A PROOFS AND THEORETICAL ANALYSIS

A.1 CONTINUOUS VERSIONS OF TSD AND DASGUPTA

Our probabilistic hierarchy model enables us to replace the discrete parent assignments in the Dasgupta cost and TSD with the parent probabilities from the relaxed adjacency matrices $\boldsymbol{A}$ and $\boldsymbol{B}$. This, in turn, leads to LCA *probabilities* which are consistent under the tree-sampling procedure. For the first time, this allows us to *directly* and efficiently optimize for hierarchical clustering quality metrics in an end-to-end fashion instead of proxy losses or heuristic algorithms.

Hence, we propose the soft Dasgupta cost **(Soft-Das)** by replacing the cardinality of the lowest common ancestor $|\mathrm{v}_i \wedge \mathrm{v}_j|$ by the expected cardinality under the probabilistic model, i.e. $c(\mathrm{z}) = \sum_v p_{\text{anc}}(\mathrm{z}|\mathrm{v})$.

$$\text{Soft-Das}(\boldsymbol{P}_{\boldsymbol{A},\boldsymbol{B}}(\mathcal{T})) = \sum_{\mathrm{v}_i,\mathrm{v}_j \in V} P(\mathrm{v}_i, \mathrm{v}_j) \sum_{\mathrm{z}} p(\mathrm{z} = \mathrm{v}_i \wedge \mathrm{v}_j) c(\mathrm{z})$$

In the same way, we propose a differentiable version of the soft Tree Sampling Divergence **(Soft-TSD)**. To this end, we replace the discrete assignments in the distributions $p(\mathrm{z})$ and $q(\mathrm{z})$ with probabilistic assignments, i.e.

$$\text{Soft-TSD}(\boldsymbol{P}_{\boldsymbol{A},\boldsymbol{B}}(\mathcal{T})) = \mathrm{KL}(p(\mathrm{z})||q(\mathrm{z})) = \sum_{\mathrm{z}} p(\mathrm{z}) \log \frac{p(\mathrm{z})}{q(\mathrm{z})}$$

$$\text{where } p(\mathrm{z}) = \sum_{\mathrm{v}_i,\mathrm{v}_j} p(\mathrm{z} = \mathrm{v}_i \wedge \mathrm{v}_j) P(\mathrm{v}_i, \mathrm{v}_j)$$

$$q(\mathrm{z}) = \sum_{\mathrm{v}_i,\mathrm{v}_j} p(\mathrm{z} = \mathrm{v}_i \wedge \mathrm{v}_j) P(\mathrm{v}_i) P(\mathrm{v}_j)$$

Note that for both metrics we recover the same score as their discrete formulations in the case of a deterministic probabilistic model, i.e. when $\boldsymbol{A}$ and $\boldsymbol{B}$ are binary-valued.

A.2 TREE-SAMPLING PROCEDURE

Recall our assumption that the internal nodes are ordered, and that $B_{ij} = 0$ if $j \leq i$. This implies that there are no possible cycles, or equivalently that $\boldsymbol{B}$ is a strictly upper-triangular matrix, i.e., it describes a directed acyclic graph (DAG). Combined with the fact that each node in a tree (except the root) has *exactly* one parent, we see that the sampled discrete hierarchy is indeed a tree. We denote this tree-sampling process by $\mathcal{T} = (\hat{\boldsymbol{A}}, \hat{\boldsymbol{B}}) \sim P_{\boldsymbol{A},\boldsymbol{B}}(\mathcal{T})$. We can also compute the probability of any tree $\mathcal{T}$ under the sampling procedure described above:

$$P_{\boldsymbol{A},\boldsymbol{B}}(\mathcal{T} = (\hat{\boldsymbol{A}}, \hat{\boldsymbol{B}})) = \prod_{i,j} A_{i,j}^{\hat{A}_{i,j}} \prod_{i',j'} B_{i',j'}^{\hat{B}_{i',j'}} \tag{10}$$

Note that $A_{i,j}$ and $B_{i',j'}$ are the probabilities of internal nodes z_j and $\mathrm{z}_{j'}$ to be a parent of leaf and internal nodes v_i and $\mathrm{z}_{i'}$ respectively while $\hat{A}_{i,j}$ and $\hat{B}_{i',j'}$ are equal to 1 if these connections exist in the tree $\mathcal{T}$, else 0.

A.3 PROOF OF THEOREM 1

Proof. For $\mathcal{M}$ to be absorbing (i) it must have at least one absorbing state, and (ii) at least one absorbing state must be reachable from any state in a finite number of steps. For (i), ω is an absorbing state since its self-transition probability $\boldsymbol{T}_{k,k} = 1$, where $k = |\mathcal{S}| + 1$ is its corresponding index in the transition matrix $\boldsymbol{T}$. Thus, once reached, a random walk cannot leave the state ω. To show (ii), we note that since the transient state transition matrix $\boldsymbol{Q}$ is a strictly upper-triangular matrix, any random walk on $\mathcal{M}$ must lead to state $\mathrm{z}_{n'}$. From state $\mathrm{z}_{n'}$, the random walk transits to ω with probability $\boldsymbol{w}_{n'} = 1$. Thus, since n' is finite, state ω can be reached from any state in a finite number of steps. Further, ω is the only absorbing state since all self-transition probabilities of states in $\mathcal{S}$ are zero as $\text{diag}(\boldsymbol{Q}) = \boldsymbol{0}$. Since $\boldsymbol{Q}$ is strictly upper-triangular, none of the transient states can be visited more than once on a random walk, and therefore $\mathcal{M}$ is acyclic. □

A.4 PROOF OF THEOREM 2

Proof. We can arbitrarily define the order in which we sample from the categorical distributions in $\boldsymbol{A}$ and $\boldsymbol{B}$ because of the independence of the sampling steps. We choose to start by sampling first from $\boldsymbol{A}_i$, i.e., the row corresponding to the leaf node v_i under consideration: $w^{(1)} \sim \text{Cat}(\boldsymbol{A}_i)$. Next, we sample from the row corresponding to $w^{(1)}$, and repeat until we reach z_k, i.e.

$$w^{(t)} \sim \text{Cat}(\boldsymbol{B}_{w^{(t-1)}}) \quad \text{for } 1 < t \leq T,$$

where $w^{(T)} = \mathrm{z}_k$. For the remaining entries, we continue in arbitrary order. Observe that $(w^{(1)}, \ldots, w^{(T)})$ are the ancestors of leaf v_i in $\mathcal{T}$. Further observe that this sampling procedure is identical to how the path $\hat{r}_i$ is generated in the random walk $\mathcal{W}(\mathrm{v}_i)$, completing the proof. □

A.5 PROOF OF THEOREM 3

Proof. First, recall that all paths end in the root node, such that r_i necessarily ends in $\mathrm{z}_{n'} = r_i^{(T)}$. When reasoning about the lowest *common* ancestors, it is no longer sufficient to consider the ancestors (or, equivalently, path to the root) of a single leaf node in isolation. Instead, we need to consider *pairs* of dependent paths $(r_i, r_j), i \neq j$ rooted in v_i and v_j, respectively. Note that r_i and r_j necessarily converge at some internal node $z_k = \mathrm{v}_i \wedge \mathrm{v}_j$ — the latest at the root node $z_{n'}$.

