# Triple Selection for Ordinal Embedding

**Anonymous Author(s)**
Affiliation
Address
email

## Abstract

An ordinal embedding positions objects in a Euclidean space to satisfy a partial or total ordering of object distances. We present an algorithm to select a small subset of distance comparisons such that an embedding which satisfies all comparisons will recover point positions with high accuracy. The number of comparisons our algorithm uses is close to the proven lower bound of $\Omega(nd\log(n))$ for the problem, and we conjecture that on datasets with certain "nice" distributional properties it always achieves nearly-perfect embeddings within a constant factor of this lower bound. We believe it capable of finding all possible triples using just $O(n^2)$ comparisons, and provide theoretical support for this belief. We validate these results with an empirical study on real and synthetic datasets.

## 1 Introduction

An ordinal embedding based on triple comparisons aims to position a set of $n$ points into $\mathbb{R}^d$ to satisfy a set $\mathcal{C} \subset [n]^3$ of ordinal constraints, where $[n]$ denotes $\{1, \ldots, n\}$. Constraints are defined in terms of some distance metric $\delta_{i,j}$, and have the form $\delta_{i,j} < \delta_{i,k}$. An embedding $\hat{X} = \{\hat{x}_1, \ldots, \hat{x}_n\}, x_i \in \mathbb{R}^d$ is sought so that, using $\hat{d}_{a,b}$ to denote the Euclidean norm $\|\hat{x}_a - \hat{x}_b\|$,

$$\delta_{i,j} < \delta_{i,k} \in \mathcal{C} \implies \hat{d}_{i,j} \leq \hat{d}_{i,k}. \tag{1}$$

Such an embedding satisfying all $O(n^3)$ possible triples is said to be *weakly isotonic*. When an embedding also satisfies all $O(n^4)$ constraints of the form $\delta_{i,j} < \delta_{k,l}$ it is *isotonic*. Ordinal embedding is of interest when only comparative object judgements can be made (e.g., which musicians have "a more similar sound?"), or when features are available but are assumed to contain no information beyond the ordering they impose (as in ranking with clickthrough data). The embedding $\hat{X}$ can be considered a latent representation of the objects for various downstream tasks.

Kleindessner and von Luxburg [2014] proved that for sufficiently large $n$ when the points are drawn from a subset $V \subset \mathbb{R}^d$ meeting certain regularity conditions, any isotonic function $\hat{X}$ is a similarity transform of the point positions in $V$. That is, that for any $\epsilon > 0$ there is some $n_0$ such that for any $n > n_0$, there is a constant scaling factor $s$ such that

$$s\|\hat{x}_i - \hat{x}_j\| - \epsilon \leq \|x_i - x_j\| \leq s\|\hat{x}_i - \hat{x}_j\| + \epsilon. \tag{2}$$

Ordinal constraints contain no information about specific point positions or orientation (e.g. scaling, rotation, reflection, and translation of all points), so this constitutes perfect recovery of the information in $\mathcal{C}$. Loosely speaking, as $X$ grows dense in $V$ the position of each point becomes bounded arbitrarily tightly, so the individual scalings $s_{i,j} = \hat{d}_{i,j}/\delta_{i,j}$ of all pairwise distances converge to the same value. Arias-Castro [2015] proved a similar large sample convergence result for triple embedding, and also proved that in the large sample limit various subsets of $\mathcal{C}$ would suffice (for example, triples providing the total ordering of each point's $k$-nearest neighbors for suitable $k$ dependent on $n$).

Submitted to 29th Conference on Neural Information Processing Systems (NIPS 2016). Do not distribute.

We are interested in selecting the smallest possible subset of triples in $\mathcal{C}$ which can be proven to produce high-quality embeddings, assuming that an embedding can be found which satisfies as many triples as possible. We treat embedding itself as a "black box," and rely on state-of-the-art procedures discussed below. Figure 1 shows perfect embeddings for two different subsets of triples as a motivating example. The baseline algorithm performs adequately for some Machine Learning tasks, but our proposed algorithm proves much stronger. Jamieson and Nowak [2011] proved that this particular baseline takes $\Omega(n^3)$ triples to achieve comparable performance.

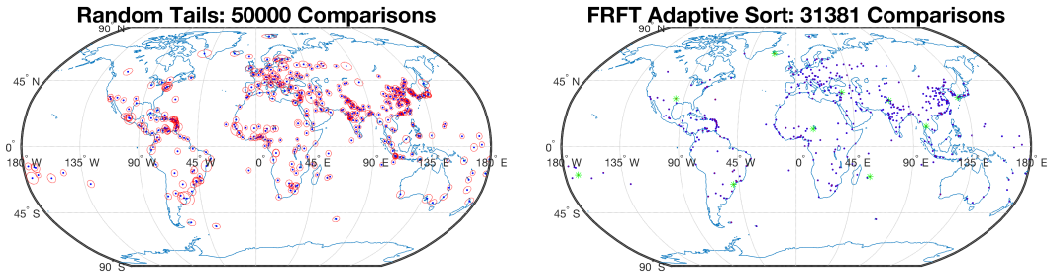


Figure 1: Embedding error comparison for two sets of triples on a 3d dataset with 500 cities of the world. The algorithms are described in Section 4 and Section 3. Circle radii show average distance error for a given city, and green asterisks denote anchors used by the FRFT algorithm. FRFT Adaptive Sort uses $3/5$ as many comparisons as the baseline and achieves a nearly-perfect embedding.

We present our algorithm in Section 2. When all triples are known in advance, our algorithm can select just $O(nd)$ triples which produce high quality embeddings. When the correct set of triples is not known *a priori*, answers to triple questions ("Is $a$ closer to $b$ or to $c$?") must be solicited from some oracle (e.g. expert assessors, A/B testing, or crowdsourcing). Jamieson and Nowak [2011] proved that at least $\Omega(nd \log n)$ triple questions must be asked adaptively from an oracle in order to reconstruct the entire set of $O(n^3)$ triples. By sorting all points by distance to each point, one obtains a trivial upper bound of $O(n^2 \log n)$ adaptive triple questions to recover all triples. Empirically, our adaptive algorithm in Section 3 can obtain all triples in only $O(n^2)$ comparisons when $d \ll n$, and we present empirical results and a theoretical argument as to why. Further, we empirically show that just $O(nd \log n)$ triples selected by our algorithm suffice to reduce the average distance error to almost zero, matching the lower bound at the cost of achieving an approximate solution. We believe that this smaller bound is tight, in agreement with the conjecture of Jamieson and Nowak [2011], but have not yet been able to prove that this is the case. Our theoretical results can be found in Section 5.

