Appendix

A Omitted Proofs

Proof of [Lemma 4] Let c_0 be the point that minimizes $\text{avg}(C, c_0)$ in S. Let $B_C(c_0, r)$ be a ball around c_0 of radius r, which contains the set of points in C with distance less than some r from c_0. Correspondingly let $\overline{B}_C(c_0, r)$ denote the ball with same center and radius, but including its boundaries. For any radius r_ϵ, to simplify the notation we denote the two balls by B_ϵ and $\overline{B}_\epsilon$ respectively. We can find a r_ϵ such that $|B_\epsilon| < \epsilon|C|$, but $\overline{B}_\epsilon \geq \epsilon|C|$. We will prove any point in $\overline{B}_\epsilon$ is a good center for C. For any point $q \in \overline{B}_\epsilon$, by the triangle inequality, $\sum_{p\in C} d(p, q) \leq \sum_{p\in C} d(p, c_0) + \sum_{p\in C} d(q, c_0) \leq \sum_{p\in C} d(p, c_0) + |C|r_\epsilon$. Notice that $\sum_{p\in C} d(p, c_0) = \sum_{p\in B_\epsilon} d(p, c_0) + \sum_{p\notin B_\epsilon} d(p, c_0) \geq (1-\epsilon)|C|r_\epsilon$, since $d(p, c_0) \geq r_\epsilon$ if $p \notin B_\epsilon$. Therefore, $|C|r_\epsilon \leq \frac{1}{1-\epsilon}\sum_{p\in C} d(p, c_0)$, giving us $\sum_{p\in C} d(p, q) \leq (1 + \frac{1}{1-\epsilon})\sum_{p\in C} d(p, c_0) \leq 2(1+\epsilon)\sum_{p\in C} d(p, c_0)$, so $\text{avg}(C, q) \leq 2(1+\epsilon)\text{avg}(C, c_0)$.

This implies that it is sufficient to sample a single point from $\overline{B}_\epsilon$. There are $\epsilon|C|$ points in the ball $\overline{B}_\epsilon$. Every time we sample we have ϵ probability of getting a point p such that $\text{avg}(C, p) \leq 2(1+\epsilon)\text{avg}(C, c_0)$. Chernoff bounds ensure a such a point is sampled with probability at least $1 - \frac{1}{n^{\Omega(1)}}$. □

B Correctness and Run Time of Average Linkage Algorithm

The following theorems show that the algorithm approximates the decision of average linkage up to a constant factor, and the algorithm runs in near-linear time. The value of ρ is asdescribed in the data strcutre in the preliminaries.

Theorem 9 (Running Time)**.** *Let $n = |S|$ and Δ be the aspect ratio in S. The run time of the algorithm is $O(\frac{1}{\epsilon}n^\rho \log^2 n \cdot \log\Delta \cdot H(n))$.*

Proof. We first analyze the time needed to construct the data structures. there are $\frac{1}{\epsilon}\log\Delta$ number of threshold δ's. For every δ, it takes $O(n^\rho \log^2 nH(n))$ time to build the data structure. The total time to construct each of these data structures is bounded by $O(\frac{1}{\epsilon}n^\rho \log^2 n \cdot \log\Delta \cdot H(n))$.

Now we analyze the total time the algorithm spends querying the data structure or deleting a point. These queries occur to discover whether a cluster can be merged. If a cluster cannot be merged, its corresponding point is removed from the data structure for the given threshold δ. Thus the total number of queries is $O(\frac{1}{\epsilon}(\log\Delta)n)$. Every query takes $O(n^\rho \log n)$ time. In total more time is spent constructing the data structures.

Next, we analyze the total time it takes to update the data structures. By Lemma 7, every time we update the center and deviation for cluster C it takes $O(\frac{1}{\epsilon}|C|\log n)$ time. The amortized cost on every point in C is $O(\frac{1}{\epsilon}\log n)$. Pick any point $p \in S$, every time p receives an amortized cost of $O(\frac{1}{\epsilon}\log n)$, the size of the cluster p belongs to grows by a factor of $1+\eta$ where $\eta = \frac{\epsilon^2}{1+\xi}$. Thus the total merging and updating cost on point p is bounded by $O(\log_{1+\eta}(n)\cdot\frac{1}{\epsilon}\log n) = O(\frac{1}{\eta}\log n \cdot \frac{1}{\epsilon}\log n)$. This is bounded by $O(\frac{1}{\epsilon^3}\log^2 n)$. Taking the sum over all points in S gives a total update time of $O(\frac{1}{\epsilon^3}n\log^2 n)$. This is a lower order term because it is bounded by the time to construct the data structures if $H(n) = \Omega(n)$.

□

The next theorem bounds the approximate ratio of the algorithm.

Theorem 10 (Approximation Guarantees)**.** *With high probability, Algorithm 1 gives an approximation ratio of $(5\gamma+4)(1+O(\epsilon))$ for average linkage and completes the hierarchical clustering tree.*

Our remaining goal is to prove Theorem 10. The theorem would be immediate from Lemma 6 if σ and ϕ were updated every time two clusters were merged. However, these are not always updated to ensure an efficient running time. Instead, we show the following key properties of the algorithm. Intuitively, they guarantee that if the cluster size does not grow significantly, "borrowing" the center and deviation from a cluster it is merged from will only cause the distance to be slightly distorted. The first property is critical and what allows us to derive the second and third.

