Expected Probabilistic Hierarchies

Marcel Kollovieh[1,2,3] **Bertrand Charpentier**[4] **Daniel Zügner**[5] **Stephan Günnemann**[1,2,3,4]

[1] School of Computation, Information and Technology, Technical University of Munich
[2] Munich Data Science Institute [3] Munich Center for Machine Learning
[4] Pruna AI [5] Microsoft Research AI for Science

Correspondence to: m.kollovieh@tum.de

Abstract

Hierarchical clustering has usually been addressed by discrete optimization using heuristics or continuous optimization of relaxed scores for hierarchies. In this work, we propose to optimize *expected* scores under a probabilistic model over hierarchies. **(1)** We show *theoretically* that the global optimal values of the expected Dasgupta cost and Tree-Sampling divergence (TSD), two unsupervised metrics for hierarchical clustering, are equal to the optimal values of their discrete counterparts contrary to some relaxed scores. **(2)** We propose Expected Probabilistic Hierarchies (EPH), a probabilistic model to learn hierarchies in data by optimizing expected scores. EPH uses *differentiable hierarchy sampling* enabling end-to-end gradient descent based optimization, and an *unbiased subgraph sampling* approach to scale to large datasets. **(3)** We evaluate EPH on synthetic and real-world datasets including *vector* and *graph* datasets. EPH outperforms all other approaches quantitatively and provides meaningful hierarchies in qualitative evaluations.

1 Introduction

A fundamental problem in unsupervised learning is clustering. Given a dataset, the task is to partition the instances into similar groups. While *flat* clustering algorithms such as k-means group data points into disjoint groups, a *hierarchical* clustering divides the data recursively into smaller clusters, which yields several advantages over a flat one. Instead of only providing cluster assignments of the data points, it captures the clustering at multiple granularities, allowing the user to choose the desired level of fine and coarseness depending on the task. The hierarchical structure can be easily visualized in a *dendrogram* (e.g., see Fig. 4), making it easy to interpret and analyze. Hierarchical clustering finds applications in many areas, from personalized recommendation [39] and document clustering [35] to gene-expression [16] and phylogenetics [17]. Furthermore, the presence of hierarchical structures can be observed in various real-world graphs in nature and society [33].

A first family of methods for hierarchical clustering are discrete approaches. They aim at optimizing some hierarchical clustering quality scores on a discrete search space, i.e.:

$$\max_{\hat{\mathcal{T}}} \text{score}(\boldsymbol{X}, \hat{\mathcal{T}}) \text{ s.t.} \hat{\mathcal{T}} \in \text{discrete hierarchies}, \tag{1}$$

where $\boldsymbol{X}$ denotes a given (vector or graph) dataset. Examples of score optimization could be the minimization of the discrete Dasgupta score [12], the minimization of the error sum of squares [37], the maximization of the discrete TSD [8], or the maximization of the modularity score [4]. Discrete approaches have two main limitations: the optimization search space of discrete hierarchies is large and constrained, which often makes the problem intractable without using heuristics, and the learning procedure is not differentiable and thus not amenable to gradient-based optimization, as done by

38th Conference on Neural Information Processing Systems (NeurIPS 2024).

 https://doi.org/10.52202/079017-0443

most deep learning approaches. To mitigate these issues, a second, more recent family of continuous methods proposes to optimize some (soft-)scores on a continuous search space of relaxed hierarchies:

$$\max_{\mathcal{T}} \text{soft-score}(\boldsymbol{X}, \mathcal{T}) \text{ s.t.} \mathcal{T} \in \text{relaxed hierarchies.} \tag{2}$$

Examples are the relaxation of Dasgupta [7, 10, 41] or TSD scores [41]. A major drawback of continuous methods is that the optimal hierarchy of soft scores might not align with their discrete counterparts.

Contributions. In this work, we propose to optimize *expected* discrete scores, called Exp-Das and Exp-TSD, moving away from prior work on *relaxed* soft scores. In particular, our contributions can be summarized as follows:

- **Theoretical contribution**: We analyze the *theoretical* properties of both soft and expected scores. We show that the minimal value of a continuous relaxation can be different from that of the discrete Dasgupta cost while the optimal values of the expected scores are equal to their optimal discrete counterparts.
- **Model contribution**: We propose a new method called Expected Probabilistic Hierarchies (EPH) to optimize Exp-Das and Exp-TSD. EPH provides an *unbiased* estimate of Exp-Das and Exp-TSD with biased gradients based on differentiable hierarchy sampling. By utilizing an *unbiased* subgraph sampling procedure, EPH *scales* to large (vector) datasets.
- **Experimental contribution**: In quantitative experiments, we show that EPH outperforms other baselines on 20/24 cases on 16 datasets, including both *graph* and *vector* datasets. In qualitative experiments, we show that EPH provides *meaningful* hierarchies.

2 Related Work

Discrete Methods. We further differentiate between agglomerative (bottom-up) and divisive (top-down) discrete algorithms. Well-established agglomerative methods are the linkage algorithms that subsequently merge the two clusters with the lowest distance into a new cluster. There are several ways to define the similarity of two clusters. The average linkage (AL) method uses the average similarity, while single linkage (SL) and complete linkage (CL) use the minimum and maximum similarity between the groups, respectively [18]. Finally, the ward linkage (WL) algorithm [37] operates on Euclidean distances and merges the two clusters with the lowest increase in the sum of squares. In contrast, Bayesian Hierarchical Clustering [19] uses statistical hypothesis tests to decide which clusters to merge. Another agglomerative approach is the Louvain algorithm [4], which iteratively maximizes the modularity score. A more recent parallelized agglomerative graph-based approach, ParHAC, enables scaling to massive datasets. In terms of quality, however, ParHAC performs inferior to AL w.r.t. the Dasgupta cost [14]. Unlike agglomerative methods, divisive algorithms work in a top-down fashion. Initially, all leaves share the same cluster and are recursively divided into smaller ones using flat clustering algorithms. Famous examples are based on the k-means algorithm [35] or use approximations of the sparsest cut [12].

Continuous Methods. In recent years, many continuous algorithms have emerged to solve hierarchical clustering. These methods minimize continuous relaxations of the Dasgupta cost using gradient descent based optimizers. Monath et al. [27] optimized a probabilistic cost version. To parametrize the probabilities, they performed a softmax operation on learnable routing functions from each node on a fixed binary hierarchy. Chierchia and Perret [10] proposed UFit, a model operating in the ultrametric space. Furthermore, to optimize their model, they presented a soft-cardinal measure to compute a differentiable relaxed version of the Dasgupta cost. Other approaches operate on continuous representations in hyperbolic space, such as gHHC [28] and HypHC [7].

Zügner et al. [41] recently presented a flexible probabilistic hierarchy model (FPH). FPH directly parametrizes a probabilistic hierarchy and substitutes the discrete terms in the Dasgupta cost and Tree-Sampling Divergence with their probabilistic counterparts. This results in a differentiable objective function, which they optimize using projected gradient descent.

Differentiable Sampling Methods. Stochastic models with discrete random variables are difficult to train as the backpropagation algorithm requires all operations to be differentiable. To address this problem, estimators such as the Gumbel-Softmax [20] or Gumbel-Sinkhorn [26] are used to retain gradients when sampling discrete variables. These differentiable sampling methods have been

used for several tasks, including DAG predictions [9], spanning trees or subset selection [30], and generating graphs [6]. In contrast to prior work, EPH enables optimizing the expected clustering scores under a probabilistic hierarchy by utilizing differentiable sampling. Note that sampling spanning trees is not applicable in our case since we have a restricted structure where the nodes of the graph correspond to the leaves of the tree.

3 Probabilistic Hierarchical Clustering

We consider a graph dataset. Let $\mathcal{G} = (V, E)$ be a graph with n vertices $V = \{\mathrm{v}_1, \ldots, \mathrm{v}_n\}$ and m edges $E = \{\mathrm{e}_1, \ldots, \mathrm{e}_m\}$. Let $w_{i,j}$ denote the weight of the edge connecting the nodes v_i and v_j if $(i, j) \in E$, 0 otherwise, and $w_i = \sum_j w_{i,j}$ the weight of the node v_i. In general, we assume a directed graph, i.e., $w_{i,j} \neq w_{j,i}$. We define the edge distribution $P(\mathrm{v}_i, \mathrm{v}_j)$ for pairs of nodes, $P(\mathrm{v}_i, \mathrm{v}_j) \propto w_{i,j}$, s.t. $\sum_{\mathrm{v}_i, \mathrm{v}_j \in V} P(\mathrm{v}_i, \mathrm{v}_j) = 1$ and equivalently the node distribution $P(\mathrm{v}_i) \propto w_i$, s.t. $\sum_{\mathrm{v}_i \in V} P(\mathrm{v}_i) = 1$. We can extend this representation to any vector dataset $\mathcal{D} = \{\mathrm{x}_1, \ldots, \mathrm{x}_n\}$ and interpret the dataset as a graph by using the data points x_i as nodes and pairwise similarities (e.g., cosine similarities) as edge weights.

Discrete Hierarchical Clustering. We define a discrete hierarchical clustering $\hat{\mathcal{T}}$ of a graph $\mathcal{G}$ as a rooted tree with n leaves and n' internal nodes. The leaves $V = \{\mathrm{v}_1, \mathrm{v}_2, \ldots, \mathrm{v}_n\}$ represent the nodes of $\mathcal{G}$, while the internal nodes $Z = \{\mathrm{z}_1, \mathrm{z}_2, \ldots, \mathrm{z}_{n'}\}$ represent clusters, with $\mathrm{z}_{n'}$ being the root node. Each internal node groups the data into disjoint sub-clusters, where edges reflect memberships of clusters. We can represent the hierarchy using two binary adjacency matrices $\hat{\boldsymbol{A}} \in \{0, 1\}^{n \times n'}$ and $\hat{\boldsymbol{B}} \in \{0, 1\}^{n' \times n'}$, i.e., $\hat{\mathcal{T}} = (\hat{\boldsymbol{A}}, \hat{\boldsymbol{B}})$. While $\hat{\boldsymbol{A}}$ describes the edges from the leaves to the internal nodes, $\hat{\boldsymbol{B}}$ specifies the edges between the internal nodes. Since every node in the hierarchy except the root has exactly one outgoing edge, we have the following constraints: $\sum_j^{n'} \hat{A}_{i,j} = 1$ for $1 \leq i \leq n$, $\sum_j^{n'} \hat{B}_{i,j} = 1$ for $1 \leq i < n'$, and $\sum_j^{n'} \hat{B}_{n',j} = 0$ for the last row. Thus, except for the last row of $\hat{\boldsymbol{B}}$, both matrices are row-stochastic. We denote the ancestors of v as $\mathrm{anc}(\mathrm{v})$, and the lowest common ancestor (LCA) of the two leaves v_i and v_j in $\hat{\mathcal{T}}$ as $\mathrm{v}_i \wedge \mathrm{v}_j$.

Probabilistic Hierarchical Clustering. Zügner et al. [41] recently proposed probabilistic hierarchies. The idea is to use a continuous relaxation of the binary adjacency matrices while keeping the row-stochasticity constraints. Thus, we end up with two matrices $\boldsymbol{A} \in [0, 1]^{n \times n'}$ and $\boldsymbol{B} \in [0, 1]^{n' \times n'}$. The entries represent parent probabilities, i.e., $A_{i,j} := p(\mathrm{z}_j \mid \mathrm{v}_i)$ describes the probability of the internal node z_j being the parent of v_i and $B_{i,j} := p(\mathrm{z}_j \mid \mathrm{z}_i)$ the probability of the internal node z_j being the parent of z_i. Together, they define a probabilistic hierarchy $\mathcal{T} = (\boldsymbol{A}, \boldsymbol{B})$. Given such a probabilistic hierarchy, one can easily obtain a discrete hierarchy by interpreting the corresponding rows of $\boldsymbol{A}$ and $\boldsymbol{B}$ as categorical distributions. We sample an outgoing edge for each leaf and internal node. Since $\boldsymbol{B}$ is restricted to be an upper triangular matrix, this tree-sampling procedure will result in a valid discrete hierarchy, denoted by $\hat{\mathcal{T}} = (\hat{\boldsymbol{A}}, \hat{\boldsymbol{B}}) \sim P_{\boldsymbol{A},\boldsymbol{B}}(\mathcal{T})$. Alternatively, instead of sampling a discrete hierarchy, one can take the most likely edge for each node. This can be efficiently done by selecting the entry with the highest probability in each row, resulting in the discrete matrices $\hat{\boldsymbol{A}}$ and $\hat{\boldsymbol{B}}$. This approach serves as a greedy approximation to extract the most likely hierarchy from the probabilistic one.

