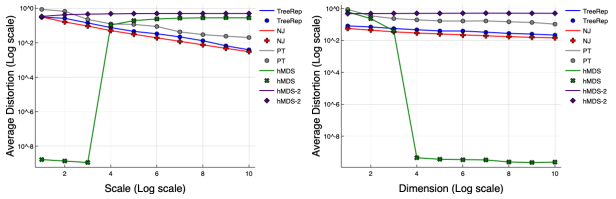

Data	MAP			Distortion			Time		
	TR	HMDS 2	HMDS 200	TR	HMDS 2	HMDS 200	TR	HMDS 2	HMDS 200
Celegans	0.473	0.110	0.400	0.197	0.527	0.142	0.014	0.03	39
Diseasesome	0.895	0.284	0.833	0.188	0.427	0.160	0.017	0.03	56
PHD	0.979	0.027	0.203	0.204	0.461	0.233	0.037	0.06	11
Yeast	0.815	0.147	0.386	0.205	0.166	0.162	0.057	0.05	74
Grid worm	0.707	0.006	0.285	0.188	0.494	0.132	0.731	0.5	381
GRQC	0.685	0.142	0.482	0.192	0.460	0.137	0.42	0.86	512

Thank you for carefully reviewing our submission.

R3"In Abstract, ... please clarify what do you want to express." **Ans: See next discussion.**

R1"The relationship of the method to hyperbolic space ... Sarkar's construction, that would make the relationship clear" **Ans: Trees and hyperbolic manifolds are inherently linked via their geometry.** We discuss the history of finding a discrete, combinatorial structure to represent the metric first and then embedding it into hyperbolic space. Sections 8, 9 in the Appendix, describe this link explicitly and mathematically. For example, in section 8, we present a well known fact from hyperbolic geometry that the asymptotic cone (a geometric tool to study large scale geometry) of a hyperbolic manifold is a tree. Furthermore, references such as "On tree-likeness of hyperbolic space"[20] present different types of geometric connections. Finally, we note that Sarkar's method gives us a way to embed a tree into hyperbolic space with as low distortion as we desire (at the cost of applying a rescaling of the original metric, followed by unscaling the embedded metric - this scale trick is used extensively by De Sa [36]).

R1"h-MDS (their Algorithm 2) ... hyperbolic space." **Ans:** For comparison purposes, we chose the other algorithm from [36] and not hmds because hmds has guaranteed good performance only if **the metric comes from a hyperbolic manifold** and we use hmds to embed into **same dimension or higher**. For completeness and to address the reviewers comment, we have now compared against hmds and the results can be seen in the figures and the table above. The only result not displayed is the performance on the bio data sets as hmds either resulted in infs or returned all 0 metric. In the legend above, HMDS 2 uses hmds to embed into 2 dimensional manifold, HMDS 200 is an embedding into a 200 dimensional manifold, and hdms is embedding into the same dimension as the original points. As we can see, for all experiments **HMDS 2 does not perform well** and the scaling of the distances has an impact numerically. HMDS 200 also has bad MAP, but good distortion. With regards to computational efficiency, HMDS 2 was faster than all except TreeRep and MST, while HMDS 200 was slower than all tree methods but faster than the optimization based methods. These results will be added to the final version. The code used to run hmds is from authors of [36]'s github repository.

R4"TreeRep provides a guarantee ... global bound on the distortion." **Ans:** We are missing an overall global bound on the distortion. We leave it as future (technical theoretical) work. We verify experimentally that we have low global distortion.

R4"Also, in the experiments...TreeRep outperforms other tree construction methods." **Ans:** The addition of Steiner nodes gives us more flexibility in creating trees over methods such as MST and low-stretch trees. We conjecture that the reason we outperform CT [1] is this algorithm uses only 3 points when placing of nodes, but the Gromov condition depends on 4 points and so the algorithm misses some geometric information. NJ is a local to global method; i.e., it makes local connections between points, whereas we are global to local. We do not know a priori, however, which method, NJ or TreeRep, will work better. Nevertheless, **our method is faster than NJ and can always be used.**

R2"My only concern ... NeurIPS community might be interested in." **Ans:** A variety of the papers on this topic have been published in Neurips/ICML with great success [30,31,36,43]

R3"Reproducibility...No" **Ans:** We provide code, pseudo-code, set ups, parameters, our data, and steps for how to run our experiments at the link in Footnote 4 on page 6 of the submission.

R1"the paper would be a stronger contender...downstream task (such as MAP)" **Ans:** For unweighted graphs, results in table 6 of the Appendix, we do compute MAP. The last two paragraphs of the paper discuss this experiment.

R4"In Definition 5 ..." **Ans:** Yes, that was a mistake, we have fixed the definition.

R3"Table 1, 2 and 3...efficiency is not that impressive" **Ans:** The tables do show our efficiency. Table 1 shows that for $n = 1611$, we are 2 orders of magnitude faster than NJ. Table 3 shows that we are at least an one order magnitude faster than all algorithms except MST (which performs poorly) and many orders of magnitude faster than PT and PM.

R3"Yu and De Sa's work in NeurIPS2019... " **Ans:**. Yu and De Sa primarily proposes a method of representing points on the hyperbolic manifold using a small number of bits rather than a new method to learn a representation. Indeed, they use the learning methods from [30,31] and use their proposed representation to avoid numerical issues. Thus, we do not directly compare against them