Exact and Approximate Hierarchical Clustering Using A*: Supplementary Material

Craig S. Greenberg[*,1] **Sebastian Macaluso**[*,2] **Nicholas Monath**[*,3]
Avinava Dubey[4] **Patrick Flaherty**[5] **Manzil Zaheer**[4] **Amr Ahmed**[4] **Kyle Cranmer**[2] **Andrew McCallum**[3]

[1]National Institute of Standards and Technology
[2]Center for Cosmology and Particle Physics & Center for Data Science, New York University
[3]College of Information and Computer Sciences, University of Massachusetts Amherst
[4]Google Research, Mountain View, CA
[5]Department of Mathematics and Statistics, University of Massachusetts Amherst

A APPENDIX

A.1 THEOREM 1 (CORRECTNESS OF A* MAP) PROOF

Proof. Assume toward contradiction that Algorithm 1 does not return the optimal hierarchical clustering, $H^\star$. It is clear (from the conditional in line 9, $|\mathsf{lvs}(H_\Xi(X))| = |X|$, and the fact that if X_L, X_R are children of X then $X_L \bigcup X_R = X$ and $X_L \bigcap X_R = \varnothing$) that Algorithm 1 returns a hierarchical clustering, $H_\Xi(X) \in \mathbb{H}(\mathbb{T})$, so the assumption is that $H_\Xi(X)$ is not optimal, i.e., $\phi(H^\star) < \phi(H_\Xi(X))$. Consider the set, S, of data sets present in both $H^\star$ and $H_\Xi(X)$ that branch to different children, i.e., $S = \{X_i | X_{L^\star}, X_{R^\star} \in H^\star \wedge X_L, X_R \in H_\Xi(X) \wedge X_{L^\star} \bigcup X_{R^\star} = X_L \bigcup X_R = X_i \wedge X_{L^\star} \neq X_L \neq X_R \neq X_{R^\star}\}$. The set S must be non empty, otherwise $H_\Xi(X) = H^\star$, a contradiction. Now consider the set, T, of ancestors of S, i.e., $T = \{X_i | X_i \in S \wedge \nexists X_j \in S \text{ s.t. } X_i \neq X_j \wedge X_i \bigcap X_j = X_i\}$. Since S is non-empty, the set T must also be non empty. Now select a dataset, $X_i \in T$, such that the sub-hierarchy rooted at X_i in $H^\star$, $H^\star(X_i)$, has lower energy than the sub-hierarchy rooted at X_i in $H_\Xi(X)$, $H_\Xi(X_i)$, i.e., $\phi(H^\star(X_i)) < \phi(H_\Xi(X_i))$. There must exist at lease one such X_i, otherwise $\phi(H_\Xi(X)) <= \phi(H^\star)$, a contradiction.

Now consider the trellis vertex corresponding to data set X_i, $\mathbb{V}_i$, with min heap $X_i[\Pi]$, and the tuple at the top of $X_i[\Pi]$, $(f_i, g_i, h_i, X_L, X_R)$. Note that because $h = 0$ in line 9 of Algorithm 1, that $h_i = 0$, therefore the f_i is equal to the energy of the sub-hierarchy of $H_\Xi(X)$ rooted at X_i, $\phi(H_\Xi(X_i))$. Note also that X_i's children in $H^\star$, $X_{L^\star}, X_{R^\star}$, must be present in $X_i[\Pi]$, along with corresponding $f^\star, g^\star, h^\star$ values, otherwise $H^\star$ is not represented in the trellis, a contradiction. Since $X_{L^\star} \neq X_L \neq X_R \neq X_{R^\star}$, the tuple $(f^\star, g^\star, h^\star, X_{L^\star}, X_{R^\star})$ is not at the top of $X_i[\Pi]$. If $h^\star = 0$, then $f^\star = \phi(H^\star(X_i))$ and $f <= f^*$, which implies $\phi(H_\Xi(X_i)) <= \phi(H^\star(X_i))$, a contradiction. If $h^\star \neq 0$, then $f^\star <= \phi(H^\star(X_i))$, otherwise $\mathbb{H}$ is not an admissible heuristic, a contradiction. This gives us $\phi(H_\Xi(X_i)) = f < f^\star <= \phi(H^\star(X_i))$, a contradiction. □

