Alpha-Stretch Probabilistic Tree Embeddings and Applications in Physics

Physics and Computer Science Interdisciplinary Project

Sarah Clapoff

Supervisor: Valerie King, *Univeristy of Victoria*

Abstract

Alpha-Stretch Probabilistic Tree Embeddings (α-PTEs) provide an approximation for the expected pairwise distance for each node pair on a graph G, within a factor of α. They have a variety of well-known and important applications in computer science, from metrical task systems to buy-at-bulk network design. We look at the construction of PTEs, the current state of PTE algorithms, including in-detail the FRT ['04] algorithm which gives an expected stretch of O(log n). Finally, we consider applications of these algorithms in physics and astronomy. We look at the problem of processing large amounts of data from an astronomical survey, which can be modeled after a buy-at-bulk network design problem.

1 Introduction

1.1 Problem Definition

Tree embeddings have appeared in a variety of graph algorithms and have been studied in depth, as approximating a graph with a more 'simple' tree is a powerful tool in maintaining information about a particular graph property. For example, the property of connectivity in a graph can be approximated by a spanning forest, and the distance to a fixed source S can be approximated by a shortest-path tree with root S. We will be looking at the problem of approximating the *pairwise distance* between any two nodes, and we will see how a probabilistic tree embedding (PTE) can provide a solution.

For a graph G, with n nodes and m edges, an $\alpha-$stretch probabilistic tree embedding ($\alpha-$PTE) algorithm returns the expected shortest pairwise distance between any two nodes, within a *stretch* factor of α. This is done through constructing a distribution of trees, τ, such that:

1 The distances in trees "dominate" [FRT04, p.4, section 2.1]. That is, $d_T(u,v) \geq d_G(u,v)$ for all $T \epsilon \tau$

2 The expected distances in the tree selected from τ is within a factor α from the distance in G.

In other words, if we want to know the distance between two nodes in G, we can find the expected distance from the α-PTE and know it is correct up to a factor of α. This has applications in a range of problems, such as metric labeling [KT02], buy-at-bulk network design [AA97], approximate distance oracles [BGS17], and more.

The goal is to minimize the stretch α. Fakcharoenphol, Rao, and Talwar [FRT04] showed that a PTE exists with stretch O(log n) for any graph G, and this bound is asymptotically tight. We will be presenting the previous and current state of $\alpha-$stretch PTE algorithms, describing in-detail the FRT algorithm, and presenting possible applications in physics. These include modeling an astronomical data processing pipeline after the buy-at-bulk network design problem.

Probabilistic tree embedding is a type of metric embedding, in which a graph with multiple paths between node pairs is represented by a more ‘simple' tree, which has only one path between each node pair. A metric space consists of a set of points and a way to define the distance, called the *metric* or *distance function* [14]. Different metrics will define distances between the same two points differently. So it is worth noting that while we focus on graphs, α-PTEs can take any n point metric space as input.

1.2 Preliminaries

Given a connected graph $G = (V, E)$ with $|V| = n$ and $|E| = m$, and a tree embedding $T = (V_T, E_T)$, we have some preliminary definitions:

$d_G(u, v) =$ Length of the shortest path between u and v on G.
"$stretch(G, T, u, v) = \frac{d_T(u,v)}{d_G(u,v)} \geq 1$"
"$stretch(G, T) = max_{(u,v)\,\epsilon\, VxV} stretch(G, T, u, v)$"

[FGH21, p. 6, 'Dynamic Tree Embeddings'] We also have the following formal definition:

Definition: A probability distribution τ over trees T_i is an α-**Probabilistic Tree Embedding** if, for each $T_i \,\epsilon\, \tau$ and for all $u, v \,\epsilon\, V$:

(1) "$d_{T_i}(u, v) \geq d_G(u, v)$"
(2) "$\mathbb{E}_{T\sim\tau}[d_T(u, v)] \leq \alpha \cdot d_G(u, v)$"
(3) $V \subseteq V_{T_i}$

[Nägele17, p. 1, section 1.2]

Note that the PTE is the *distribution* τ, not a single tree. Furthermore, an $\alpha-$stretch PTE algorithm returns a tree *sampled* from the distribution; it does not explicitly construct the full PTE. Since the input can be any n point metric space, G can be a weighted graph, in fact [BGS17] and [FGH21] both present their algorithms for weighted graphs.