Lemma 11 (Properties of Merging)**.** *During any iteration of Algorithm 2 for some fixed δ, these properties always hold.*

(1) For any cluster $C \in \mathcal{C}$, use $\widehat{C}$ to denote the subcluster of C whose center and deviation are used by C. For any cluster $C_i \in \mathcal{C}$, and another cluster $C_j \neq C_i \in \mathcal{C}$. Then we have the following two equations,

$$(1 - \tfrac{|C_i|-|\widehat{C}_i|}{|C_i|})\text{avg}(\widehat{C}_i, C_j) \leq \text{avg}(C_i, C_j) \leq (1 + \tfrac{1+\xi}{\epsilon}\cdot\tfrac{|C_i|-|\widehat{C}_i|}{|C_i|})\text{avg}(\widehat{C}_i, C_j) \quad (1)$$

and

$$\begin{aligned}&(1-\frac{|C_i|-|\widehat{C}_i|}{|C_i|})(1-\frac{|C_j|-|\widehat{C}_j|}{|\widehat{C}_j|})\text{avg}(\widehat{C}_i,\widehat{C}_j)\\&\leq \text{avg}(C_i,C_j)\\&\leq (1+\frac{1+\xi}{\epsilon}\cdot\frac{|C_i|-|\widehat{C}_i|}{|C_i|})\\&(1+\frac{1+\xi}{\epsilon}\frac{|C_j|-|\widehat{C}_j|}{|C_j|})\text{avg}(\widehat{C}_i,\widehat{C}_j) \qquad (2)\end{aligned}$$

(2) At the end of the iteration, no pairs of unmerged clusters have average distance less than $\frac{\delta}{(1+\epsilon)^2}$.

(3) During this iteration, every pair of clusters we merge has distance at most $(5\gamma+4)(1+\epsilon)^3\delta$.

In Section D in the appendix, we will see the detailed proof of this lemma. Notice that Theorem 10 is a corollary of the lemma.

C Algorithm for Single Linkage

Proof of [Theorem 3**]** First we analyze the approximation ratio. Notice that after all merges for threshold δ, there is no edge of weight at most δ that can be added without creating a cycle. Indeed, say such an edge (p,q) exists. Assume p gets added into the query list before q. Consider the time when p is deleted from the query list. q is not in the same component with p, so since $d(p,q)\leq\delta$, querying p should have returned another point and p should not be deleted at this time, contradiction. Thus for threshold δ, we have guarantee that the shortest edge that could be added has distance at least $\frac{\delta}{1+\epsilon}$. Since all queries are (δ,γ)-NN queries, the edges found have weights bounded by $\gamma\delta$, giving the approximation ratio $\gamma(1+\epsilon)$.

Now we analyze the running time. There are $\log_{1+\epsilon}\Delta=\Theta(\frac{1}{\epsilon}\log\Delta)$ threshold values in total. Fix any threshold value δ. The construction of data structure takes $\log^2 n\cdot n^\rho H(n)$. For every point p, every query of p either results in merging the component containing p with some other component, or deleting p from the query list and thus never querying p again for the current threshold δ. So the total number of queries performed for any point in S is bounded by $O(n)$, taking $O(nQ(n,\gamma))=O(n^{1+\rho}\log^2 n)$ time in total. Any point $p\in S$ gets deleted once from the data structure, so the algorithm performs $O(n)$ deletions, taking $O(nD(n,\gamma))=O(n^{1+\rho}\log^2 n)$ time. The time spent on adding and deleting points to the query list can be in $O(n)$. Assuming $H(n)=\Omega(n)$, the data construction time dominates the run time of algorithm, thus the run time is $O(n^\rho\log^2 n\log\Delta H(n))$. □

D Algorithm for Average Linkage

The aim of this section is to prove Theorem 10 and Theorem 9. Before we prove correctness and run time, we need to establish the properties in Lemma 11 first. The proof follows logic similar to that of Lemma A.6 in (1).

Proof of [Lemma 11**]**

We prove this by induction. Assume we choose to merge C_i with C_j in the k-th iteration, for threshold $\delta=(1+\epsilon)^{k-1}$. Assume all properties hold in all iterations for $\delta=(1+\epsilon)^t$ for $t=0,1,\ldots,k-2$, and all the time in the current iteration, until the merge. Specifically, we assume the algorithm finds the pair C_i and C_j correctly before merging them. Let $\mathcal{C}$ be the current partition of S. We will prove that after the merge of C_i and C_j, 1) the approximation ratio in property one still holds, 2) the next pair of clusters we found are close to each other, and eventually 3) there will be no clusters with average linkage less than $\frac{\delta}{(1+\epsilon)^2}$ by the end of iteration for current threshold δ.

Notice that the following always holds. Let $\xi=(1+\epsilon)^6(5\gamma+4)$. For any other cluster $C\in\mathcal{C}$, $\text{avg}(C_i,C_j)\leq\xi\min\{\text{avg}(C_i,C),\text{avg}(C_j,C)\}$, and $\text{avg}(\widehat{C}_i,\widehat{C}_j)\leq\xi\min\{\text{avg}(\widehat{C}_i,\widehat{C}),\text{avg}(\widehat{C}_j,\widehat{C})\}$. This will be used in the proof.

Since we choose to merge C_i and C_j, by third property in Lemma 11 we must have the following by induction $\text{avg}(C_i,C_j)\leq(1+\epsilon)^3(5\gamma+4)\delta$. We also have $\text{avg}(\widehat{C}_i,\widehat{C}_j)\leq(1+\epsilon)(5\gamma+4)\delta$. This is because the (r,γ)-NN query picked the pair $(\widehat{C}_i,\widehat{C}_j)$. Assume the cluster C comes from merging $C'_1,\ldots,C'_\ell$, where $\{C'_1,\ldots,C'_\ell\}$ are clusters in the partition at the end of the iteration for the previous threshold $\frac{\delta}{1+\epsilon}$.