A.2 COROLLARY 1 (OPTIMAL CLUSTERING) PROOF

Proof. Theorem 1 states that Algorithm 1 returns the optimal hierarchical clustering, thus proving that all hierarchical clusterings are represented in a full trellis would prove the corollary.

Assume toward contradiction that there is a hierarchical clustering, $H^\star$, that is not present in a full trellis, $\mathbb{T}$. This implies either that (1) there is a data set $X_i \in H^\star$ that is not in the trellis, i.e., $X_i \notin \mathbb{T}$, or that (2) there exists data sets $X_i, X_j, X_k \in H^\star$ s.t. $X_i = X_j \bigcup X_k \wedge X_j \bigcap X_k = \varnothing$ and $X_i, X_j, X_k \in \mathbb{T}$, but $X_j, X_k \notin \mathbb{C}(X_i)$. Regarding (1), $X_i \in H^\star$ s.t. $X_i \notin \mathbb{T}$ implies that $X \notin \mathbb{P}(X)$, a contradiction. Regarding (2), since each node in a full trellis has all possible children, i.e., $\forall \mathbb{V}_i \in \mathbb{T}, \mathbb{C}(\mathbb{V}_i) = \mathbb{P}(X_i)$, this implies that $X_j, X_k \notin \mathbb{P}(X_i)$, a contradiction.

□

A.3 PROPOSITION 1 (SPACE COMPLEXITY OF A* MAP) PROOF

Proof. Each vertex, $\mathbb{V}_i$, in a trellis, $\mathbb{T}$, stores a dataset, X_i, a min heap, $X_i[\Pi]$, and a pointer to the children associated with the best split, $X[\Xi]$. Since X_i and each pointer in $X[\Xi]$ can be represented as integers, they are stored in $O(1)$ space. The min heap, $X_i[\Pi]$, stores a five tuple, $(f_{LR},\ g_{LR},\ h_{LR}, X_L,\ X_R)$ for each of X_i's child pairs, (X_L, X_R). Each five tuple can be stored in $O(1)$ space, so the min heap is stored in $O(|\mathbb{C}(X_i)|)$ space, where $\mathbb{C}(X_i)$ is X_i's children. Thus each vertex, $\mathbb{V}_i$, takes $O(|\mathbb{C}(X_i)|)$ space to store. In a full trellis,$|\mathbb{C}(X_i)| = 2^{|X_i|-1} - 1$, making the total space complexity $\sum_{\mathbb{V}\in\mathbb{T}} 2^{|X_i|} = O(3^{|X|})$. In a sparse trellis, the largest number of parent/child relationships occurs when the trellis is structured such that the root, $\mathbb{V}_1$, is a parent of all the remaining $|\mathbb{T}| - 1$ vertices in the trellis, the eldest child of the root, $\mathbb{V}_2$, is a parent of all the remaining $|\mathbb{T}| - 2$ vertices in the trellis, and so on, thus the space complexity of a sparse trellis is $\sum_{\mathbb{V}_i\in\mathbb{T}} O(|\mathbb{C}(X_i)|) <= \sum_{i=1\ldots|\mathbb{T}|} (|\mathbb{T}| - i) = O(|\mathbb{T}|^2)$. □

A.4 THEOREM 2 (TIME COMPLEXITY OF A* MAP) PROOF

Proof. During each iteration of the loop beginning at line 4, Algorithm 1 descends from the root down to the leaves of the partial hierarchical clustering with minimum energy, $H_\Xi(X)$, such that the internal nodes have all been explored and the leaf nodes have not. Upon arriving at the leaves of the partial hierarchical clustering (line 11), each leaf node is explored, creating and populating a min heap stored at the node. Finally in line 19, every node in $H_\Xi(X)$ updates its min heap.

