in E(G[P_i])} \text{cost}_{T^*}(e) \geq \sum_{i=1}^{l} OPT_{G[P_i]}.$$

Denote by

$$h(\alpha, k) = \left(\left(\log\left(\frac{1}{1-\alpha}\right) + 1\right) + \frac{2\alpha}{1-\alpha}\left(1 + \log \frac{k}{1-\alpha}\right)\right).$$

We have

$$\begin{aligned}
\text{cost}^T(G) &= \sum_{e\in E_1}\text{cost}^T(e)+\sum_{e\in E_2}\text{cost}^T(e)\\
&\le h(\alpha,k)\cdot\sum_{j=1}^{k}\frac{\text{vol}(G[P_i])\cdot\log(\text{vol}(G[P_i]))}{2}\\
&\le h(\alpha,k)\cdot\sum_{j=1}^{k}\frac{\text{vol}(G[P_i])\cdot\log(\text{vol}(G[P_i]))}{2\Phi_{G[P_i]}\cdot\frac{1}{3}\text{vol}(G[P_i])\cdot\log(\frac{2}{3}\text{vol}(G[P_i]))}OPT_{G[P_i]}\\
&\le h(\alpha,k)\cdot\max_i\frac{3\log(\text{vol}(G[P_i]))}{2\Phi_{G[P_i]}\cdot\log(\frac{2}{3}\text{vol}(G[P_i]))}\sum_{j=1}^{k}OPT_{G[P_i]}\\
&\le h(\alpha,k)\cdot\max_i\frac{3\log(\text{vol}(G[P_i]))}{2\Phi_{G[P_i]}\cdot\log(\frac{2}{3}\text{vol}(G[P_i]))}OPT_G\\
&\le h(\alpha,k)\cdot\frac{3}{2\beta\log(\frac{4}{3})}OPT_G
\end{aligned}$$

Lemma 4 follows. □

Note that $h(\alpha,k)=O\left(\frac{1}{(1-\alpha)}\log\frac{k}{1-\alpha}\right)$, Theorem 2 follows. □

E SYNTHETIC DATASETS AND LINKAGE-BASED METHODS

In this section, we introduce the datasets generated from SBM [Ana and Jain, 2003] and HSBM [Lyzinski et al., 2017], and the settings of linkage-based baseline algorithms.

E.1 SBM DATASETS AND LINKAGE-BASED METHODS

SBM is a classic random graph model for two-level clustering structures. In this model, clusters are preset and two probabilities p_0 and p_1 for intra-cluster and inter-cluster connections of vertices are required. Usually, $p_0 < p_1$. For example, in Table 1 of the full paper, we preset five clusters with $p_0 = 0.1$ and $p_1 = 0.9$.

We compare our algorithm BBM on SBM graphs against four linkage-based methods [Cohen-Addad et al., 2019]: average-linkage (AL), single-linkage (SL), complete-linkage (CL) and Linkage++ (L++). AL, SL and CL are agglomerative methods based on different greedy strategy. For two clusters C_1 and C_2 in a weighted graph (denote by w the weight function on edges), their similarity $\text{sim}(C_1, C_2)$ are defined as:

(1) $\frac{1}{|C_1||C_2|}\sum_{u\in C_1,v\in C_2}w(u,v)$ for AL;

(2) $\max_{u\in C_1,v\in C_2}\{w(u,v)\}$ for SL;

(3) $\min_{u\in C_1,v\in C_2}\{w(u,v)\}$ for CL.

These three algorithms have the same agglomerative framework. For a weighted graph of n nodes, starting from n singleton trees with labels $\{v_i\}$, $i = 1, 2, \ldots, n$, in each iteration, two trees T_1, T_2 with root labels C_1, C_2 that have maximum similarity are merged into a new tree, whose root label is $C_1 \cup C_2$, until a single tree is left.

Note that in unweighted (or uniformly weighted) graphs used in our experiments, SL and CL probably have a large number of ties on similarity to break in each iteration. An arbitrary breaking strategy makes the resulting tree varies wildly. To break these ties in a mild fashion, we introduce a random and slight perturbation on edge weights, that is, each edge weight is set to be $1 + \varepsilon$ where ε is uniformly and randomly chosen from interval $[0, 10^{-6}]$. Intuitively, a slight perturbation should not affect the construction of a cluster tree and the output trees of SL and CL will not varies a lot. The balance indices and costs of resulting trees in our experiments are calculated on the original unweighted graphs.