2 Related Work

2.1 Motivating Example: Tree Embedding of a n-Cycle

As a motivating example for the probabilistic nature of PTEs, consider the tree embedding of a n-cycle. Rabinovich [Rab95] showed deterministically approximating a n-cycle in a tree gives a stretch of $\alpha = \Omega(n)$. In 1989, Karp [Karp89] showed that by randomly choosing one edge to delete, a PTE can be built that gives a stretch of $\alpha = 2$.

Lemma: [Karp89] There exists a 2-stretch PTE for the n-cycle C_n.

Proof by induction:

Basis: Take any two nodes separated by $i = 1$ edges, that is $d_{C_n}(u, v) = 1$. For each edge $e = (x, y)$ in $C_n : T_e = C_n/e$ is a tree. $stretch(C_n, T_e, x, y) = \frac{n-1}{1} = n - 1$
Let $\tau = T_e, e\epsilon C_n$
Let D be the distribution over $\tau : \Pr(T_e) = \frac{1}{n}$
Take any $T \sim D$, for every $e \ \epsilon \ C_n$:

$$\mathbb{E}_{T\sim D}[d_T(x, y)] = \frac{1}{n}(n-1) + \frac{n-1}{n}(1) = \frac{2(n-1)}{n} \leq 2$$
$$d(C_n, x, y) = 1$$

Thus when $i = 1$:

$$\mathbb{E}_{T\sim D}[stretch(C_n, T, x, y)] \leq 2$$

Induction Hypothesis: Assuming $i = k$ s.t. $\mathbb{E}_{T\sim D}[stretch(C_n, T, u, v)] \leq 2$ for $d_{C_n}(u, v) = k$ is true when $k \geq 1$
Induction Step: Show that $\mathbb{E}_{T\sim D}[stretch(C_n, T, x, y)] \leq 2$ for $d_{C_n}(u, v) = k + 1$
For $i = k + 1$, we are considering any (u, v) that are separated by $k + 1$ edges. That is, $d_{C_n}(u, v) = k + 1$

$$d_{T_e}(u, v) = \begin{cases} n - (k+1) = n - 1 - k, & \text{if } e \text{ is on the shortest path between } (u, v) \\ k + 1, & \text{otherwise, the shortest path stays the same} \end{cases}$$

Let A_e be the event that the tree T_e is chosen from the distribution, where e is on the shortest path between (u, v). This occurs $k + 1$ times. Then,

$$\mathbb{E}[stretch(C_n, T_e, u, v)] = Pr(A_e)(k+1) \cdot \frac{n-1-k}{k+1} + Pr(\overline{A_e})(n-(k+1)) \cdot (1)$$
$$= \frac{(k+1)}{n}\frac{n-k-1}{k+1} + \frac{n-k-1}{n} = \frac{2(n-k)}{n} - \frac{2}{n}$$

From IH, $\frac{2(n-k)}{n} \leq 2,$ *and* $2/n > 0$

$$\frac{2(n-k)}{n} - \frac{2}{n} \leq 2$$

Thus, $\mathbb{E}_{T\sim D}[stretch(C_n, T, u, v)] \leq 2$ for $d_{C_n}(u, v) = k + 1$ is true when $k \geq 1$.
Therefore, by induction, there exists a 2-stretch PTE for the n-cycle C_n. □

Thus, the expected stretch is improved greatly (from $\Omega(n)$ to 2) when a probabilistic distribution of trees is taken, rather than deterministically.

2.2 Related Results

Results of related α-PTE algorithms can be found in Table 1.
Bartal [Bar96] presented the first formal algorithm for probabilistic metric embeddings, and also provided the following theorem:

Theorem: [Bar96] There exists a connected graph G such that if G is α-probabilistically-approximated by tree metrics, then $\alpha = \Omega(\log n)$.

The implication of this theorem is that the results of FRT are tight; we cannot expect to obtain a tighter approximation than $\alpha = \log n$. The runtime of these algorithms were all polynomial until [MS09]. Blelloch, Guh, and Sun [2017] provide an algorithm with a runtime of $O(m \log n)$, which is a substantial improvement.

