1 Summarize the paper

This paper proposed a cost function for the hierarchical clustering problem. Specifically, given a set of points represented by a weighted undirected graph, this paper proposes to perform hierarchical clustering in the form of building a rooted tree. The cost function of building the rooted tree is given as the summation of graph edges cut costs. To illustrate this in more detail, given a subset of nodes, several edge cuts are needed to separate these nodes into disjoint clusters. The cost of cutting these cross-cluster edges obtains a penalty proportional to the node-set size. The paper also proves some exciting properties of the proposed cost function and shows that this criterion can work sensibly in canonical instances.

This paper proves that building an optimal rooted tree that minimizes the cost function is NP-hard. To solve this problem, the author proposed an algorithm to build an approximate optimal tree in a top-down construction procedure and proves it can achieve a good approximation ratio.

2 What are the main contributions of the paper?

The main contribution of the paper is twofold:

(1) The author defined a cost function for hierarchical clustering for the first time. This cost function can be used to compare different hierarchical clustering schemes and could also help to improve the current methods.

(2) The paper provides an approximated algorithm to find the optimal solution (under the defined optimal cost function) with a provably good approximation ratio.

3 What are the limitations of the paper?

The limitation I can think of this paper include the following points,

(1) Although this paper proposed a cost function for hierarchical clustering on points, its variables are discrete, i.e., different possible structures of the rooted trees. Thus it is hard to optimize the proposed cost function (It is impossible to use SGD based optimization algorithm).

(2) The paper didn't perform experiments on real-world data sets.

(3) There is a minor error when explaining the toy example shown in Figure 2, if I understood it correctly, $cost_G(T) = 6 + 3 \cdot 3 + 3 \cdot 1 = 21$ should be revised to $cost_G(T) = 6 + 3 \cdot 3 + 3 \cdot 2 = 24$. What is more, some of the writing are not easy to understand (which actually is not technical limitation but really small flaw compared to the contribution of this paper!).

4 Make a research proposal on the topic of the paper

This paper gives the first try to propose a cost function of the hierarchical clustering problem, which is a significant contribution for evaluating hierarchical clustering performance and possibly help improve the current algorithms.

(1) Although optimizing the objective cost is NP-hard, we can always search for a better approximation of the optimal solution. So a research proposal would be, based on the proposed cost function, to find a better algorithm to improve the approximation ratio factor.

(2) Although the paper proved MakeTree(V) could find the approximate optimal result by a good approximation ratio, MakeTree(V) is still a greedy algorithm. A simple try might be: Add beam search strategy on the proposed MakeTree(V) to find a possible better solution (with a trade-off of spending more time).

(3) As said in section 3, the proposed method can not be optimized by SGD. So an interesting but challenging future research direction is finding a continuous version of the cost function, and after that, we can find the optimal solution by applying SGD optimization. (Right now, I do not have a clear proposal in my mind on how to do this, but a possible way to do this is to use a Gaussian mixture model, which assigns a node to different clusters with different probabilities. We can combine this with the cost function proposed in the paper).

5 Prove Equation (1) (at the top of page 4)

The key to prove Equation(1) is to focus on the internal node of tree T. Let us use $Cost_1(T)$ and $Cost_2(T)$ to denote the two cost function, where

$$Cost_1(T) = \sum_{\{i,j\}\in E} w_{ij}|leaves(T[i \vee j])|$$
$$Cost_2(T) = \sum_{splits\ S\to(S_1,\cdots,S_k)\in T} |S|w(S_1,\cdots,S_k)$$

From the definition of *split*, we can get that each internal node, denoted as u, corresponds to a *split* denoted as $S_u \in S$

Let use $\mathcal{U}$ to denote all internal nodes in T and E_u to denote all edges that exsit among $leaves(T(u))$, let use $S_u \to \{S_{u1},\cdots,S_{uk}\}$ to denote the *split* at internal node u. For each internal node u, it holds that

(1) $|leaves(T[u])|$ = $|S_u|$, which is obvious.

(2) $\sum_{\{i,j\}\in E_u} w_{ij} = w(S_{u1},\cdots,S_{uk})$. Here we need to discuss two cases: (1) If the direct children of an internal node u are all leaves, then $leaves(T[u])$ denote a smallest cluster and $\sum_{\{i,j\}\in E_u} w_{ij} = w(S_{u1},\cdots,S_{uk})$ is the number of edges in this cluster. (b) On the other hand, if the directed children of an internal node include at least one internal node, then $leaves(T[u])$ consists of more than one clusters of leaves. In this case $\sum_{\{i,j\}\in E_u} w_{ij} = w(S_{u1},\cdots,S_{uk})$ denotes the number of cross-cluster edges.