Linkage++ [Cohen-Addad et al., 2019] is proposed for unweighted graphs. It takes a positive integer d as a parameter and projects vertices into a d-dimensional subspace of $\mathbb{R}^n$. The Euclidean distance based SL algorithm are invoked to construct

d clusters, and then based on a newly defined similarity (the same as that for AL) for these d clusters, SL is invoked again to merge them into a single tree. In our experiments, the ground truth d of each SBM graph is given as input for Linkage++.

E.2 HSBM DATASETS

HSBM is a generalization of SBM to the hierarchical case. In this model, k is set to be an intrinsic number of hierarchies of a graph. Let $\vec{p} = (p_0, p_1, \ldots, p_{k-1})$ be a probability vector, for which p_i is the probability of generating edges for vertex pairs whose LCA on the ground-truth cluster tree has depth i. The 0-depth node is the root, and typically, $p_{i-1} \leq p_i$ for each i.

Our experiments for HCSE utilize k-level HSBM with $k = 4$ and 5. we set the graph sizes for $k = 4, 5$ to be 2500 and 6000 respectively. For simplicity, in each ground-truth cluster tree, the number of internal nodes on each level is fixed. On each level, we group all the tree nodes uniformly and randomly into a fixed number of groups, and then allocate each group accordingly to the tree nodes on the next upper level. For $k = 4$, we set the tree node numbers from leaves to the root to be $[2500, 250, 25, 5, 1]$, and for $k = 5$, to be $[6000, 600, 125, 25, 5, 1]$.

F ADDITIONAL EXPERIMENTAL RESULTS

In this section, we introduce the balance indices and additional results on binary and non-binary clustering algorithms.

F.1 MORE COST AND BALANCEDNESS RESULTS FOR BINARY CLUSTERING ON SBM GRAPHS

We conduct more experiments on cost and balancedness results on SBM graphs. As similar settings in Table 1 of the full paper, we choose the two strategies "Biased" and "Random" for generating cluster sizes for SBM. For the Biased strategy, five clusters are generated, and their sizes are set to be $[32, 32, 64, 128, 256]$ with ± 5 random perturbation on each figure. The probability of presence for intra-cluster edges is also $p_1 = 0.9$, but compared to $p_0 = 0.1$ for inter-cluster edges in the full paper, we choose $p_0 = 0.2$ and 0.01 respectively. The results are shown in Table 2.

Table 2: Cost and balancedess results on SBM graphs for the Biased strategy. All values are averaged over 100 rounds of trials.

	$(p_1, p_0) = (0.9, 0.2)$					$(p_1, p_0) = (0.9, 0.01)$				
Method	B_{size}	B_{vol}	B_{dep}	cost(Das)	cost(SE)	B_{size}	B_{vol}	B_{dep}	cost(Das)	cost(SE)
AL	0.663	0.702	8.70	1.51E7	8.75E5	0.351	0.407	2.33	6.21E6	5.73E5
SL	0.217	0.225	1.29	1.92E7	8.92E5	0.244	0.251	1.55	1.37E7	6.15E5
CL	0.975	0.976	60.2	1.81E7	9.02E5	0.972	0.976	60.1	9.59E6	6.13E5
L++	0.894	0.913	29.7	**1.32E7**	8.65E5	0.603	0.655	5.05	**6.12E6**	5.74E5
BBM	**0.0146**	**0.0556**	**0.178**	1.33E7	**8.55E5**	**0.0148**	**0.0148**	**0.181**	6.13E6	**5.72E5**

Our algorithm BBM always achieves the best cost(SE) and balance indices compared with the baselines, while keeps competitive (quite near) to the cost(Das) of Linkage++. Combined with the results in Table 1 of the full paper, it can be observed that the costs and balance indices are better for SBM graphs with more obvious clustering structure.

For the Random strategy, we choose different cluster numbers 10 and 15 while adjusting the inter-link probabilities accordingly to $p_0 = 0.01$ and 0.005, respectively. The sizes of each clusters are generated uniformly and randomly from integers in the interval $[2^4, 2^6)$ as we do in the full paper. The probability of presence for intra-cluster edges is $p_1 = 0.9$. The results are shown in Table 3.

It can be observed that our algorithm BBM always achieves the best cost(SE) and balance indices compared with the baselines, while keeps competitive to cost(Das) of Linkage++.

F.2 NMI RESULTS FOR NON-BINARY CLUSTERING ON HSBM GRAPHS

The NMI results for our non-binary clustering algorithm HCSE on HSBM graphs are summarized in Table 4.