For completeness we also mention the state-of-the-art work of Forester, Goranci, and Henzinger [2021]. They provide an algorithm that maintains a $O(\log^4 n)-$stretch PTE for a *dynamic* graph undergoing insertions and deletions, with an update time of $m^{1/2+O(1)}$. Maintaining dynamic $\alpha-$stretch PTEs is still very much an open problem.

expected stretch α	**runtime**	**reference**
$\mathcal{O}(\log^2 n)$	polynomial	[Bartal '96]
$\mathcal{O}(\log n \log\log n)$	polynomial	[Bartal '98]
$\mathcal{O}(\log n)$	polynomial	[FRT '04]
$\mathcal{O}(\log n)$	$\mathcal{O}(m \log^3 n)$	[Mendel Schwob '09]
$\mathcal{O}(\log n)$	$\mathcal{O}(m \log n)$	[Blelloch Guh Sun '17]
	update time	
$\mathcal{O}(\log^4 n)$	$m^{1/2+\mathcal{O}(1)}$	[Forester Goranci Henzinger '21]

Table 1: Expected stretch and runtime/update time of $\alpha-$PTE algorithms. All algorithms are for static graphs, except for [FGH21] which maintains an $O(\log^4 n)-$stretch PTE for a dynamic graph undergoing insertions and deletions.

3 The Fakcharoenphol, Rao, and Talwar [FRT04] Algorithm

In this section, we will first present the algorithm used by FRT to construct their PTE, and then the procedure used to decompose the graph at each step. While FRT present their algorithm in terms of embedding a n point metric, we present it directly from a graph instead. The procedure is still the same, in which the graph is decomposed hierarchically into a laminar family, and then converted into a tree [FRT04]. The theorem is as follows:

Theorem: [FRT04] For any graph G, there exists a PTE τ with stretch $\alpha = O(\log n)$ such that, for all $(u, v) \epsilon V$ and $T_i \epsilon \tau$:

$$d_G(u,v) \leq d_{T_i}(u,v) \tag{1}$$

$$\mathbb{E}_{T\sim\tau}[d_{T_i}(u,v)] \leq O(\log n) \cdot d_G(u,v) \tag{2}$$

[Nägele17, p.1, section 1.1]

3.1 Algorithm Preliminaries

We will first define some more preliminaries. Let $Diam(G) = D$ be the *diameter* of G, that is the longest path between any two nodes in G. For a pair of nodes (u, v), we will say they are *separated* at level i if the decomposition procedure places them in different subgroups at this level (ie. G_i is the last subgroup in which they both are members of); we can also say they have been *split* or "*cut*" [FRT04, p. 7].

The decomposition algorithm relies on building *balls*, which are defined as:
"$B(u, r) = \{n \epsilon V | d(n, u) < r\}$" [15], And are the collection of all nodes within radius r, centered around u .

The trees constructed by this algorithm are *"hierarchical well separated,"* [FRT04, p. 8, section 2.4] the edges connecting each level decrease by a factor of 2. This is achieved through the hierarchical cut procedure, which creates a laminar family that is then converted to a tree. A *laminar family* is a "family of subsets of V such that for any two subsets, $A, B :\ A \subset B$ or $B \subset A$, or $A \cap B = \emptyset$." [FRT04, p. 4, footnote] At each level, the decomposition procedure splits the graph into a subgroups of diameter $\leq D/2$, which is then converted into a tree. Since the total number of divisions (and thus the height of the tree) is bound by $O(\log n)$, the upper bound on the stretch is also $O(\log n)$ (See Figure 1b) [FRT04].

3.2 PTE Construction Algorithm

Given our graph $G = (V, E)$ with diameter D, the procedure given by FRT to construct the PTE *iteratively* decomposes G into disjoint subgraphs $G_1, G_2, ..., G_\ell$. It uses a randomized decomposition procedure, which will be presented in the next section. The randomized procedure has the following properties, as summarized (along with the algorithm) in Martin Nägele's scribed lecture notes [Nägel17]:

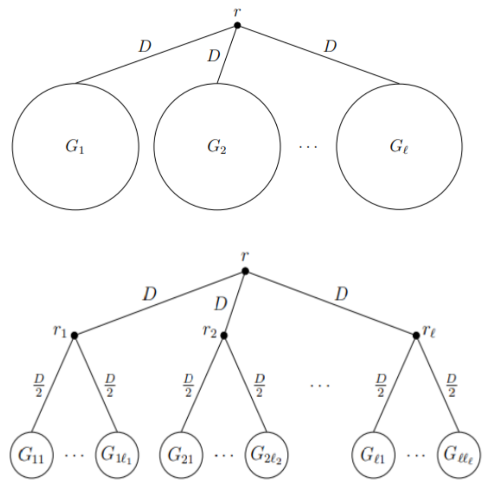
(a) "for all $i \epsilon [\ell]$, $\text{diam}(G_i) \leq \frac{D}{2}$"
(b) "for all $u, v \epsilon V$, the probability that u and v are assigned to different parts G_i is at most $\frac{d_G(u,v)}{D} \cdot \alpha$"

[Nägele17, p.1, section 1.2]

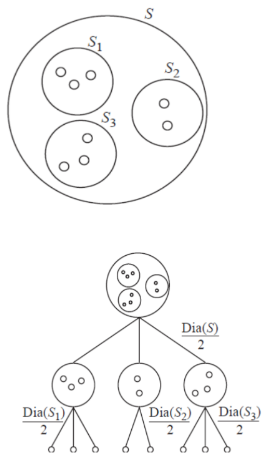
The algorithm proceeds as follows (See Figure 1):

`ConstructPTE(`G, D`): [Nägel17]`

```
1: Decompose $G$ into subgraphs $G_1, ..., G_\ell$
2: Introduce root $r$ and connect to subgraphs by edges with weight $D$
3: For $i \epsilon \ell$ : call ConstructPTE($G_i$, $Diam(G_i)$): Decompose each $G_i$ into subgraphs and
   connect to $r_i$ with edges of weight $Diam(G_i)$.
4: Continue until $G$ has been fully decomposed into single nodes.
```



(a) Two steps of ConstructPTE(G, D) [Nägele17, p. 2, Figure 1]


S
S_1
S_2
S_3
$\frac{\text{Dia}(S)}{2}$
$\frac{\text{Dia}(S_1)}{2}$
$\frac{\text{Dia}(S_2)}{2}$
$\frac{\text{Dia}(S_3)}{2}$

(b) Converting a laminar family into a tree [FRT04, p. 5, Fig. 1]

Figure 1: Two visualizations of the FRT $\alpha - PTE$ construction.

In other words, G is partitioned into subgroups $G_1, G_2, ...G_\ell$ according to the random decomposition algorithm, and each of these subgraphs are attached to a new root node (not in G) r, by edges of weight D. Now, the diameter of each G_i is *at most* $D/2$. Then, the process is repeated to each subgraph: G_1 is partitioned into $G_{11}, G_{12}, ..., G_{1\ell_1}$ and connected to a new node r_1 with edges of weight $diam(G_1)/2$, G_2 is partitioned into $G_{12}, G_{22}, ..., G_{2\ell_2}$ and connected to a new node r_2 with weight $diam(G_2)/2$, and so on. Since the diameters of $G_1, G_2, ..., G_\ell$ are at most $D/2$, the edge weights and diameters of G_{ij} are both *at most* $D/4$. At each step i the diameter of the subgraphs is at most $Diam(G_i)/2$, and the weights connecting the subgraphs are $Diam(G_i)$. This continues until G has been fully partitioned into individual nodes, at which point the leaves of T are exactly the nodes of G. In the decomposition procedure, the partitioning of subgraphs is randomized, and so T is a random tree produced as part of the distribution τ.

3.3 Proof of property (1) of PTEs:

We will now show that this algorithm satisfies both properties of the PTE definition.
Recall the first property:

$$\forall u, v \,\epsilon V,\ \forall T_i \,\epsilon\, \tau : \quad d_G(u, v) \leq d_{T_i}(u, v) \tag{3}$$

[Nägele17, p.1, section 1.1]

Proof by induction:
Basis: u, v are separated at the root/level 0 (Figure 1a, top).
This means that they belong to different partitions after the first step, and thus different subtrees in T, T_u, T_v. Let $h(T_i)$ = the height of tree T_i. The distance in T :
$d_T(u,v) = h(T_u) + D + h(T_v) + D$
$d_T = 2D + h(T_u) + h(T_v), \quad h(T_u), h(T_v) \geq 0$
$d_T(u,v) \geq 2D$
By definition, $d_G(u,v) \leq D$
$\therefore d_T(u,v) \geq d_G(u,v)$ if u, v are separated at the root/level 0.