Our algorithm is easy to understand and implement. We strategically choose some *anchor* $a \in [n]$ and sort all the other points by their distances to $a$. We then use the ordinal information learned so far to choose the next anchor and repeat the process. By selecting widely-spread anchors, we rapidly learn about the ordering in all regions and dimensions of the space. For the first $O(d)$ anchors, we use $O(n \log n)$ triples per anchor to sort all points by the distance to the anchor. After $O(d)$ anchors, the current embedding gives a good partially-sorted list of all points by embedded distance to the new anchor, thus we are able to adaptively sort using only $O(n)$ triples per anchor. Thus, empirically, our algorithm needs $O(nd \log(n))$ adaptive triple questions in order to achieve a "good" embedding (matching the lower bound), and linear comparisons per anchor thereafter to improve the embedding.

## 2 A Near-Optimal Subset of Triples

In this section we present our triple selection algorithm. For now, we treat sorting as a "black box" subroutine and consider the remainder of the algorithm. We reexamine the choice of sorting algorithms in Section 3. When all triples are known in advance, one could replace the sort algorithm with a method to retrieve the $n-2$ triples needed to express the ranking of all $n-1$ other objects by distance from the current anchor. We assume here that whatever sorting algorithm is employed, it returns this correct and minimal set of $n-2$ triples.

**Building $\epsilon$-nets.** Our algorithm requires the anchors to be widely spread through $X$. To achieve this, we choose anchors which are likely to form $\epsilon$-nets. These nets are widely studied, and have

many applications in machine learning and computational geometry. We review their definition here. For any set $X$ of $n$ points in a metric space with distance function $\delta$, an $\epsilon$-net is a subset $N \subset [n]$ such that (1) for all $a, b \in N, \delta_{a,b} \geq \epsilon$, and (2) for all $b \in [n] \setminus N, \min_{a \in N} \delta_{a,b} < \epsilon$.

Exact distances are not available in our setting, so it is not clear how to build exact $\epsilon$-nets. However, we can build a good approximation using a *farthest-rank-first traversal* (FRFT) of $X$. We define a FRFT as follows. The first anchor $a \in X$ is chosen arbitrarily. Each additional anchor is then chosen at random from the set $M$ whose minimum rank from the prior anchors is maximized.

**Proposition 1.** *Any prefix of a farthest-rank-first traversal forms a good approximation of an $\epsilon$-net.*

*Proof.* (of Proposition 1)

It is well known that for any $k \in [n]$, the first $k$ members of a (farthest-first) traversal [Gonzalez, 1985] by $M' \equiv \operatorname{argmax}_{a \in [n] \setminus N} \min_{b \in N} \delta_{a,b}$ forms an $\epsilon$-net, where $\epsilon$ is $\min_{j<k} \delta_{N[k],N[j]}$.

We prove here that the set $M$ which we choose from is a superset of $M'$. Let $r$ be the max min rank in some iteration of the traversal, achieved by the members of $M$. Let $O := [n] \setminus (N \cup M)$ be the set of points which have some smaller minimum rank. Since $r$ is minimal for the members of $M$, for each $o \in O$ and $m \in M$ there must exist some corresponding $n \in N$ such that $\delta_{n,o} < \delta_{n,m}$. Thus, no member of $O$ can be a member of $M'$, and $M'$ is a subset of $M$. In particular, in any round $k$ when $|M| = 1$, the first $k$ members of $N$ forms an exact $\epsilon$-net. □

Empirically, on most of the data sets we have tested on, the size of $M$ is quite often one. Additionally, any embedding of a FRFT net which is consistent with the rankings of all net members must necessarily contain a FRFT net comprised of the same members, and whenever we know the original net was an exact $\epsilon$-net it must be the case that the embedded net is also an exact $\epsilon$-net. This already suggests that the embedding quality will be high when $N$ has sufficiently many members.

**Our algorithm.** We present FRFT Ranking as Algorithm 1, which is complete apart from treating the sort algorithm as a black box. We select anchors in FRFT order and sort points for each selected anchor. After each sort operation, we produce an embedding $\hat{x}_i$ which we use to produce a guess $\hat{r}_{N[i]}$ of the ranking for the next anchor. We terminate either when all anchors have been sorted or when our guess $\hat{r}_{N[i]}$ is sufficiently similar to the true ranking $r_{N[i]}$. We will discuss the `Disorder()` function used for this in a moment. When embedding in each round is prohibitively expensive, it is adequate to terminate when some comparison budget is exhausted or after some multiple of $d + 1$ anchors have been considered. For theoretical reasons discussed in Section 5, at least $d + 1$ anchors are necessary to obtain a good embedding. When an insufficient number of anchors has been considered, there are multiple solutions to the embedding objective which are widely divergent from each other.

**Algorithm 1:** FRFT Ranking

**Input** : The number $n$ of objects to embed, the dimensionality $d$, and the disorder tolerance $\tau$.
**Output** : The complete rankings $r$ for all members of $N$.

$N[1] \leftarrow$ random member of $[n]$ ;
$r_{N[1]} \leftarrow$ `SortForAnchor(RandomPerm(`$n-1$`),` $N[1]$`)` ;
**for** $i \leftarrow 2 : n$ **do**
  $M \leftarrow \operatorname{argmax}_{a \in [n] \setminus N} \min_{b \in N} r_b(a)$ ;
  $N[i] \leftarrow$ random member of $M$ ;
  $\hat{X} \leftarrow$ `Embed(`$r, n, d$`)` ;
  $\hat{r}_{N[i]} \leftarrow$ ranking for $N[i]$ in $\hat{X}$ ;
  $r_{N[i]} \leftarrow$ `SortForAnchor(`$\hat{r}_{N[i]}, N[i]$`)` ;
  **if** `Disorder(`$r_{N[i]}, \hat{r}_{N[i]}$`)` $< \tau$ **then**
    **return** $r$ ;
  **end**
**end**
**return** $r$ ;

Adding points to the net improves performance very rapidly; on every dataset we have attempted, $d \log(n)$ net members are more than enough to achieve a nearly-perfect embedding, provided that

the embedding algorithm can find an embedding which satisfies all triples. We discuss possible explanations of our algorithm's performance in Section 5.