Table 3: Cost and balancedness results on SBM graphs for the Random strategy. All values are averaged over 100 rounds of trials.

	block num = 10, $(p_1, p_0) = (0.9, 0.01)$					block num = 15, $(p_1, p_0) = (0.9, 0.005)$				
Method	B_{size}	B_{vol}	B_{dep}	cost(Das)	cost(SE)	B_{size}	B_{vol}	B_{dep}	cost(Das)	cost(SE)
AL	0.321	0.351	2.09	4.71E5	8.27E4	0.333	0.362	2.28	8.87E5	1.37E5
SL	0.337	0.346	2.36	2.04E6	1.02E5	0.382	0.390	2.76	5.28E6	1.76E5
CL	0.957	0.959	37.6	1.63E6	9.90E4	0.969	0.970	50.6	3.83E6	1.67E5
L++	0.725	0.735	7.10	**4.16E5**	8.26E4	0.769	0.779	9.12	**7.74E5**	1.37E5
BBM	**0.108**	**0.0866**	**0.723**	4.24E5	**8.19E4**	**0.102**	**0.0749**	**0.724**	7.99E5	**1.36E5**

Table 4: The NMI results for $k = 4, 5$ respectively. Each group picks three probability vectors. The probability $p_{k-1} = 0.9$ for the innermost hierarchy is default and omitted from the table. $p_{k-2}, \ldots, p_0$ of each vector are listed sequentially. Those choices of $\vec{p}$ guarantee that the clusters on each level are clear and the generated graphs are connected with high probability. For each k, $\vec{p}$ makes the clustering structure become increasingly clear one by one. "$-$" means that the algorithm does not find the corresponding level. HCSE and all baselines have no hyper-parameters.

			$(k = 4)$							$(k = 5)$				
$\vec{p}$	HC_{ave}	HC_{sin}	HC_{com}	HLP	LOU	HCSE	$\vec{p}$	HC_{ave}	HC_{sin}	HC_{com}	HLP	LOU	HCSE
4.5E(-2)	0.904	0.821	0.815	0.899	**0.997**	0.902	1.5E(-2)	0.927	0.857	0.863	0.993	**1.00**	0.957
1.5E(-3)	0.958	0.611	0.642	0.792	**1.00**	0.825	1.5E(-3)	0.843	0.731	0.743	0.703	**0.961**	0.857
6.0E(-6)	–	0.557	0.894	0.598	–	**1.00**	7.5E(-5)	0.875	0.593	0.601	–	0.877	**0.885**
							1.0E(-6)	–	0.438	0.523	–	–	**0.984**
5.5E(-2)	0.876	0.819	0.816	0.864	**0.994**	0.886	4.5E(-2)	0.940	0.846	0.849	0.962	**0.999**	0.946
1.5E(-3)	0.914	0.595	0.606	0.799	**1.00**	0.860	4.5E(-3)	0.915	0.723	0.748	0.822	**0.951**	0.892
4.0E(-6)	–	0.657	0.840	0.653	–	**1.00**	7.5E(-5)	0.925	0.639	0.648	0.771	0.903	**0.948**
							6.0E(-7)	–	0.508	0.542	–	–	**0.959**
4.5E(-2)	0.908	0.823	0.824	0.879	**0.978**	0.901	1.5E(-2)	0.917	0.857	0.866	0.994	**0.999**	0.935
7.5E(-4)	0.947	0.630	0.652	0.824	**1.00**	0.977	1.5E(-3)	0.813	0.735	0.757	0.879	**0.958**	0.860
2.5E(-6)	0.656	0.574	0.718	0.677	–	**1.00**	5.0E(-5)	0.872	0.631	0.671	0.841	0.877	**0.915**
							3.0E(-7)	0.762	0.686	0.761	0.786	–	**0.953**

Our experiments demonstrate that HCSE outperforms those algorithms that incorporate other efficient and non-parametric linkage-based binary clustering methods into our framework. This demonstrates the effectiveness of our new cost function. HCSE also outperforms the representative heuristic non-parametric algorithms LOUVAIN [Blondel et al., 2008] and HLP [Rossi et al., 2020]. These two algorithms proceed simply by recursively invoking flat clustering algorithms based on modularity and label propagation, respectively, and the hierarchy number is solely determined by the number of rounds when the algorithm terminates. They follow the main framework of present non-binary clustering algorithms whose interpretability is poor. Our experimental results show that HCSE has a great advantage in both finding the intrinsic number of hierarchies and reconstructions.