Thus, we denote with $\underline{r_i} = (r_i^{(1)}, \ldots, \mathrm{v}_i \wedge \mathrm{v}_j)$ the first part of the path r_i until (and including) its lowest common ancestor with r_j, i.e., $\mathrm{v}_i \wedge \mathrm{v}_j$. Analogously, $\overline{r}_i = (r_i^{(|\underline{r_i}|+1)}, \ldots, \mathrm{z}_{n'})$, such that $r_i = (\underline{r_i}, \overline{r}_i)$. Further, note that the paths r_i and r_j are on the same underlying hierarchy $\mathcal{T}$, thus $\overline{r}_i = \overline{r}_j$, as both paths have the same trajectory to the root once they have reached their lowest common ancestor, i.e., they are dependent.

The probability of observing the pair of paths (r_i, r_j) under the tree-sampling perspective is

$$p^{(\mathcal{T})}((r_i, r_j)) = p(r_i^{(1)}|\mathrm{v}_i) \cdot \prod_{t=2}^{|\underline{r_i}|} p(r_i^{(t)}|r_i^{(t-1)}) \cdot p(r_j^{(1)}|\mathrm{v}_j) \cdot \prod_{t=2}^{|\underline{r_j}|} p(r_j^{(t)}|r_j^{(t-1)}) \cdot \prod_{t=|\underline{r_i}|+1}^{|r_i|} p(r_i^{(t)}|r_i^{(t-1)}) \tag{11}$$

More compactly,

$$p^{(\mathcal{T})}((r_i, r_j)) = p(\underline{r_i}) \cdot p(\underline{r_j}) \cdot p(\overline{r_i}) = p((\underline{r_i}, \underline{r_j})) \cdot p(\overline{r_i}) = p((\underline{r_i}, \underline{r_j})) \cdot p(\overline{r_j}) \tag{12}$$

Importantly, we can see from Eq. (12) that, in general, $p^{(\mathcal{T})}((r_i, r_j)) \neq p(r_i) \cdot p(r_j)$ i.e., the paths r_i and r_j are *not* independent. We denote $p_{\text{anc}}^{(\mathcal{T})}(\mathrm{z}_k|\mathrm{v}_i, \mathrm{v}_j)$ the probability of the internal node z_k to be the ancestor of leaf nodes v_i and v_j under the tree-sampling perspective. Hence, the probability of the internal node z_k to be the ancestor of leaf nodes v_i and v_j under dependent and independent random walks are different i.e. $p_{\text{anc}}^{(\mathcal{T})}(\mathrm{z}_k|\mathrm{v}_i, \mathrm{v}_j) \neq p_{\text{anc}}(\mathrm{z}_k|\mathrm{v}_i) \cdot p_{\text{anc}}(\mathrm{z}_k|\mathrm{v}_j)$. This makes intuitive sense because knowing that $\mathrm{z}_{k'}$ is an ancestor of v_i and v_j in a tree $\mathcal{T}$, additional knowledge that $\mathrm{z}_k, k > k'$ is an ancestor of v_i implies that z_k is also an ancestor of v_j.

However, for the parts of r_i and r_j *before* they converge, Eq. (12) shows that $p((\underline{r_i}, \underline{r_j})) = p(\underline{r_i}) \cdot p(\underline{r_j})$. This is because all transitions in $\underline{r_i}$ and $\underline{r_j}$ are disjoint thus independent. This is an important insight because it means that the probability of observing two paths both converging at an internal node z_k factorizes. □

A.6 PROOF OF THEOREM 4

Proof. We start by reorganizing Eq. (5):

$$\overbrace{p_{\text{anc}}(\mathrm{z}_k|\mathrm{v}_i)p_{\text{anc}}(\mathrm{z}_k|\mathrm{v}_j)}^{(i)} = \overbrace{p^{(\mathcal{M})}(\mathrm{z}_k = \mathrm{v}_i \wedge \mathrm{v}_j)}^{(ii)} + \overbrace{\sum_{k'=1}^{k-1} p^{(\mathcal{M})}(\mathrm{z}_{k'} = \mathrm{v}_i \wedge \mathrm{v}_j) p_{\text{anc}}(\mathrm{z}_k|\mathrm{z}_{k'})^2}^{(iii)} \tag{13}$$

In words, we can split the event "z_k is an ancestor of v_i and v_j" (i) into two mutually exclusive events: (ii) z_k is the lowest common ancestor of v_i and v_j; or (iii) some internal node lower in the topological

order is the LCA of v_i and v_j, and further, both random walks also traverse through z_k. (ii) and (iii) are mutually exclusive since exactly one internal node is the LCA for v_i and v_j on any two random walks.

Since the two random walks are independent, the probability of z_k being traversed on both walks factorizes. Thus, $p^{(\mathcal{M})}_{\text{anc}}(\mathrm{z}_k|\mathrm{v}_i, \mathrm{v}_j) = p_{\text{anc}}(\mathrm{z}_k|\mathrm{v}_i)p_{\text{anc}}(\mathrm{z}_k|\mathrm{v}_j)$ and therefore (i) is the probability of z_k being an ancestor of v_i and v_j in our Markov chain $\mathcal{M}$.