We first prove $\text{avg}(C_i,C_j)\leq\xi\text{avg}(C_i,C)$. By property one, the average distance between C_i and any of $C'_1,\ldots,C'_\ell$ is at least $\frac{\delta}{(1+\epsilon)^3}$. By an averaging argument we have $\text{avg}(C_i,C)=\frac{\sum_{t=1}^{\ell}|C'_t|\text{avg}(C_i,C'_t)}{|C|}\geq\frac{\delta}{(1+\epsilon)^3}$. Letting $\xi=(1+\epsilon)^6(5\gamma+4)$ gives $\text{avg}(C_i,C_j)\leq\xi\text{avg}(C_i,C)$. Likewise we have $\text{avg}(C_i,C_j)\leq\xi\text{avg}(C_j,C)$.

Next we prove $\text{avg}(\widehat{C}_i,\widehat{C}_j)\leq\xi\text{avg}(\widehat{C}_i,\widehat{C})$. By property one $\text{avg}(\widehat{C}_i,\widehat{C})\geq(1-\epsilon)^2\text{avg}(C_i,C)\geq(1-\epsilon)^2\cdot\frac{\delta}{(1+\epsilon)^3}\simeq\frac{\delta}{(1+\epsilon)^5}$. Since $\text{avg}(\widehat{C}_i,\widehat{C}_j)\leq(5\gamma+4)(1+\epsilon)\delta$, the inequality holds.

Now we prove the property one still holds after the merge. Let $C''=C_i\cup C_j$ be the new cluster. Assume first that $(\phi(C''),\sigma(C''))$ got recalculated. Then $\widehat{C}''=C''$, $s(C'')=|C''|$ and for any cluster C, $\text{avg}(\widehat{C}'',C)=$

$\text{avg}(C'', C)$. Both Inequality (1) and (2) are reduced to:

$$(1 - \frac{|C| - |\widehat{C}|}{|C|})\text{avg}(C'', \widehat{C}) \leq \text{avg}(C'', C)$$
$$\leq (1 + \frac{1+\xi}{\epsilon} \cdot \frac{|C| - |\widehat{C}|}{|C}|)\text{avg}(C'', \widehat{C})$$

By induction we have

$$(1 - \frac{|C| - |\widehat{C}|}{|C|})\text{avg}(C_i, \widehat{C}) \leq \text{avg}(C_i, C)$$
$$\leq (1 + \frac{1+\xi}{\epsilon} \cdot \frac{|C| - |\widehat{C}|}{|C|})\text{avg}(C_i, \widehat{C})$$

and

$$(1 - \frac{|C| - |\widehat{C}|}{|C|})\text{avg}(C_j, \widehat{C}) \leq \text{avg}(C_j, C)$$
$$\leq (1 + \frac{1+\xi}{\epsilon} \cdot \frac{|C| - |\widehat{C}|}{|C|})\text{avg}(C_j, \widehat{C})$$

Taking the weighted average gives the inequality we want.

For the following argument, assume that $(\phi(C''), \sigma(C''))$ is not recalculated. WLOG, assume $s(C_i) \geq s(C_j)$ and we used the center of deviation of C_i, so $\widehat{C}'' = \widehat{C}_i$. Now, given another cluster $C \in \mathcal{C}$, we have that $\text{avg}(C, C_j) \leq \text{avg}(C_i, C) + \text{avg}(C_i, C_j) \leq (1+\xi)\text{avg}(C_i, C)$ by triangle inequality of average distances. We first prove Inequality (1). That is,

$$(1 - \frac{|C| - |\widehat{C}|}{|C|})\text{avg}(C'', \widehat{C}) \leq \text{avg}(C'', C)$$
$$\leq (1 + \frac{1+\xi}{\epsilon} \cdot \frac{|C| - |\widehat{C}|}{|C|})\text{avg}(C'', \widehat{C})$$

And

$$(1 - \frac{|C''| - |\widehat{C}''|}{|C''|})\text{avg}(\widehat{C}'', C) \leq \text{avg}(C'', C)$$
$$\leq (1 + \frac{1+\xi}{\epsilon} \cdot \frac{|C''| - |\widehat{C}''|}{|C''|})\text{avg}(\widehat{C}'', C)$$

The former inequality can be proved in the same way as the case where C'' is recalculated by taking weighted average over two inequalities for C_i and C_j. Thus we only need to prove the second one. This is done by expanding the expression of $\text{avg}(C'', \widehat{C})$ as weighted average of $\text{avg}(C_i, \widehat{C})$ and $\text{avg}(C_j, \widehat{C})$ and then leverage the fact that $\text{avg}(C_j, C) \leq (1+\xi)\text{avg}(C_i, C)$. For upper bound,

$$\text{avg}(C'', C) = \frac{|C_i|\text{avg}(C_i, C) + |C_j|\text{avg}(C_j, C)}{|C_i| + |C_j|}$$
$$\leq \frac{|C_i|\text{avg}(C_i, C) + |C_j|(1+\xi)\text{avg}(C_i, C)}{|C_i| + |C_j|}$$
$$\leq \frac{|C_i| + |C_j|(1+\xi)}{|C_i| + |C_j|}(1 + \frac{1+\xi}{\epsilon}\frac{|C_i| - s(C_i)}{|C_i|})\text{avg}(\widehat{C}_i, C)$$
$$= (1 + \frac{\xi|C_j|}{|C_i| + |C_j|})(1 + \frac{1+\xi}{\epsilon}\frac{|C_i| - s(C_i)}{|C_i|})\text{avg}(\widehat{C}_i, C)$$

We prove that

$$(1 + \tfrac{\xi|C_j|}{|C_i|+|C_j|}) \cdot (1 + \tfrac{1+\xi}{\epsilon} \cdot \tfrac{|C_i| - s(C_i)}{|C_i|})$$
$$\leq 1 + \tfrac{1+\xi}{\epsilon} \cdot \tfrac{|C''| - s(C'')}{|C''|} \quad (3)$$