4 Expected Probabilistic Hierarchical Clustering

4.1 Expected Metrics

Unlike flat clusterings, there has been a shortage of objective functions for hierarchical clusterings. Thus, many algorithms to derive hierarchies were developed without a precise objective. An objective function not only allows us to evaluate the quality of a hierarchy but also yields possibilities for optimization techniques. Recently, the two unsupervised functions Dasgupta cost **(Das)** [12] and Tree-Sampling Divergence **(TSD)** [8] were proposed, triggering the development of a new generation of hierarchical clustering algorithms. The Dasgupta cost is a well-established metric for graphs and vector data, while the TSD is a recent metric specifically tailored to graphs. In addition to being unsupervised, i.e., applicable in cases where the data is unlabeled, both metrics have intuitive motivations. The metrics can be written as:

$$\mathrm{Das}(\hat{\mathcal{T}}) = \sum_{\mathrm{v}_i, \mathrm{v}_j \in V} P(\mathrm{v}_i, \mathrm{v}_j) c(\mathrm{v}_i \wedge \mathrm{v}_j) \qquad \text{and} \qquad \mathrm{TSD}(\hat{\mathcal{T}}) = \mathrm{KL}(p(\mathrm{z}) \| q(\mathrm{z})), \tag{3}$$

https://doi.org/10.52202/079017-0443

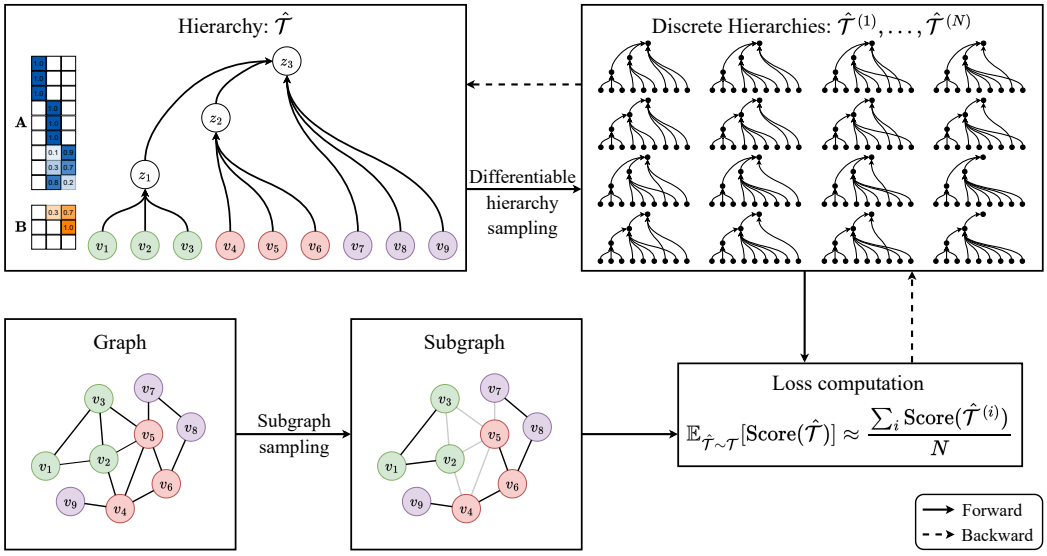


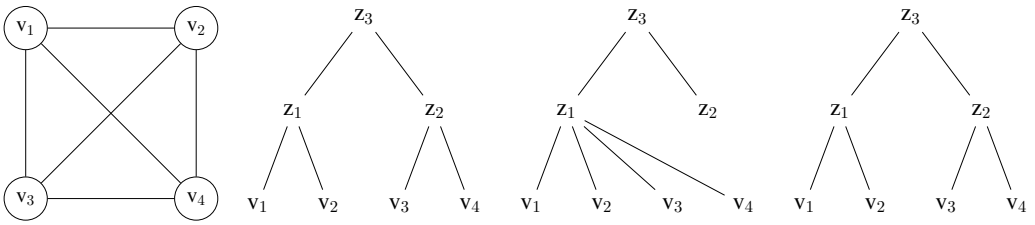
Figure 1: Overview of our proposed EPH model. During training, EPH first samples discrete hierarchies using our differentiable hierarchy sampling (Sec. 4.3) and a subgraph using our subgraph sampling procedure (Sec. 4.4). The expected scores are then computed and averaged. Finally, the probabilistic hierarchy is updated via backpropagation. A formal description is given in App. C.7.

where $c(\mathrm{z})$ is the number of leaves whose ancestor is z, i.e., $c(\mathrm{z}) = \sum_{\mathrm{v}_i \in V} \mathbf{1}_{[\mathrm{z} \in \mathrm{anc}(\mathrm{v}_i)]}$, and $p(\mathrm{z})$ and $q(\mathrm{z})$ are two distributions induced by the edge and node distributions, i.e., $p(\mathrm{z}) = \sum_{\mathrm{v}_i, \mathrm{v}_j} \mathbf{1}_{[\mathrm{z}=\mathrm{v}_i \wedge \mathrm{v}_j]} P(\mathrm{v}_i, \mathrm{v}_j)$ and $q(\mathrm{z}) = \sum_{\mathrm{v}_i, \mathrm{v}_j} \mathbf{1}_{[\mathrm{z}=\mathrm{v}_i \wedge \mathrm{v}_j]} P(\mathrm{v}_i) P(\mathrm{v}_j)$. Dasgupta favors similar leaves to have their lowest common ancestor low in the hierarchy [12]. TSD quantifies the ability to reconstruct the graph from the hierarchy in terms of information loss [8]. Hence, any hierarchy achieving a good score provides a good compression of the original graph. In practice, both metrics are good indicators of meaningful hierarchies [12, 8, 7]. Recently, Zügner et al. [41] proposed the Flexible Probabilistic Hierarchy (FPH) method. FPH substitutes the indicator functions with their corresponding probabilities under the tree-sampling procedure (see App. A.1), obtaining cost functions for probabilistic hierarchies called Soft-Das and Soft-TSD. These two metrics correspond to the scores of the *expected hierarchies* under the tree-sampling procedure. In contrast, we propose to optimize the *expected metrics* in this work. Intuitively, this corresponds to moving the expectation from inside the metric functions to outside, reflecting the natural way of performing Monte-Carlo approximation via (tree-) sampling. More specifically, our objectives are:

$$\min_{\boldsymbol{A},\boldsymbol{B}} \mathbb{E}_{\hat{\mathcal{T}}}\left[\text{Das}(\hat{\mathcal{T}})\right] \text{ s.t. } \hat{\mathcal{T}} \sim P_{\boldsymbol{A},\boldsymbol{B}}(\mathcal{T}) \quad \text{and} \quad \max_{\boldsymbol{A},\boldsymbol{B}} \mathbb{E}_{\hat{\mathcal{T}}}\left[\text{TSD}(\hat{\mathcal{T}})\right] \text{ s.t. } \hat{\mathcal{T}} \sim P_{\boldsymbol{A},\boldsymbol{B}}(\mathcal{T}), \tag{4}$$

which we denote as **Exp-Das** and **Exp-TSD**. Note that we optimize over $\boldsymbol{A}$ and $\boldsymbol{B}$, which parametrize a probabilistic hierarchy, while the edge weights are given by the dataset and used to compute the node and edge distribution. We show in Sec. 4.2 that the optimal values of the expected scores share the same intuitive meaning as their discrete counterparts. While the probabilities used in the FPH computation are consistent, their relaxed scores are not consistent with the expected scores under the tree-sampling procedure. In Fig. 2, we show a simple case where Soft-Das does not align with the global optimal value, whereas Exp-Das does.

Note that while the Dasgupta cost favors binary branches [12], EPH and the probabilistic hierarchies are not limited to these. While binary hierarchies offer fine-grained clustering, non-binary hierarchies provide more flexibility by allowing nodes to have more than two children, which often aligns better with real-world clustering tasks.



(a) K_4 Graph (b) A minimizing hierarchy (c) FPH (d) EPH

Figure 2: An example where FPH fails to infer a minimizing hierarchy. A hierarchy minimizing the Dasgupta cost and the inferred hierarchies by FPH and EPH on the unweighted K_4 graph, i.e., every normalized edge weight is equal to $\frac{1}{6}$. While FPH achieves a Dasgupta cost of 4.0 after discretization, the continuous hierarchy has a Soft-Das score below 3.0. On the other hand, EPH finds a minimizing hierarchy with a cost of $\frac{10}{3}$. A weighted example is shown in Fig. 8.

4.2 Theoretical Analysis of EPH and FPH

The main motivation to use the expected metrics is the property that their global *optimal value*, i.e., the *score* obtained by the globally optimal hierarchy (the *optimizer*), is equal to their discrete counterparts, as we show in Prop. 4.1.

Proposition 4.1. *Let $\boldsymbol{A}$ and $\boldsymbol{B}$ be transition matrices describing a probabilistic hierarchy. Then, the following equalities hold:*

$$\min_{\boldsymbol{A},\boldsymbol{B}} \mathbb{E}_{\hat{\mathcal{T}}\sim P_{\boldsymbol{A},\boldsymbol{B}}(\mathcal{T})}\big[\mathrm{Das}(\hat{\mathcal{T}})\big] = \min_{\hat{\mathcal{T}}} \mathrm{Das}(\hat{\mathcal{T}}) \quad \text{and} \quad \max_{\boldsymbol{A},\boldsymbol{B}} \mathbb{E}_{\hat{\mathcal{T}}\sim P_{\boldsymbol{A},\boldsymbol{B}}(\mathcal{T})}\big[\mathrm{TSD}(\hat{\mathcal{T}})\big] = \max_{\hat{\mathcal{T}}} \mathrm{TSD}(\hat{\mathcal{T}}). \tag{5}$$

(See proof in App. A.5)

Consequently, optimizing our cost function aims to find the optimal discrete hierarchy. Furthermore, we prove in Prop. 4.2 that Soft-Das is a lower bound of Exp-Das, therefore its minimum is a lower bound of the optimal discrete Dasgupta cost.

Proposition 4.2. *Let $\boldsymbol{A}$ and $\boldsymbol{B}$ be transition matrices describing a probabilistic hierarchy. Then, Soft-Das will be lower than or equal to the expected Dasgupta cost under the tree-sampling procedure:*

$$\text{Soft-Das}(\mathcal{T}) \leq \mathbb{E}_{\hat{\mathcal{T}}\sim P_{\boldsymbol{A},\boldsymbol{B}}(\mathcal{T})}\big[\mathrm{Das}(\hat{\mathcal{T}})\big]. \tag{6}$$

(See proof in App. A.4)

In Fig. 2, we provide a specific example illustrating a case where the minimizer of Soft-Das is continuous, and FPH fails to find the optimal hierarchy, i.e., obtaining a minimal Dasgupta cost. We know an integral solution exists for EPH since Exp-Das and Exp-TSD are convex combinations of their discrete counterparts. Furthermore, Exp-Das is neither convex nor concave, as we show in App. A.6. In Tab. 1, we provide an overview of properties of the cost functions of FPH and EPH.

Table 1: Properties of Soft-Das, Exp-Das, Soft-TSD, and Exp-TSD.

Property	Problem Type	Convex/Concave	Integral	Optimal	Consistent
Soft-Das	Min.	Neither w.r.t. $\boldsymbol{A}$ and $\boldsymbol{B}$ (see Fig.14 (left))	✗	✗	✗
Exp-Das	Min.	Neither w.r.t. $\boldsymbol{A}$ and $\boldsymbol{B}$ (see App.A.6)	✓	✓	✓
Soft-TSD	Max.	Convex w.r.t. LCA probabilities [41]	✓	✓	✗
Exp-TSD	Max.	-	✓	✓	✓

4.3 Unbiased Computation of Expected Scores via Differentiable Sampling

In order to compute the expected scores, one could attempt to use a closed-form expression. To derive these for Exp-Das and Exp-TSD, however, we would need to calculate the probability $p\left(\mathrm{z} = \mathrm{v}_i \wedge \mathrm{v}_j, \mathrm{z} \in \mathrm{anc}(\mathrm{v})\right)$ for which no known solution exists, and the expectation of a logarithm, respectively (see Eq. 13 and Eq. 14). An alternative to the closed-form solution is to approximate the

https://doi.org/10.52202/079017-0443

expectations via the Monte Carlo method. We propose to approximate Exp-Das and Exp-TSD with N *differentiably* sampled hierarchies $\{\hat{\mathcal{T}}^{(1)}, \dots, \hat{\mathcal{T}}^{(N)}\}$ (see "Loss computation" in Fig. 1):

$$\text{Exp-Score}(\mathcal{T}) \approx \frac{1}{N} \sum_{i=1}^{N} \text{Score}(\hat{\mathcal{T}}^{(i)}). \tag{7}$$

However, differentiable sampling of discrete structures like hierarchies is often complex. To this end, our differentiable hierarchy sampling algorithm consists of three steps integrating the tree-sampling procedure and the straight-through Gumbel-Softmax estimator [20]: **(1)** We sample the parents of the leaf nodes by interpreting the columns of $\boldsymbol{A}$ as parameters of straight-through Gumbel-Softmax estimators. **(2)** We sample the parents of the internal nodes by interpreting the columns of $\boldsymbol{B}$ as parameters of straight-through Gumbel-Softmax estimators. This procedure is *differentiable* — each step is differentiable — and *expressive* — it can sample any hierarchy with n leaves and n' internal nodes. **(3)** We use the Monte Carlo method to approximate the expectation by computing the arithmetic mean of the scores of the sampled hierarchies. We reuse the *differentiable* computation of Soft-Das and Soft-TSD, which match the discrete scores for discrete hierarchies while providing gradients w.r.t. $\boldsymbol{A}$ and $\boldsymbol{B}$ (see Fig. 1 for an overview).

Complexity. Since we sample N hierarchies from $n' + n - 1$ many categorical distributions with $\mathcal{O}(n')$ classes, the sampling process can be done with a complexity of $\mathcal{O}(N \times n \times n' + N \times n'^2)$. The dominating term is the computation of the Das and TSD scores with a complexity of $\mathcal{O}(N \times m \times n'^2)$ for graph datasets and $\mathcal{O}(N \times n^2 \times n'^2)$ for vector datasets [41]. This is often efficient as we typically have $n' \ll n$ and for graphs $m \ll n^2$. In Sec. 4.4, we propose a subgraph sampling approach to reduce the complexity to $\mathcal{O}(N \times M \times n'^2 + n^2)$ for large vector datasets, where $M < n^2$.

4.4 Scalable Expected Dasgupta Cost Computation via Subgraph Sampling

As discussed in the complexity analysis, the limiting factor is $\mathcal{O}(n^2 \times n'^2)$, corresponding to the evaluation of the Dasgupta cost, which becomes prohibitive for large datasets. To reduce the complexity, we propose an *unbiased* subgraph sampling approach. First, we note that we can interpret the normalized similarities $P(\mathrm{v}_i, \mathrm{v}_j)$ as a probability mass function of a categorical distribution. This interpretation allows us to rewrite the Dasgupta cost as an expectation and approximate it via a sampling procedure. More specifically,

$$\text{Das}(\hat{\mathcal{T}}) = \mathbb{E}_{(\mathrm{v}_i, \mathrm{v}_j) \sim P(\mathrm{v}_i, \mathrm{v}_j)} \left[c(\mathrm{v}_i \wedge \mathrm{v}_j) \right] \approx \frac{1}{M} \sum_{k=1}^{M} c(\mathrm{v}_i^{(k)} \wedge \mathrm{v}_j^{(k)}), \tag{8}$$

where $\{(\mathrm{v}_i^{(1)}, \mathrm{v}_j^{(1)}), \dots, (\mathrm{v}_i^{(M)}, \mathrm{v}_j^{(M)})\}$ are M edges sampled from the edge distribution $P(\mathrm{v}_i, \mathrm{v}_j)$, which can be done in $\mathcal{O}(M + n^2)$ [24]. We refer to this sampling approach as subgraph sampling (see Fig. 1). We can approximate the expected Dasgupta cost using the same procedure. In contrast to Exp-Das, Exp-TSD cannot be easily viewed as an expectation of edges, thus making the approximation via subgraph sampling impractical. However, since TSD is a metric originally designed for graphs, which are generally sparse, it would yield little benefits.