The events in (iii) can indeed be expressed:

$$p^{(\mathcal{M})}\left(\mathrm{z}_{k'} = \mathrm{v}_i \wedge \mathrm{v}_j, \mathrm{z}_k \in \text{anc}\left(\mathrm{v}_i, \mathrm{v}_j\right)\right) = p^{(\mathcal{M})}\left(\mathrm{z}_{k'} = \mathrm{v}_i \wedge \mathrm{v}_j\right) \cdot p^{(\mathcal{M})}\left(\mathrm{z}_k|\mathrm{z}_{k'} \in \text{anc}\left(\mathrm{v}_i, \mathrm{v}_j\right)\right) \tag{14}$$

where in the last step we exploit that $\mathrm{z}_{k'} = \mathrm{v}_i \wedge \mathrm{v}_j$ implies $\mathrm{z}_{k'} \in \text{anc}\left(\mathrm{v}_i, \mathrm{v}_j\right)$ as well as the Markov property of the random walks. Further, note that

$$\begin{aligned} p^{(\mathcal{M})}_{\text{anc}}(\mathrm{z}_k|\mathrm{z}_{k'} \in \text{anc}(\mathrm{v}_i, \mathrm{v}_j)) &= p_{\text{anc}}(\mathrm{z}_k|\mathrm{z}_{k'} \in \text{anc}(\mathrm{v}_i)) \cdot p_{\text{anc}}(\mathrm{z}_k|\mathrm{z}_{k'} \in \text{anc}(\mathrm{v}_j)) \\ &= p_{\text{anc}}(\mathrm{z}_k|\mathrm{z}_{k'})^2, \end{aligned} \tag{15}$$

where we first exploit factorization due to independence and in the last step again the Markov property of the random walks. □

A.7 Proof of Theorem 5

Proof. Let $\hat{r}_i \in \mathcal{P}(\mathrm{v}_i)$, $\hat{r}_j \in \mathcal{P}(\mathrm{v}_j)$ be two independent random walks on $\mathcal{M}$ rooted in v_i and v_j, respectively. Then,

$$p^{(\mathcal{M})}(\mathrm{z}_k = \mathrm{v}_i \wedge \mathrm{v}_j) = \sum_{(\hat{r}_i, \hat{r}_j):\mathrm{z}_k = \mathrm{v}_i \wedge \mathrm{v}_j} p((\underline{\hat{r}_i}, \underline{\hat{r}_j})), \tag{16}$$

where $\underline{\hat{r}_i} = (\hat{r}_i^{(1)}, \ldots, \mathrm{z}_k)$ is the first part of $\hat{r}_i$ until it reaches z_k. Note that the second part of the paths, $\overline{\hat{r}_i}$ and $\overline{\hat{r}_j}$, which are theoretically independent under our Markov chain model, are marginalized out in the LCA formula.

However, due to the independence of the first part of the paths $\underline{\hat{r}_i}$ and $\underline{\hat{r}_j}$ under both models (see Eq. (4)), we can write:

$$\begin{aligned} p^{(\mathcal{M})}(\mathrm{z}_k = \mathrm{v}_i \wedge \mathrm{v}_j) &= \sum_{(\hat{r}_i, \hat{r}_j):\mathrm{z}_k = \mathrm{v}_i \wedge \mathrm{v}_j} p((\underline{\hat{r}_i}, \underline{\hat{r}_j})) \\ &= \sum_{(\hat{r}_i, \hat{r}_j):\mathrm{z}_k = \mathrm{v}_i \wedge \mathrm{v}_j} p(\underline{\hat{r}_i}) \cdot p(\underline{\hat{r}_j}) \\ &= p^{(\mathcal{T})}(\mathrm{z}_k = \mathrm{v}_i \wedge \mathrm{v}_j). \quad \square \end{aligned} \tag{17}$$

A.8 Number of pairs of LCA paths.

Theorem 7. *Let $\mathcal{M}$ be a Markov chain as defined in Definition 1. The number of pairs of paths from two leaves v_i, v_j for which an internal node z_k is the lowest common ancestor is 3^{k-1}.*

Proof. Proof by induction over k. For the base case $k = 1$ we have one pair of paths for which z_k is the LCA, i.e. directly from v_i to z_k and v_j to z_k. Assume that for internal node z_k there are 3^{k-1} unique pairs of paths for which z_k is the LCA. For each of these paths we can generate three unique paths for which z_{k+1} is the LCA. (1) rewire the last transition of v_i's path to go to z_{k+1} instead of z_k. (2) do the same but for v_j. (3) rewire both v_i's and v_j's last transition to go to z_{k+1} instead of z_k. Thus, the number of pairs of paths from v_i and v_j to z_{k+1} is $3 \cdot 3^{k-1} = 3^k$. □

A.9 Proof of Theorem 6

We provide here the proof for the fast vectorized computation of $\boldsymbol{P}^{\text{LCA}}_{\mathrm{v}_i, \mathrm{v}_j}$ in Theorem. 6.

Proof. We start by reorganizing Eq. (9):

$$\boldsymbol{P}^{\text{anc}}_{\text{v}_i} \odot \boldsymbol{P}^{\text{anc}}_{\text{v}_j} = \boldsymbol{P}^{\text{LCA}}_{\text{v}_i,\text{v}_j} + \boldsymbol{P}^{\text{LCA},T}_{\text{v}_i,\text{v}_j} \cdot \tilde{\boldsymbol{P}}^{\text{anc}} \odot \tilde{\boldsymbol{P}}^{\text{anc}} \tag{18}$$

Note that the inverse $(\boldsymbol{I} + \tilde{\boldsymbol{P}}^{\text{anc}} \odot \tilde{\boldsymbol{P}}^{\text{anc}})^{-1}$ is guaranteed to exist and is efficient to compute because $\tilde{\boldsymbol{P}}^{\text{anc}} \odot \tilde{\boldsymbol{P}}^{\text{anc}}$ is a strictly upper triangular and therefore nilpotent matrix. The k-th entry is thus:

$$\left[\boldsymbol{P}^{\text{anc}}_{\text{v}_i} \odot \boldsymbol{P}^{\text{anc}}_{\text{v}_j}\right]_k = \boldsymbol{P}^{\text{LCA}}_{\text{v}_i,\text{v}_j,\text{v}_k} + \sum_{k'=1}^{n'} \boldsymbol{P}^{\text{LCA}}_{\text{v}_i,\text{v}_j,\text{v}_{k'}} \cdot \left[\tilde{\boldsymbol{P}}^{\text{anc}} \odot \tilde{\boldsymbol{P}}^{\text{anc}}\right]_{k',k}.$$