There are many possible rank correlation functions which one could use for a stopping criterion. We concern ourselves in Section 3 with adaptive sort algorithms which can produce the correct ranking in $O(n)$ comparisons when the input list is adequately sorted, so we employ the disorder measure `Reg`, discussed in Petersson and Moffat [1992] and Moffat et al. [1996]. Any sort algorithm which is optimal with respect to this measure is also optimal with respect to all the other commonly-used disorder measures (e.g. the number of inversions or of monotonic runs). `Reg` is a function of the true ordering $r$ and an ordering $s$ to evaluate against $r$, where $r$ and $s$ both map object indexes in $[n]$ to their ranks. It is defined in terms of the rank distance $d_{i,j}$ in $r$ from an item $i$ to item $j$ which appears before it in $s$, and measures the degree to which items which are nearby in $s$ are also nearby in $r$. Objects are penalized when their distances in $r$ and in $s$ differ.

$$d_{i,j} = \max(r(i), r(j)) - \min(r(i), r(j)) + 1, \text{ where } s(i) > s(j) \quad (3)$$

$$\texttt{Disorder}(r, s) = \texttt{Reg}(r, s) = \prod_{i=2}^{n} \min\{t + d_{i,i-t} - 1 : 1 \leq t < i\} \quad (4)$$

The value of `Reg` ranges from 1, for sorted and reverse-sorted lists, to $O(n^{O(n)})$. Because of its large range, it is more practical to calculate $\log(\texttt{Reg})$ instead.

# 3 Efficient Sorting

We now turn to the question of efficient sorting when no triples are known in advance. We will use *triple question* to mean the question posed to some oracle, "Is $a$ closer to $b$ or to $c$?" This concept is distinct from a *triple*, asserting that "$a$ is closer to $b$ than to $c$." Sorting all points naively with respect to each possible head would require $\Theta(n^2 \log n)$ triple questions to obtain all $\Theta(n^2)$ triples required to express all possible ordinal information (via transitivity). However, when the ranking inferred from an embedding $\hat{X}$ of the triples obtained thus far is close enough to the true ranking, an adaptive sort algorithm will be able to obtain the correct ranking with just $O(n)$ questions. To sort, we produce an embedding of the previous triples, infer the ranking of all points w.r.t. a new anchor (henceforth called "the ranking for an anchor"), and pass that permutation to an adaptive sort algorithm.

Although there are many adaptive sort algorithms which take $O(n \log n)$ triple question for random lists and $O(n)$ comparisons for nearly-sorted lists, the constant factors hidden in the asymptotic analysis can make a large practical difference in the number of anchors one can visit within a given comparison budget. We experimented with several algorithms, including binary insertion sort, quicksort, natural mergesort, Neatsort [La Rocca and Cantone, 2014], Timsort [Peters, 2002], and Splaysort [Moffat et al., 1996]. We found that the minimum number of comparisons was achieved by a hybrid algorithm which runs a basic implementation of mergesort until the minimum disorder for any inferred list drops below some threshold, and then uses splaysort for all future anchors. We occasionally had problems when no good embedding could be found, even though by construction a perfect embedding existed. In these cases, it is helpful to repeat the embedding process several times and select the embedding which satisfies the most triples.

We show in Figure 2 that the value of $\log(\texttt{Reg})$ rapidly drops after the first $d + 1$ anchors. This is the empirical basis of our belief that the algorithm can find all triples in $O(n^2)$ comparisons.

# 4 Empirical Results

**Our Algorithms.** We call our algorithm "FRFT Ranking" when it emits $O(n)$ triples per head, as in the case when triples are known in advance. We call it "FRFT Adaptive Sort" when triple questions must be answered by some oracle, using $O(n \log(n))$ triples for early anchors and $O(n)$ triples later.

**Evaluation.** We show learning curves for several datasets in Figure 3 using the Soft Ordinal Embedding algorithm by Terada and von Luxburg [2014]. We embed each set of triples up to ten times with random initialization, and select the embedding with the smallest value achieved for the SOE objective. We mark with an asterisk (*) points on the plot where an embedding satisfied all the triples it was optimizing for.

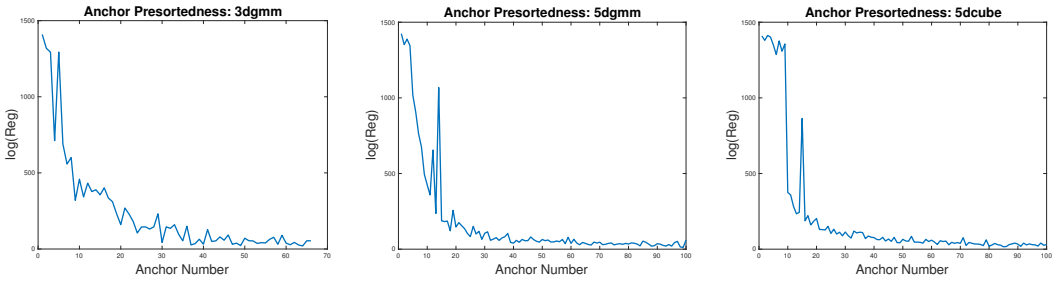


Figure 2: Disorder of predicted rankings for anchors. `Reg` drops quickly and stays low, provided a good embedding is found. The left plot is in 3 dimensions, and the other two are in 5. The large spikes correspond to embeddings which did not satisfy all provided triples. The algorithm could compensate by repeating the embedding process.

We evaluate embeddings using two measures: one based on distance scaling and one based on ranking prediction. Our first measure, *Distance RMSE* (root mean squared error), is based on the fact that in a perfect embedding all pairwise distances would be scaled by the same constant. That is, there is some $s \in \mathbb{R}$ such that for all points $i, j \in [n]$, $\delta_{i,j} \approx s\hat{d}_{i,j}$. We know the exact pairwise distances for our datasets, so we fit an optimal $\hat{s}$ to the embedding distances and report the RMSE of the residuals,

$$drmse(X, \hat{X}) \equiv \min_{\hat{s}} \left( \sum_{i<j} (\delta_{i,j} - \hat{s}\hat{d}_{i,j})^2 / n \right)^{1/2} \tag{5}$$

This value measures the average "warping" of the embedding compared to the real positions. Smaller is better, and zero is perfect.

Distance RMSE tends to be more affected by errors for larger distances, and we wish to also measure performance on the other end of the spectrum. We are also interested in the ability to predict the ranking of $X$ by distance from any point in the embedding. Our second measure achieves both. $\tau_{AP}$, introduced by Yilmaz et al. [2008] and commonly used for Information Retrieval, is a top-heavy rank correlation coefficient similar to Kendall's $\tau$, but which places more weight on correctly ranking the beginning of the list (e.g. at shorter distances). Like Kendall's $\tau$, the perfect ranking has $\tau_{AP} = 1$, a random permutation has $\tau_{AP}$ close to zero, and a reverse permutation has $\tau_{AP} = -1$. We report the mean $\tau_{AP}$ value of the rankings of all points in an embedding.