Note that we end up with two different sampling procedures. First, we have the *differentiable hierarchy sampling* (see Eq. 7). This is necessary to approximate the expectations. Since we do not have a closed-form expression of Exp-Das and Exp-TSD, we sample discrete hierarchies from the probabilistic ones and average the scores. Secondly, we have the *subgraph sampling* (see Eq. 8), which interprets the Dasgupta cost as an expectation. This is done to reduce the computational overhead for vector datasets since the number of pairwise similarities grows quadratically in the number of data points. This estimation is unbiased and introduces an additional parameter, i.e., the number of sampled edges, which allows a trade-off between computational cost and quality. By inserting the probabilistic edge sampling approach into the tree sampling, we estimate Exp-Das to scale it to large vector datasets. An overview of our model is shown in Fig. 1, and a formal description is given in App. C.7.

In contrast to other approaches, the loss function of EPH is consistent with the discrete scores, meaning that probabilistic hierarchies with a better training score encode better discrete hierarchies. Following from Prop. 4.1, we know that any discrete hierarchy sampled from the optimal probabilistic hierarchy is an optimal discrete hierarchy. While global optimality is not guaranteed, the training loss provides a reliable indicator of the discrete evaluation performance since the loss functions, Exp-Das and Exp-TSD, are computed on discrete hierarchies.

5 Experiments

5.1 Experimental Setup

Datasets. We evaluate our method on both graph and vector datasets. *Graph datasets:* We use the datasets Polblogs [1], Brain [2], Citeseer [34], Genes [11], Cora-ML [25, 5], OpenFlight [29], WikiPhysics [3], and DBLP [38]. To preprocess the graph, we first collect the largest connected component. Secondly, every edge is made bidirectional and unweighted. An overview of the graphs is shown in Tab. 7 in App. B.1. *Vector datasets:* We test our method on vector data for the Dasgupta cost. Here, we selected the seven datasets Zoo, Iris, Glass, Digits, Segmentation, Spambase, and Letter from the UCI Machine Learning repository [15]. Furthermore, we also use Cifar-100 [23]. Digits and Cifar-100 are image datasets, the remaining are vector data. While we only flatten the images of Digits, we preprocess Cifar-100 using the ResNet-101 BiT-M-R101x1 by Kolesnikov et al. [22], which was pretrained on ImageNet-21k [13]. More specifically, we use the 2048 dimensional activations of the final layer for each image in Cifar-100 as feature vectors. Furthermore, we normalize all features to have a mean of zero and a standard deviation of one. We compute cosine similarities between all pairs of data points using their normalized features. This results in dense similarity matrices. Finally, we remove the self-loops. Note that in contrast to the graph datasets, the vector data similarities are weighted. An overview is shown in Tab. 8 in App. B.1. Since we are in an unsupervised setting, we have no train/test split, i.e., we train and evaluate on the whole graph.

Baselines. We compare our model against both discrete and continuous approaches. For discrete approaches, we use the single, average, complete [18], and ward linkage [37] algorithms, respectively referred to as **SL**, **AL**, **CL**, and **WL**. We do not report the results of SL and CL on the graph datasets that do not have edge weights since these methods are not applicable to unweighted graphs. In addition to the linkage algorithms, we also compare to the recursive sparsest cut (**RSC**) [12] and the Louvain method (**Louv.**) [4]. For continuous approaches, we use the gradient-based optimization approaches Ultrametric Fitting (**UF**) [10], Hyperbolical Hierarchical Clustering (**HypHC**) [7], gradient-based Hyperbolic Hierarchical Clustering (**gHHC**) [28], and Flexible Probabilistic Hierarchy (**FPH**) [41]. While the linkage algorithms derive a hierarchy based on heuristics or local objectives, UF, HypHC, gHHC, and FPH aim to optimize a relaxed Dasgupta cost or TSD score. We set a time limit of 120 hours for all the methods and provided a budget of 512GB of memory for each experiment.

Practical Considerations. We repeat the randomized methods with five random seeds and report the best score of the discrete hierarchies. We use the same experimental setup as Zügner et al. [41], i.e., we use $n' = 512$ internal nodes, compress full hierarchies using the scheme presented by Charpentier and Bonald [8], and use DeepWalk embeddings [32] on the graphs for methods that require features. We train EPH using PAdamax (projected Adamax [21]), reduce the learning rate for $\boldsymbol{B}$ by a factor of 0.1 every 1000 epochs, and reset the probabilistic hierarchy to the so-far best discrete hierarchy. To approximate the expectation of EPH, we use 20 samples, except for Spambase, Letter, and Cifar-100, where we use 10, 1, and 1, respectively, to reduce the runtime. On the datasets Digits, Segmentation, Spambase, Letter, and Cifar-100, we train EPH and FPH by sampling $n\sqrt{n}$ edges. For the remaining datasets and during all validation steps, we use the whole graph. Both EPH and FPH are initialized using the average linkage algorithm. We train FPH with its original setting and our proposed scheduler and report the minimum of both for each dataset. For the remaining methods, we use the recommended hyperparameters. An overview of the hyperparameters is shown in Tab. 10, and an ablation study in App. C.4. Finally, all methods are evaluated on discrete hierarchies. For EPH, we take the most likely edge for each node as explained in Sec. 3.

5.2 Results

Graph Dataset Results. We report the discrete Dasgupta costs and TSD scores for the graph datasets in Tab. 2. EPH achieves best scores across 13/16 settings and second-best otherwise. In particular, EPH, which optimizes Exp-Das, consistently achieves a better Dasgupta cost than FPH, which optimizes Soft-Das. This observation aligns with the theoretical advantages of Exp-Das compared to Soft-Das (see Sec. 4.2). EPH and FPH, which both use the tree-sampling probabilistic framework, consistently achieve the best results. This underscores the effectiveness of tree sampling for hierarchical clustering.

Code available at https://www.cs.cit.tum.de/daml/expected-probabilistic-hierarchies

https://doi.org/10.52202/079017-0443

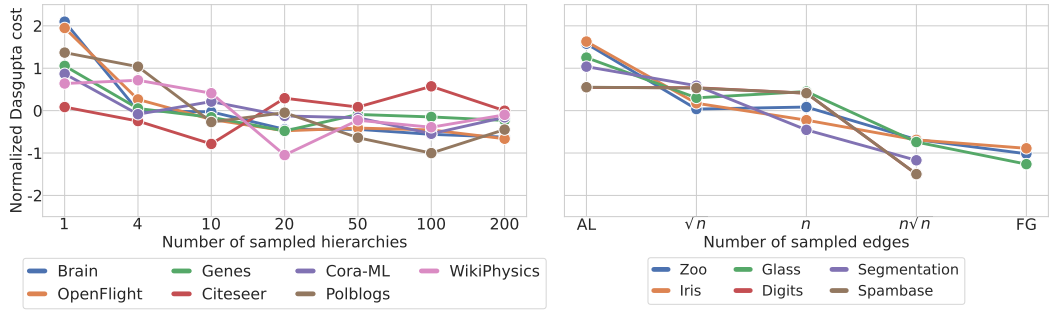


Figure 3: Hyperparameter study. Normalized Dasgupta costs for different numbers of sampled hierarchies (left) and different number of sampled edges (right) after the EPH training, including the average linkage algorithm (AL) and a training on the full graph (FG). The scores are normalized such that each dataset has a mean of zero and a standard deviation of one.

Table 2: Results for the graph datasets. Best scores in **bold**, second best underlined.

	Dasgupta cost (↓)								Tree-sampling divergence (↑)							
Dataset	PolBl.	Brain	Cites.	Genes	Cora-ML	OpenF.	WikiP.	DBLP	PolBl.	Brain	Cites.	Genes	Cora-ML	OpenF.	WikiP.	DBLP
WL	338.52	567.90	137.80	270.18	301.68	379.68	660.12	OOM	26.59	25.13	62.14	60.93	52.76	50.59	42.18	OOM
AL	355.61	556.68	83.69	196.50	292.77	363.40	658.04	36,463	25.25	28.91	67.80	66.72	55.30	52.02	43.15	38.99
Louv.	344.47	582.45	158.79	247.27	335.57	501.29	798.75	40,726	28.86	30.74	68.09	67.51	58.18	52.97	47.01	41.40
RSC	307.70	526.17	85.41	188.82	264.62	367.36	630.53	OOM	28.04	29.19	67.39	66.28	56.14	52.01	44.86	OOM
UF	331.79	508.30	91.86	208.51	305.43	410.17	560.45	OOM	21.77	24.49	60.13	59.45	48.42	47.64	42.37	OOM
gHHC	349.71	595.70	147.17	308.42	313.29	390.21	672.84	87,344	24.70	25.62	59.53	54.20	49.56	51.36	41.08	16.29
HypHC	272.81	519.96	416.38	632.02	594.23	529.04	678.45	OOM	19.65	7.26	18.98	13.00	19.18	26.82	23.92	OOM
FPH	238.65	425.70	76.03	182.91	257.42	355.61	482.40	31,687	31.37	32.75	**69.38**	**67.78**	**59.55**	57.58	49.87	41.62
EPH	**235.50**	**400.20**	**74.01**	**176.57**	**238.28**	**312.31**	**456.26**	**30,600**	**32.05**	**34.24**	69.36	67.75	59.41	**57.83**	**50.23**	**42.74**

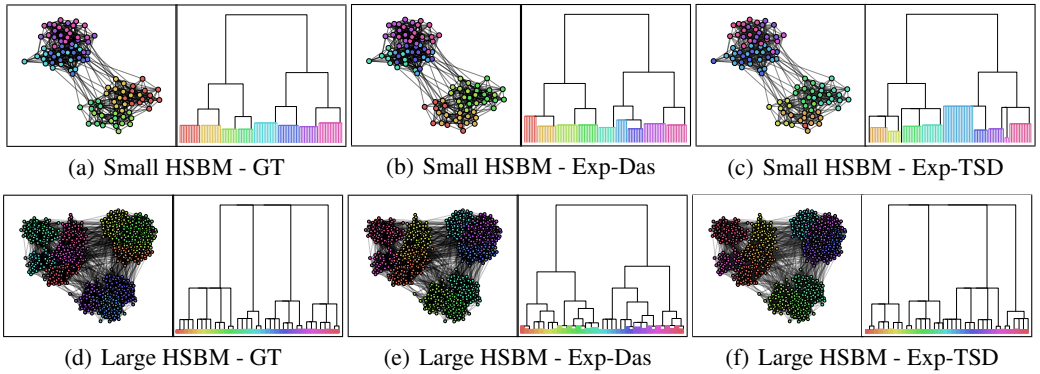
Furthermore, the performance of the discrete approaches, i.e., the linkage algorithms WL and AL, the Louvain method, and RSC, is competitive with the other continuous methods, even though they use heuristics or local objectives to infer the hierarchies. The inferior performance of gHHC and HypHC can be intuitively explained by the fact that these methods were originally designed for vector datasets. WL, UF, and HypHC could not scale to DBLP within the memory budget, as they require the computation of a dense n^2 similarity matrix, leading to out-of-memory (OOM) issues.

Vector Dataset Results. We report the discrete Dasgupta costs of several methods on the vector datasets in Tab. 3. Consistent with the results on the graph datasets, EPH outperforms all baselines and achieves the best scores across 7/8 datasets. These results demonstrate the capacity of EPH to not only cluster graphs but also adapt to vector datasets. Further, EPH consistently outperforms FPH, again emphasizing the benefit of optimizing expected scores over soft scores. In contrast with graph datasets, HypHC performs competitively on vector datasets. This is reasonable since this method was originally designed for vector datasets. FPH performs slightly worse than HypHC on most datasets and is only better on Iris.

Table 3: Dasgupta costs (↓) for the vector datasets. Best scores in **bold**, second best underlined.

Dataset	Zoo	Iris	Glass	Digits	Segm.	Spam.	Letter	Cifar
WL	56.28	69.98	122.16	1126.77	1266.17	2962.62	12241	32979
AL	56.31	69.48	121.64	1121.68	1258.22	2952.21	12181	32972
SL	57.67	70.71	126.33	1166.05	1368.21	3042.28	13166	33314
CL	55.78	70.10	123.02	1140.26	1277.48	2971.59	12396	33131
Louv.	56.26	72.31	125.94	1126.48	1238.35	2916.48	11946	32940
RSC	55.94	**69.10**	121.45	1119.43	1237.20	2917.07	11895	**32907**
UF	56.28	69.40	122.43	1137.53	1322.86	2998.17	13090	OOM
gHHC	60.09	69.63	123.33	1119.74	1269.28	3018.44	12151	33089
HypHC	56.05	69.22	121.52	1118.08	1233.07	2921.38	11930	OOM
FPH	56.13	69.13	122.00	1132.84	1238.45	2933.56	12197	33224
EPH	**55.77**	**69.10**	**120.94**	**1117.58**	**1230.60**	**2916.17**	**11894**	32913

Hyperparameter Study. We show in Fig. 3 (left) the effect of the number of sampled hierarchies on the EPH performances. On the one hand, we observe that a large number of sampled hierarchies (i.e., $N \geq 20$) generally yields better results than a small number of sampled hierarchies (i.e., $N \leq 10$) except for Citeseer. Intuitively, a higher number of sampled hierarchies should lead to a more accurate expected score approximation. On the other hand, a very large number for sampled hierarchies (i.e., $N \geq 100$) might not lead to significant improvements while requiring more computational resources. Intuitively, the randomness induced by a lower number of sampled hierarchies could be beneficial to escape local optima. In general, we found that 20 samples led to satisfactory results for all datasets, thus achieving a good trade-off between approximation accuracy, optimization noise, and computational requirements.



(a) Small HSBM - GT (b) Small HSBM - Exp-Das (c) Small HSBM - Exp-TSD

(d) Large HSBM - GT (e) Large HSBM - Exp-Das (f) Large HSBM - Exp-TSD

Figure 4: Ground truth clusters and dendrograms compared to the inferred ones for the HSBMs.

Table 4: Results of EPH for the HSBMs with $n' = \#\text{Cluster}$.