Plugging in the definitions of Eq. (8), using Theorem 2, and observing that due to the upper triangular structure $\tilde{\boldsymbol{P}}^{\text{anc}}_{k',k} = 0$ for $k' > k$ we obtain

$$p_{\text{anc}}(\text{z}_k|\text{v}_i) \cdot p_{\text{anc}}(\text{z}_k|\text{v}_j) = p(\text{z}_k = \text{v}_i \wedge \text{v}_j) + \sum_{k'=1}^{k-1} p(\text{z}_{k'} = \text{v}_i \wedge \text{v}_j) \cdot p_{\text{anc}}(\text{z}_k|\text{z}_{k'})^2. \qquad \square$$

A.10 Vectorized computations

All quantities involved in Soft-Das and Soft-TSD can be computed in closed-form based on the Markov chain $\mathcal{M}$ and its fundamental matrix. However, their computation should not be done naively, as this involves unnecessary computations. Constructing the full tensor of LCA probabilities is expensive since $\boldsymbol{P}^{\text{LCA}} \in \mathbb{R}^{n \times n \times n'}$. Note, however, that to compute the Soft Dasgupta loss or the distribution $p(\text{z})$ in TSD we only require the LCA probabilities $p(\text{z} = \text{v}_i \wedge \text{v}_j)$ for pairs of leaves connected by an edge (i.e., $P(\text{v}_i, \text{v}_j) > 0$). That is, we only need to construct an LCA probability matrix of shape $\mathbb{R}^{m \times n'}$. Thus, we can exploit the sparsity of real world graphs, as typically $m \ll n^2$.

In a similar way, the computation of the distribution $q(\text{z})$ does also not require the expensive explicit computation of $p(\text{z}_k = \text{v}_i \wedge \text{v}_j)$ for all pairs of leaf nodes. Instead, we can again exploit insights from the Markov chain $\mathcal{M}$. First, observe that the equation of $q(z)$ in Soft-TSD describes an expectation:

$$q(\text{z}) = \mathbb{E}_{\text{v}_i,\text{v}_j \sim P(\text{v})} \left[p(\text{z} = \text{v}_i \wedge \text{v}_j)\right]. \tag{19}$$

Defining $\hat{\boldsymbol{p}}^{\text{anc}} = \boldsymbol{p}^T \cdot \boldsymbol{P}^{\text{anc}}$, the computation of this expectation can be vectorized similarly to Theorem 6 (see derivation in App. A.11): $\boldsymbol{q} = (\hat{\boldsymbol{p}}^{\text{anc}} \odot \hat{\boldsymbol{p}}^{\text{anc}})^T \cdot (\boldsymbol{I} + \tilde{\boldsymbol{P}}^{\text{anc}} \odot \tilde{\boldsymbol{P}}^{\text{anc}})^{-1}$.

A.11 Vectorized $\boldsymbol{q}$ computation.

We provide here the proof for the fast vectorized computation of $\boldsymbol{q}$ in Eq. 19.

Proof. We first rewrite the expectation Eq. 19 in vectorized form:

$$\boldsymbol{q} = \sum_{i,j} \boldsymbol{p}_{\text{v}_i} \boldsymbol{P}^{\text{LCA}}_{\text{v}_i,\text{v}_j} \boldsymbol{p}_{\text{v}_j}$$

where we denote $P(\text{v}_i) = \boldsymbol{p}_{\text{v}_i}$. Subsequently, we can plug the $\boldsymbol{P}^{\text{LCA}}$ formula Eq.. (9) and pull $\boldsymbol{p}_{\text{v}_i}$ into the Hadamard product which is done over the internal node dimension:

$$\begin{aligned}
\boldsymbol{q} &= \sum_{i,j} \boldsymbol{p}_{\text{v}_i} \boldsymbol{p}_{\text{v}_j} (\boldsymbol{P}^{\text{anc}}_{\text{v}_i} \odot \boldsymbol{P}^{\text{anc}}_{\text{v}_j})^T \cdot (\boldsymbol{I} + \tilde{\boldsymbol{P}}^{\text{anc}} \odot \tilde{\boldsymbol{P}}^{\text{anc}})^{-1} \\
&= \sum_i \boldsymbol{p}_{\text{v}_i} \left(\boldsymbol{P}^{\text{anc}}_{\text{v}_i} \odot \sum_j \boldsymbol{p}_{\text{v}_j} \boldsymbol{P}^{\text{anc}}_{\text{v}_j} \right)^T \cdot (\boldsymbol{I} + \tilde{\boldsymbol{P}}^{\text{anc}} \odot \tilde{\boldsymbol{P}}^{\text{anc}})^{-1} \\
&= \left(\sum_i \boldsymbol{p}_{\text{v}_i} \boldsymbol{P}^{\text{anc}}_{\text{v}_i} \odot \sum_j \boldsymbol{p}_{\text{v}_j} \boldsymbol{P}^{\text{anc}}_{\text{v}_j} \right)^T \cdot (\boldsymbol{I} + \tilde{\boldsymbol{P}}^{\text{anc}} \odot \tilde{\boldsymbol{P}}^{\text{anc}})^{-1} \\
&= (\hat{\boldsymbol{p}}^{\text{anc}} \odot \hat{\boldsymbol{p}}^{\text{anc}})^T \cdot (\boldsymbol{I} + \tilde{\boldsymbol{P}}^{\text{anc}} \odot \tilde{\boldsymbol{P}}^{\text{anc}})^{-1}.
\end{aligned}$$

where $\hat{\boldsymbol{p}}^{\text{anc}} = \boldsymbol{p}^T \cdot \boldsymbol{P}^{\text{anc}}$. $\square$

A.12 COMPLEXITY ANALYSIS.

Both Dasgupta and TSD computations require to compute the ancestor probabilities (Eq. 8) which can be done in $O(n \times n'^2)$ (i.e. inverse of triangular matrix $(\boldsymbol{I} - \boldsymbol{B}) \in \mathbb{R}^{n' \times n'}$, plus matrix multiplication with $\boldsymbol{A} \in \mathbb{R}^{n \times n'}$). Then, Soft-Das. or the distribution $p(\mathrm{z})$ for the Soft-TSD loss require the LCA probabilities (Eq. 9) for all leaves connected by an edge only, amounting to $O(m \times n'^2)$ operations, where m is the number of edges in the graph. Note that similarly to Eq. (8), the inverse computation in Eq. (9) can be done in $O(n'^2)$. Additionally, Soft-TSD requires the computation of $q(\mathrm{z})$ (complexity $O(n \times n'^2)$). Both Soft-Das. and Soft-TSD computations are dominated by the $O(m \times n'^2)$ term. This leads to an efficient time complexity as long as we assume a small number of internal nodes $n' \ll n$, which is reasonable in practice.