**Datasets.** We evaluate against several datasets. Due to the time it takes to embed many points in many dimensions, we were only able to provide comprehensive results for data of small dimension. However, our results suggest good performance on higher-dimensional data.
*Simulated data ($n = 500, d = 3, 5$).* We generate 500 points from Gaussian mixture models in three and five dimensions, with 10 components having random means and variance but designed with some overlap. We also test on a uniform sample from a five dimensional cube.
*Cities ($n = 500, d = 3$).* We select 500 cities from a large dataset by choosing the most populous city in each country, and then choosing additional cities in order of decreasing population. Note that the convex hull of this set consists of all 500 cities.

**Baselines.** We compare against the following baselines.
*Random Tails* iterates over all points in round-robin fashion, adding a randomly selected triple $(a, b, c) : \delta_{a,b} < \delta_{a,c}$ for each.
*kNN* also iterates over all points in round-robin fashion. In the $k^{\text{th}}$ iteration, it adds the triple $\delta_{a,b} < \delta_{a,c}$, where the ranks $r_a(b) = k$ and $r_a(c) = k + 1$. Thus, $kn$ triples express the total ordering of each point's $k$-nearest neighbors.
*Landmarks* visits points ("landmarks") in FRFT order. When each point is visited, $2n$ triples are added to insert the new point into the correct position in the rankings of all other points with respect to the previous landmarks.
*Crowd Kernel* is the authors' implementation of the "Crowd Kernel" algorithm [Tamuz et al., 2011]. This algorithm, designed for use with crowdsourcing, is an approximate active learning algorithm which selects questions aimed to minimize the expected KL divergence between prior and posterior

embeddings using a particular probabilistic model of worker responses. Instead of embedding using the authors' algorithm, we embed with SOE for comparable results.

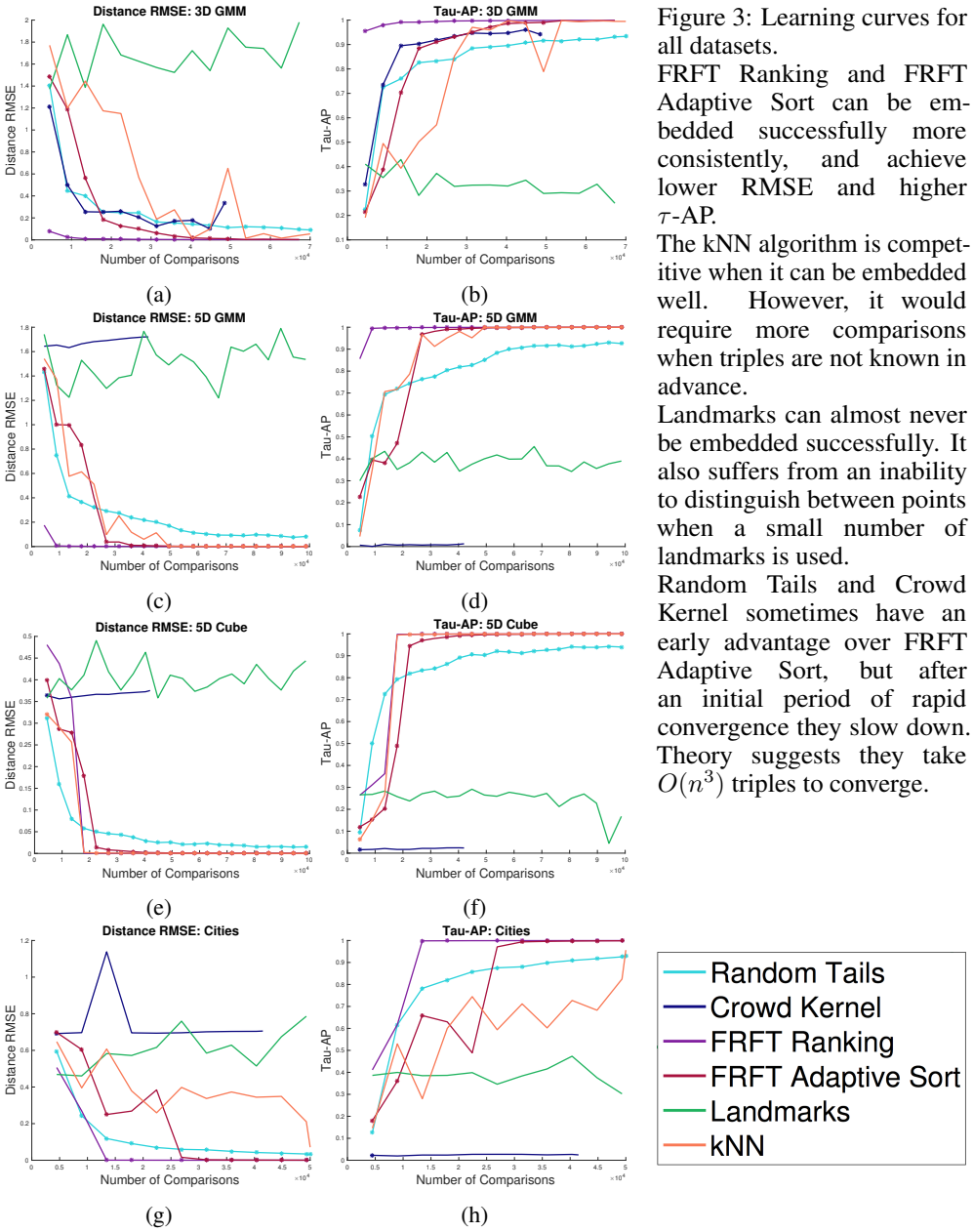


Figure 3: Learning curves for all datasets.
FRFT Ranking and FRFT Adaptive Sort can be embedded successfully more consistently, and achieve lower RMSE and higher $\tau$-AP.
The kNN algorithm is competitive when it can be embedded well. However, it would require more comparisons when triples are not known in advance.
Landmarks can almost never be embedded successfully. It also suffers from an inability to distinguish between points when a small number of landmarks is used.
Random Tails and Crowd Kernel sometimes have an early advantage over FRFT Adaptive Sort, but after an initial period of rapid convergence they slow down. Theory suggests they take $O(n^3)$ triples to converge.

## 5 Theory

In an optimal embedding of a set of points in $\mathbb{R}^d$, all pairwise distances will be scaled by the same constant. We believe that our algorithm works well because it constrains most of the distances in an embedding to be scaled by almost the same amount. While much of the prior work on ordinal embedding focuses on locating individual points, we believe that an alternative focus on constraining distance scalings to converge to the same value may lead to improved results. We argue here that our algorithm is effectively using sets of triples to construct constraints on the ratios of distance scalings between points spread throughout the embedding.