Note that when running Algorithm 1, each node in the trellis is explored at most once (a given node, $\mathbb{V}_i$, is explored when $\mathbb{V}_i \in lvs(X[\Xi])$ at line 12). If every node in a trellis is explored, $H_\Xi(X) = \text{argmin}_{H\in\mathbb{H}(X)}$, at which point Algorithm 1 computes $H_\Xi(X)$ at line 6 in $O(\log n)$ time and halts at line 10. Therefore, the time it takes Algorithm 1 to explore every node in trellis $\mathbb{T}$ gives an upper bound on the time it takes Algorithm 1 to find the tree with minimum energy hierarchical clustering encoded by $\mathbb{T}$.

The time it takes Algorithm 1 to explore every $\mathbb{V}_i \in \mathbb{T}$ is the sum of (1) the time $\mathbb{V}_i$ contributes to computation of $H_\Xi(X)$ at line 6 when $\mathbb{V}_i \in lvs(X[\Xi])$, (2) the time it takes to populate the min heap for each $\mathbb{V}_i$ at line 13, and (3) the time $\mathbb{V}_i$ contributes to updating the min heap of every node in $H_\Xi(X)$ at line 19 when $\mathbb{V}_i \in lvs(X[\Xi])$. We consider each of these values in turn:

(1) The time it takes for Algorithm 1 to compute $H_\Xi(X)$ in a given iteration is the sum of the length of the root to leaf paths in $lvs(X[\Xi])$, since reading X[Ξ] takes $O(1)$ time. Therefore the amount of time $\mathbb{V}_i$ contributes to computation of $H_\Xi(X)$ is the length of the path in $H_\Xi(X)$ from the root to $\mathbb{V}_i$.

In a full trellis, the maximum possible path length from the root to $\mathbb{V}_i$ is $|X| - |X_i|$. Thus, the total amount of time spent computing $H_\Xi(X)$ at line 6 in a full trellis when every vertex is explored is $\sum_{k=1\ldots n} \binom{n}{k}(n-k) = \frac{1}{2}\frac{(2^n-2)}{n} = O(2^n)$.

In a sparse trellis, the path lengths are maximized when the root, $\mathbb{V}_1$, is a parent of all the remaining $|\mathbb{T}| - 1$ vertices in the trellis, the eldest child of the root, $\mathbb{V}_2$, is a parent of all the remaining $|\mathbb{T}| - 2$ vertices in the trellis, and so on. Thus the sum of the total path lengths is bounded from above by $\sum_{i=1\ldots|\mathbb{T}|} (|\mathbb{T}| - i) = O(|\mathbb{T}|^2)$, and the total amount of time spent computing $H_\Xi(X)$ at line 6 in a sparse trellis when every vertex is explored is $O(|\mathbb{T}|^2)$.

(2) The time it takes for Algorithm 1 to populate the min heap for $\mathbb{V}_i$ is the amount of time it takes to compute f_{LR}, g_{LR}, h_{LR} (lines 14-16) and to enqueue $(f_{LR}, g_{LR}, h_{LR}, X_L, X_R)$ onto the min heap (line 17) for all children, X_L, X_R.

It is possible to compute g_{LR} in $O(1)$ time, since the first term in equation 4 is a value look up in the model, and second and third terms are the g_{LR} values at X_L and X_R, respectively, and are memoized at those nodes. It is possible to compute f_{LR} in $O(1)$ time, since it is just the sum of two numbers, g_{LR} and h_{LR}. The time it takes to compute h_{LR} depends on the objective-specific heuristic being used. Note that h_{LR} is the sum of a function of a single X_i in Equation 5, i.e.,

$$h_{LR} = \sum_{X_\ell\in \text{lvs}(H_\Xi(X_L\cup X_R))} \mathbb{H}(X_\ell) = \sum_{X_\ell\in \text{lvs}(H_\Xi(X_L))} \mathbb{H}(X_\ell) + \sum_{X_\ell\in \text{lvs}(H_\Xi(X_R))} \mathbb{H}(X_\ell)$$

so we compute $\mathbb{H}(X_i)$ once per node, memoize it, and compute h_{LR} in $O(1)$ time given $\mathbb{H}(X_L)$ and $\mathbb{H}(X_R)$ by summing the two values.