	Dasgupta cost (↓)		Tree-sampling divergence (↑)	
Method	HSBM Small	HSBM Large	HSBM Small	HSBM Large
GT	26.29	130.16	43.14	51.50
EPH	26.19	121.08	43.56	51.53
Level	Normalized Mutual Information		Normalized Mutual Information	
Level 1	1.0	1.0	1.0	1.0
Level 2	1.0	1.0	1.0	1.0
Level 3	0.77	0.81	0.87	0.99

We show in Fig. 3 (right) the effect of the number of sampled edges on the EPH performances on vector datasets. Using more edges consistently leads to better results. In particular, going from n to $n\sqrt{n}$ shows a significant performance improvement while going from $n\sqrt{n}$ to n^2 yields only minor improvements. Hence, controlling the amount of sampled edges allows us to scale our method to large datasets while maintaining high performance. On the small datasets Zoo, Iris, and Glass, we use the whole graph, while for the other datasets, we sample $n\sqrt{n}$ edges as a trade-off between runtime and quality of the hierarchical clustering. We provide further experiments in Sec. C.4, including a minimized version of EPH outperforming FPH in terms of runtime and results.

External Evaluation. As the Dasgupta cost and Tree-Sampling Divergence are internal metrics and do not necessarily measure how closely a derived hierarchy aligns with an external ground truth, we complement our unsupervised quantitative evaluation with an external evaluation. Given that ground-truth hierarchies are typically unavailable in real-world data, we evaluate EPH on synthetic graph datasets with known ground-truth hierarchies. Additionally, we analyze to what extent the inferred hierarchies preserve the flat class labels for the vector datasets in Sec. C.3 in the appendix.

Table 5: Cophenetic correlations (↑) of EPH for different distances on the HSBMs with $n' = \#\text{Cluster}$.

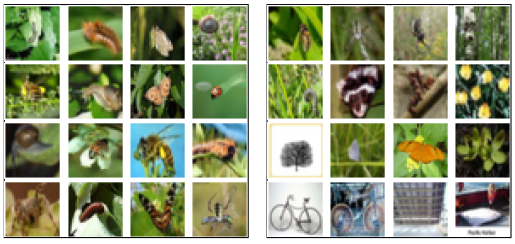
	Cophenetic Correlation (↑)			
	Shortest path distance		DeepWalk Distance	
Method	HSBM Small	HSBM Large	HSBM Small	HSBM Large
GT	**0.77453**	**0.67839**	0.94001	0.90920
Exp-Das	0.77174	0.67692	0.93973	0.89788
Exp-TSD	0.77440	0.67838	**0.94027**	**0.90922**

We augment the graph datasets with two hierarchical stochastic block models (**HSBMs**), which enables us to compare the inferred hierarchy with the ground truth. As the HSBM graphs are generated in a stochastic process, the ground-truth hierarchy is not necessarily optimal in terms of the Dasgupta cost or Tree-Sampling Divergence. Furthermore, we compute the normalized mutual information (**NMI**) across the different levels of the hierarchies, and the cophenetic correlations for the shortest path distance (**SPD**) and the Euclidean distance of DeepWalk embeddings (see Tab. 4 and Tab. 5). We observe that EPH recovers the first three levels of the ground-truth hierarchy almost perfectly. Moreover, the inferred hierarchies by EPH obtain even better scores than the ground-truth hierarchies on the HSBMs, underlining the remarkable capacity of EPH to optimize the Dasgupta and TSD scores. Additionally, the TSD objective proves to be a more suitable metric for recovering the ground-truth levels of the HSBM in terms of NMI. The cophenetic correlations observed between Exp-Das and Exp-TSD are notably high

https://doi.org/10.52202/079017-0443

and closely align with the ground truth, with Exp-TSD even surpassing the ground truth on DeepWalk distances. We attribute this to the fact that a minimal Dasgupta cost favors binary branches, which does not reflect the hierarchies of the HSBMs. Our visual analysis of the graphs and hierarchies (see Fig. 4) confirms this observation. Additional results for FPH are provided in App. C.1.

Qualitative Evaluation. We visualize the largest cluster inferred on Cifar-100 using EPH. More specifically, we select the internal nodes with the most directly connected leaves. Additionally, we sort the images by their probability, i.e., their entry in the matrix $\boldsymbol{A}$. We show the 16 images with the highest probability and the 16 with the lowest probability for the largest cluster in Fig. 5. We observe that the images with high probabilities are all similar and related to insects. This shows that EPH is able to group similar images together. In contrast, the last images with the lowest probability do not fit into the group. This demonstrates the capacity of EPH to measure the uncertainty in the cluster assignments. We provide additional results with the same behavior for other clusters in App. C.2 (see Fig. 9, Fig. 10, and Fig. 11). Furthermore, we visualize the graph and inferred hierarchies of EPH for OpenFlight in Fig. 12 in the appendix. Both minimizing Exp-Das and Exp-TSD generate reasonable clusters and successfully distinguish different world regions. Finally, we visualize the ground truth classes and clusters of selected vector datasets in Fig. 13.



(a) Highest Probability (b) Lowest Probability

Figure 5: Largest derived cluster on Cifar-100.

6 Discussion

Limitations. While the Gumbel-Softmax estimators of the expectations are unbiased in the forward pass, the estimation of the gradients is not [31] and thus impacts the EPH optimization. Furthermore, even though the global optimal values of the expected and discrete scores match, EPH does not guarantee convergence into a global optimum when optimizing using gradient descent methods.

Conclusion. In this work, we propose EPH, a novel end-to-end learnable approach to infer hierarchies in data. EPH operates on probabilistic hierarchies and directly optimizes the expected Dasgupta cost and Tree-Sampling Divergence using *differentiable hierarchy sampling*. We show that the global optima of the expected scores are equal to their discrete counterparts. Furthermore, we present an *unbiased subgraph sampling* approach to scale EPH to large datasets. We demonstrate the capacity of our model by evaluating it on several synthetic and real-world datasets. EPH outperforms traditional and recent state-of-the-art baselines.

References

[1] Lada A Adamic and Natalie Glance. The political blogosphere and the 2004 us election: divided they blog. In *Proceedings of the 3rd international workshop on Link discovery*, pages 36–43, 2005.

[2] Katrin Amunts, Claude Lepage, Louis Borgeat, Hartmut Mohlberg, Timo Dickscheid, Marc-Étienne Rousseau, Sebastian Bludau, Pierre-Louis Bazin, Lindsay B Lewis, Ana-Maria Oros-Peusquens, et al. Bigbrain: an ultrahigh-resolution 3d human brain model. *Science*, 340(6139): 1472–1475, 2013.

[3] Nicolas Aspert, Volodymyr Miz, Benjamin Ricaud, and Pierre Vandergheynst. A graph-structured dataset for wikipedia research. In *Companion Proceedings of The 2019 World Wide Web Conference*, pages 1188–1193, 2019.

[4] 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.

[5] Aleksandar Bojchevski and Stephan Günnemann. Bayesian robust attributed graph clustering: Joint learning of partial anomalies and group structure. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

[6] Aleksandar Bojchevski, Oleksandr Shchur, Daniel Zügner, and Stephan Günnemann. Netgan: Generating graphs via random walks. In *International conference on machine learning*, pages 610–619. PMLR, 2018.

[7] Ines Chami, Albert Gu, Vaggos Chatziafratis, and Christopher Ré. From trees to continuous embeddings and back: Hyperbolic hierarchical clustering. *Advances in Neural Information Processing Systems*, 33:15065–15076, 2020.

[8] Bertrand Charpentier and Thomas Bonald. Tree sampling divergence: an information-theoretic metric for hierarchical graph clustering. In *IJCAI-19*, 2019.

[9] Bertrand Charpentier, Simon Kibler, and Stephan Günnemann. Differentiable dag sampling. *arXiv preprint arXiv:2203.08509*, 2022.

[10] Giovanni Chierchia and Benjamin Perret. Ultrametric fitting by gradient descent. *Advances in neural information processing systems*, 32, 2019.

[11] Ara Cho, Junha Shin, Sohyun Hwang, Chanyoung Kim, Hongseok Shim, Hyojin Kim, Hanhae Kim, and Insuk Lee. Wormnet v3: a network-assisted hypothesis-generating server for caenorhabditis elegans. *Nucleic acids research*, 42(W1):W76–W82, 2014.

[12] Sanjoy Dasgupta. A cost function for similarity-based hierarchical clustering. In *Proceedings of the forty-eighth annual ACM symposium on Theory of Computing*, pages 118–127, 2016.

[13] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[14] Laxman Dhulipala, David Eisenstat, Jakub Lacki, Vahab Mirrokni, and Jessica Shi. Hierarchical agglomerative graph clustering in poly-logarithmic depth. *Advances in Neural Information Processing Systems*, 35:22925–22940, 2022.

[15] Dheeru Dua and Casey Graff. UCI machine learning repository, 2017. URL http://archive.ics.uci.edu/ml.

[16] 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.

[17] Joseph Felsenstein. *Inferring phylogenies*, volume 2. Sinauer associates Sunderland, MA, 2004.

[18] Trevor Hastie, Robert Tibshirani, Jerome H Friedman, and Jerome H Friedman. *The elements of statistical learning: data mining, inference, and prediction*, volume 2. Springer, 2009.

[19] Katherine A Heller and Zoubin Ghahramani. Bayesian hierarchical clustering. In *Proceedings of the 22nd international conference on Machine learning*, pages 297–304, 2005.

[20] Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144*, 2016.

[21] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[22] Alexander Kolesnikov, Lucas Beyer, Xiaohua Zhai, Joan Puigcerver, Jessica Yung, Sylvain Gelly, and Neil Houlsby. Big transfer (bit): General visual representation learning. In *European conference on computer vision*, pages 491–507. Springer, 2020.

[23] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

[24] Richard A. Kronmal and Arthur V. Peterson. On the alias method for generating random variables from a discrete distribution. *The American Statistician*, 33(4):214–218, 2022/09/29/ 1979. ISSN 00031305. doi: 10.2307/2683739. Full publication date: Nov., 1979.

 https://doi.org/10.52202/079017-0443

[25] Andrew Kachites McCallum, Kamal Nigam, Jason Rennie, and Kristie Seymore. Automating the construction of internet portals with machine learning. *Information Retrieval*, 3(2):127–163, 2000.

[26] Gonzalo Mena, David Belanger, Scott Linderman, and Jasper Snoek. Learning latent permutations with gumbel-sinkhorn networks. *arXiv preprint arXiv:1802.08665*, 2018.

[27] Nicholas Monath, Ari Kobren, Akshay Krishnamurthy, and Andrew McCallum. Gradient-based hierarchical clustering. In *31st Conference on neural information processing systems (NIPS 2017), Long Beach, CA, USA*, 2017.

[28] Nicholas Monath, Manzil Zaheer, Daniel Silva, Andrew McCallum, and Amr Ahmed. Gradient-based hierarchical clustering using continuous representations of trees in hyperbolic space. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 714–722, 2019.

[29] Jani Patokallio. Openflights. URL https://openflights.org/.

[30] Max Paulus, Dami Choi, Daniel Tarlow, Andreas Krause, and Chris J Maddison. Gradient estimation with stochastic softmax tricks. *Advances in Neural Information Processing Systems*, 33:5691–5704, 2020.

[31] Max B Paulus, Chris J. Maddison, and Andreas Krause. Rao-blackwellizing the straight-through gumbel-softmax gradient estimator. In *International Conference on Learning Representations*, 2021.

[32] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 701–710, 2014.

[33] Erzsébet Ravasz and Albert-László Barabási. Hierarchical organization in complex networks. *Physical review E*, 67(2):026112, 2003.

[34] Prithviraj Sen, Galileo Namata, Mustafa Bilgic, Lise Getoor, Brian Galligher, and Tina Eliassi-Rad. Collective classification in network data. *AI magazine*, 29(3):93–93, 2008.

[35] Michael Steinbach, George Karypis, and Vipin Kumar. A comparison of document clustering techniques. 2000.

[36] Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(11), 2008.

[37] Joe H Ward Jr. Hierarchical grouping to optimize an objective function. *Journal of the American statistical association*, 58(301):236–244, 1963.

[38] Jaewon Yang and Jure Leskovec. Defining and evaluating network communities based on ground-truth. *Knowledge and Information Systems*, 42(1):181–213, 2015.

[39] Yuchen Zhang, Amr Ahmed, Vanja Josifovski, and Alexander Smola. Taxonomy discovery for personalized recommendation. In *Proceedings of the 7th ACM international conference on Web search and data mining*, pages 243–252, 2014.

[40] Sheng Zhou, Hongjia Xu, Zhuonan Zheng, Jiawei Chen, Jiajun Bu, Jia Wu, Xin Wang, Wenwu Zhu, Martin Ester, et al. A comprehensive survey on deep clustering: Taxonomy, challenges, and future directions. *arXiv preprint arXiv:2206.07579*, 2022.

[41] Daniel Zügner, Bertrand Charpentier, Morgane Ayle, Sascha Geringer, and Stephan Günnemann. End-to-end learning of probabilistic hierarchies on graphs. In *International Conference on Learning Representations*, 2022.