Induction Hypothesis: Assume that $d_T(u,v) \geq d_G(u,v)$ when u, v are separated at level $i, i \geq 0$. At this level, by property (a), G has been partitioned i times, and each subgraph has a diameter of $\frac{D}{2^i}$ at most. Thus, u, v belong to the same partition of G that has diameter $\frac{D}{2^i}$. The weights on T connecting level i and $i+1$ are $\frac{D}{2^i}$.

Induction Step: Assuming $d_T(u,v) \geq d_G(u,v)$ when u, v are separated at level i, show $d_T(u,v) \geq d_G(u,v)$ if u, v are separated at level $i+1$, when $i \geq 0$.

u, v belong to the same partition of G with diameter $\frac{D}{2^{i+1}}$, thus at most they are separated by $\frac{D}{2^{i+1}}$. That is,
$d_G(u,v) \leq \frac{D}{2^{i+1}}$
The weights connecting the subtrees containing u and v, T_u and T_v, have weights of $\frac{D}{2^{i+1}}$. So,
$d_T(u,v) = \frac{D}{2^{i+1}} + h(T_u) + \frac{D}{2^{i+1}} + h(T_v)$
$d_T(u,v) = 2\frac{D}{2^{i+1}} + h(T_u) + h(T_v), \quad h(T_u), h(T_v) \geq 0$
$d_T(u,v) \geq \frac{D}{2^i}$

$\frac{D}{2^{i+1}} \leq \frac{D}{2^1}, \quad$ and $d_G(u,v) \leq \frac{D}{2^{i+1}}$, $d_T(u,v) \geq \frac{D}{2^i}$
$\therefore d_G(u,v) \leq d_T(u,v)$

Conclusion: By induction, $d_G(u,v) \leq d_T(u,v)$ when u, v are separated at all levels. Thus, $d_G(u,v) \leq d_T(u,v)$ for all $u, v \,\epsilon V$.

Further, the procedure for tree construction is the same for each tree $T_1, ..., T_k$ in τ so that: $\forall u, v \,\epsilon V$, $\forall\, i \,\epsilon [k] : d_G(u,v) \leq d_{T_k}(u,v)$ □

3.4 Proof of Property (2) of PTEs:

Recall Property (2):

$$\forall u, v \,\epsilon V : \mathbf{E}_{T \sim \tau}[d_T(u,v)] \leq O(\log n) \cdot d_G(u,v)$$

[Nägele17, p.1, section 1.1]

Proof: Let A_i be the event that (u,v) are separated at level i on T [Nägel17].
Note the height of T is bound by $D + \frac{D}{2} + \frac{D}{4} + ... \leq 2D$, thus $Diam(T) \leq 4D$ [Nägel17]. Further, for (u,v) separated at level i, they belong to the same subtree rooted at i, with

diameter $\frac{D}{2^i}$. So for (u, v) separated at level i, $d_T^i(u, v) \leq \frac{D}{2^i}$.
From property (b) of the decomposition procedure,
$Pr(u, v$ are assigned to different subgroups$) \leq \frac{d_G(u,v)}{D} \cdot O(\log n)$
For (u, v) separated at level i, they belong to the same subgroup G_i with $Diam(G_i) \leq \frac{D}{2^i}$.
Thus, $Pr(A_i) = \frac{d_G(u,v)}{D/2^i} \cdot O(\log n)$
Then, the expected distance on T becomes

$$\mathbb{E}_T[d_T(u, v)] = Pr(A_0) \cdot d_T^0(u, v) + Pr(A_1) \cdot d_T^1(u, v) + ... = \sum_i \frac{\alpha \cdot d_G(u, v)}{D/2^i} d_T^i(u, v)$$

$$\leq \sum_i \frac{\alpha \cdot d_G(u, v)}{D/2^i} \frac{D}{2^i} = 4\alpha \cdot d_G(u, v)$$