The time complexity of creating a heap of all tuples is $O(|\mathbb{C}(|\mathbb{V}_i|)$, where $|\mathbb{C}(\mathbb{V}_i)|$ is the number of children $\mathbb{V}_i$ has in trellis $\mathbb{T}$.

Thus it takes $O(|\mathbb{C}(\mathbb{V}_i)|) + O(|\mathbb{H}(X_i)|)$ time to create the min heap for $\mathbb{V}_i$, where $O(|\mathbb{H}(X_i)|)$ is the time complexity for computing $\mathbb{H}(X_i)$.

Therefore, creating the min heaps for every node in a full trellis takes $\sum_{k=1\ldots n} \binom{n}{k} O(2^k) = O(3^n)$ time, when $O(\mathbb{H}(X_i)) = O(2^{|X_i|})$. In a sparse trellis, $\sum_{\mathbb{V}_i \in \mathbb{T}} O(|\mathbb{C}(\mathbb{V}_i)|) + O(|\mathbb{H}(X_i)|) = \sum_{\mathbb{V}_i \in \mathbb{T}} O(|\mathbb{C}(\mathbb{V}_i)|) + \sum_{\mathbb{V}_i \in \mathbb{T}} O(|\mathbb{H}(X_i)|) = \sum_{i=1\ldots|\mathbb{T}|} (|\mathbb{T}| - i) = O(|\mathbb{T}|^2)$, when each trellis vertex, $\mathbb{V}_i$, has the maximum possible number of children and $O(\mathbb{H}(X_i)) = O(|X_i|)$.

(3) When exploring node $\mathbb{V}_i$, Algorithm 1 pops and enqueues an entry from the min heap of every node on the path from $\mathbb{V}_i$ to the root in $H_\Xi(X)$, which takes $O(log(|\mathbb{C}(\mathbb{V}_k)|)$ for each ancestor $\mathbb{V}_k$ in the path. In a full trellis, the maximum possible path length from the root to $\mathbb{V}_i$ is $|X| - |X_i|$. Thus, in the worst case it takes Algorithm 1 $\sum_{j=||X_i|\ldots n} O(log(2^j))$ time to update the min heaps when $\mathbb{V}_i$ is explored. Therefore, the total amount of time spent updating the min heaps in $H_\Xi(X)$ at line 19 in a full trellis when every vertex is explored is $\sum_{k=1\ldots n} \binom{n}{k} \sum_{j=k\ldots n} log(2^k) = O(n^2 2^n)$. In a sparse trellis, the path lengths are maximized when the root, $\mathbb{V}_1$, is a parent of all the remaining $|\mathbb{T}| - 1$ vertices in the trellis, the eldest child of the root, $\mathbb{V}_2$, is a parent of all the remaining $|\mathbb{T}| - 2$ vertices in the trellis, and so on. Thus the sum of the total path lengths is bounded from above by $\sum_{i=1\ldots|\mathbb{T}|} \sum_{j=i\ldots|\mathbb{T}|} log(j) = O(log(|\mathbb{T}|^3))$

Therefore, the time complexity of running Algorithm 1 on a full trellis is $O(3^n)$, and the time complexity of running Algorithm 1 on a sparse trellis is $O(|\mathbb{T}|^2)$.