We can rewrite $Cost_1(T)$ **as the sum of cost over all internal node, and** $Cost_2(T)$ **the sum of cost over all internal split as follow,**

$$Cost_1(T) = \sum_{u\in\mathcal{U}} |leaves(T[u])| \sum_{\{i,j\}\in E_u} w_{ij} \tag{1}$$
$$Cost_2(T) = \sum_{S_u\in S} |S_u| \sum_{splits\ S_u\to\{S_{u1},\cdots,S_{uk}\}} w(S_{u1},\cdots,S_{uk}) \tag{2}$$

Let us use $C_1(u)$ to denote $|leaves(T[u])|\sum_{\{i,j\}\in E_u} w_{ij}$ and $C_2(S_u)$ to denote $|S_u|\sum_{splits\ S_u\to\{S_{u1},\cdots,S_{uk}\}} w(S_{u1},\cdots,S_{uk})$, then we can rewrite the cost function as

$$Cost_1(T) = \sum_{u\in\mathcal{U}} C_1(u) \tag{3}$$
$$Cost_2(T) = \sum_{S_u\in S} C_2(S_u). \tag{4}$$

For each internal node u and its corresponding split S_u, $C_1(u) = C_2(u)$, thus we can proof $Cost_1(T) = Cost_2(T)$. That is to say, we can affirm that Equation (1) in the paper holds.

6 Prove that dT (·; ·) as defined in at the top of page 4 is indeed an ultrametric

Let us consider a general case shown in Figure 1. Circles in the tree represent the internal nodes, and squares represent the leaves. For two randomly chosen node i and j (colored in green) in the tree, we can find their lowest common ancestor $i \vee j$ (colored in blue). The subtree rooted at $i \vee j$, $T[i \vee j]$, is covered in yellow for observation ease. We omit the rest part of the tree with a triangle.

For any another leave k, k either lies in the subtree $T[i \vee j]$ (yellow region) or outside $T[i \vee j]$ (triangle region). We will discuss this two condition separately. Note that for all conditions, i, j, k can be used interchangeably

Condition 1: k lies outside subtree $T[i \vee j]$

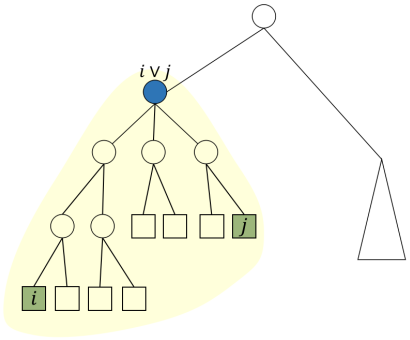


Figure 1: An example to proof Q5 and Q6

In this case, the lowest common ancestor of node i, k and j, k are the same, i.e., $[i \vee k] = [j \vee k]$, and $[i \vee k]$ is the ancestor of $[i \vee j]$. This is shown in Figure 2. In this case,

$$\begin{aligned} d_T(i,k) - d_T(i,j) &= d_T(j,k) - d_T(i,j) \\ &= |leaves(T[i \vee k])| - |leaves(T[i \vee j])| \\ &\geq 1 \end{aligned}$$

$d_T(i,k) - d_T(i,j) \geq 1$ always holds because there are at least 1 leave in the subtree where k lies. Thus $d_T(i,j) \leq max(d_T(i,k), d_T(j,k))$ holds obviously.