F.3 ABLATION FOR COST(SE)

For ablation, we replace cost(SE) with cost(Das) in the stretch and compress processes, respectively. We define Das-Das to be the non-binary clustering algorithm that replaces the structural entropy cost used in both stretch and compress processes with cost(Das), Das-SE the one that replaces with cost(Das) in stretch only, and SE-Das the one that replaces with cost(Das) in compress only. In these settings, HCSE is in fact SE-SE. The NMI results for $k = 4, 5$ with the same settings of probability vectors with Table 4 are demonstrated in Table 5.

Table 5: The ablation NMI results for $k = 4, 5$ respectively.

($k = 4$)					($k = 5$)				
$\vec{p}$	Das-Das	Das-SE	SE-Das	HCSE	$\vec{p}$	Das-Das	Das-SE	SE-Das	HCSE
4.5E(-2)	0.795	0.813	0.825	**0.902**	1.5E(-2)	0.843	0.845	0.928	**0.957**
1.5E(-3)	0.579	0.593	0.718	**0.825**	1.5E(-3)	0.710	0.713	0.845	**0.857**
6.0E(-6)	0.287	0.382	**1.00**	**1.00**	7.5E(-5)	0.544	0.551	0.876	**0.885**
					1.0E(-6)	0.317	0.353	0.965	**0.984**
5.5E(-2)	0.814	0.823	0.808	**0.886**	4.5E(-2)	0.832	0.847	0.885	**0.946**
1.5E(-3)	0.587	0.584	0.758	**0.860**	4.5E(-3)	0.706	0.714	0.833	**0.892**
4.0E(-6)	0.395	0.369	**1.00**	**1.00**	7.5E(-5)	0.563	0.553	0.932	**0.948**
					6.0E(-7)	0.443	0.368	0.943	**0.959**
4.5E(-2)	0.814	0.821	0.830	**0.901**	1.5E(-2)	0.839	0.846	0.934	**0.935**
7.5E(-4)	0.594	0.591	0.938	**0.977**	1.5E(-3)	0.710	0.713	**0.872**	0.860
2.5E(-6)	0.403	0.396	**1.00**	**1.00**	5.0E(-5)	0.555	0.557	**0.925**	0.915
					3.0E(-7)	0.447	0.442	0.949	**0.953**

It can be observed that, for almost all settings of probability vectors, HCSE outperforms all other strategies with stretch or compress process that uses cost(Das). When we keep to use structural entropy in stretch, the NMIs of HCSE and SE-Das are much better than Das-Das and Das-SE that use cost(Das) in stretch. This implies that structural entropy always has more advantages in binary hierarchical clustering than cost(Das), because structural entropy based stretch process reconstructs on the binary cluster tree more precise intermediate truth clusters which can be preserved after compress.

F.4 INFLECTION POINTS

We show in Figure 2 the inflection points of HCSE for the three probability vectors with $k = 4$ in Table 4 of the full paper. $\delta_t(\mathcal{H})$ has the sharpest drop for each graph after the ground-truth sparsest levels have been stratified at $t = 4$, which terminates the stratification of HCSE. This verifies the rationality of our strategy of stratification in finding the intrinsic hierarchy numbers.

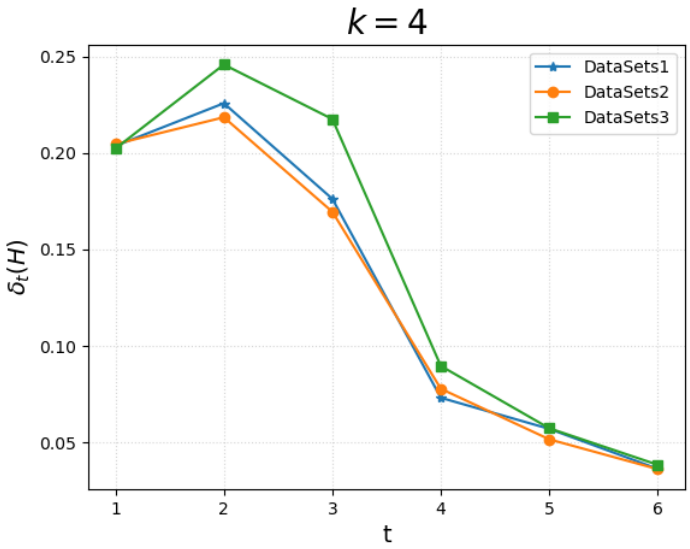


Figure 2: $\delta_t(\mathcal{H})$ variations for HCSE and $k = 4$. The inflection points for all probability vectors appear correctly on $t = 4$.