A.13 CONVEXITY OF SOFT-TSD

Theorem 8. *Let $\boldsymbol{H} \in [0,1]^{n' \times n \times n}$ be a tensor whose elements $\boldsymbol{H}_{kij} = p(\mathrm{z}_k = \mathrm{v}_i \wedge \mathrm{v}_j)$ are the LCA probabilities of internal nodes w.r.t. pairs of leaf nodes.* Soft-TSD$(\boldsymbol{H})$ *is convex in $\boldsymbol{H}$.*

Proof. Let $\boldsymbol{H}^{(1)}$, $\boldsymbol{H}^{(2)}$ be two LCA probability tensors defined as above, and $0 \leq \alpha \leq 1$. We first compute the distribution p induced by the edge distribution. We have that

$$\begin{aligned} p(\alpha \boldsymbol{H}_k^{(1)} + (1-\alpha)\boldsymbol{H}_k^{(2)}) &= \sum_{\mathrm{v}_i, \mathrm{v}_j} P(\mathrm{v}_i, \mathrm{v}_j)(\alpha \boldsymbol{H}_{kij}^{(1)} + (1-\alpha)\boldsymbol{H}_{kij}^{(2)}) \\ &= \alpha \sum_{\mathrm{v}_i, \mathrm{v}_j} P(\mathrm{v}_i, \mathrm{v}_j)\boldsymbol{H}_{kij}^{(1)} + (1-\alpha) \sum_{\mathrm{v}_i, \mathrm{v}_j} P(\mathrm{v}_i, \mathrm{v}_j)\boldsymbol{H}_{kij}^{(2)} \\ &= \alpha \cdot p(\boldsymbol{H}_k^{(1)}) + (1-\alpha) \cdot p(\boldsymbol{H}_k^{(2)}). \end{aligned} \tag{20}$$

Analogously we compute the distribution q induced by the independent ndoe distribution. We have that

$$q(\alpha \boldsymbol{H}_k^{(1)} + (1-\alpha)\boldsymbol{H}_k^{(2)}) = \alpha \cdot q(\boldsymbol{H}_k^{(1)}) + (1-\alpha) \cdot q(\boldsymbol{H}_k^{(2)}). \tag{21}$$

We combine this with the well-known fact that KL-divergence is convex w.r.t. pairs of distributions, i.e.,

$$\mathrm{KL}(\alpha p_1(z) + (1-\alpha)p_2(z), \alpha q_1(z) + (1-\alpha)q_2(z)) \leq \alpha \mathrm{KL}(p_1(z), q_1(z)) + (1-\alpha)\mathrm{KL}(p_2(y), q_2(z)),$$

to obtain the desired result:

$$\text{Soft-TSD}(\alpha \boldsymbol{H}^{(1)} + (1-\alpha)\boldsymbol{H}^{(2)}) \leq \alpha \cdot \text{Soft-TSD}(\boldsymbol{H}^{(1)}) + (1-\alpha) \cdot \text{Soft-TSD}(\boldsymbol{H}^{(2)}). \tag{22}$$

□

Theorem 9. *Let $\mathcal{H}_{\text{FPH}} = \{\boldsymbol{H} : \exists \boldsymbol{A}, \boldsymbol{B} \in \Phi(n, n') : \boldsymbol{H} = \text{FPH}(\boldsymbol{A}, \boldsymbol{B})\}$ denote the set of probabilistic hierarchies which can be represented by FPH. Here, $\Phi(n, n')$ are the constraints FPH places on $\boldsymbol{A}, \boldsymbol{B}$ (see Sec. 2), and FPH $(\boldsymbol{A}, \boldsymbol{B})$ is shorthand the mapping from transition matrices to lowest common ancestor probability tensors defined in Theorems 4 and 6. Here, $\boldsymbol{H}_{kij} = p(\mathrm{z}_k = \mathrm{v}_i \wedge \mathrm{v}_j)$ are the LCA probabilities of internal nodes w.r.t. pairs of leaf nodes. This set $\mathcal{H}_{\text{FPH}}$ is convex.*

Proof. We start by recalling form Theorems 4 and 5 that FPH computes lowest common ancestor probabilities $p(\mathrm{z}_k = \mathrm{v}_i \wedge \mathrm{v}_j)$ from the continuous parent probability matrices $\boldsymbol{A}$ and $\boldsymbol{B}$ such that the LCA probabilities are consistent with the expected result from the tree-sampling procedure described in Sec. A.2. More formally,

$$\begin{aligned} \boldsymbol{H}_{kij} := p(\mathrm{z}_k = \mathrm{v}_i \wedge \mathrm{v}_j) &= \mathbb{E}_{\hat{\boldsymbol{H}} \sim (\boldsymbol{A}, \boldsymbol{B})} \left[\mathbb{I}[\mathrm{z}_k = \mathrm{v}_i \wedge \mathrm{v}_j] \right] \\ &= \mathbb{E}_{\hat{\boldsymbol{H}} \sim (\boldsymbol{A}, \boldsymbol{B})} \left[\hat{\boldsymbol{H}}_{kij} \right] = \mathbb{E} \left[\hat{\boldsymbol{H}} \right]_{kij}, \end{aligned} \tag{23}$$

where $\hat{\boldsymbol{H}} \in \{0,1\}^{n' \times n \times n}$ is a discrete hierarchy obtained via tree-sampling from $\boldsymbol{A}$ and $\boldsymbol{B}$. By definition of the expectation we write

$$\boldsymbol{H} = \mathbb{E}_{\hat{\boldsymbol{H}} \sim (\boldsymbol{A}, \boldsymbol{B})} \left[\hat{\boldsymbol{H}} \right] = \sum_{\hat{\boldsymbol{H}} \in \mathcal{H}(n, n')} p(\hat{\boldsymbol{H}} | \boldsymbol{A}, \boldsymbol{B}) \cdot \hat{\boldsymbol{H}}, \tag{24}$$

where $\mathcal{H}(n, n')$ is the set of all valid discrete hierarchies with n leafs and n' internal nodes. Thus, any continuous hierarchy $\boldsymbol{H}$ learned by FPH is a convex combination of discrete hierarchies $\hat{\boldsymbol{H}}$. This completes the proof. □