A Appendix

A.1 Equations of Soft-Das and Soft-TSD

In the following, we show the equations of Soft-Das and Soft-TSD.

$$\text{Soft-Das}(\mathcal{T}) = \sum_{\mathrm{v}_i,\mathrm{v}_j \in V} P(\mathrm{v}_i, \mathrm{v}_j) \sum_{\mathrm{z} \in Z} \sum_{v \in V} p\left(\mathrm{z} = \mathrm{v}_i \wedge \mathrm{v}_j\right) p\left(\mathrm{z} \in \mathrm{anc}(\mathrm{v})\right) \tag{9}$$

$$\text{Soft-TSD}(\mathcal{T}) = \sum_{\mathrm{z} \in Z} p(\mathrm{z}) \log \frac{p(\mathrm{z})}{q(\mathrm{z})} \tag{10}$$

$$\text{where } p(\mathrm{z}) = \sum_{\mathrm{v}_i,\mathrm{v}_j \in V} P(\mathrm{v}_i, \mathrm{v}_j) p\left(\mathrm{z} = \mathrm{v}_i \wedge \mathrm{v}_j\right) \tag{11}$$

$$q(\mathrm{z}) = \sum_{\mathrm{v}_i,\mathrm{v}_j \in V} P(\mathrm{v}_i) P(\mathrm{v}_j) p\left(\mathrm{z} = \mathrm{v}_i \wedge \mathrm{v}_j\right) \tag{12}$$

A.2 Closed Form Solutions of Exp-Das and Exp-TSD

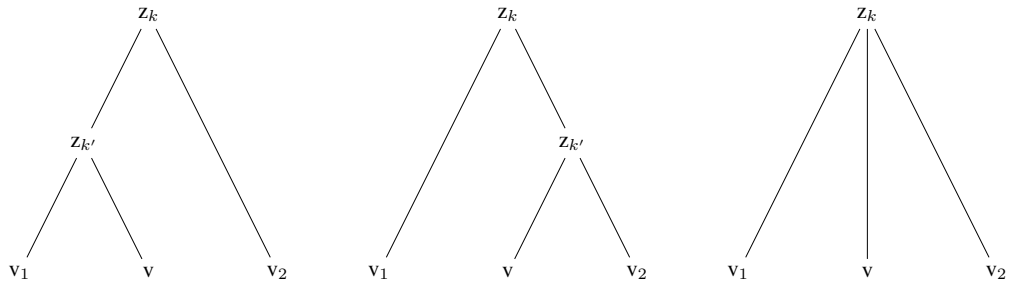
To compute closed-form solutions of the expectations, the following equations need to be solved:

$$\text{Exp-Das}(\mathcal{T}) = \sum_{\mathrm{v}_i,\mathrm{v}_j \in V} P(\mathrm{v}_i, \mathrm{v}_j) \sum_{\mathrm{z} \in Z} \sum_{v \in V} p\left(\mathrm{z} = \mathrm{v}_i \wedge \mathrm{v}_j, \mathrm{z} \in \mathrm{anc}(\mathrm{v})\right) \tag{13}$$

$$\text{Exp-TSD}(\mathcal{T}) = \sum_{\mathrm{z} \in Z} \mathbb{E}_{\hat{\mathcal{T}} \sim P_{\boldsymbol{A},\boldsymbol{B}}(\mathcal{T}=(\boldsymbol{A},\boldsymbol{B}))} \left[p(\mathrm{z}) \log \frac{p(\mathrm{z})}{q(\mathrm{z})} \right]. \tag{14}$$

A.3 Relation between Joint and Independent LCA and Ancestor Probabilities

While the LCA probabilities are crucial to compute Soft-Das, Exp-Das requires the joint LCA and ancestor probabilities, i.e., $p(\mathrm{z}_k = \mathrm{v}_i \wedge \mathrm{v}_j, \mathrm{v} \in \mathrm{anc}(\mathrm{z}_k))$, for the leaves v_i, v_j and v and the internal node z_k. In Prop. A.1, we show that the joint probabilities are an upper bound of the product of the single terms.



(a) $r_{\mathrm{v}}^{\mathrm{z}_k}$ and $r_{\mathrm{v}_1}^{\mathrm{z}_k}$ meet at node $\mathrm{z}_{k'}$. (b) $r_{\mathrm{v}}^{\mathrm{z}_k}$ and $r_{\mathrm{v}_2}^{\mathrm{z}_k}$ meet at node $\mathrm{z}_{k'}$. (c) $r_{\mathrm{v}}^{\mathrm{z}_k}$, $r_{\mathrm{v}_1}^{\mathrm{z}_k}$ and $r_{\mathrm{v}_2}^{\mathrm{z}_k}$ meet at node z_k.

Figure 6: The different cases of the event $p\left(\mathrm{z}_k = \mathrm{v}_1 \wedge \mathrm{v}_2 \mid \mathrm{z}_k \in \mathrm{anc}(\mathrm{v})\right)$. While the LCA of v_1 and v_2 is z_k in every case, the LCA of v_1 and v and the LCA of v_2 and v are different. We have three cases: either the paths from v_1 or v_2 and v meet before z_k at node $\mathrm{z}_{k'}$ (shown in (a) and (b)), or all paths meet for the first time at z_k (shown in fig. (c)).

Proposition A.1. *Let p describe the probability under the tree-sampling procedure, z_k an internal node, v_1, v_2, and v leaves. Then, the following inequality holds:*

$$p(\mathrm{z}_k = \mathrm{v}_1 \wedge \mathrm{v}_2) p(\mathrm{z}_k \in \mathrm{anc}(\mathrm{v})) \leq p\left(\mathrm{z}_k = \mathrm{v}_1 \wedge \mathrm{v}_2, \mathrm{z}_k \in \mathrm{anc}(\mathrm{v})\right) \tag{15}$$

Proof. First, we observe that the right-hand side of the inequality can be rewritten as:

$$p\left(\mathrm{z}_k = \mathrm{v}_1 \wedge \mathrm{v}_2, \mathrm{z}_k \in \mathrm{anc}(\mathrm{v})\right) = p\left(\mathrm{z}_k = \mathrm{v}_1 \wedge \mathrm{v}_2 \mid \mathrm{z}_k \in \mathrm{anc}(\mathrm{v})\right) p(\mathrm{z}_k \in \mathrm{anc}(\mathrm{v})). \tag{16}$$

https://doi.org/10.52202/079017-0443

To prove the non-trivial case $p\left(\mathrm{z}_k \in \mathrm{anc}(\mathrm{v})\right) \neq 0$, we need to show that the following holds:

$$p\left(\mathrm{z}_k = \mathrm{v}_1 \wedge \mathrm{v}_2\right) \leq p\left(\mathrm{z}_k = \mathrm{v}_1 \wedge \mathrm{v}_2 \mid \mathrm{z}_k \in \mathrm{anc}(\mathrm{v})\right). \tag{17}$$

Let $r_{\mathrm{v}_i}^{\mathrm{z}_j} = (\mathrm{v}_i, \ldots, z_j)$ denote a path from a leaf v_i to an internal node z_j and let $\mathrm{z}_{n'}$ be the root node. Recalling from [41] that the paired path probability under the tree-sampling procedure is

$$p((r_{\mathrm{v}_1}^{\mathrm{z}_{n'}}, r_{\mathrm{v}_2}^{\mathrm{z}_{n'}})) = p(r_{\mathrm{v}_1}^{\mathrm{z}_k}) p(r_{\mathrm{v}_2}^{\mathrm{z}_k}) p(r_{\mathrm{z}_k}^{\mathrm{z}_{n'}}), \tag{18}$$

with $\mathrm{z}_k = \mathrm{v}_1 \wedge \mathrm{v}_2$, we can rewrite the LCA probabilities as

$$p\left(\mathrm{z}_k = \mathrm{v}_1 \wedge \mathrm{v}_2\right) = \sum_{(r_{\mathrm{v}_1}^{\mathrm{z}_k}, r_{\mathrm{v}_2}^{\mathrm{z}_k}) : \mathrm{z}_k = \mathrm{v}_1 \wedge \mathrm{v}_2} p(r_{\mathrm{v}_1}^{\mathrm{z}_k}) p(r_{\mathrm{v}_2}^{\mathrm{z}_k}). \tag{19}$$

Adding the condition $\mathrm{z}_k \in \mathrm{anc}(\mathrm{v})$ means there exists a path from the leaf v to the internal node z_k. There are three different cases: first, the path meets $r_{\mathrm{v}_1}^{\mathrm{z}_k}$ and $r_{\mathrm{v}_2}^{\mathrm{z}_k}$ at z_k for the first time, or the path meets the path $r_{\mathrm{v}_1}^{\mathrm{z}_k}$ or $r_{\mathrm{v}_2}^{\mathrm{z}_k}$ in a lower node $\mathrm{z}_{k'}$, with $k' < k$. The cases are shown in Fig. 6. In the first case, all three paths are independent. Thus, the LCA probabilities do not change. In the other two cases, they are only independent up to the node $\mathrm{z}_{k'}$. The probability for the path $r_{\mathrm{z}_{k'}}^{\mathrm{z}_k}$ is equal to 1 since we know that $\mathrm{z}_k \in \mathrm{anc}(\mathrm{v})$. More formally, the conditional probability is

$$p\left(\mathrm{z}_k = \mathrm{v}_1 \wedge \mathrm{v}_2 \mid \mathrm{z}_k \in \mathrm{anc}(\mathrm{v})\right) = \sum_{(r_{\mathrm{v}_1}^{\mathrm{z}_k}, r_{\mathrm{v}_2}^{\mathrm{z}_k}) : \mathrm{z}_k = \mathrm{v}_1 \wedge \mathrm{v}_2} p(r_{\mathrm{v}_1}^{\mathrm{z}_k} \mid \mathrm{z}_k \in \mathrm{anc}(\mathrm{v})) p(r_{\mathrm{v}_2}^{\mathrm{z}_k} \mid \mathrm{z}_k \in \mathrm{anc}(\mathrm{v})). \tag{20}$$

Assuming that the path from v to z_k meets the path from v_1 to z_k in the node $\mathrm{z}_{k'}$ with $k' \leq k$, we have

$$p(r_{\mathrm{v}_1}^{\mathrm{z}_k} \mid \mathrm{z}_k \in \mathrm{anc}(\mathrm{v})) p(r_{\mathrm{v}_2}^{\mathrm{z}_k} \mid \mathrm{z}_k \in \mathrm{anc}(\mathrm{v})) = p(r_{\mathrm{v}_1}^{\mathrm{z}'_k}) p(r_{\mathrm{v}_2}^{\mathrm{z}_k}) \geq p(r_{\mathrm{v}_1}^{\mathrm{z}_k}) p(r_{\mathrm{v}_2}^{\mathrm{z}_k}). \tag{21}$$

The last inequality follows since $r_{\mathrm{v}_1}^{\mathrm{z}'_k}$ is a subpath of $r_{\mathrm{v}_1}^{\mathrm{z}_k}$ and therefore has a higher probability. This concludes the proof. □

A.4 Proof of Proposition 4.2

In the following, we provide the proof of the inequality shown in Prop. 4.2.

Proof. To prove it, we first write out the definitions of Soft-Das and the expected Dasgupta cost.

$$\text{Soft-Das}(\mathcal{T}) = \sum_{\mathrm{v}_1, \mathrm{v}_2} P(\mathrm{v}_1, \mathrm{v}_2) \sum_z \sum_{\mathrm{v}} P(\mathrm{z} = \mathrm{v}_1 \wedge \mathrm{v}_2) P(\mathrm{z} \in \mathrm{anc}(\mathrm{v})) \tag{22}$$

and

$$\begin{aligned}
\mathbb{E}_{\hat{\mathcal{T}} \sim P_{\boldsymbol{A},\boldsymbol{B}}(\mathcal{T})} \left[\mathrm{Das}(\hat{\mathcal{T}})\right] &= \mathbb{E}_{\hat{\mathcal{T}} \sim P_{\boldsymbol{A},\boldsymbol{B}}(\mathcal{T})} \left[\sum_{\mathrm{v}_1, \mathrm{v}_2} P(\mathrm{v}_1, \mathrm{v}_2) \sum_z \sum_{\mathrm{v}} \mathbb{I}_{[z = \mathrm{v}_1 \wedge \mathrm{v}_2]} \mathbb{I}_{[\mathrm{z} \in \mathrm{anc}(\mathrm{v})]}\right] && (23)\\
&= \mathbb{E}_{\hat{\mathcal{T}} \sim P_{\boldsymbol{A},\boldsymbol{B}}(\mathcal{T})} \left[\sum_{\mathrm{v}_1, \mathrm{v}_2} P(\mathrm{v}_1, \mathrm{v}_2) \sum_z \sum_{\mathrm{v}} \mathbb{I}_{[z = \mathrm{v}_1 \wedge \mathrm{v}_2, \mathrm{z} \in \mathrm{anc}(\mathrm{v})]}\right] && (24)\\
&= \sum_{\mathrm{v}_1, \mathrm{v}_2} P(\mathrm{v}_1, \mathrm{v}_2) \sum_z \sum_{\mathrm{v}} \mathbb{E}_{\hat{\mathcal{T}} \sim P_{\boldsymbol{A},\boldsymbol{B}}(\mathcal{T})} \left[\mathbb{I}_{[z = \mathrm{v}_1 \wedge \mathrm{v}_2, \mathrm{z} \in \mathrm{anc}(\mathrm{v})]}\right] && (25)\\
&= \sum_{\mathrm{v}_1, \mathrm{v}_2} P(\mathrm{v}_1, \mathrm{v}_2) \sum_z \sum_{\mathrm{v}} P(z = \mathrm{v}_1 \wedge \mathrm{v}_2, z \in \mathrm{anc}(\mathrm{v})) && (26)
\end{aligned}$$

The proof follows by using Prop. A.1. □

A.5 Proof of Proposition 4.1

Here we provide the proof of Prop. 4.1.

Proof. To prove the left-hand side, we first observe that the expected Dasgupta cost can be rewritten as a convex combination of the Dasgupta costs of all possible hierarchies under the tree-sampling procedure. More formally,

$$\mathbb{E}_{\hat{\mathcal{T}} \sim P_{\boldsymbol{A},\boldsymbol{B}}(\mathcal{T})} \left[\text{Das}(\hat{\mathcal{T}}) \right] = \sum_{\hat{\mathcal{T}} \in \mathcal{H}(n,n')} P_{\boldsymbol{A},\boldsymbol{B}}(\hat{\mathcal{T}}) \text{Das}(\hat{\mathcal{T}}) \tag{27}$$

where $\mathcal{H}(n, n')$ describes the set of all valid hierarchies with n leaves and n' internal nodes. Thus, the minimizer of the expected Dasgupta cost is a convex combination of all minimizing hierarchies, with the minimum being equal to the optimal Dasgupta cost. The equation on the right-hand side for TSD can be proved equivalently. □

Note that since the expectation operator is convex, any *discrete optimizer* (i.e., discrete hierarchies achieving the optimum value) of the discrete scores will be an optimizer of the expected scores and vice-versa. In this case, discrete hierarchies are represented by deterministic $\boldsymbol{A}$, $\boldsymbol{B}$ matrices. Only probabilistic hierarchies, which are optimizers of the expected scores, described by non-discrete $\boldsymbol{A}$ and $\boldsymbol{B}$ matrices, are not optimizers of the discrete scores. This is expected since those probabilistic hierarchies do not belong to the valid input domain of the discrete scores. In addition, any sample we draw from these probabilistic optimizers is also a discrete optimizer of Dasgupta or TSD because of the convexity of the expectation operator.