We believe the following conjecture lies at the heart of why the algorithm performs well. We state the conjecture first, and then provide some intuition. A partial proof is provided in the supplemental material, but we have not yet been able to make it rigorous. Let $s_{i,j} \equiv \hat{d}_{i,j}/\delta_{i,j}$ refer to the *scaling* of the distance between points $i$ and $j$ in a given embedding.

**Conjecture 1** (Remote Scaling)**.** *Let $N = \{a_1, \ldots, a_{d+1}\} \subset [n]$ be an $\epsilon$-net of $X$, and let $i, j \in [n] \setminus N$ be two other points in $X$. For each $k \in [d+1]$, choose some $l_k, u_k \in [n]$ that are not in $N$ such that for some small constant $c_1$, (1) $\delta_{i,j} \geq c_1 \delta_{l_k,u_k}$, (2) $i$ and $j$ are ranked after $l_k$ and before $u_k$ in $r_{a_k}$, (3) for at least $d$ members $a_k$ of $N$, $\delta_{a_k,u_k} < \epsilon$, and (4) if there is some $a_k \in N$ with $\delta_{a_k,u_k} \geq \epsilon$, then $l_k$ is ranked after all other members of $N$ in $r_{a_k}$.*

*Then in any embedding $\hat{X}$ which preserves the rankings $R(N)$, $s_{i,j} \leq \frac{2\sqrt{2}}{c_1} \max_k s_{l_k,u_k}$.*

In broad terms, the conjecture states that when the full rankings are known for an $\epsilon$-net containing $d+1$ members and an embedding $\hat{X}$ is produced consistent with those rankings, the embedded distance between any pair of points which is "bracketed" between some other pairs of points in the rankings of each anchor cannot be embedded with a much greater distance than the distances between the bracketing points. The conditions stated in the lemma are used to constrain the distance by the distances between points which are not themselves anchors. Thus, by enforcing constraints on distances which involve only the anchors, we are limiting the scalings of distances which do not involve the anchors and which are not explicitly mentioned to the embedding algorithm.

The conjecture does not typically hold when $d$ or fewer anchors are used. To see why, consider the constraints imposed by the ranking for a single anchor. If the rank $r_a(i) < r_a(j)$ for some $a, i, j \in [n]$, then $j$ must be embedded outside the $d$-dimensional sphere centered at $\hat{x}_i$ with radius $\hat{d}_{a,i}$, and $i$ must be embedded inside the corresponding sphere of radius $\hat{d}_{a,j}$. When $r_a(i) < r_a(j) < r_a(k)$, $j$ is constrained to lie outside the sphere for $i$ but inside the sphere for $k$. If this is the only constraint known, the distances from other points to $j$ are not constrained very much more than if only $r_a(j) < r_a(k)$ was known: they can vary by up to $2\hat{d}_{a,k}$. In order for distances to $j$ to be bounded by some value related to $\hat{d}_{a,k} - \hat{d}_{a,i}$, we must add additional constraints for other anchors. In fact, $d+1$ anchors are needed to adequately constrain $j$. This is not surprising: it is well known that when exact distances are known to a set of $d+1$ points which are sufficiently distinct (e.g. in general position) the position of a point $p$ can be directly obtained. Our lemma is based on a relaxation of this fact which can be applied to ordinal constraints without knowledge of exact distances.

Importantly, the bracketing points may be anywhere in the space; this constrains remote regions of the space to be scaled to roughly the same degree. We believe that when the true positions $X$ meet certain regularity conditions, any embedding consistent with these triples must have all points in roughly the correct positions. We further believe that adding more points to the net will rapidly force all points to be tightly constrained. When this happens, the predicted rankings used by our algorithm can be sorted in linear time and any embedding will have low distance error and high $\tau_{AP}$.

The rankings for new anchors depend only on distances which do not appear in the set of triples passed to the embedding algorithm in a given round. We believe that Conjecture 1 or something like it explains why `Reg` drops so quickly for these rankings.

# 6 Related Work

Ordinal Embedding, also called non-metric embedding or non-metric multidimensional scaling, has been studied by various communities for well over sixty years. Kleindessner and von Luxburg [2014] and Arias-Castro [2015] have answered long-standing questions about convergence to correctly-scaled distances in the large sample limit. These papers and their references should be consulted for more on the history of the field. Jamieson and Nowak [2011] contribute the $\Omega(nd \log n)$ lower bound on adaptive triple selection and that non-adaptive triple selection is $\Omega(n^3)$.

Adaptive triple selection seems to be less studied. Often all triples are known in advance, in which case practitioners either use them all, select a subset at random, or employ the kNN or Landmarks algorithms we use as baselines. Large-sample convergence for these latter cases was proven in the above works, and the embedding algorithms of Terada and von Luxburg [2014] are well-suited to embedding these triples. We have found their Soft Ordinal Embedding algorithm to generally produce

better results at a smaller computational cost as compared to most prior embedding algorithms. Jamieson and Nowak [2011] suggest using an embedding before each question to determine whether it can be inferred from the prior answers. This is often computationally infeasible, however, and the present work shows that it is not necessary.

The only algorithm we have found which was tailored for use in crowdsourcing is the Crowd Kernel algorithm by Tamuz et al. [2011], which we use as a baseline. This algorithm is based on a greedy approximation of the difficult problem of calculating the KL divergence of an objective function over the space of possible embeddings. While its question selection method outperforms random selection (by selecting tails closer than average to the head), its embedding method performs very poorly compared to Soft Ordinal Embedding.

The problem of embedding has been heavily studied, particularly by the metric and kernel learning communities. These communities focus on learning transformations of object features to satisfy ordinal constraints, and their methods reduce to Ordinal Embedding when an identity matrix is used in place of features. Early approaches to metric and kernel learning employed semidefinite programming [Weinberger et al., 2006, Xing et al., 2003] and/or required eigenvalue decompositions. More recent approaches have focused on minimizing the Bregman divergence [Davis et al., 2007, Kulis et al., 2009, Jain et al., 2012], which is guaranteed to find a positive semidefinite (PSD) kernel, or on ignoring semidefiniteness until convergence and then calculating a final projection of the output matrix to the nearest PSD matrix [Chechik et al., 2010]. Ordinal embedding without object features has also been studied by Agarwal et al. [2007, 2010], who provide a flexible and modular algorithm with proven convergence guarantees. McFee and Lanckriet [2011] considers how to learn a similarity function which is as consistent as possible with multiple feature sets as well as ordinal constraints.