Theorem 10. *The Soft-TSD optimization problem solved by FPH is integral. That is, the global maximum of the Soft-TSD optimization problem solved by FPH is the same as the global optimum of the* discrete *optimization problem of optimizing TSD over discrete hierarchies.*

Proof. This follows from Theorems 8 and 9. Theorem 8 establishes that the Soft-TSD objective function is convex in the hierarchy tensors $\boldsymbol{H}$; Theorem 9 proves that the set of hierarchies FPH optimizes over is convex. When maximizing a convex function over a convex set, we are guaranteed to find the global optimum at a vertex of the constraint set, which are discrete hierarchies in the case of FPH. Thus, the global maximizer is a discrete hierarchy; this discrete hierarchy must also be the maximizer of the discrete TSD optimization problem, since our relaxation optimizes over a superset of all discrete hierarchies. □

Implications of Theorems 8, 9, and 10. In the previous theorems, we have shown that we are *maximizing* a convex function over a convex set. In general, maximizing a convex function over a convex set is NP-hard (Benson, 1995). Thus, we cannot hope to efficiently recover the global optimum. However, our continuous relaxation brings several practical benefits for the optimization.

First, observe that directly optimizing over the convex set of continuous hierarchies described in Theorem 9 is not practical. This is because there are exponentially many corners of the set, and encoding the constraints of the set is very difficult. Our parameterization of (continuous) hierarchies via $\boldsymbol{A}$, $\boldsymbol{B}$ and being able to efficiently compute the *expected* lowest common ancestor probabilty tensor enables us to optimize over a fairly low-dimensional and convex set. The constraints on $\boldsymbol{A}, \boldsymbol{B}$, i.e., entries in $[0, 1]$, unit row sums and upper-triangular structure of $\boldsymbol{B}$, are easy to encode and enforce during optimization. This comes at the cost that mapping from $\boldsymbol{A}$ and $\boldsymbol{B}$ to the LCA tensor $\boldsymbol{H}$ is nonconvex (yet describes, as per Theorem 9, a convex set over hierarchies). Thus, we can solve the optimization problem with off-the-shelf methods such as projected gradient descent and benefit from the elaborate techniques from nonconvex optimization. While it is possible that FPH gets stuck in a non-discrete local optima during optimization, we can easily obtain a discrete and valid hierarchy given the non-discrete local optimizer via tree-sampling or selecting the most likely parent for all leaves and internal nodes under $\boldsymbol{A}$ and $\boldsymbol{B}$, as described in Sec. 2.2.

B Experiment Information

B.1 Link Prediction with Soft-TSD

The TSD can be interpreted in terms of retrieved information when reconstructing the original graph from the tree representation (Charpentier & Bonald, 2019). In this case, the reconstruction scheme for the edge weights of the reconstructed graph $\hat{\mathcal{G}}$ is:

$$\hat{w}(\mathrm{v}_i, \mathrm{v}_j) = w(\mathrm{v}_i) w(\mathrm{v}_j) \frac{p(\mathrm{v}_i \wedge \mathrm{v}_j)}{q(\mathrm{v}_i \wedge \mathrm{v}_j)} \tag{25}$$

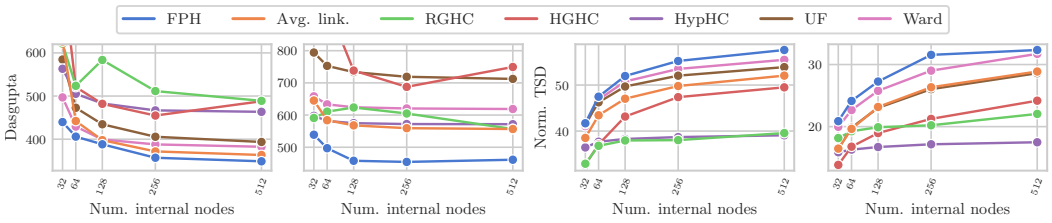
B.2 Link prediction setup

For all datasets, we randomly select 10% of edges to hold out for testing while making sure that the graph remains connected. Further, we set $n' = 256$ and minimize Soft-TSD via FPH. For DC-SBM, we use the Python package 'graph-tool' and follow the documentation[3] with default parameters to learn the model. For VGAE, we use the default hyperparameters by the authors (one hidden layer, latent dimensions $[32, 16]$, learning rate 0.01, training for 200 epochs). We use the variant described in the paper which replaces the node attributes by the $n \times n$ identity matrix. For DeepWalk, we set the embedding dimension to 10.

[3] https://graph-tool.skewed.de/static/doc/demos/inference/inference.html

Dataset	Nodes (LCC)	Edges (LCC)	MI (LCC)	License
PolBlogs, (Adamic & Glance, 2005)	1,222	16,715	2.39	n/a
Brain, (Amunts et al., 2013)	1,770	8,957	3.37	n/a
Citeseer, (Sen et al., 2008)	2,110	3,694	5.69	n/a
Genes, (Cho et al., 2014)	2,194	2,688	6.12	n/a
Cora-ML, (McCallum et al., 2000; Bojchevski & Günnemann, 2018)	2,810	7,981	5.23	n/a
WikiPhysics, (Aspert et al., 2019)	3,309	31,251	3.44	n/a
OpenFlight, (Patokallio,)	3,097	18,193	3.44	ODbL
Ogbn-products, (Hu et al., 2020)	2,385,902	61,806,367	9.29	Amazon license
Ogbn-arxiv, (Hu et al., 2020; Wang et al., 2020)	169,343	1,157,799	7,40	ODC-BY
Ogbl-collab, (Hu et al., 2020; Wang et al., 2020)	232,865	961,883	9.02	ODC-BY
DBLP, (Yang & Leskovec, 2015)	317,080	1,049,866	9.64	n/a

Table 7: Dataset summary; we convert directed datasets to undirected and select the largest connected component (LCC).



(a) Dasgupta - OpenFlight (b) Dasgupta - Brain (c) TSD - OpenFlight (d) TSD - Brain

Figure 3: Results on hierarchical clustering measured by Dasgupta cost (lower is better) and TSD (higher is better).

B.3 Dataset summary

See Table 7 for an overview of the datasets we used.