We have:

$$LHS \leq \quad (1 + \frac{1+\xi}{\epsilon} \cdot \frac{|C_i| - s(C_i)}{|C_i|}$$
$$+ \frac{\xi|C_j|}{|C_i| + |C_j|} \cdot (1 + \frac{1+\xi}{\epsilon} \cdot \frac{\epsilon^2}{1+\xi}))$$
$$= (1 + \frac{1+\xi}{\epsilon} \cdot \frac{|C_i| - s(C_i)}{|C_i|} + (1+\epsilon)\frac{\xi|C_j|}{|C_i| + |C_j|})$$

We prove that $(1+\epsilon)\frac{\xi|C_j|}{|C_i|+|C_j|} \leq \frac{1+\xi}{\epsilon} \cdot \frac{|C_j|s(C_i)}{|C_i|(|C_i|+|C_j|)}$. Since $\frac{s(C_i)}{|C_i|} \geq \frac{1+\xi-\epsilon^2}{1+\xi}$, $\frac{1+\xi}{\epsilon} \cdot \frac{|C_j|s(C_i)}{|C_i|(|C_i|+|C_j|)} \geq \frac{1+\xi-\epsilon^2}{\epsilon} \cdot \frac{|C_j|}{|C_i|+|C_j|} \geq (1+\epsilon)\frac{\xi|C_j|}{|C_i|+|C_j|}$. Therefore,

$$\text{avg}(C'', C) \leq \text{avg}(\widehat{C}_i, C) \cdot (1 + \frac{1+\xi}{\epsilon} \cdot \frac{|C_i| - s(C_i)}{|C_i|}$$
$$+ (1+\epsilon)\frac{\xi|C_j|}{|C_i| + |C_j|})$$
$$\leq \text{avg}(\widehat{C}_i, C) \cdot (1 + \frac{1+\xi}{\epsilon} \cdot \frac{|C_i| - s(C_i)}{|C_i|}$$
$$+ \frac{1+\xi}{\epsilon}\frac{|C_j|s(C_i)}{|C_i|(|C_i| + |C_j|)})$$
$$= \text{avg}(\widehat{C}_i, C)(1 + \frac{1+\xi}{\epsilon} \cdot \frac{|C_i| + |C_j| - s(C_i)}{|C_i| + |C_j|})$$
$$= \text{avg}(\widehat{C}'', C)(1 + \frac{1+\xi}{\epsilon} \cdot \frac{|C''| - s(C_i)}{|C''|})$$

For lower bound,

$$\text{avg}(\widehat{C}'', C) = \frac{|C_i|\text{avg}(C_i, C) + |C_j|\text{avg}(C_j, C)}{|C_i| + |C_j|}$$
$$\geq \frac{|C_i|\text{avg}(C_i, C)}{|C_i| + |C_j|}$$
$$\geq \text{avg}(\widehat{C}_i, C) \cdot \frac{s(C_i)}{|C_i|} \cdot \frac{|C_i|}{|C_i| + |C_j|}$$
$$= \text{avg}(\widehat{C}_i, C) \cdot \frac{s(C_i)}{|C_i| + |C_j|}$$
$$= \text{avg}(\widehat{C}'', C) \cdot (1 - \frac{|C''| - s(C'')}{|C''|})$$

Thus we've proved that the Inequality (1) holds after the new merge. Using the same logic, we can prove Inequality (2).

For upper bound,

$$\begin{aligned}
&\text{avg}(C'', C) = \frac{|C_i|\text{avg}(C_i, C) + |C_j|\text{avg}(C_j, C)}{|C_i| + |C_j|} \\
&\leq \frac{|C_i|\text{avg}(C_i, C) + |C_j|(1+\xi)\text{avg}(C_i, C)}{|C_i| + |C_j|} \\
&\leq \frac{|C_i| + |C_j|(1+\xi)}{|C_i| + |C_j|} \cdot (1 + \frac{1+\xi}{\epsilon} \cdot \frac{|C_i| - s(C_i)}{|C_i|}) \\
&\cdot (1 + \frac{1+\xi}{\epsilon} \cdot \frac{|C| - s(C)}{|C|})\text{avg}(\widehat{C}_i, \widehat{C})
\end{aligned}$$

Plugging Inequality (3) into the RHS gives us:

$$\begin{aligned}
\text{avg}(C'', C) \leq (1 + \frac{1+\xi}{\epsilon} \cdot \frac{|C''| - s(C'')}{|C''|}) \\
(1 + \frac{1+\xi}{\epsilon} \cdot \frac{|C| - s(C)}{|C|})\text{avg}(\widehat{C}_i, \widehat{C})
\end{aligned}$$

For lower bound,

$$\begin{aligned}
&\text{avg}(C'', C) = \frac{|C_i|\text{avg}(C_i, C) + |C_j|\text{avg}(C_j, C)}{|C_i| + |C_j|} \\
&\geq \frac{|C_i|\text{avg}(C_i, C)}{|C_i| + |C_j|} \\
&\geq \text{avg}(\widehat{C}_i, \widehat{C}) \cdot (1 - \frac{|C_i| - s(C_i)}{|C_i|}) \cdot (1 - \frac{|C| - s(C)}{|C|}) \\
&\cdot \frac{|C_i|}{|C_i| + |C_j|} \\
&= \text{avg}(\widehat{C}_i, \widehat{C}) \cdot \frac{s(C_i)}{|C_i|} \cdot \frac{s(C)}{|C|} \cdot \frac{|C_i|}{|C_i| + |C_j|} \\
&= \text{avg}(\widehat{C}_i, \widehat{C}) \cdot \frac{s(C_i)}{|C_i| + |C_j|} \frac{s(C)}{|C|} \\
&= \text{avg}(\widehat{C}'', \widehat{C}) \cdot (1 - \frac{|C''| - s(C'')}{|C''|})(1 - \frac{|C| - s(C)}{|C|})
\end{aligned}$$

The first property directly implies $\text{avg}(C_i, C_j) = (1 \pm \epsilon)^2\text{avg}(\widehat{C}_i, \widehat{C}_j)$ for all pairs $C_i, C_j \in \mathcal{C}$.