Our algorithm reduces the problem to sorting, so when an unreliable oracle (e.g. crowdsourcing workers) is used it is natural to consider the deep literatureon crowdsourcing sort algorithms [Marcus et al., 2011, Niu et al., 2015] and on noise-tolerant sorting [Ajtai et al., 2009, Braverman and Mossel, 2008, Hadjicostas and Lakshmanan, 2011].

# 7 Discussion

We present a simple and highly competitive algorithm for selecting triples for ordinal embedding, achieving excellent performance within a constant factor of the proven lower bound. We provide theoretical results to suggest that the reason our algorithm performs well is that it implicitly constrains pairs of points at similar distances in the original space to be placed at similar distances in the embedding. We suggest that our algorithm is essentially using sets of triples to create constraints of the form $\hat{d}_{i,j} \approx \hat{d}_{k,l}$. We suspect that further focus on such constraints may lead to better embedding and triple selection algorithms in the future. We are also interested in the potential of using side information such as "$N$ is an $\epsilon$-net" directly by an embedding algorithm to impose further structure and speed convergence to a global optimum.

There are several clear paths forward for improving our algorithm. It is essential to minimize constant factors when sorting for the first $d + 1$ anchors, and further exploration here may be fruitful. The algorithm can readily be adapted to handle noisy triples by employing a sort algorithm tailored for crowdsourcing or some reasonable model of response noise, but it is likely that a sort algorithm could be tailored to make use of the previous rankings and space dimensionality information to infer answers to some triple questions. We speculate that it is possible to apply an argument using distance convexity or the triangle inequality for this purpose and to use $o(n)$ triple questions to collect rankings in later stages of the algorithm. This may provide a path to an algorithm matching the proven lower bound.

# References

Arvind Agarwal, Jeff M. Phillips, and Suresh Venkatasubramanian. Universal multi-dimensional scaling. *SIGKDD*, 2010. doi: 10.1145/1835804.1835948. URL `http://doi.acm.org/10.1145/1835804.1835948`.

Sameer Agarwal, Josh Wills, Lawrence Cayton, Gert Lanckriet, David Kriegman, and Serge Belongie. Generalized non-metric multidimensional scaling. *AISTATS*, 2007.

Miklós Ajtai, Vitaly Feldman, Avinatan Hassidim, and Jelani Nelson. Sorting and Selection with Imprecise Comparisons. *ICALP*, 2009.

Ery Arias-Castro. Some theory for ordinal embedding. *arXiv.org*, January 2015. URL http://arxiv.org/abs/1501.02861v2.

Mark Braverman and Elchanan Mossel. Noisy sorting without resampling. *SODA*, 2008. URL http://dl.acm.org/citation.cfm?id=1347082.1347112.

Gal Chechik, Varun Sharma, Uri Shalit, and Samy Bengio. Large scale online learning of image similarity through ranking. *JMLR*, 2010. URL http://dl.acm.org/citation.cfm?id=1756006.1756042.

Jason V. Davis, Brian Kulis, Prateek Jain, Suvrit Sra, and Inderjit S. Dhillon. Information-theoretic metric learning. *ICML*, 2007. URL http://doi.acm.org/10.1145/1273496.1273523.

Teofilo F Gonzalez. Clustering to minimize the maximum intercluster distance. *Theoretical Computer Science*, 38:293–306, 1985.

Petros Hadjicostas and K B Lakshmanan. Recursive merge sort with erroneous comparisons. *Discrete Applied Mathematics*, 159(14):1398–1417, August 2011.

Prateek Jain, Brian Kulis, Jason V. Davis, and Inderjit S. Dhillon. Metric and kernel learning using a linear transformation. *JMLR*, 2012. URL http://dl.acm.org/citation.cfm?id=2503308.2188402.

Kevin G Jamieson and Robert D Nowak. Low-dimensional embedding using adaptively selected ordinal data. *49th Annual Allerton Conference on Communication, Control, and Computing*, 2011. doi: 10.1109/Allerton.2011.6120287.

M Kleindessner and U von Luxburg. Uniqueness of Ordinal Embedding. *COLT*, 2014.

Brian Kulis, Mátyás A. Sustik, and Inderjit S. Dhillon. Low-rank kernel learning with bregman matrix divergences. *JMLR*, 2009. URL http://dl.acm.org/citation.cfm?id=1577069.1577082.

Marcello La Rocca and Domenico Cantone. NeatSort - A practical adaptive algorithm. *arXiv.org*, July 2014.

Adam Marcus, Eugene Wu, David Karger, Samuel Madden, and Robert Miller. Human-powered sorts and joins. *Proc. VLDB Endow.*, 2011. URL http://dx.doi.org/10.14778/2047485.2047487.

Brian McFee and Gert Lanckriet. Learning Multi-modal Similarity. *JMLR*, 12, February 2011.

Alistair Moffat, Gary Eddy, and Ola Petersson. Splaysort: Fast, Versatile, Practical. *Software Practice and Experience*, 26(7):781–797, July 1996.

Shuzi Niu, Yanyan Lan, Jiafeng Guo, Xueqi Cheng, Lei Yu, and Guoping Long. Listwise Approach for Rank Aggregation in Crowdsourcing. *WSDM*, 2015.

Tim Peters. Timsort, 2002. URL http://bugs.python.org/file4451/timsort.txt. (Visited on 2016-05-15).

Ola Petersson and Alistair Moffat. A Framework for Adaptive Sorting. *SWAT*, 1992.

Omer Tamuz, Ce Liu, Serge Belongie, Ohad Shamir, and Adam Tauman Kalai. Adaptively Learning the Crowd Kernel. *arXiv.org*, May 2011.

Yoshikazu Terada and Ulrike von Luxburg. Local ordinal embedding. *ICML*, 2014.

Kilian Q. Weinberger, John Blitzer, and Lawrence K. Saul. Distance metric learning for large margin nearest neighbor classification. *NIPS*, 2006.

Eric P. Xing, Andrew Y. Ng, Michael I. Jordan, and Stuart Russell. Distance metric learning, with application to clustering with side-information. *NIPS*, 2003.

Emine Yilmaz, Javed Alexander Aslam, and Stephen E Robertson. A new rank correlation coefficient for information retrieval. *SIGIR*, 2008.

# A Partial Proof of Conjecture 1

We first show a trivial lemma which bounds one scaling by another based on the ratios of two points' true and embedded distances.