□

A.5 PROPOSITION 2 (OPTIMAL EFFICIENCY OF A* MAP) PROOF

Proof. We define a mapping between paths in the trellis representing the state space and the standard space of tree structures via $H_\Xi(X)$ (Eq. 2). This mapping is bijective. We have describe a method to find the neighboring states of $H_\Xi(X)$ in the standard space of trees using the trellis in Algorithm 1 (▷Explore new leaves). Given the additive nature of the cost functions in the family of costs (Eq. 2), we can maintain the splits/merges in the aforementioned nested min heap in the trellis structure and have parent nodes' f values be computed from their child's min heaps. The result is that Algorithm 1 exactly follows A*'s best-first according to f with an admissible heuristic in search over the entire space of tree structures. Therefore the optimal efficiency of Algorithm 1 follows as a result of the optimal efficiency of the A* algorithm. □

A.6 PROPOSITION 3 (ADMISSIBILITY OF HIERARCHICAL CORRELATION CLUSTERING COST HEURISTIC) PROOF

Proof. We wish to prove that $\mathbb{H}_{hcc1}(X) <= \text{argmin}_{H \in \mathbb{H}(X)} \phi_{hcc}(H)$. Note that in every hierarchical clustering, every element is eventually separated (at the leaves), thus every edge eventually crosses a cut in every tree, thus $\forall H \in \mathbb{H}(X)$,

$$\sum_{X_L, X_R \in \textsf{sibs}(H)} \sum_{x_i, x_j \in X_L \times X_R} w_{ij} \mathbb{I}[w_{ij} > 0] = \sum_{x_i, x_j \in X} w_{ij} \mathbb{I}[w_{ij} > 0]) \tag{1}$$

Therefore,

$$\begin{aligned}
\phi(H) &= \sum_{X_L, X_R \in \textsf{sibs}(H)} \psi(X_L, X_R) && \text{(by definition)} \\
&= \sum_{X_L, X_R \in \textsf{sibs}(H)} (\sum_{x_i, x_j \in X_L \times X_R} w_{ij}\mathbb{I}[w_{ij} > 0] + \sum_{\substack{x_i, x_j \in X_L \times X_L, \\ i<j}} |w_{ij}|\mathbb{I}[w_{ij} < 0] + \sum_{\substack{x_i, x_j \in X_R \times X_R, \\ i<j}} |w_{ij}|\mathbb{I}[w_{ij} < 0]) && \text{(given HCC objective)} \\
&>= \sum_{X_L, X_R \in \textsf{sibs}(H)} \sum_{x_i, x_j \in X_L \times X_R} w_{ij}\mathbb{I}[w_{ij} > 0] && (\forall X, \sum_{x_i, x_j \in X \times X} |w_{ij}|\mathbb{I}[w_{ij} < 0] >= 0) \\
&= \sum_{x_i, x_j \in X} w_{ij}\mathbb{I}[w_{ij} > 0] && \text{(Eq. 1)} \\
&= \mathbb{H}_{hcc1}(X) && \text{(given HCC Heuristic)}
\end{aligned}$$

□

A.7 HEURISTIC FUNCTIONS FOR GINKGO JETS

We want to find heuristic functions to set an upper bound on the likelihood of Ginkgo hierarchies (for more details about Ginkgo see [19]). We split the heuristic into internal nodes and leaves.

A.7.1 Internal nodes

We want to find the smallest possible upper bound to

$$f(t|\lambda, t_{\mathrm{P}}) = \frac{1}{1-e^{-\lambda}} \frac{\lambda}{t_{\mathrm{P}}} e^{-\frac{\lambda}{t_{\mathrm{P}}} t} \tag{2}$$

with λ a constant. We first focus on getting an upper bound for t_{P} in the exponential. The upper bound for all the parent masses is given by the squared mass of the root vertex, t_{root} (top of the tree).