For $\alpha = O(\log n)$:

$$\mathbb{E}_T[d_T(u, v)] \leq d_G(u, v)\ O(\log n)$$

□

3.5 Randomized Graph Decomposition

Now, we focus on the decomposition algorithm used to create the subgroups G_i in *each step* of the iterative algorithm above. FRT [04] base their algorithm on the decomposition procedure of Carlinescu et. al [CKR01]. This algorithm is as follows, as summarized by Martin Nägele's scribe notes [Nägel17]:

```
GraphDecomposition($G$): [CKR01]
1: "Choose a radius $\theta$ at random on the interval $[\frac{D}{8}, \frac{D}{4}]$."
2: "Choose a random permutation, $\pi$ of all nodes, $v_1, ..., v_n$."
3: "Create the subgroup $G_i$ by taking the set of nodes $B(\pi(i), \theta) \setminus \bigcup_{j<i} B(\pi(j), \theta)$. The $i^{th}$
   ball is still built, even if $\pi(i)$ is already included" (because it hasn't yet been the center
   of a ball).
```

[Nägele17, p. 2, 'Randomised Graph Decomposition']

In other words, to divide each subgroup into $G_1, ..., G_\ell$, first the nodes within that subgroup are randomly partitioned into $\pi = v_1, ..., v_n$ and a radius θ is randomly selected between $[\frac{D}{8}, \frac{D}{4}$. Then, for each node i in π, the subgroup G_i is constructed of all nodes within radius θ of $\pi(i)$. Subtracted from this is all balls of radius θ that have a center which has come before i in the permutation (ie. $\pi(j)$, $j < i$), which gets rid of already assigned nodes. The i^{th} ball is still built, even if $\pi(i)$ already belongs to a ball, so that nodes will be partitioned individually.

For example, the first subgroup G_1 is built from the ball $B(\pi(1), \theta)$. Then G_2 is built by creating $B(\pi(2), \theta)$, if $\pi(1)$ is inside this ball, then the entirety of G_1 is removed from G_2.

To show property (a) of the FRT algorithm is satisfied: the radius θ is at most $\frac{D}{4}$, so the diameter is at most $\frac{D}{2}$ [Nägel17].
To show property (b), we want to create an upper limit on the probability that u, v are not assigned to the same subgroup G_i. Consider the ball $B(u,r)$ as this upper bound; if all nodes within the ball have not assigned to the same subgroup at level i, we say the ball has been "*cut*" [Nägele17, p.2] (and therefore u, v, which both belong to the ball, will be assigned to different components).
There is some node, w, $\pi(i) = w$ that cuts $B(u,r)$ at level i. That is, the component G_i cuts $B(u,r)$ into two pieces (we can say without loss of generality that u and v belong to the two pieces) and it is the first to do so (otherwise $B(u,r)$ would not contain both u, v) [Nägel17].
In other words:

(a) "$G_j \cap B(u,r) = \emptyset$ for all $j < i$" [Nägele17, p. 3]
(b) $G_i \cap B(u,r) \neq \emptyset$ and $B(u,r) \nsubseteq G_i$

Now, we will show that property (b) holds, following the proof of Nägele's scribe notes [Nägel17]:

Theorem: for all $u, v \epsilon V$, Pr$[u, v$ are assigned to different parts in $G_i] \leq \frac{d_G(u,v)}{D} \cdot O(logn)$.
Proof: Take $v_1, v_2, ...$ to be the nodes of G increasingly sorted by their distance to u. Then,

$$Pr[B(u,r) \text{ is cut}] = \sum_{i=1}^{n} \Pr[B(u,r) \text{ is cut by } v_i]$$

[Nägele17, p.3, Eq. (4)]

For $B(u,r)$ to be cut by v_i, the radius θ must be in the interval $(d(v_i,u) - r, d(v_i,u) + r)$, which has size $2r$. The probability of this is $\frac{2r}{D\backslash 8}$ at most (the $2r$ comes from the size of the interval, and $D\backslash 8$ comes from (1.) in the algorithm). Additionally, all nodes $v_1, ..., v_{i-1}$ must come after v_i in π (these are closer to u than v_i, and $B(u,r)$ will have already been cut if they came before). The probability of this is $\frac{1}{i}$ (because the probability of choosing v_i out of i possible values is $1/i$). The total probability is then

$$Pr[B(u,r) \text{ is cut}] \leq \sum_{i=1}^{n} \frac{2r}{D\backslash 8} \frac{1}{i} \leq \frac{16r \cdot logn}{D}$$

[Nägele17, p.3, Eq. (5)]