For the second property, assume there is one (C'_i, C'_j) where $\text{avg}(C'_i, C'_j) \leq \frac{\delta}{(1+\epsilon)^2}$. So $\text{avg}(\widehat{C}'_i, \widehat{C}'_j) \leq \delta$. This implies that the deviations $\sigma(\widehat{C}'_i), \sigma(\widehat{C}'_j) \leq 2(1+\epsilon)\delta$, so the two centers $\phi(\widehat{C}'_i), \phi(\widehat{C}'_j)$ cannot be deleted until the query returns empty set. Since $\text{avg}(\widehat{C}'_i, \widehat{C}'_j) \leq \delta$, $d(\phi(\widehat{C}'_i), \phi(\widehat{C}'_j)) \leq \text{avg}(\widehat{C}'_i, \widehat{C}'_j) + \sigma(\widehat{C}'_i) + \sigma(\widehat{C}'_j) \leq 5(1+\epsilon)\delta$. Since $r = 5(1+\epsilon)\delta$, a valid r, γ-NN query can't return empty set for $\phi(C'_i)$ and $\phi(C'_j)$, contradiction.

For the third property, since after the merge $(\widehat{C}'_i, \widehat{C}'_j)$ is returned by a valid (r, γ)-NN query where $r = 5(1+\epsilon)\delta$, $\text{avg}(\widehat{C}'_i, \widehat{C}'_j) \leq (1+\epsilon)(5\gamma+4)\delta$. Using the first property gives us $\text{avg}(C'_i, C'_j) \leq (1+\epsilon)^3(5\gamma+4)\delta$.

□

E Omitted Experiment Results

This section contains the experiment results omitted in the main body. We show the performance table and running time plots for Proxy-Hash-SL And Proxy-Hash-AL on the three road map datasets and `seizure`. These results lead to the same conclusions as in the main body: both Proxy-Hash-SL and Proxy-Hash-AL are more accurate and efficient than directly using the proxy metric.

E.1 Performance And Running Time for Proxy-Hash-SL And Proxy-Hash-AL

We defer the performance data for Proxy-Hash-SL on road map dataset in Bay Area and Great Lakes until the next subsection, where we will show the complete data table containing performance data for a range of ϵ values in the LSH algorithm, gathered from the road map datasets in all three cities. Here we show the performance for Proxy-Hash-AL on road maps in Bay Area and Great Lakes in Table 6 and 7.

Running-time wise, Figure 4 and 5 show the growth in number of distance computations in Proxy-Hash-AL for road map in Bay Area and the Great Lakes, versus growth in sample size. The sample sizes are plotted on log scale. Both curves are strictly sub-quadratic as it is dominated by $y = cx^{1.3}$ where c is a constant.

In the main body, we have also omitted the running time plots for Proxy-Hash-SL for all data sets. See Figure 6 for them. Clearly, all running time curves are strictly subquadratic, and even only slightly super-linear on all road map datasets.

E.2 Robustness of Performance Against Sample Sizes And Parameter Tuning

There are a lot of parameters in the implementation of both Proxy-Hash-SL and Proxy-Hash-AL, which might affect the accuracy and efficiency of our algorithms. One of the most important parameters is ϵ - in every round of LSH the threshold merging value grows by a factor of $1+\epsilon$. In this section we show that the performance of our algorithms are robust against different ϵ values and the growth of sample sizes. We focus on Proxy-Hash-SL for now.

See Table 3, 9, 10 for partial statistics of Proxy-Hash-SL performance for Bay Area and the Great Lakes. The sample sizes are picked to grow by approximately a factor of 2. Here

Table 6: Comparing the performance of different average linkage methods, Bay Area

Sample Size	299	440	481	830	909	1065	1280	1363
Aprx Ratio, mean, Proxy-AL	3.608	2.491	3.423	2.830	2.431	3.017	2.787	2.175
Aprx Ratio, mean, Proxy-Hash-AL	1.484	1.647	1.493	1.725	1.629	1.709	1.821	1.730
Aprx Ratio, 90%, Proxy-AL	6.554	4.327	5.324	3.759	3.816	5.156	4.711	3.418
Aprx Ratio, 90%, Proxy-Hash-AL	2.024	2.300	1.987	2.402	2.214	2.313	2.550	2.353
Aprx Ratio, max, Proxy-AL	85.784	126.459	89.782	116.732	44.144	96.401	215.839	64.011
Aprx Ratio, max, Proxy-Hash-AL	2.553	3.223	3.184	3.759	3.816	3.646	4.507	3.418
Global Obj, Proxy-AL	0.994	0.990	0.990	0.998	0.999	0.994	0.963	1.001
Global Obj, Proxy-Hash-AL	1.004	0.970	0.985	0.975	0.999	0.987	0.966	0.971

Table 7: Comparing the performance of different average linkage methods, the Great Lakes

Sample Size	86	211	363	496	632	713	1064	1885
Aprx Ratio, mean, Proxy-AL	1.539	1.457	1.935	1.971	1.975	1.742	1.828	1.837
Aprx Ratio, mean, Proxy-Hash-AL	1.362	1.545	1.562	1.592	1.603	1.675	1.766	1.797
Aprx Ratio, 90%, Proxy-AL	2.158	1.732	2.476	2.516	3.280	2.839	2.466	2.658
Aprx Ratio, 90%, Proxy-Hash-AL	1.890	2.311	2.112	2.172	2.158	2.268	2.523	2.412
Aprx Ratio, max, Proxy-AL	14.361	8.603	55.200	45.255	28.929	17.767	65.448	51.816
Aprx Ratio, max, Proxy-Hash-AL	2.428	3.367	3.043	2.971	3.079	3.137	5.073	3.943
Global Obj, Proxy-AL	0.992	0.996	0.990	1.001	0.988	0.997	0.990	0.996
Global Obj, Proxy-Hash-AL	0.983	0.998	0.996	0.983	0.994	0.995	0.992	0.997