A.6 Non-Convexity and Non-Concavity of Exp-Das

Minimizing a convex function using gradient descent is easier than a concave one. Minimizing a concave function heavily depends on the initialization in a constrained setting. Exp-Das($\mathcal{T} = (\boldsymbol{A}, \boldsymbol{B})$) is neither convex nor concave with respect to $\boldsymbol{A}$ and $\boldsymbol{B}$. For both, a counter-example exists. This implies that we can not tell whether Exp-Das converges into a local or global minimum when training. To show that Exp-Das is not concave, it is sufficient to find two hierarchies $\mathcal{T}_1 = (\boldsymbol{A}_1, \boldsymbol{B}_1)$ and $\mathcal{T}_2 = (\boldsymbol{A}_2, \boldsymbol{B}_2)$ such that:

$$\frac{1}{2}\text{Exp-Das}(\mathcal{T}_1) + \frac{1}{2}\text{Exp-Das}(\mathcal{T}_2) \geq \text{Exp-Das}\left(\frac{1}{2}(\mathcal{T}_1 + \mathcal{T}_2)\right), \tag{28}$$

and equivalently to show that it is not convex:

$$\frac{1}{2}\text{Exp-Das}(\mathcal{T}_1) + \frac{1}{2}\text{Exp-Das}(\mathcal{T}_2) \leq \text{Exp-Das}\left(\frac{1}{2}(\mathcal{T}_1 + \mathcal{T}_2)\right), \tag{29}$$

where $\mathcal{T}_1 + \mathcal{T}_2 = (\boldsymbol{A}_1 + \boldsymbol{A}_2, \boldsymbol{B}_1 + \boldsymbol{B}_2)$. In Fig. 7, we show these two examples. In (a) and (b), we show two hierarchies, and in (c), a linear interpolation of these two. The graph in (d) satisfies Eq. 28, while the graph in (e) satisfies Eq. 29. We report the Dasgupta costs for all hierarchy and graph combinations in Tab. 6.

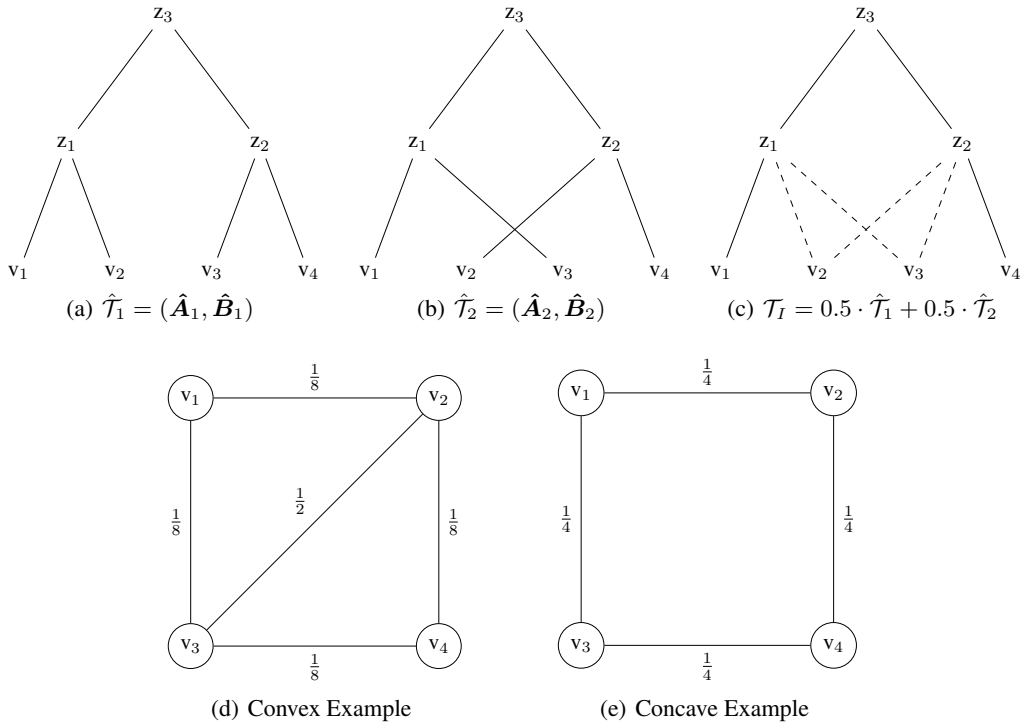
Table 6: Dasgupta costs for all combinations of hierarchies and graphs from Fig. 7.

Hierarchy	$\hat{\mathcal{T}}_1$	$\hat{\mathcal{T}}_2$	$\mathcal{T}_I$
Convex Example	3.5	3.5	3.375
Concave Example	3.0	3.0	3.25

A.7 Weighted Failing Soft-Das Example

In addition to the example in Fig. 2, we show a weighted graph where FPH fails to find the minimizing hierarchy in Fig. 8.

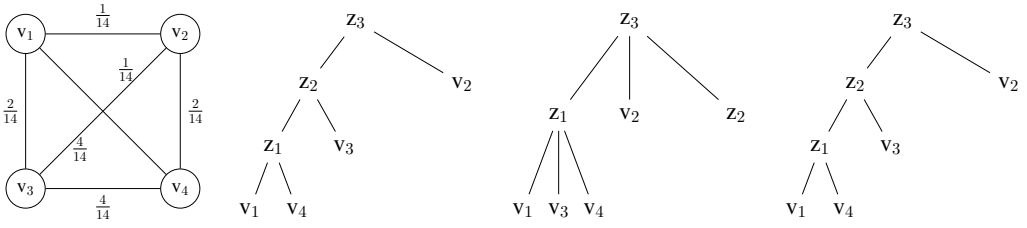
https://doi.org/10.52202/079017-0443



(a) $\hat{\mathcal{T}}_1 = (\hat{\boldsymbol{A}}_1, \hat{\boldsymbol{B}}_1)$ (b) $\hat{\mathcal{T}}_2 = (\hat{\boldsymbol{A}}_2, \hat{\boldsymbol{B}}_2)$ (c) $\mathcal{T}_I = 0.5 \cdot \hat{\mathcal{T}}_1 + 0.5 \cdot \hat{\mathcal{T}}_2$

(d) Convex Example (e) Concave Example

Figure 7: Three hierarchies and two graphs that show that Exp-Das is neither convex nor concave with respect to $\boldsymbol{A}$ and $\boldsymbol{B}$. The hierarchy in (c) is a linear interpolation of the hierarchies in (a) and (b). The graphs in (d) and (e) are counter-examples, with convex and concave behavior, respectively.



(a) K_4 Graph (b) A minimizing hierarchy (c) FPH (d) EPH

Figure 8: An example of a weighted K_4 graph where FPH fails to infer a minimizing hierarchy. While FPH achieves a Dasgupta cost of $\frac{23}{7}$ after discretization, the continuous hierarchy has a Soft-Das score below 3.0. On the other hand, EPH finds a minimizing hierarchy with a cost of 3.0.

B Experiments Information

B.1 Datasets

An overview of the graph and vector datasets is given in Tab. 7 and Tab. 8. The details of the HSBMs are shown in Tab. 9.

Table 7: Overview of the graph datasets.

Dataset	Number of Nodes	Number of Edges	License
PolBlogs	1222	16715	n/a
Brain	1770	8957	n/a
Citeseer	2110	3694	n/a
Genes	2194	2688	n/a
Cora-ML	2810	7981	n/a
OpenFlight	3097	18193	OBdL
WikiPhysics	3309	31251	n/a
DBLP	317080	1049866	n/a

Table 8: Overview of the vector datasets.

Dataset	Number of Data Points	Number of Attributes	Number of Classes	License
Zoo	101	17	7	CC BY 4.0
Iris	150	4	3	CC BY 4.0
Glass	214	10	6	CC BY 4.0
Digits	1797	8×8	10	CC BY 4.0
Segmentation	2310	19	7	CC BY 4.0
Spambase	4601	57	2	CC BY 4.0
Letter	20000	16	26	CC BY 4.0
Cifar-100	50000	2048	100	n/a

Table 9: Overview of the HSBMs.

Dataset	Number of Nodes	Number of Edges	Number of Clusters
Small HSBM	101	1428	15
Large HSBM	1186	27028	53

https://doi.org/10.52202/079017-0443

B.2 Hyperparameters

We show an overview of the hyperparameters we used in Tab. 10. Furthermore, we compare the original FPH results with the tuned results, i.e., the minimum of the original, and using the aforementioned scheduler in Tab. 11.

Table 10: Overview of the Hyperparameters.

Method	Hyperparameter	Value
EPH	LR	Scheduler
	Initialization	Average Linkage
	Num. Samples	20
	Num. Samples*	10
	Num. Samples**	1
FPH	LR	$\min\{\text{Scheduler}, 0.05\}$
	LR	$\min\{\text{Scheduler}, 150\}$
	Initialization	Average Linkage
	Epochs	1000
HypHC	LR	$\min\{1e^{-3}, 5e^{-4}, 1e^{-4}\}$
	Temp.	$\min\{1e^{-1}, 5e^{-2}, 1e^{-2}\}$
	LR***	$1e^{-3}$
	Temp.***	$1e^{-1}$
	Epochs	50
	Num. Triplets	n^2
UF	Loss	$\min\{\text{Dasgupta, Closest+Size}\}$
	LR	0.1
	Epochs	500
Scheduler	LR_A (Exp-Das)	0.1
	LR_A (Exp-Das)****	0.05
	LR_B (Exp-Das)	0.1
	LR_B (Exp-Das)*****	0.01
	LR_A (Exp-TSD)	150
	LR_B (Exp-TSD)	500
	Sampling frequency	1000
	Sampling frequency*****	2000
	Epochs (Exp-Das)	10000
	Epochs (Exp-TSD)	3000
DeepWalk	Embedding Dim.	10
	Embedding Dim.*****	32

* Used for DBLP and Spambase
** Used for Letter and Cifar-100
*** Used for Letter
**** Used for Cifar-100
***** Used for DBLP

Table 11: Dasgupta costs of the original FPH (orig.) and with our modifications (tuned). Improvements are highlighted in bold.

	Dasgupta cost								Tree-sampling divergence							
Dataset	PolBl.	Brain	Cites.	Genes	Cora-ML	OpenF.	WikiP.	DBLP	PolBl.	Brain	Cites.	Genes	Cora-ML	OpenF.	WikiP.	DBLP
FPH (orig.)	262.48	503.67	77.16	183.63	257.42	355.61	537.95	31,687	31.37	32.23	69.37	67.69	58.02	57.58	49.87	41.62
FPH (tuned)	**238.65**	**425.70**	**76.03**	**182.91**	257.42	355.61	**482.40**	31,687	**31.41**	**32.75**	**69.38**	**67.78**	**59.55**	57.58	49.87	41.62

C Additional Results

C.1 HSBM Results for FPH

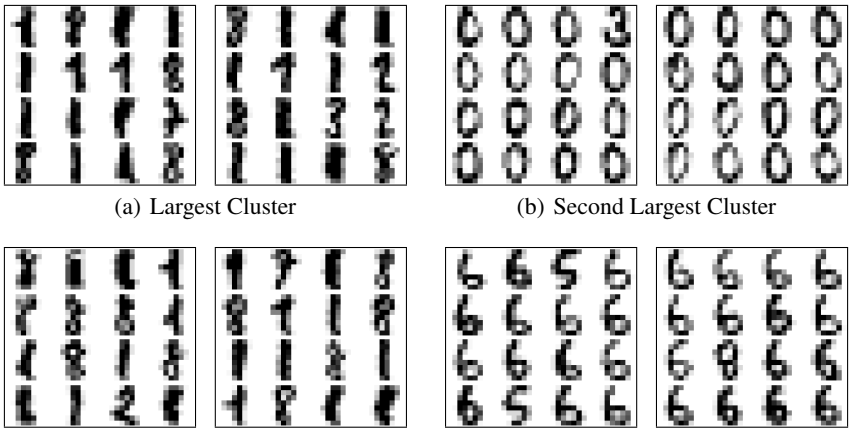
We show the results for FPH on the HSBM graphs in Tab 12.

Table 12: Results of FPH for the HSBMs with n'=# Cluster.

	Dasgupta cost		Tree-sampling divergence	
Method	HSBM Small	HSBM Large	HSBM Small	HSBM Large
GT	26.29	130.16	43.14	51.50
FPH	27.84	127.99	43.53	51.53
Level 1	1.0	0.99	1.0	1.0
Level 2	1.0	0.95	1.0	1.0
Level 3	0.77	0.81	0.87	0.99

C.2 Additional Visualizations

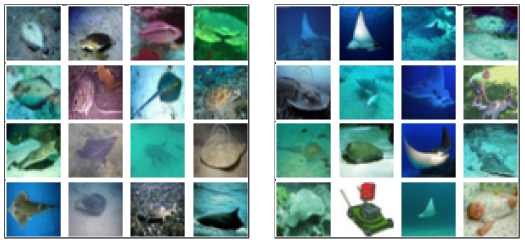
We show the largest clusters for the Digits and Cifar100 datasets in Fig. 9, Fig. 10, and Fig. 11. We observe that EPH provides qualitatively reasonable clusters and groups similar images.



(a) Largest Cluster (b) Second Largest Cluster

(c) Third Largest Cluster (d) Fourth Largest Cluster

Figure 9: Largest derived clusters on Digits. On the left in each subplot the 16 images with the highest probability, on the right the 16 images with the lowest probability.

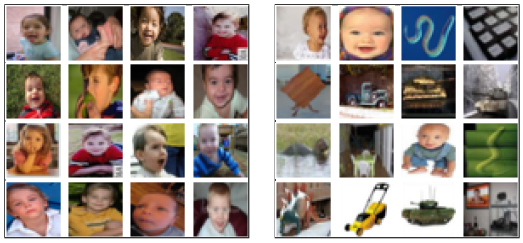


(a) Highest Probability (b) Lowest Probability

Figure 10: Second largest derived cluster on Cifar-100.