B.4 Additional result figures

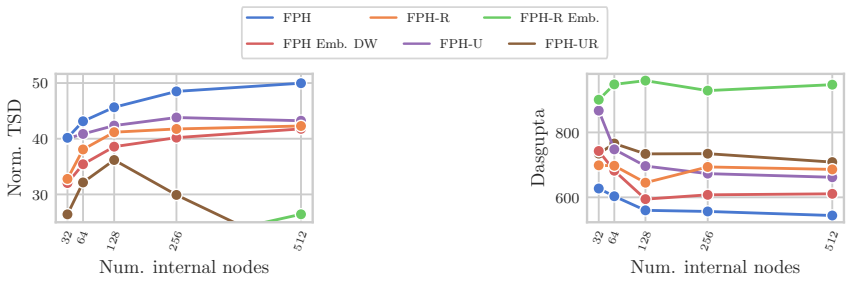
In Fig. 3 we show results for four more datasets.

B.5 Ablation study

See Fig 4 for the comparison of different FPH model variants, and our full discussion in Sec. 4.1.

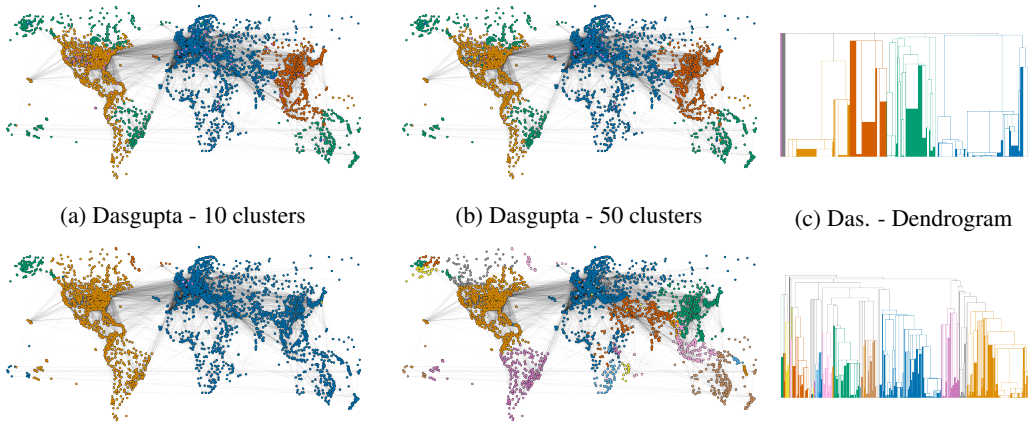
B.6 Hierarchy Visualization

We conduct a qualitative study of the structure discovered by FPH. In Figure 5 we compare the hierarchies learned by FPH when optimizing for TSD or Dasgupta, respectively. We cut at different levels of the dendrogram to obtain a coarse hierarchy (10 clusters) and fine-grained structure (50 clusters). Comparing Figure 5 (a) and (d), we notice that the coarse structure learned by optimizing Soft-Dasgupta looks more appealing, as TSD essentially splits only into the Americas and the rest



(a) TSD - WikiPhysics (b) Dasgupta - WikiPhysics

Figure 4: Ablation study results.



(a) Dasgupta - 10 clusters (b) Dasgupta - 50 clusters (c) Das. - Dendrogram

(d) TSD - 10 clusters (e) TSD - 50 clusters (f) TSD - Dendrogram

Figure 5: Visual comparison of trees obtained after Soft-TSD and Soft-Dasgupta optimization on OpenFlight.

of the world. At 50 clusters, however, we observe the opposite: TSD splits the airports across the world into meaningful, coherent geographical regions, whereas Dasgupta looks mostly unchanged from the coarse version, highlighting the complementarity of both quality metrics. In addition, the dendrogram learned by TSD in (f) appears to be of higher quality and more balanced than the Dasgupta dendrogram in (c).

B.7 Hyperparameters

We use the hyperparameters for models and baselines described in Tab. 8. Note that we train FPH for 1,000 epochs and restore the best hierarchy after training. For `ogbn-products` we train for 2,000 epochs. Further, to ensure convergence we reduce the learning rate to 0.02 and introduce weight decay of 1e-5 for FPH (Das.) for $n' \geq 128$ on Polblogs, Cora-ML, and Brain, and train for 2,000 epochs. Similarly, we use different learning rates for $\boldsymbol{A}$ and $\boldsymbol{B}$ for FPH (Das.) on ogbn-arxiv, ogbl-collab, and DBLP ($\text{lr}_{\boldsymbol{A}} = 1e-2$, $\text{lr}_{\boldsymbol{B}} = 1e-5$).

B.8 Computing infrastructure

We train all models on a single GPU (NVIDIA GTX 1080 Ti or NVIDIA GTX 2080 Ti, 11 GB memory) in our own in-house compute cluster. The machines have 10-core Intel CPUs. We use Python 3 and PyTorch for all our experiments.

B.9 HSBM graphs

We generated HSBMs with $n = 100$ leaf nodes and $n = 1000$ leaf nodes for our external evaluation. The small HSBMs have 3 levels with edge probabilities in $[.01, .1, .3, .6]$, a branching factor of 2 and core community sizes in $[10, 15]$. The large HSBMs have 3 levels with edge probabilities in $[.001, .01, .1, .4]$, branching factor in $[2, 3, 4]$ and core community sizes in $[30, 35]$. In Fig. 6 and Fig. 7, we plot one of the five synthetic HSBM graphs used in our experiments (right), and their corresponding dendrograms (left). The graphs have $n = 100$ and $n = 1000$ leaf nodes and three levels of hierarchy.

C Additional results

Model	Hyperparameter	Value
FPH (TSD)	Learning rate	150
	Batch size K*	10,000
	Batch cutoff C*	200,000
FPH-R (TSD)	Learning rate	200
FPH (Das.)	Learning rate	0.05
	Batch size K*	10,000
	Batch cutoff C*	200,000
FPH Emb.	Learning rate	0.1
RGHC	Routing NN dim	128
	Iterations	5000
	Learning rate	0.0001
HGHC	Init. method	K-means + agglom. linkage
	Iterations	10
	Learning rate	0.1
UF	Loss	Closest + cluster size
	Epochs	500
	Learning rate	0.1
HypHC	Num. triples	50M
	Epochs	50
	Learning rate	0.001
	Temperature	0.1
DeepWalk	Embedding dim	10
	Embedding dim*	32

* Used for ogbn-products, ogbn-arxiv, ogbl-collab, DBLP.

Table 8: Hyperparameter settings.