Next, we look for the biggest lower bound on the squared mass t in the exponential of Equation 2. We consider a fully unbalanced tree as the topology with the smallest per level values of t and we bound each level in a bottom up approach (singletons are at the bottom). For each internal node, t has to be greater than t_{cut} (t_{cut} is a threshold below which binary splittings stop in Ginkgo). Thus, for each element, we find the smallest parent invariant squared mass t_{Pi} that is above the threshold t_{cut}, else set this value to t_{cut} and save it to a list named t_{min}, that we sort. Maximizing the number of nodes per level (having a full binary tree) minimizes the values of t. We start at the first level, taking the first $N//2$ entries of t_{min} and adding them to our candidate minimum list, t_{bound} (N is the number of elements to cluster). For the second level, we take t_p^0 as the minimum value in t_{min} that is greater than t_{cut} and bound t by $[\tilde{t}(i\%2) + t_p^0(i//2)](j//2)$ with $i = 3$ and $j = N\%2 + N//2$, where $\tilde{t}$ is the minimum invariant squared mass among all the leaves. The reason is that $(j//2)$ bounds the total possible number of nodes at current level and i the minimum number of elements of the tree branch below a given vertex at current level. We calculate this bound level by level, maximizing the possible number of nodes at each level. This is described in Algorithm 1.

Algorithm 1 Lower Bound on t in Equation 2

```
1:  function LOWERBOUND($t_i$, $t_{min}$, $N$)
2:      Input: Elements invariant mass squared: $t_i$, minimum value in $t_{min}$ that is greater than $t_{cut}$: $t_p^0$, list with minimum
    parent mass squared above $t_{cut}$ for each element: $t_{min}$, and number of elements: $N$.
3:      Output: List that bounds $t$: $t_{bound}$.
4:      ▷ Set initial variables
5:      $m^2 = sort([t_i$ for i in N$])$
6:      $t_{bound} = sort(t_{min})[0 : N//2]$
7:      $i = 3$
8:      $j = N\%2 + N//2$
9:      do
10:         $t_{bound} += [m^2[0](i\%2) + t_p^0(i//2)](j//2))$
11:         $j = j\%2 + j//2$
12:         $i += 1$
13:     while $len(t_{bound}) < N - 2$
```

Finally, to get a bound for t_{P} (denoted as $\tilde{t}_P$) in the denominator of Equation 2 we want the minimum possible values. Here we consider two options for a heuristic:

- Admissible heuristic: Thus we take $\tilde{t}_P = t_{bound} + \tilde{t}$ (given that the parent of any internal node contains at least one more element) except when $t_{bound} < \max\{t_i\}_{i=0}^{N}$ where we just keep $t_{\mathrm{P}} = t_{bound}$ (see [19] for more details).
- h1: $\tilde{t}_P = t_{bound} + 2\ t_p^0$. Even though we do not have a proof for h1 to be admissible, we gained confidence about it from checking it on a dataset of 5000 Ginkgo jets with up to 9 elements (compared to the exact trellis solution for the MAP tree), as well as the fact that exact A* with h1 agrees with the exact trellis within 6 significant figures (see Figure 5).

Putting it all together, the upper bound for Equation 2 is

$$\sum \log f^{bound} = [-\log(1-e^{-\lambda}) + \log(\lambda)] \\ - \sum_{i=0}^{N-2} \left(\log(\tilde{t}_P) + \lambda * \frac{t^i_{bound}}{t_{root}} \right)$$

A.7.2 Leaves

We want to find the smallest possible upper bound to

$$f(t|\lambda, t_{\mathrm{P}}) = \frac{1}{1-e^{-\lambda}} \left(1 - e^{-\frac{\lambda}{t_{\mathrm{P}}} t_{\mathrm{cut}}} \right) \tag{3}$$

with λ and t_{cut} constants.