Table 8: Ratio between total road distance of the tree and real MST, New York

Sample Size	169	330	727	1166	1825	3765	6710	14428	28985
Euclidean	1.760	1.304	1.554	1.784	1.412	1.805	1.753	1.883	1.511
Proxy-Hash-SL-0.1	1.035	1.016	1.021	1.024	1.027	1.030	1.029	1.022	1.024
Proxy-Hash-SL-0.2	1.048	1.035	1.055	1.038	1.045	1.040	1.066	1.044	1.055
Proxy-Hash-SL-0.3	1.109	1.030	1.055	1.052	1.062	1.060	1.063	1.052	1.059
Proxy-Hash-SL-0.4	1.082	1.036	1.075	1.064	1.068	1.064	1.063	1.074	1.078
Proxy-Hash-SL-0.5	1.074	1.094	1.081	1.091	1.065	1.068	1.090	1.072	1.077

Table 9: Ratio between total road distance of the tree and real MST, Bay Area

Sample Size	251	424	909	1804	3732	7135	10106	22861
Euclidean	2.241	1.643	1.954	1.407	1.905	2.252	2.198	2.397
Proxy-Hash-SL-0.1	1.027	1.039	1.032	1.026	1.035	1.026	1.028	1.027
Proxy-Hash-SL-0.2	1.058	1.045	1.038	1.046	1.058	1.038	1.049	1.047
Proxy-Hash-SL-0.3	1.050	1.045	1.034	1.051	1.066	1.064	1.054	1.070
Proxy-Hash-SL-0.4	1.098	1.042	1.058	1.049	1.073	1.063	1.068	1.069
Proxy-Hash-SL-0.5	1.105	1.070	1.061	1.049	1.081	1.082	1.069	1.071

Table 10: Ratio between total road distance of the tree and real MST, the Great Lakes

Sample Size	616	1289	3063	6564	14056	26004
Euclidean	1.434	1.586	1.433	1.623	1.799	1.824
Proxy-Hash-SL-0.1	1.054	1.036	1.029	1.036	1.030	1.031
Proxy-Hash-SL-0.2	1.053	1.067	1.038	1.046	1.057	1.049
Proxy-Hash-SL-0.3	1.111	1.081	1.042	1.079	1.052	1.071
Proxy-Hash-SL-0.4	1.108	1.100	1.043	1.084	1.074	1.082
Proxy-Hash-SL-0.5	1.081	1.094	1.062	1.085	1.118	1.083

Figure 7, 8 and 9 show the performance of our algorithm versus sample size and different ϵ values. Here Proxy-Hash-SL-ϵ refers to spanning tree constructed by Proxy-Hash-SL using ϵ as parameter.

Figure 11 and 12 show the number of distance computations that would be needed by our algorithm, versus sample size and ϵ. As is the case with New York, it grows only slightly super-linearly with sample size.

Figure 10, 11 and 12 show the number of distance evaluations done by the algorithm. The naive implementation this grows quadratically. We see that Proxy-Hash grows slightly superlinearly with input size.

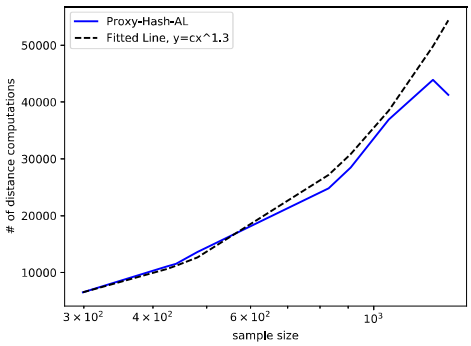


Figure 4: Growth of distance computation, Proxy-Hash-AL, Bay Area

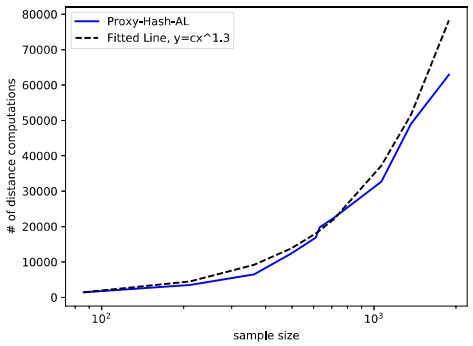
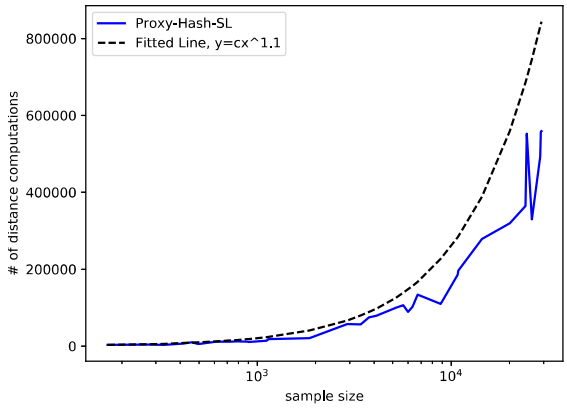
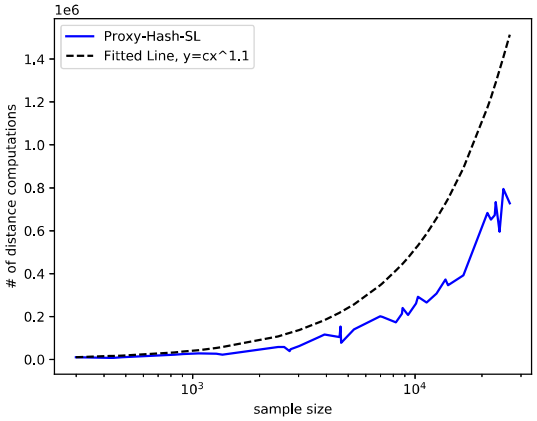