F.5 JACCARD INDEX RESULTS ON AMAZON NETWORK

We do our non-binary clustering experiments on the Amazon network [1] for which the set of ground-truth clusters has been given. There are a huge number of overlaps and variations on the sizes of these clusters, and we suppose that they are some parts of a hierarchical cluster tree. We evaluate the resulting cluster trees by Jaccard index. For two sets A, B, the *Jaccard Index* of them is defined as $J(A, B) = |A \cap B|/|A \cup B|$. We pick the largest cluster which is a subgraph with 58283 vertices and 133178 edges. For each ground-truth cluster c that appears in this subgraph, we find from the resulting cluster tree an internal node that has maximum Jaccard index with c. Then we calculate the average Jaccard index $\overline{J}$ over all such c. The scores are summarized in Table 6.

Table 6: Comparisons of the average Jaccard index ($\overline{J}$). We choose the best results in five runs for the two baselines HLP and LOUVAIN, respectively.

index	HCSE	HLP	LOUVAIN
$\overline{J}$	**0.20**	0.16	0.17

G SUPPLEMENTARY FIGURES AND PSEUDOCODES

We list all the supplementary figures and pseudocodes in this section.

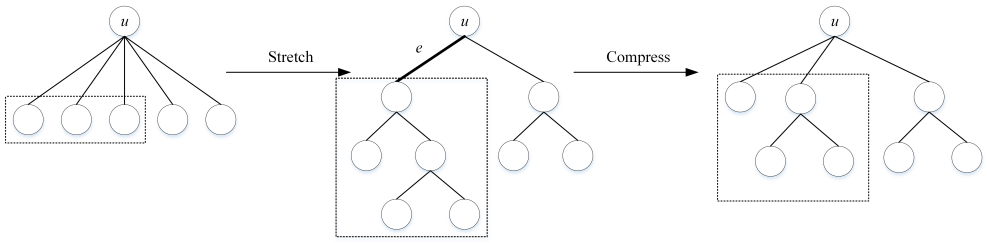
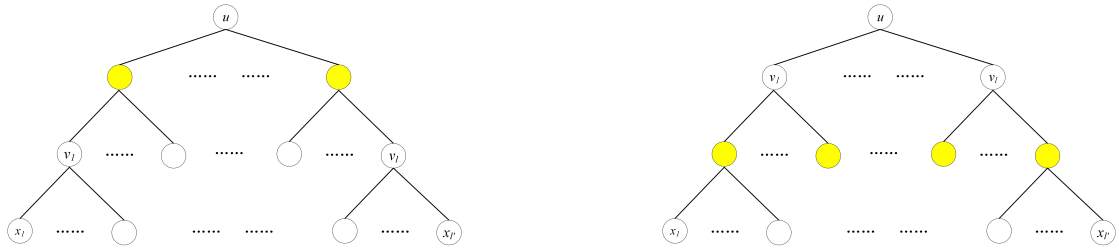


Figure 3: Illustrations of stretch and compress for a u-triangle. A binary cluster tree is constructed first by stretch, and then edge e is compressed, which yields a non-binary tree.

[1] http://snap.stanford.edu/data/



(a) (b)

Figure 4: Illustrations of stratification for a 2-level cluster tree. The preference of (a) and (b) depends on the average sparsity of triangles at each level.

Algorithm 5 k-Hierarchical clustering based on structural entropy (k-HCSE)

Input: a graph $G = (V, E)$, $k \in \mathbb{Z}^+$.
Output: a k-level cluster tree T.
Initialize T to be the 1-level cluster tree.
$h =$ height(T).
while $h < k$ **do**
  // Find the sparsest level of T (breaking ties arbitrarily).
  $j' \leftarrow \arg\max_j\{\overline{\text{Spar}}_j(T)\}$.
  if $\overline{\text{Spar}}_{j'}(T) = 0$ **then**
    // No cost will be saved by any further clustering.
    break.
  for $u \in U_{j'}$ **do**
    $T_u \leftarrow$ Stretch(u-triangle T_u).
    Compress(T_u).
  $h \leftarrow h + 1$
  for $j \in [j' + 1, h]$ **do**
    Update U_j.
return T.

Algorithm 6 Hierarchical clustering based on structural entropy (HCSE)

Input: a graph $G = (V, E)$.
Output: a cluster tree T.
$t \leftarrow 2$.
while $\Delta_t\delta < \Delta_{t-1}\delta$ or $\Delta_t\delta < \Delta_{t+1}\delta$ **do**
  if $\max_j\{\overline{\text{Spar}}_j(T)\}$=0 **then**
    break.
  $t \leftarrow t + 1$.
return t-HCSE(T).