**Lemma 1** (Scaling Ratios)**.** *Let $i, j, k, l \in [n]$ be embedded points so constrained that $\delta_{i,j} \geq c_1 \delta_{k,l}$ and $\hat{d}_{i,j} \leq c_2 \hat{d}_{k,l}$. Then $s_{i,j} \leq \frac{c_2}{c_1} s_{k,l}$.*

*Proof.* (Of Lemma 1) This follows because $s_{i,j} \equiv \frac{\hat{d}_{i,j}}{\delta_{i,j}} \leq \frac{\hat{d}_{i,j}}{c_1 \delta_{k,l}} \leq \frac{c_2 \hat{d}_{k,l}}{c_1 \delta_{k,l}} = \frac{c_2}{c_1} s_{k,l}$. □

We next explore the shape of the subspace $I$ wherein $\hat{x}_i$ and $\hat{x}_j$ are constrained to lie, defined explicitly below. Since $\hat{x}_i, \hat{x}_j \in I$, their distance can be upper bounded by the supremum distance between two points in $I$. We therefore need to show that this supremum distance is no more than some constant multiple of $\max_k \hat{d}_{l_k,u_k}$.

**Defining $I$.** For any two points $a, b \in [n]$, define ball $B(a, b)$ as the set of points in the embedding space which are at most $\hat{d}_{a,b}$ from $\hat{x}_a$: a ball centered at $\hat{x}_a$ with radius $\hat{d}_{a,b}$. Similarly, let shell $S(c, l, u) \equiv B(c, u) \setminus B(c, l)$ be the set of points of distance more than $\hat{d}_{c,l}$ but no more than $\hat{d}_{c,u}$ from $\hat{x}_c$.

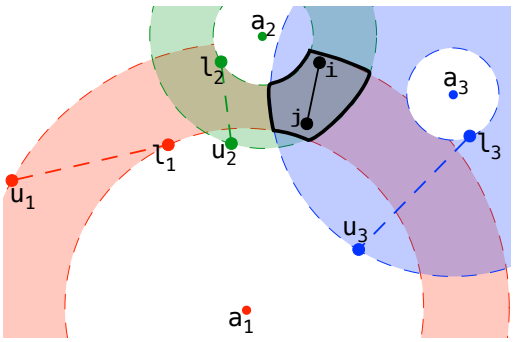


Figure 4: The intersection of three anchor shells, $I$, limits the embedding distance $\hat{d}_{i,j}$

We will call $t_k \equiv \hat{d}_{c,u} - \hat{d}_{c,l}$ the *thickness* of this shell; observe that by the triangle inequality $t_k \leq \hat{d}_{l,u}$. We will also be interested in the maximum shell thickness, $t^* := \max_k t_k$. Both $i$ and $j$ are constrained to lie within the intersection of shells

$$I \equiv \bigcap_{k \in [d+1]} S(a_k, l_k, u_k) = \left( \bigcap_{k \in [d+1]} B(a_k, u_k) \right) \setminus \left( \bigcup_{k \in [d+1]} B(a_k, l_k) \right). \tag{6}$$

The set $I$ must be non-empty because $i$ and $j$ must take positions within $I$ in order for $\hat{X}$ to satisfy all order constraints.

**$I$ is located within halfspaces formed by the members of $N$.** $I$ is constrained by (3) and (4) to lie entirely within one of the two halfspaces to one side of the hyperplane containing any subset of $d$ or fewer points of $N$, as we will show here. This implies that $I$ excludes more than half of the volume of any of its component shells.

*Case 1:* When condition (3) holds for all net members, and since $N$ is an $\epsilon$-net, all $B(a_k, u_k)$ have radii less than $\epsilon$ while their centers are separated by more than $\epsilon$. In this case, no ball includes the center of another ball, and $I$ lies entirely in the convex hull of $N$.

*Case 2:* When condition (4) holds, $I$ is within $\epsilon$ of all but one net member $a_k$. In this case, the intersection of the first $d$ balls $B(a_j, u_j)$ forms a symmetric shape with its bisecting hyperplane containing the first $d$ net members, and containing portions both inside and outside of the convex hull of $N$. For the final member $a_k$, all other net members are ranked before its lower bound point $l_k$, so we we exclude the bisecting hyperplane and the portion in the convex hull of $N$.

Since $I$ excludes more than half of each shell, it is easy to show that for all $a_k \in N$, $\hat{d}_{i,j} < 2\hat{d}_{a_k,u_k}$. This also implies that that the maximum arc $I$ can contain from any sphere is less than $\pi$ radians.

**$I$ is connected.** Although $I$ is not convex, it is a connected subspace. That is, any two points in $I$ can be connected by a curve lying entirely in $I$. To see why, observe that the non-convexity of the space is caused by subtracting the balls $B(s_k, l_k)$ from $I$. In order for $I$ to be disconnected, these balls would need to be able to intersect in such a way that different subsets of $I$ could be separated by a curve leading from one ball to another. These balls must intersect from the line connecting two net members outward, so the "negative" regions grow from the bisecting hyperplane of the union of the $B(a_k, u_k)$ balls toward the outer edges. Since $I$ includes only points on one side of this bisecting hyperplane, it must contain only a connected region.

**Distances in $I$ are bounded by its "corners."** The supremum distance between any two points in $I$ is at most the maximum distance between any two points which lie on the "corners" of $I$; that is, on the intersection of at least $d$ of the spheres which define the inner and outer boundaries of the intersecting shells. We prove this by contradiction. Suppose that $\hat{x}_i$ and $\hat{x}_j$ attain the maximal distance in $I$. Without loss of generality, assume that $\hat{x}_i$ does not lie on such a corner.

*Case 1:* If $\hat{x}_i$ is not on the surface of any boundary sphere, then it can be moved away from $\hat{x}_j$ and toward some sphere to increase the distance, contradicting that the distance was maximal.

*Case 2:* $\hat{x}_i$ is on the surface of fewer than $d$ spheres, including some $B(s_k, l_k)$. Then the sphere curves away from $\hat{x}_j$, and the distance could be increased by moving $\hat{x}_i$ along the boundary for a contradiction.

*Case 3:* $\hat{x}_i$ is on the surface of fewer than $d$ spheres, and all are $B(s_k, u_k)$ boundaries. Because $\hat{d}_{i,j} < 2\hat{d}_{s_a,u_k}$ (due to the halfspace argument above), the circle with diameter $\hat{d}_{i,j}$ curves more sharply than the boundary it rests against, and we can again increase the distance between points by moving $\hat{x}_i$ along the boundary for a contradiction.

As long as we are resting against at most $d-1$ boundaries, we can always move $\hat{x}_i$ along any one of the boundaries. Since the maximum arc one can travel along any sphere is less than $\pi$ radians, this will always move $\hat{x}_i$ further from $\hat{x}_j$. It only becomes trapped when it encounters the intersection of $d$ boundaries, because there are no more "directions" in which it could move. Because $I$ is connected, this process will lead to the supremum distance in $I$.