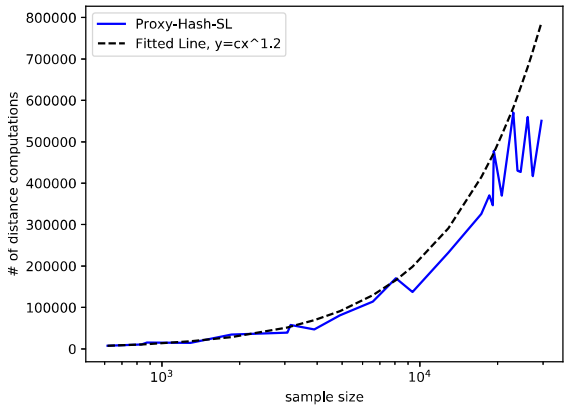
Figure 5: Growth of distance computation, Proxy-Hash-AL, the Great Lakes



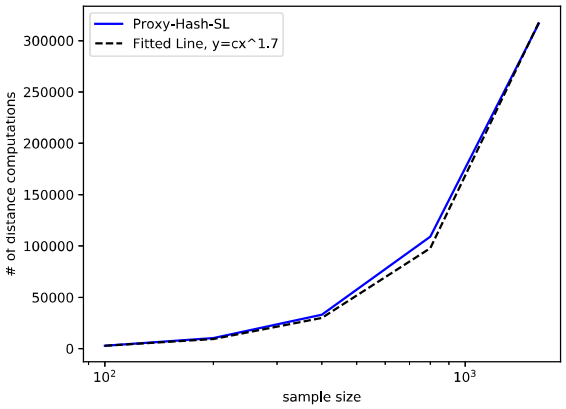
(a) New York



(b) Bay Area



(c) The Great Lakes



(d) Seizure

Figure 6: Number of distance computation, versus sample size (log), Proxy-Hash-SL