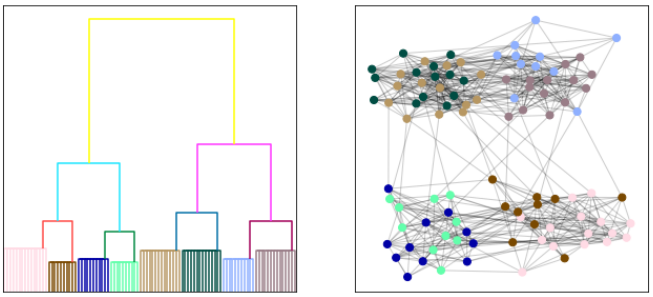


Figure 6: Example HSBM graph with $n = 100$, $n' = 7$, and three levels in the hierarchy.

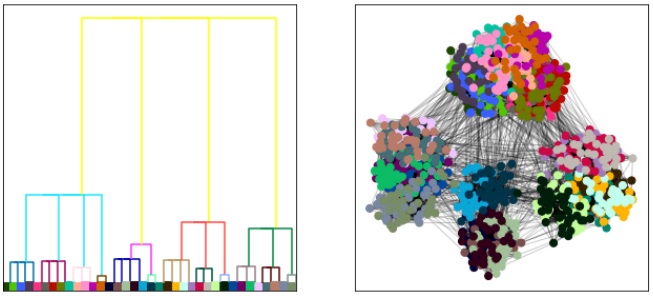


Figure 7: Example HSBM graph with $n = 1000$, $n' = 53$, and three levels in the hierarchy.

	Dasgupta	Norm. TSD
CoraML	336.86	57.51
Citeseer	178.23	68.45
Polblogs	443.48	25.93
Brain	777.14	29.28
Genes	247.26	67.47
WikiPhysics	986.32	46.03
Openflight	633.66	51.51
Ogbn-arxiv	31,655	37.75
Ogbl-collab	20,664	46.12
DBLP	40,744	40.92

Table 9: Hierarchical clustering results for Louvain.

	Dasgupta cost (lower is better)					Normalized TSD (higher is better)				
Alg.	Ward	UF	HypHC	HGHC	RGHC	Ward	UF	HypHC	HGHC	RGHC
Brain	596.73	938.49	568.18	894.87	650.53	32.43	26.28	17.68	17.31	16.54
OpenFlight	416.05	643.45	423.80	477.51	469.98	55.59	49.88	40.06	47.52	45.84
Genes	221.76	258.21	467.12	482.11	444.82	66.87	63.73	23.94	50.59	40.98
Citeseer	105.12	280.66	271.80	224.63	200.91	69.28	62.95	31.74	52.53	47.66
Cora-ML	301.47	673.27	441.21	516.87	499.49	57.22	47.96	29.10	42.10	35.19
PolBlogs	383.51	726.34	334.69	428.52	376.94	27.01	10.73	21.60	20.30	20.78
WikiPhysics	808.87	958.20	701.14	919.27	790.39	45.54	41.55	33.85	34.51	36.51
ogbn-arxiv	22,046	64,950	OOM	37,177	26,286	37.43	26.22	OOM	17.55	25.20
ogbl-collab	14,834	101,562	OOM	112,048	17,964	45.20	30.50	OOM	11.11	37.73
DBLP	33,349	160,742	OOM	171,975	41,796	38.87	22.62	OOM	5.61	29.9

Table 10: Hierarchical clustering results ($n' = 512$, $d = 128$).

Model	Citeseer	Cora	Polblogs	DBLP	ogbn-arxiv
RGHC	0.218	0.394	0.756	0.510	0.358
HGHC	0.304	0.362	0.604	0.655	0.385
Ward	0.363	0.445	0.436	0.587	0.402
UF	0.180	0.242	0.102	0.395	0.143
HypHC	0.285	0.390	0.740	-	-

Table 11: NMI results for $d = 128$ DeepWalk embeddings.

Model	Citeseer	Cora	Polblogs	ogbn-arxiv	DBLP
Avg.	0.367	0.420	0.507	0.216	0.526
RGHC	0.281	0.400	**0.730**	0.286	0.510
HGHC	0.365	0.379	0.177	0.290	0.408
Ward	0.368	0.504	0.702	**0.411**	0.591
UF	0.347	0.428	0.676	0.254	<u>0.598</u>
HypHC	0.270	0.121	0.691	OOM	OOM
Louvain	0.329	0.500	0.640	0.395	0.558
FPH	**0.398**	0.462	0.680	0.251	0.560
FPH (Louv.)	0.380	<u>0.507</u>	0.614	0.399	0.564
FPH (Ward)	<u>0.393</u>	**0.516**	<u>0.708</u>	<u>0.401</u>	**0.604**

Table 12: NMI results on real-world datasets. FPH (Louv.) and FPH (Ward) refer to FPH initialized from the solutions of Louvain and Ward, respectively.

Alg.	TSD standard deviation			Dasgupta standard deviation		
	HypHC	HGHC	RGHC	HypHC	HGHC	RGHC
Brain	0.53 (3%)	0.06 (<0.5%)	0.6 (3%)	19.96 (3%)	38.56 (5%)	14.18 (3%)
OpenFlight	1.15 (3%)	0.4 (1%)	0.7 (2%)	26.61 (6%)	31.14 (6%)	23.9 (5%)
Genes	0.61 (3%)	0.12 (<0.5%)	1.19 (2%)	13.3 (3%)	2.88 (1%)	17.31 (7%)
Citeseer	0.19 (1%)	0.09 (<0.5%)	1.29 (3%)	6.93 (3%)	5.56 (4%)	8.02 (6%)
Cora-ML	0.96 (3%)	0.13 (<0.5%)	0.61 (1%)	17.39 (4%)	21.78 (5%)	12.98 (4%)
PolBlogs	0.13 (1%)	0.09 (<0.5%)	0.18 (1%)	4.04 (1%)	1.92 (1%)	3.4 (1%)
WikiPhysics	0.47 (1%)	0.14 (<0.5%)	0.56 (1%)	5.77 (1%)	6.83 (1%)	30.88 (4%)
ogbn-arxiv	-	0.15 (1%)	1.31 (5%)	-	405 (2%)	765 (3%)
ogbl-collab	-	0.16 (1%)	0.61 (2%)	-	1,592 (5%)	625 (3%)
DBLP	-	0.47 (3%)	0.81 (3%)	-	2,482 (3%)	1,005 (2%)

Table 13: Standard deviations of non-deterministic baselines. In parentheses we report the standard deviation in relation to the best value reported in Table 1 in percent.