Where $\log(n)$ comes from the identity $\sum_{i=1}^{n} \frac{1}{i} = logn$.
For $r = d_G(u,v)$ (which is the case for u,v belonging to farthest points in $B(u,r)$), this becomes $\frac{16 d_G(u,v)}{D} \cdot logn = \frac{d_G(u,v)}{D} \cdot O(logn)$.
As stated above, if the ball containing u, v is cut, then u and v belong to different components.
Thus, for all nodes $u, v \epsilon V$:

$$\Pr[\ u, v \text{ belong to different parts in } G_i] \leq \frac{d_G(u,v)}{D} \cdot O(logn)$$

□

So, this graph decomposition procedure supports the properties stated by FRT. In sections 3.3 and 3.4, we also showed that the FRT algorithm produces a PTE for any graph G. Thus, we have summarized a procedure to randomly hierarchically decompose a graph G and construct an $\alpha-$PTE with stretch $\alpha = O(\log n)$.

4 Applications in Physics and Astronomy

For any instance where we want approximations of pairwise shortest distances for all nodes in a network, we can apply an $\alpha-$PTE. In this section, we present an application of the FRT $\alpha-$PTE algorithm on processing large amounts of data from astronomical surveys, modeled after the buy-at-bulk network problem.

In computer science, common applications include metric labelling problems [KT02], group Steiner tree problems [GKR00], vehicle routing [CCGG98], approximate distance oracles [BGS17], and the buy-at-bulk network problems [AA97][FGH21]. Many problems in Physics and astronomy can be modeled to fit these already-known applications. For example, an approximate distance oracle (ADO) is a compact data structure that answers pairwise distance queries, and PTEs can be used to speed up ADO construction [BGS17]. An interesting application of ADOs could be in the simulations of systems such as galaxy mergers, and the interactions of physical bodies with one another. We instead choose to focus on the buy-at-bulk network design application.

4.1 Buy-At-Bulk Network Design: Astronomical Data Processing

Astronomical surveys collect large amounts of data, which requires processing before releasing it to the public for scientific research. For example, consider the Rubin Observatory's Legacy Survey of Space and Time (LSST). This survey, scheduled to start in 2025, will collect 20 terabytes per night for 10 years, all of which will need to be processed at different times and in different ways [LSST]. Following the ideas of [AA97] and [FGH21], we can model the large data processing network after the buy-at-bulk network design problem in order to give an O(log$n$) approximation of the fastest overall processing route.

The buy-at-bulk network design problem from Awerbuch and Azar [1997] consists of a weighted, undirected graph G with source-sink pairs of nodes. Each pair has a demand associated with it. Additionally, there is a price function for any edge in G, which gives the cost for purchasing some quantity on that edge [FGH21]. This function is non-decreasing and sub-additive, so that purchasing the sum of two units costs less than purchasing the two units individually (ie. $f(x+y) \leq f(x) + f(y)$). Solutions are "collections of paths $P_1, P_2, ..., P_k$, where P_i is a path from s_i to s_t, routing the $dem(i)$ units of commodity" [FGH21, p. 1, section 1]. When an $\alpha-$PTE algorithm is applied, it returns an approximation to the solution which minimizes the total cost of routing the total demand through all pairs [FGH21].

We can model the data processing network of the LSST after this problem as follows. The data management network is represented as **G=(V,E,ℓ)**, a network of $|V| = n$ machines sending and receiving data in different stages of being processed. The **edges**, $e \epsilon E$, represent different parts of the overall processing procedure, with lengths ℓ (how long it will take to move between different processing steps). There are k sender-receiver node pairs, the **demand** $dem(i)$ of each pair (s_i, t_i) is the amount of data that they need to be processed. The **cost function** $f(u)$ is the time required to process one unit u. An **acceptable solution** is a set of paths $P_1, ...P_k$, where P_i is a path that processes the amount of data demanded by (s_i, t_i). For each solution, "$c_e = \sum_{i:e\epsilon P_i} dem(i)$" [FGH21, p. 25, section 5.2] is the cost per edge, which is the total amount of data processed using the section e of the overall processing procedure [FGH21]. The goal is finding the solution that minimizes the total cost for paths between all pairs, resulting in the fastest time. When G is represented as a tree, solving the buy-at-bulk network problem is much faster [AA97]. Thus, we can apply the FRT algorithm to built a tree embedding of the network, which will then give an approximate solution within a factor of O(log$n$).