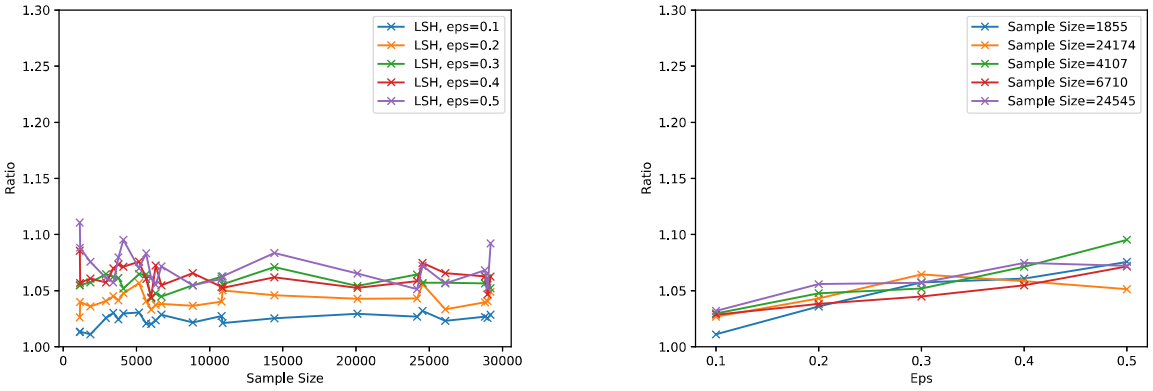


Figure 7: Approximation ratio versus sample size and ϵ value for New York

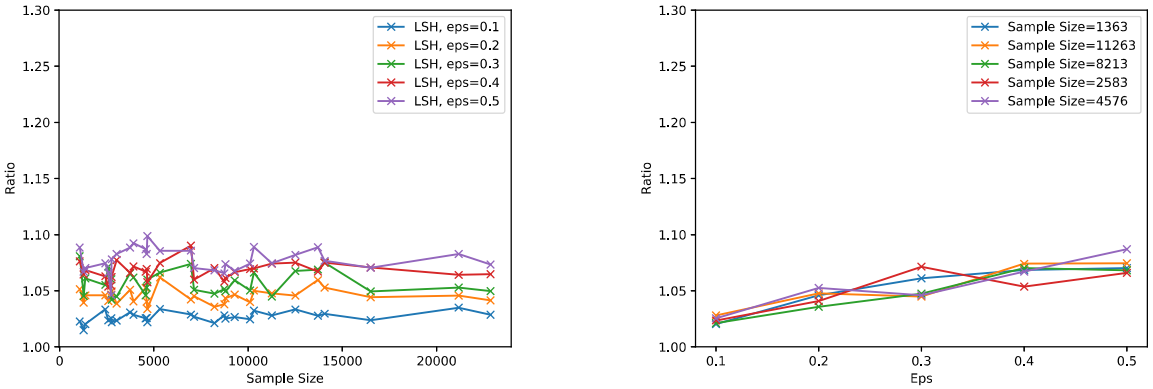


Figure 8: Approximation ratio versus sample size and ϵ value for Bay Area

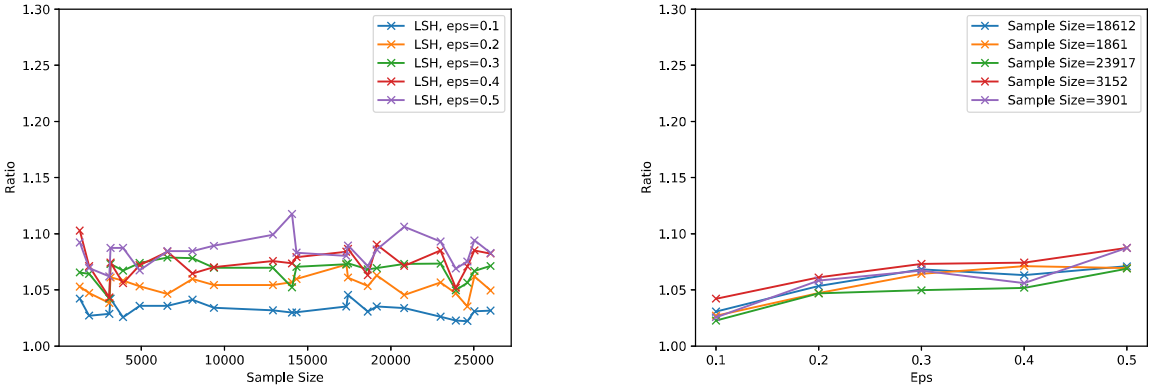


Figure 9: Approximation ratio versus sample size and ϵ value for the Great Lakes

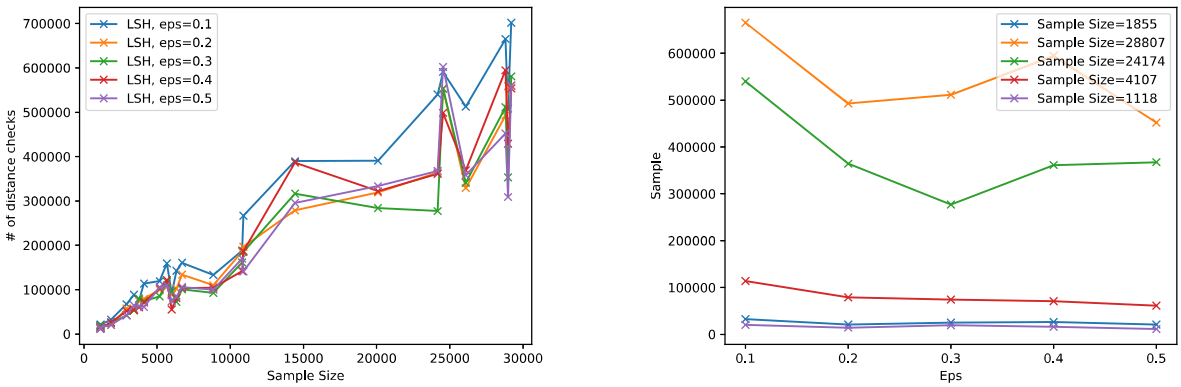


Figure 10: Number of distance computations versus sample size and ϵ for New York.

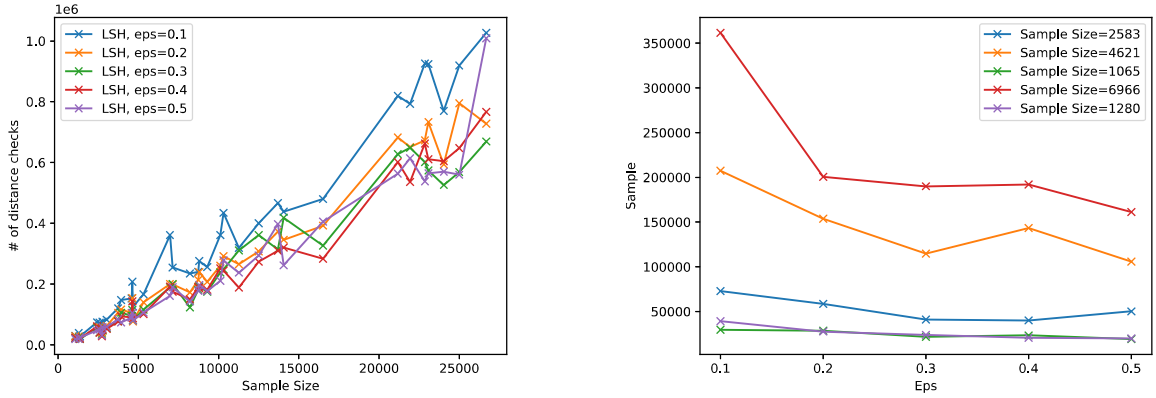


Figure 11: Number of distance computations versus sample size and ϵ for Bay Area.

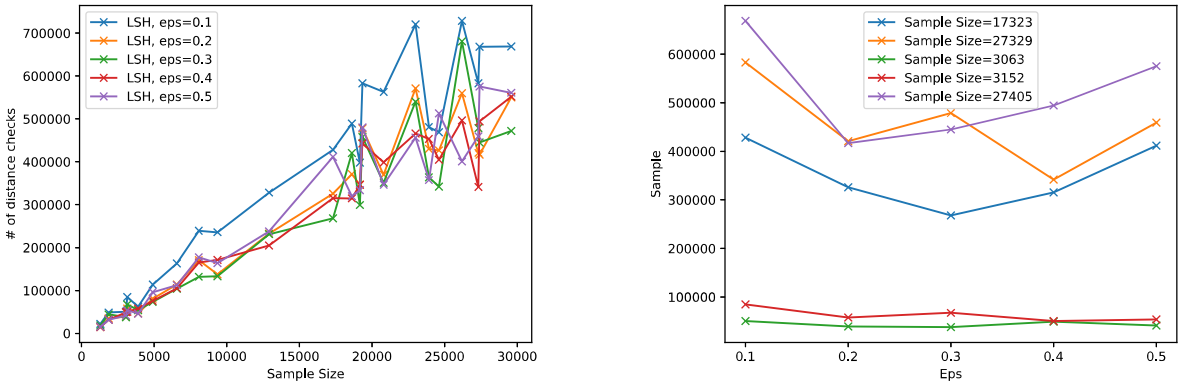


Figure 12: Number of distance computations versus sample size and ϵ for the Great Lakes