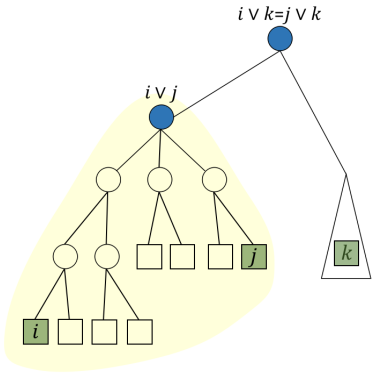


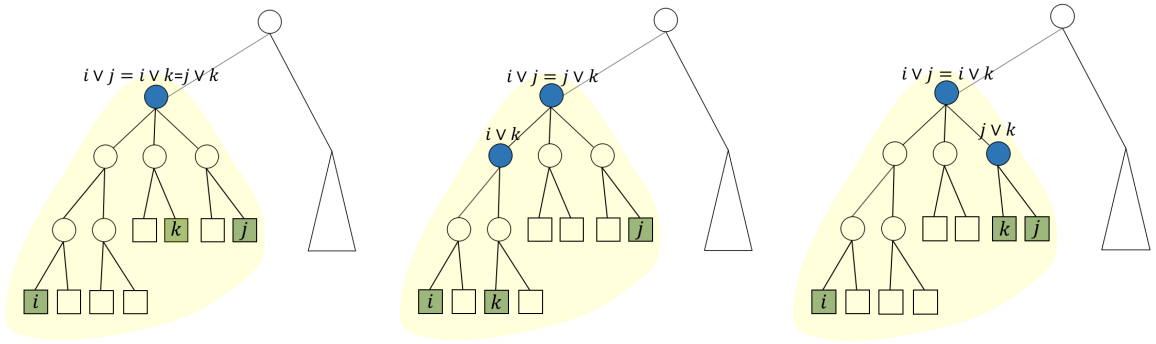
Figure 2: Condition 1: k lies outside $T[i \vee j]$

Condition 2: k lies in subtree $T[i \vee j]$

In this case, $i \vee k$ (or $j \vee k$) are either the same as $i \vee j$ (e.g. Figure 3(a)), or the child of $i \vee j$ (Figure 3(b) and 3(c)). Note that i, j, k can be used interchangeably, so we only discuss the cases shown in Figure 3. Figure 3(b) and Figure 3(c) are actually the same case if we use i and j interchangeably, but we discuss them separately for ease of understanding.

In Figure 3(a), $d_T(i,j) = d_T(i,k) = d_T(j,k)$, thus $d_T(i,j) \leq max(d_T(i,k), d_T(j,k))$ holds.
In Figure 3(b), $d_T(i,k) + 1 \leq d_T(i,j) = d_T(k,j)$, thus $d_T(i,j) \leq max(d_T(i,k), d_T(j,k))$ holds.
In Figure 3(c), $d_T(i,j) = d_T(i,k) \geq d_T(j,k) + 1$, thus $d_T(i,j) \leq max(d_T(i,k), d_T(j,k))$ holds.

There are no other conditions, **so we proved that** $d_T(i,j) \leq max(d_T(i,k), d_T(j,k))$ **holds for all conditions.**



(a) *(b)* *(c)*

Figure 3: Condition 2: k lies in subtree $T[i \vee j]$

The definition of ultrametric is a distance defined on a space M that satisfies for all $x, y, z \in M$.

$$\begin{aligned} d(x,y) &\geq 0 \\ d(x,y) &= d(y,x) \\ d(x,x) &= 0 \\ if\ d(x,y) = 0\ then\ x &= y \\ d(x,z) &\leq max\ \{d(x,y), d(y,z)\}. \end{aligned}$$

Till now we have proved $d_T(.,.)$ have the last property of an ultrametric.

To prove $d_T(.,.)$ has the first four properties, we use the same conclusion we get from section 6 to proof $d_T(.,.)$ is an ultrametric. Note that or all conditions, i, j, k can be used interchangeably.

We can conclude from section 6 that there are two possible condition: (a) $d_T(i,j) = d_T(i,k) = d_T(j,k)$. (b) $d_T(i,j) = d_T(i,k) \geq d_T(j,k) + 1$. For both condition, we can proof $d_T(.,.)$ has the first four properties of a ultrametric.

(A) For any $i \neq j$, there are at least two leaves in $leaves(T[i \vee j])$, i.e., $d_T(i,j) > 1$, the first property holds;

(B) For any i, j, we have $d_T(i,j) = 0 \rightarrow |leaves(i \vee j)| = 1 \rightarrow i = j$, and $i = j \rightarrow d_T(i,j) = 0$, the third and fourth property holds;

(B) For any i, j, we have $d_T(i,j) = d_T(j,i)$, the second property holds;

Proven that $d_T(.,.)$ satisfy all five properties of an ultrametric, we can affirm $d_T(.,.)$ is a ultrametric.

7 Prove that an ultrametric is also a metric

First of all, there are four properties of a metric.

Property (1), non-negativity : $x \neq y \rightarrow d(x,y) > 0$;

Property (2), identity of indiscernibles: $d(x,y) = 0 \Leftrightarrow\ x = y$;

Property (3), symmetry: $d(x,y) = d(y,x)$;

Property (4), triangle inequality: $d(x,z) \leq d(x,y) + d(y,z)$.