Further work could also be done to model this data processing network after the open problem of dynamic buy-at-bulk network design presented by Forester, Goranci, and Henzinger [2021]. In this case, the addition and removal of processing sections e would reflect the amount of processing resources and time available.

5 Conclusion

We have presented $\alpha-$Stretch Probabilistic Tree Embeddings, past and present work, and applications in physics and astronomy. We presented the Fakcharoenphol, Rao, and Talwar [2004] algorithm, which provides an expected stretch of $\alpha = O(\log n)$. [BGS17] showed that the runtime can be improved to O(m log n), and [FGH21] have presented a dynamic algorithm with expected stretch $\alpha = O(\log^4 n)$ and update time $O(m^{1/2+O(1)})$, and these are open problems. The O(log$n$)-stretch result of FRT is asymptotically tight though, and cannot be improved.

6 References

References

[AA97] Baruch Awerbuch and Yossi Azar. Buy-at-bulk network design. In Proc. of the Sympo- sium on Foundations of Computer Science (FOCS), pages 542–547, 1997.

[Bar96] Y. Bartal. Probabilistic approximation of metric spaces and its algorithmic applications. In Proc. of the Symp. on Found. of Comp. Sci. (FOCS), pages 184–, 1996.

[BGS17] Guy E. Blelloch, Yan Gu, and Yihan Sun. Efficient construction of probabilistic tree embeddings. In Proc. of the International Colloquium on Automata, Languages, and Programming (ICALP), pages 26:1–26:14, 2017.

[CCGG98] Moses Charikar, Chandra Chekuri, Ashish Goel, and Sudipto Guha. Rounding via trees: Deterministic approximation algorithms for group steiner trees and k-median. In Proc. of the Symposium on the Theory of Computing (STOC), pages 114–123, 1998.

[CKR01] Gruia Calinescu, Howard Karloff, and Yuval Rabani. Approximation algorithms for the 0- extension problem. In Proceedings of the Twelfth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA '01, pages 8–16, Philadelphia, PA, USA, 2001. Society for Industrial and Applied Mathematics.

[FGH21] Forster, S., Goranci, G., and Henzinger, M. Dynamic maintenance of low-stretch probabilistic tree embeddings with applications. In Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms (SODA) (pp. 1226-1245). Society for Industrial and Applied Mathematics, 2021.

[FRT04] Jittat Fakcharoenphol, Satish Rao, and Kunal Talwar. A tight bound on approximating arbitrary metrics by tree metrics. In Proc. of the Symp. on Theory of Comp. (STOC), pages 448–455, 2003. `https://www.sciencedirect.com/science/article/pii/S0022000004000637`

[GKR00] Naveen Garg, Goran Konjevod, and R. Ravi. A polylogarithmic approximation algorithm for the group steiner tree problem. J. Algorithms, 37(1):66–84, 2000.

[Karp89] Richard Karp. A 2k-competitive algorithm for the circle, Manuscript, August 1989.

[KT02] Jon M. Kleinberg and Éva Tardos. Approximation algorithms for classification problems with pairwise relationships: metric labeling and markov random fields. J. ACM, 49(5):616–639, 2002.

[MS09] Manor Mendel and Chaya Schwob. Fast C-K-R Partitions of Sparse Graphs. In Chicago Journal of Theoretical Computer Science, number 2, volume 2009. University of Chicago, July 2009.

[Rab95] Rabinovich, Y., Raz, R. Lower Bounds on the Distortion of Embedding Finite Metric Spaces in Graphs. Discrete Comput Geom 19, 79–94 (1998). `https://doi.org/10.1007/PL00009336`

[Nägel17] Nagele, M. Scribe Notes, `https://people.inf.ethz.ch/gmohsen/AA19/Notes/S7.pdf`

[14] Wikipedia contributors, "Metric space," Wikipedia, The Free Encyclopedia, `https://en.wikipedia.org/w/index.php?title=Metric_space&oldid=1187360893` (accessed December 20, 2023).

[15] Wikipedia contributors, "Ball (mathematics)," Wikipedia, The Free Encyclopedia, `https://en.wikipedia.org/w/index.php?title=Ball_(mathematics)&oldid=1188592058` (accessed December 20, 2023).

[LSST] Large Synoptic Survey Telescope. "Data Management." Rubin Observatory, April 14, 2020. `https://www.lsst.org/about/dm`