We want the maximum possible value of t_{P} while still getting an upper bound in Equation 3. A parent t_{P} consists of a pairing of two elements (leaves). We find and sort the same list t_{min} as we did for the internal nodes. We do not know the order in which the two children are sampled, and the minimum value happens when each child is sampled last, $t_{\mathrm{Pi}} \to (\sqrt{t_{\mathrm{Pi}}} - \sqrt{t_i})^2$ (see [19] for more details). We take the minimum value greater than t_{cut} in t_{min} and refer to it as t^0_{Pi}. Thus we get $t_{\mathrm{Pi}} = (\sqrt{t^0_{\mathrm{Pi}}} - \sqrt{t_i})^2$. For the element with greatest invariant squared mass t_i we just keep $t_{\mathrm{Pi}} = t^0_{\mathrm{Pi}}$. Finally, sum the likelihood over each element i.

A.8 DASGUPTA'S COST

(Dasgupta's Cost)[20] Given a graph with vertices of the dataset X and weighted edges representing pairwise similarities between points $\mathcal{W} = \{(i, j, w_{ij}) | i, j \in \{1, ..., |X|\} \times \{1, ..., |X|\}, i < j, w_{ij} \in \mathbb{R}^+\}$. Dasgupta's cost is defined as:

$$\psi(X_i, X_j) = (|X_i| + |X_j|) \sum_{x_i, x_j \in X_i \times X_j} w_{ij} \tag{4}$$

This is equivalent to the cut-cost definition of Dasgupta's cost with the restriction to binary trees [20].

We are given the similarity graph, $\mathcal{W}(X)$, where the nodes of the graph refer to the dataset X and edges (denoted $E_{\mathcal{W}(X)}$), $w_{i,j}$ give similarity between points $x_i, x_j \in X$. Given a subset $X' \subseteq X$, we hope to design a heuristic $h(\cdot)$ such that $h(X') \leq \min_{H \in \mathbb{H}(X')} J(H)$.

Dasgupta's cost is a sum over weighted edges $E_{\mathcal{W}(X)}$ with each edge's similarity multiplied by the number of descendant nodes. This leads to a heuristic in which the number of descendants is lower-bounded by having a single descendant:

$$h_{\mathsf{d1}}(X) = \sum_{(i,j) \in E_{\mathcal{W}(X)}} w_{i,j} \tag{5}$$

Proposition 1. (Admissible Heuristic for Dasgupta) *Equation 5 is an admissible heuristic for Dasgupta's cost, that is* $h_{\mathit{d1}}(X) \leq J_{\mathit{Dasgupta}}(X)$.

Proof. We wish to prove that $\mathbb{H}_{dasgupta}(X) <= \mathrm{argmin}_{H \in \mathbb{H}(X)} \phi_{dasgupta}(H)$. Note that in every hierarchical clustering, every element is eventually separated (at the leaves), thus every edge eventually crosses a cut in every tree, thus $\forall H \in \mathbb{H}(X)$,

$$\sum_{X_L, X_R \in \mathsf{sibs}(H)} \sum_{x_i, x_j \in X_L \times X_R} w_{ij} = \sum_{x_i, x_j \in X} w_{ij} \tag{6}$$

Since $\forall X, |X| \in \mathbb{Z}^+$, we have

$$
\begin{aligned}
\phi(H) &= \sum_{X_L, X_R \in \textsf{sibs}(H)} \psi(X_L, X_R) && \text{(by definition)} \\
&= \sum_{X_L, X_R \in \textsf{sibs}(H)} ((|X_L| + |X_R|) \sum_{x_i, x_j \in X_L \times X_R} w_{ij}) && \text{(given Dasgupta objective)} \\
&>= \sum_{X_L, X_R \in \textsf{sibs}(H)} \sum_{x_i, x_j \in X_L \times X_R} w_{ij} && ((|X_L| + |X_R|) \geq 1 \wedge w_{ij} \in \mathbb{R}^+) \\
&= \sum_{x_i, x_j \in X} w_{ij} && \text{(by Equation 6)} \\
&= \mathbb{H}_{Dasgupta}(X) && \text{(given Dasgupta heuristic)}
\end{aligned}
$$

□