**The corners for some net member pairs define a rectangle on any plane intersecting $I$.** Consider any plane which intersects $I$, and the projection of any $a_j, a_k \in N$ onto that plane. For simplicity, when we refer to $a_j, a_k, l_j, l_k, u_j$, and $u_k$ in the remainder of our argument in two dimensions we mean their projections onto the plane. Define $e$ as the distance between (the planar projections) of $a_j$ and $a_k$. Let $u$ and $v$ be the distances from $a_j$ and $a_k$ to $l_j$ and $l_k$, respectively, and similarly define the shell thicknesses $t_j$ and $t_k$ based on $u_j$ and $u_k$. See Figure 5 for reference.

When $u + v > e$ (which may not always be the case), the constraints for $a_j$ and $a_k$ will intersect at four points, which we label $p, q, r$, and $s$. Let $p$ and $q$ have distance $u$ to $a_j$, and let $r$ and $s$ have distance $u + t_j$ to $a_j$. Similarly, let $q$ and $r$ have distance $v$ to $a_k$, and let $p$ and $s$ have distance $v + t_k$ to $a_k$. Note that all four points $p, q, r$ and $s$ are bounded to lie on the same side of $\overline{a_j a_k}$, so the diagram applies down to scaling and reflection.

This property does not always hold: the distances $u, v$, and $e$ are not constrained to guarantee that they satisfy the triangle inequality. When this does not hold, one can use the constraints from a third anchor (when $d \geq 2$ a third anchor will exist) and consider the partial triangle formed by all six constraints. We leave the proper analysis of this approach for future work.

The polygon with vertices $pqrs$ is a rectangle. To see why, first note that the triangles $\triangle a_j pq, \triangle a_j sr, \triangle a_k qr$, and $\triangle a_k ps$ are all isosceles. This implies that the line from $a_j$ which passes through the midpoint of $\overline{pq}$ does so at a right angle, and is thus a perpendicular bisector of $\overline{pq}$. For the same reason, the perpendicular bisector of $\overline{sr}$ goes to $a_j$, and the perpendicular bisectors of $\overline{ps}$ and $\overline{qr}$ go to $a_k$. Since the perpendicular bisectors of opposite sides of the quadrilateral $pqrs$ coincide, it is a rectangle. We will define the length of edges $\overline{pq} = \overline{sr}$ to be $x_k$, and the length of edges $\overline{ps} = \overline{qr}$ to be $x_j$. No two points in $I$ which lie on any plane containing $a_j$ and $a_k$ are more distant than $\sqrt{x_i^2 + x_j^2}$.

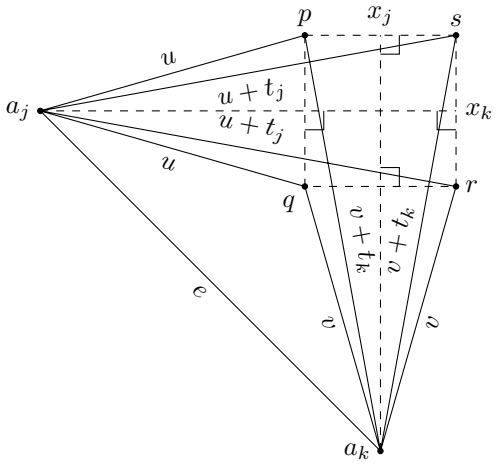


Figure 5: A plane intersecting $I$ and two net members. $pqrs$ forms a rectangle, so the maximum distance between any two corners is $\sqrt{x_j^2 + x_k^2}$.

**The rectangle side lengths are at most $2t^*$.** We will now bound the size of the larger rectangle size. Without loss of generality, suppose it is $x_k \geq x_j$. By the triangle inequality on $\triangle sa_k r$ we already have that $x_k > t_k$. With a little more work, we can prove an upper bound relating $x_k$ to $t^*$. See Figure 6 for reference.

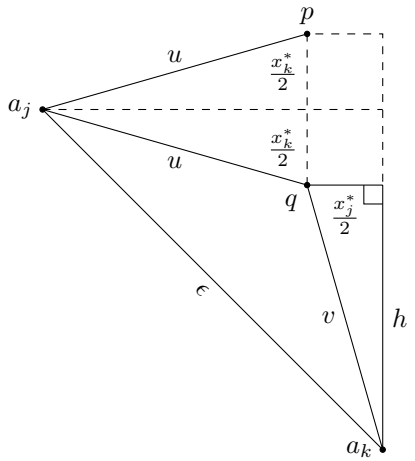


Figure 6: Rectangle $pqrs$ in a configuration which illustrates its edge bounds. We have replaced all $t_k$ with $t^*$, causing possibly-increased edges $x_k^*$.

Since $t_k \leq t^*$, we can upper bound any edge length $x_k$ by increasing the thicknesses of all shells to $t^*$ and bounding the corresponding edge length $x_k^*$ in the larger rectangle. Working in this larger rectangle, we have that $\overline{a_k p} = v + t^*$. Although $x_k^* > t^*$, by the triangle inequality $x_k^*$ is smaller than the portion of $\overline{a_k p}$ which lies inside rectangle $pqrs$. Let $h$ be the height of the line bisecting isosceles triangle $\triangle a_k qr$. Since the length of $\overline{a_k p}$ is $v + t^*$, the length of the portion lying in the rectangle is less than $t^* + (v - h)$. By the triangle inequality, we have that $v - h < x_j^*/2$, so we have that $x_k^* < t^* + (v - h) < t^* + x_j^*/2$. Since $x_k^* \geq x_j^*$, we have that $t^* > x_k^* - x_j^*/2 > x_k^*/2$, so we conclude that $2t^* > x_k^* \geq x_k$ and $2t^* > x_k^* > x_j^* \geq x_j$. Thus, for any rectangle edge, we have the following bound.

$$t_k < x_k < 2t^*, \forall k \in [d+1] \tag{7}$$

**$i$ and $j$ lie within a square with diagonal $t^*\sqrt{2}$.** Consider any plane containing $i$ and $j$. By definition, this plane intersects $I$, and by the above argument all points in $I$ which are on this plane lie within a square with edge length $2t^*$. Therefore, $\hat{d}_{i,j} < 2\sqrt{2}t^*$.

Let $k^* = \text{argmax}_k \hat{d}_{l_k,u_k}$. Since $\delta_{i,j} > c_1\delta_{l_k^*,u_k^*}$ and $\hat{d}_{i,j} < 2\sqrt{2}t^*$, we can apply Lemma 1 to conclude the result.