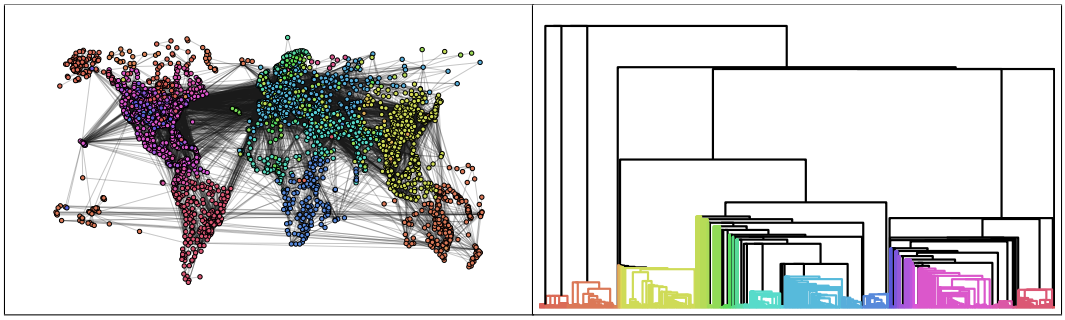
Additionally, we provide a visualization of the OpenFlight dataset in Fig. 12, showing that both Exp-Das and Exp-TSD yield meaningful hierarchies and clusters. Finally, we apply t-SNE [36] to selected datasets and compare the ground-truth labels with clusters inferred by EPH in Fig. 13.

https://doi.org/10.52202/079017-0443

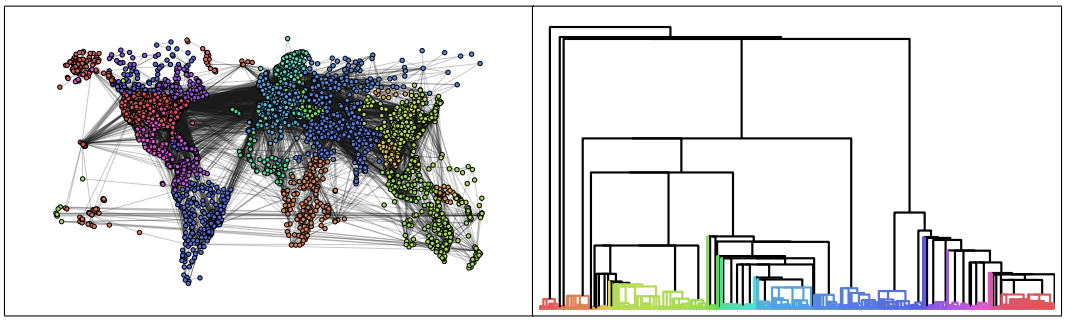


(a) Highest Probability (b) Lowest Probability

Figure 11: Third largest derived cluster on Cifar-100.



(a) Das - 64 clusters



(b) TSD - 64 clusters

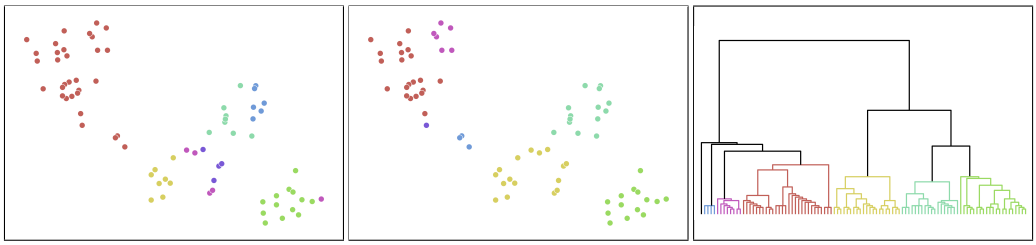
Figure 12: Inferred clusters inferred by EPH optimizing Exp-Das and Exp-TSD. 64 clusters are highlighted in the graphs and dendrograms.

C.3 Additional External Evaluation

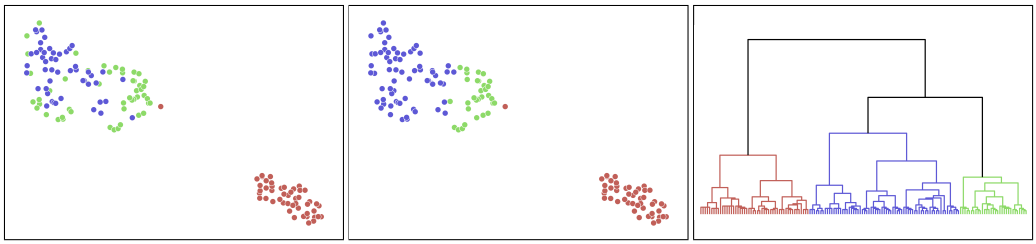
In addition to the unsupervised metrics, we investigate whether the inferred hierarchies on the vector datasets preserve the flat ground truth class labels. To do this, we flatten the derived hierarchies and compare inferred clusters with the available ground-truth labels by applying the Hungarian algorithm to align the cluster assignments with the labels, as explained by Zhou et al. [40]. Using this procedure, we compute the accuracies, shown in Tab. 13. While the linkage algorithms were inferior to the continuous optimization algorithms in terms of Dasgupta cost, they dominate here. EPH, trained on Exp-Das, yields the best accuracies only on Iris and Spambase. As the linkage algorithms and Louvain generate hierarchies using heuristics while the continuous methods aim to minimize the Dasgupta cost, the results are not surprising since the Dasgupta cost and other metrics do not necessarily go hand in hand.

C.4 Ablation

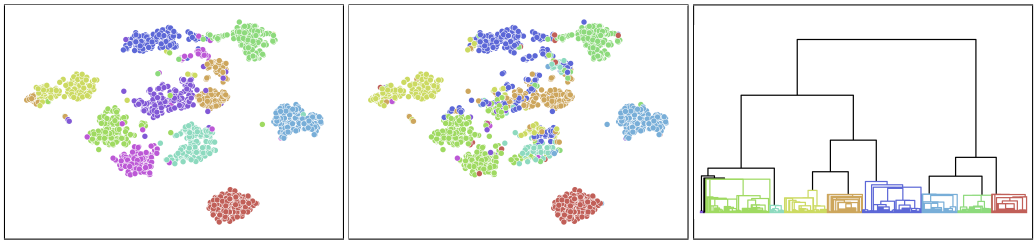
For our ablation study, we use a simplified optimization scheme. More specifically, we use a fixed learning rate of 0.05 and only train for 1000 epochs.



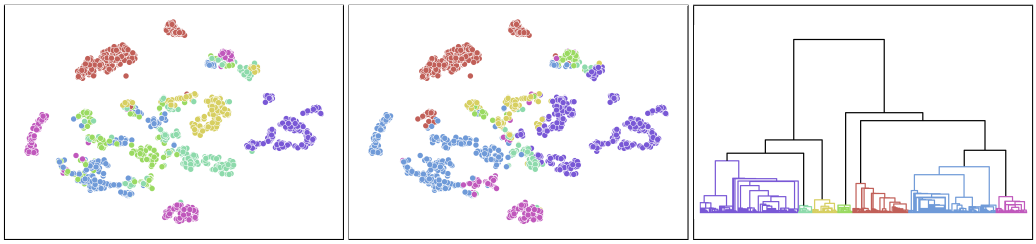
(a) Zoo



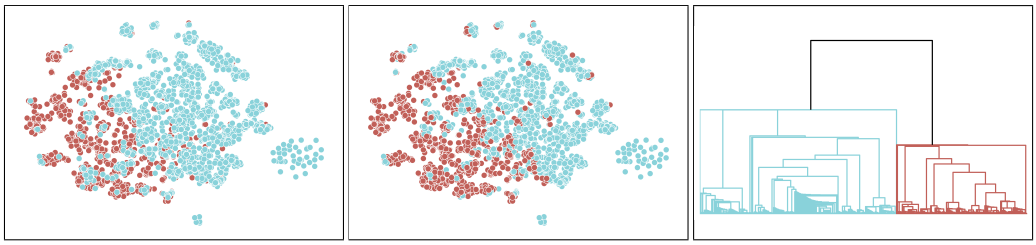
(b) Iris



(c) Digits



(d) Segmentation



(e) Spambase

Figure 13: Ground truth clusters (left) compared to inferred flattened clusters (middle) and dendrograms (right) of EPH for the datasets Zoo, Iris, Digits, Segmentation, and Spambase with $n' = \min\{n-1, 512\}$ internal nodes after applying the t-SNE algorithm.

https://doi.org/10.52202/079017-0443

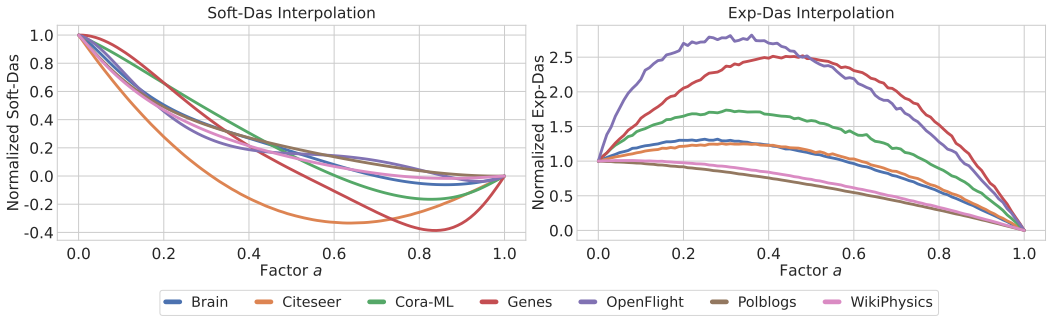


Figure 14: Linear interpolation of Soft-Das and Exp-Das scores from the average linkage hierarchy to the hierarchy inferred by Exp-Das.

Table 13: Accuracies for the vector datasets. Best scores in bold, second best underlined.

Dataset	Zoo	Iris	Glass	Digits	Segm.	Spam.	Letter	Cifar
WL	0.74	0.76	0.44	**0.82**	**0.61**	0.76	**0.33**	**0.49**
AL	0.80	0.82	0.43	0.65	0.48	0.82	0.31	0.16
SL	**0.84**	0.67	0.37	0.10	0.15	0.61	0.04	0.01
CL	0.81	0.77	0.40	0.59	0.51	0.74	0.26	0.33
Louv.	0.60	**0.83**	0.42	0.68	0.29	**0.86**	0.18	0.05
RSC	0.41	0.35	0.38	0.39	0.51	0.84	0.11	0.05
UF	0.60	0.55	0.43	0.43	0.54	0.54	0.04	OOM
gHHC	0.59	0.71	0.42	0.51	0.30	0.69	0.08	0.01
HypHC	0.79	0.82	**0.52**	0.42	0.29	0.60	0.10	OOM
FPH	0.58	**0.83**	0.40	0.20	0.44	0.61	0.06	0.03
EPH	0.70	**0.83**	0.40	0.65	0.53	**0.86**	0.14	0.18

Constrained vs. Unconstrained Optimization. We require the rows of the matrices $\boldsymbol{A}$ and $\boldsymbol{B}$ to be row-stochastic. There are several possibilities to enforce this. Either we can perform constrained optimization using projections onto the probabilistic simplex or perform a softmax operation over the rows. In Tab. 14, we compare the Dasgupta costs on the graph datasets for several graph datasets. We can observe that the constrained optimization,i.e., using projections after each step, yields better

Table 14: Dasgupta costs for constrained and unconstrained optimization on several graph datasets with $n' = 512$ internal nodes.

Dataset	PolBlogs	Brain	Citeseer	Genes	Cora-ML	OpenFlight	WikiPhysics
Constrained	**252.55**	**428.40**	**74.84**	**178.90**	**242.38**	**324.45**	**481.92**
Unconstrained	272.60	457.62	79.47	188.02	269.10	349.20	526.99

results than the unconstrained optimization on every graph. This aligns with the findings of Zügner et al. [41]. Therefore, we recommend using constrained optimization.

Initialization. The initialization of a model can play a crucial role. Zügner et al. [41] found that using the AL algorithm as initialization yields substantial improvements. Therefore, we compare both initializations and additionally test using their algorithm FPH as initialization. We show the Dasgupta costs for several graph datasets in Tab. 15. As expected, using the AL algorithm or FPH

Table 15: Dasgupta costs for different initializations on several graph datasets with $n' = 512$ internal nodes. In the first three rows the initial Dasgupta costs and in the last three rows the Dasgupta costs after the training. Best scores in bold, second best underlined.

Dataset	PolBlogs	Brain	Citeseer	Genes	Cora-ML	OpenFlight	WikiPhysics
Random	914.11	1285.68	1574.76	1621.82	2107.65	2302.13	2479.51
AL	355.60	556.68	83.69	196.50	292.77	363.40	658.04
FPH	262.47	453.17	77.05	179.55	257.42	355.61	538.47
Exp-Das (Random)	275.44	499.12	307.52	368.66	564.35	502.22	624.27
Exp-Das (AL)	252.55	**428.40**	**74.84**	178.90	**242.87**	**324.45**	**481.92**
Exp-Das (FPH)	**251.55**	431.15	74.88	**177.32**	245.79	326.46	527.02

as initialization yields significant improvements over a random initialization. Even though the FPH initialization starts with a better hierarchy, the resulting hierarchies are inferior to the AL initialization. This could be caused by local minima, in which the model gets stuck. We recommend using AL as initialization since it performs best on most datasets and has a lower computational cost than FPH.

Direct vs. Embedding Parametrization. Additionally to the direct parametrization of the matrices $\boldsymbol{A}$ and $\boldsymbol{B}$, we test an embedding parametrization for each node in the hierarchy. More specifically, we use d-dimensional embeddings for the leaves and internal nodes. We perform a softmax operation with an additional learnable temperature parameter t_i to infer $\boldsymbol{A}$ and $\boldsymbol{B}$ over the cosine similarities between the embeddings. The main advantage of the embedding approach is that, in addition the hierarchical clustering, we gain node embeddings that can be used for downstream tasks such as classification or regression. We test the embedding parametrization with $d = 128$ on several graph datasets. Once we let t_i be learnable, and once we freeze it to $t_i = 1$. We compare the results to the constrained optimization. While we train the direct parametrization for 1000 epochs, the embedding approach is trained for 20000 epochs. This is done to ensure convergence since it is randomly initialized. We show the results in Tab. 16. First, we observe that not using a temperature parameter yields substantially worse

Table 16: Dasgupta costs for the direct and embedding parametrization on several graph datasets with $n' = 512$ internal nodes. Best scores in bold, second best underlined.

Dataset	PolBlogs	Brain	Citeseer	Genes	Cora-ML	OpenFlight	WikiPhysics
Direct	<u>252.55</u>	**428.40**	**74.84**	**178.90**	**242.38**	**324.45**	**481.92**
Embedding ($t_i = 1$)	451.63	659.11	1008.23	1146.86	1261.91	968.41	1108.42
Embedding	**249.84**	<u>440.29</u>	<u>213.99</u>	<u>290.19</u>	<u>409.68</u>	<u>373.07</u>	<u>514.91</u>

results. Furthermore, the embedding parametrization is inferior to the direct parametrization, even though it was trained for 20000 epochs, while the constrained optimization was only trained for 1000. Only on the dataset PolBlogs the embedding approach is slightly better than the direct parametrization. We attribute the inferior performance to the random initialization and the fact that we have to use a softmax operation instead of projections. Our results are in line with the ablation study of Zügner et al. [41]. They also parametrized their model using embeddings and used the softmax function on the negative Euclidean distances to infer the matrices $\boldsymbol{A}$ and $\boldsymbol{B}$. Since the embedding approach yields worse results with longer training times, we recommend using the direct parametrization.