From the definition of ultrametric, we can easily prove an ultrametric has property (1), (2), (3) of a metric.

To prove an ultrametric have property (4) of a metric, we first need to verify the following Lemma 1.

Lemma 1: For any x, y, z in space M, at least one of the three equality $d(x,y) = d(x,z)$, $d(x,y) = d(y,z)$, $d(y,z) = d(x,z)$ holds. If only one equality holds, for example, $d(x,y) = d(x,z)$, it must holds that $d(x,y) = d(x,z) > d(y,z)$. Otherwise, $d(x,y) = d(x,z)$= $d(x,y)$ holds.

To prove Lemma 1, we can use contradiction. We prove this by using x, y, z interchangeably: if none of the three equality holds, there must be a rank, e.g., $d(x,y) < d(x,z) < d(y,z)$, such that $d(y,z) > max\ (d(x,y), d(x,z))$, which is contradictory to the definition of an ultrametric. Thus Lemma 1 holds.

With Lemma 1 holds, we can easily prove that an ultrametric has property (4) of a metric. Thus an ultrametric is also a metric.

8 Prove Lemma 1

As shown in Equation (2) from section 5, the cost of a tree is the sum of *split* cost over all internal node.

Pick any internal node u, suppose the original subtree rooted at u is $T[u]$ and we replace it with some other subtree $T^{'}[u]$. It is given in the paper that $|leaves(T[u])| = |leaves(T^{'}[u])|$.

For internal node that lies outside $T[u]$, we denote them by $\mathcal{U}\backslash subtree\{u\}$. It is easy to see that for any internal node $v \in \{\mathcal{U}\backslash subtree\{u\}\}$, the *split* cost at v does not get influenced by the change of $T[u]$ because cross edges between $leaves(T[u])$ and the rest leaves $\{leaves(T)\backslash leaves(T[u])\}$ do not change.

For internal node u **and other internal nodes that lie in tree** $T[u]$, the sum of their *split* cost, denoted by $cost(T[u])$ and $cost(T^{'}[u])$, differs because of the structure change.

We can rewrite Equation (2) from section 5 as

$$Cost(T) = \sum_{v \in \{\mathcal{U}\backslash subtree\{u\}\}} |S_v| \sum_{splits\ S_v \to \{S_{v1}, \cdots, S_{vk}\}} w(S_{v1}, \cdots, S_{vk}) + cost(T[u]) \tag{5}$$

$$Cost(T^{'}) = \sum_{v \in \{\mathcal{U}\backslash subtree\{u\}\}} |S_v| \sum_{splits\ S_v \to \{S_{v1}, \cdots, S_{vk}\}} w(S_{v1}, \cdots, S_{vk}) + cost(T^{'}[u]) \tag{6}$$

Subtract these Equation (5) and Equation 6 we can get Lemma 1,

$$cost(T^{'}) = cost(T) - cost(T[u]) + cost(T_u^{'})$$

9 What is the approximation guarantee of the algorithm?

As given in the paper, the approximation guarantee of the algorithm MakeTree(V), is $\alpha_n \log n$, where α_n is the approximation ratio of the sparsest cut algorithm. In the paper, the author cite a paper that proposed a $\alpha_n = O(logn)$ approximation algorithm of sparset cut. But there are $\alpha_n = O(\sqrt{logn})$ times approximation algorithms available [Arora, Rao, and Vazirani(2009)].

10 What is the running time of the algorithm proposed in the paper?

The algorithm's running time is decided by the time complexity of sparsest cut and the number of top-down split. For each split, sparsest cut can be done in linear time by some fast approximation algorithm [Madry(2010), Spielman and Teng(2013)], and it needs O(logn) time to split the tree in a top-down processor. So in total, the running time of the proposed algorithm, MakeTree(V), is O(nlogn) time.

References

[Arora, Rao, and Vazirani(2009)] Arora, S.; Rao, S.; and Vazirani, U. 2009. Expander flows, geometric embeddings and graph partitioning. *Journal of the ACM (JACM)* 56(2): 1–37.

[Madry(2010)] Madry, A. 2010. Fast approximation algorithms for cut-based problems in undirected graphs. In *2010 IEEE 51st Annual Symposium on Foundations of Computer Science*, 245–254. IEEE.

[Spielman and Teng(2013)] Spielman, D. A.; and Teng, S.-H. 2013. A local clustering algorithm for massive graphs and its application to nearly linear time graph partitioning. *SIAM Journal on computing* 42(1): 1–26.