Number of Internal Nodes. As in many real-world problems, we do not know the number of internal nodes n' beforehand in our experiments. While increasing n' generally leads to more refined and expressive hierarchies, it reduces interpretability and comes with a higher computational cost. To select the hyperparameter n', we test various choices on several datasets. We show the corresponding Dasgupta costs and TSD scores in Fig. 15 and Fig. 16. We found that $n' = 512$ is sufficient to capture most information. In practice, we recommend using the Elbow method.

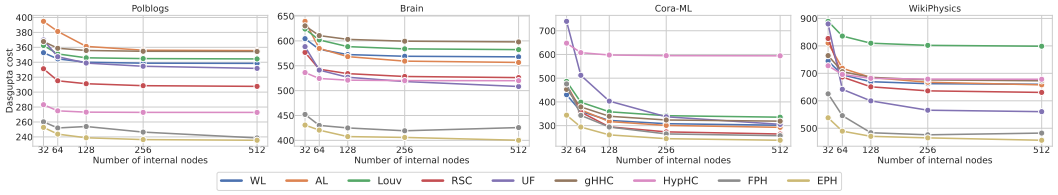


Figure 15: Dasgupta costs for different numbers of internal nodes.

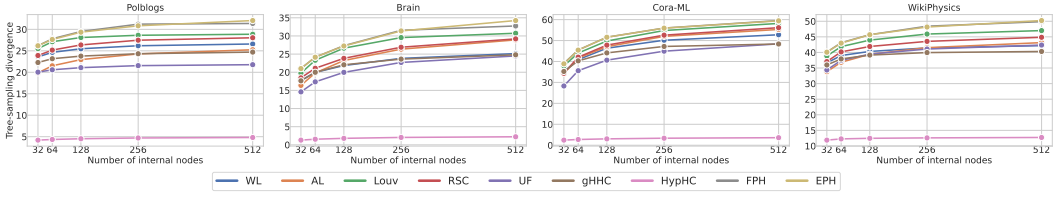


Figure 16: TSD scores for different numbers of internal nodes.

https://doi.org/10.52202/079017-0443

Number of Sampled Hierarchies. Another crucial hyperparameter for EPH is the number of sampled hierarchies. Additionally to Fig. 3, we provide the raw Dasgupta costs and standard errors after the training in Fig. 17. Furthermore, we show the influence of the number of samples to approximate the expected Dasgupta cost on randomly initialized hierarchies in Fig. 18.

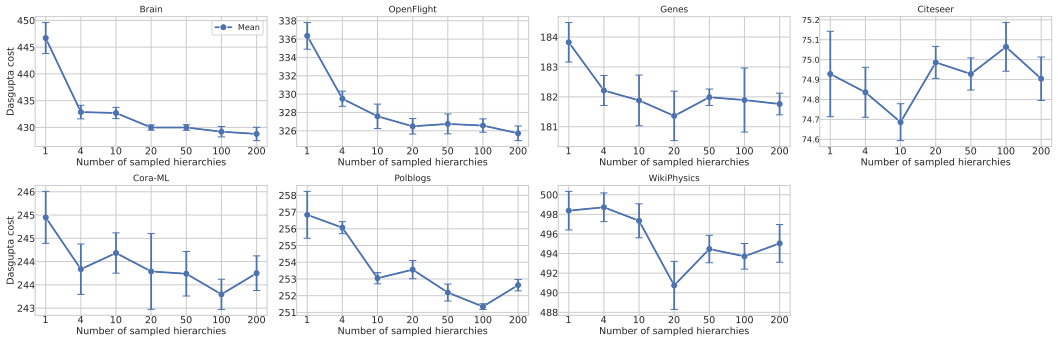


Figure 17: Dasgupta costs and standard error for different numbers of sampled hierarchies after the EPH training.

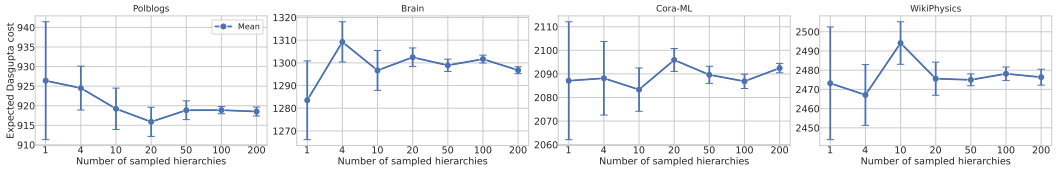


Figure 18: Approximated Expected Dasgupta costs for different numbers of sampled hierarchies for randomly initialized probabilistic hierarchies.

Minimized Version. To reduce EPH's runtime, we introduce a minimized version of our algorithm that samples a single hierarchy and omits validation steps. The final hierarchy is selected on the training loss, enabled by the alignment between expected and discrete scores. We compare this minimized version to the original and FPH on the graph datasets in Tab. 17, with all methods trained using the original setup described in Sec. 5.

Table 17: Results of the minimized version of EPH for the graph datasets. Best scores in **bold**, second best underlined.

	Dasgupta cost (↓)								Tree-sampling divergence (↑)							
Dataset	PolBl.	Brain	Cites.	Genes	Cora-ML	OpenF.	WikiP.	DBLP	PolBl.	Brain	Cites.	Genes	Cora-ML	OpenF.	WikiP.	DBLP
FPH	238.65	425.70	76.03	182.91	257.42	355.61	482.40	31,687	31.37	32.75	**69.38**	67.78	**59.55**	57.58	49.87	41.62
EPH	**235.50**	**400.20**	74.01	176.57	238.28	312.31	**456.26**	**30,600**	**32.05**	34.24	69.36	67.75	59.41	57.83	**50.23**	**42.74**
EPH (minimized)	236.86	404.30	**72.81**	**173.56**	**238.11**	**309.33**	463.63	30,644	31.92	**34.26**	69.36	**67.85**	59.54	**57.92**	50.19	42.70

Our minimized version outperforms FPH in 14 out of 16 cases while achieving a shorter runtime, as shown in Tab. 20.

C.5 Standard Deviations

We show the standard deviations of the randomized models on the graph datasets in Tab. 18 and for the vector datasets in Tab. 19.

C.6 Runtimes

We report the runtimes for EPH and the baselines in Tab. 20 and Tab. 21. While HypHC, FPH, and EPH are executed on a GPU (NVIDIA A100), the remaining methods do not support or do not require GPU acceleration. Since gHHC has a lower computational runtime than the other randomized methods, we run it with 50 random seeds instead of 5.

https://doi.org/10.52202/079017-0443

Table 18: Standard Deviations for the graph datasets.

	Dasgupta cost								Tree-sampling divergence							
Dataset	PolBl.	Brain	Cites.	Genes	Cora-ML	OpenF.	WikiP.	DBLP	PolBl.	Brain	Cites.	Genes	Cora-ML	OpenF.	WikiP.	DBLP
HypHC	6.37	9.05	22.93	16.54	51.74	28.83	36.64	OOM	0.47	0.64	0.71	0.35	1.47	1.02	1.22	OOM
EPH	0.43	1.31	0.10	2.17	1.02	2.49	2.95	35.89	0.31	0.39	0.01	0.07	0.02	0.12	0.15	0.03

Table 19: Standard Deviations for the vector datasets.

	Dasgupta cost								Accuracy							
Dataset	Zoo	Iris	Glass	Digits	Segmentation	Spambase	Letter	Cifar	Zoo	Iris	Glass	Digits	Segmentation	Spambase	Letter	Cifar
HypHC	0.08	0.58	0.27	1.37	0.46	2.04	45.10	OOM	0.05	0.01	0.05	0.07	0.06	<0.01	0.01	OOM
FPH	-	-	-	5.29	3.07	19.31	96.52	528.68	-	-	-	0.04	0.06	0.01	0.02	0.01
EPH	0.01	0.01	0.01	0.11	0.22	1.11	2.42	5.73	0.03	<0.01	0.01	0.06	0.05	<0.01	0.01	0.01

C.7 Pseudocodes

In the following, we provide a formal description of our EPH algorithm, the subgraph sampling, and how we normalize graphs.

Algorithm 1 EPH

Input: $\mathcal{G} = (V, E)$: Graph
Input: $\mathcal{T} = (\boldsymbol{A}, \boldsymbol{B})$: Initial hierarchy
Input: α: Learning rate
Input: K: Number of sampled hierarchies
for $t = 1, \dots$ **do**
  $g_t \leftarrow 0$
  for $k = 1, \dots, K$ **do**
    $\hat{\mathcal{G}} \leftarrow$ SampleSubgraph($\mathcal{G}$)
    $\hat{\mathcal{T}} \sim P_{\boldsymbol{A},\boldsymbol{B}}(\mathcal{T})$
    $g_t \leftarrow g_t + \nabla_{\mathcal{T}}$Score$(\hat{\mathcal{G}}, \hat{\mathcal{T}})$
  end for
  $\mathcal{T}_t \leftarrow \mathcal{T}_{t-1} - \frac{\alpha}{K} g_t$
  $\mathcal{T}_t \leftarrow P(\mathcal{T}_t)$ //simplex projection
end for
return $\hat{\mathcal{T}}_t$

Algorithm 2 NormalizeGraph

Input: $\mathcal{G} = (V, E)$: Graph
$P(\mathrm{v}_i, \mathrm{v}_j) \leftarrow \frac{w_{i,j}}{\sum_{\mathrm{u},\mathrm{v} \in V} w_{u,v}}$
$P(\mathrm{v}_i) \leftarrow \frac{w_i}{\sum_j w_j}$

Algorithm 3 SampleSubgraph

Input: $\mathcal{G} = (V, E)$: Graph
Input: M: Number of sampled edges
$\hat{E} \leftarrow$ MultiSet() //allow duplicate edges
for $m = 1 \dots$ M **do**
  $e = (\mathrm{v}_i, \mathrm{v}_j) \sim P(\mathrm{v}_i, \mathrm{v}_j)$
  $\hat{E}$.add(e)
end for
$\hat{\mathcal{G}} \leftarrow (V, \hat{E})$
NormalizeGraph($\hat{\mathcal{G}}$)
return $\hat{\mathcal{G}}$

https://doi.org/10.52202/079017-0443

Table 20: Runtime in seconds for the graph datasets with $n' = 512$ internal nodes.

	PolBlogs	Brain	Citeseer	Genes	Cora-ML	OpenFlight	WikiPhysics	DBLP
# Nodes	1222	1770	2110	2194	2810	3097	3309	317080
# Edges	16715	8957	3694	2688	7981	18193	31251	1049866
WL	1	<1	<1	<1	<1	1	1	OOM
AL	1	1	<1	<1	1	1	2	101
Louv.	1	1	1	1	1	1	1	1031
RSC	92	78	104	427	280	626	863	OOM
UF	9	3	1	1	2	4	6	OOM
gHHC	75	79	73	78	73	79	83	15630
HypHC	2043	4163	5981	6816	11557	14278	16778	OOM
FPH	452	547	345	373	644	592	667	6647
EPH	1496	1066	749	521	1196	1654	2325	224331
EPH (minimized)	111	100	72	79	85	116	157	4160

Table 21: Runtime in seconds for the vector datasets with $n' = \min\{n - 1, 512\}$ internal nodes.

	Zoo	Iris	Glass	Digits	Segmentation	Spambase	Letter	Cifar-100
# Points	101	150	214	1797	2310	4601	20000	50000
WL	<1	<1	<1	8	13	51	983	8316
AL	<1	<1	<1	8	13	51	985	8564
SL	<1	<1	<1	8	13	51	975	8494
CL	<1	<1	<1	8	13	52	986	8594
Louv.	<1	<1	<1	8	14	55	1065	7324
RSC	1	1	2	27	40	127	2009	14110
UF	<1	<1	<1	9	14	57	1132	OOM
gHHC	47	57	59	83	66	89	110	8462
HypHC	47	60	77	3385	5814	26933	250792	OOM
FPH	87	93	144	2586	3963	13876	134845	427557
EPH	445	763	1337	4328	6645	19974	130227	430322

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract accurately describes the contributions and claims of the paper. The methodology is explained in Sec. 4, and theoretical properties and methodology are shown and discussed in Sec. 4.2. Our subgraph sampling approach is discussed in Sec. 4.4. The results on the graph and vector datasets are presented in Sec. 5.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: The limitations are discussed in Sec. 6.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory Assumptions and Proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

https://doi.org/10.52202/079017-0443

Answer: [Yes]

Justification: All theoretical results include assumptions and proofs and are presented in App. A.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental Result Reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: All training details and hyperparameters are disclosed. We provide all hyperparameters in Tab. 10, pseudeocodes in Alg. 1 and Alg. 3, and explain the experimental setup in Sec. 5.1.

Guidelines:

- The answer NA means that the paper does not include experiments.
- If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: The project page of EPH is attached and the datasets (except the synthetic HSBMs) are publicly available. The baselines are not attached since these are all publicly available. Furthermore, the submitted code includes information about the environment used for the experiments.

   Guidelines:

   - The answer NA means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (https://nips.cc/public/guides/CodeSubmissionPolicy) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (https://nips.cc/public/guides/CodeSubmissionPolicy) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental Setting/Details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: Experimental details are provided in Sec. 5.1 and Tab. 10.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment Statistical Significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: Standard deviations of our experiments are provided in Tab. 18 and Tab. 19.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.

https://doi.org/10.52202/079017-0443

- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments Compute Resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Machine and runtimes are stated in Sec. C.6.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code Of Ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics https://neurips.cc/public/EthicsGuidelines?

   Answer: [Yes]

   Justification: The authors have reviewed the NeurIPS Code of Ethics and ensured that the paper is conform with the code.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader Impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [NA]

    Justification: This paper proposes a hierarchical clustering algorithm for graphs and vector data. There are no no broader societal impacts.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.

- If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: The proposed models do not pose a risk for misuse.

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: The licenses of the used datasets are shown in Tab. 7 and Tab. 8. Furthermore, all baselines and methods are cited accordingly.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.

https://doi.org/10.52202/079017-0443

- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New Assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: All hyperparameters are shown in Tab. 10, pseudeocodes in Alg. 1 and Alg. 3, and the experimental setup is explained in Sec. 5.1

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and Research with Human Subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification:

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional Review Board (IRB) Approvals or Equivalent for Research with Human Subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification:

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.

- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

https://doi.org/10.52202/079017-0443