

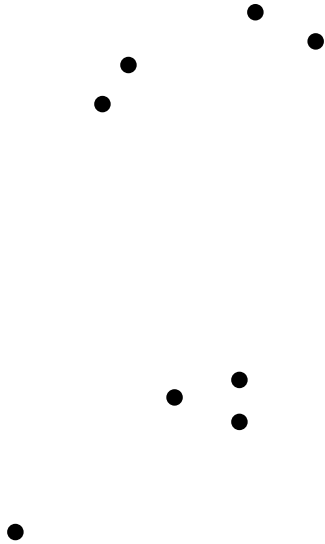
On Efficient Low Distortion Ultrametric Embedding

Vincent Cohen-Addad -- CNRS & Google Zürich

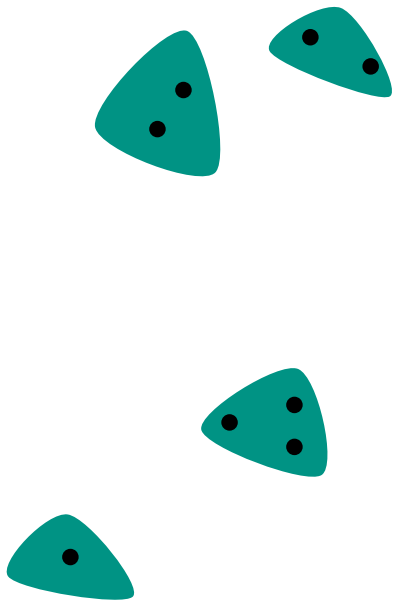
Karthik C. S. -- Tel Aviv University

Guillaume Lagarde -- LaBRI

“Flat” clustering

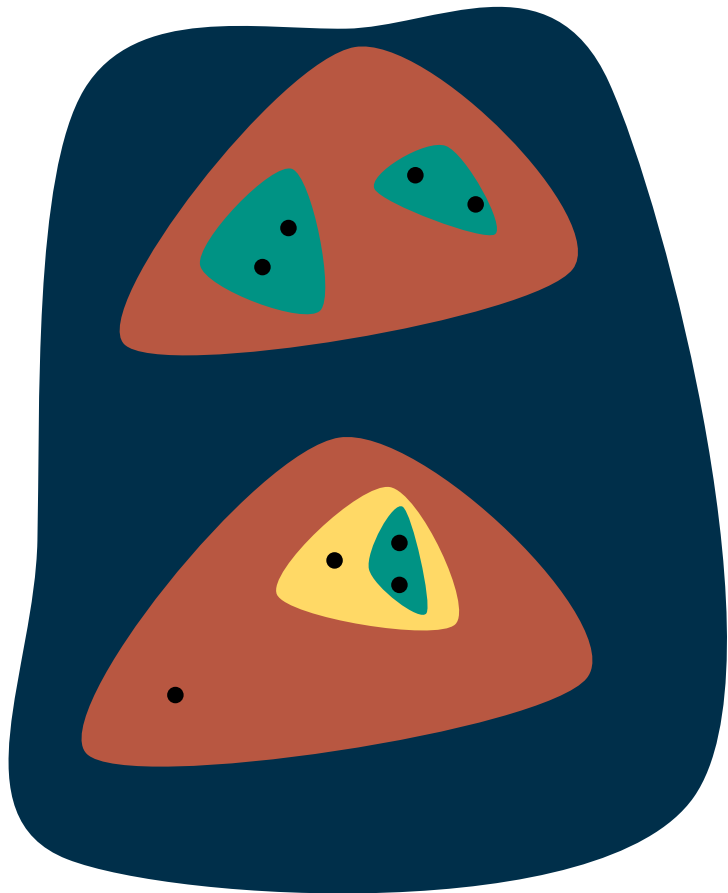


“Flat” clustering

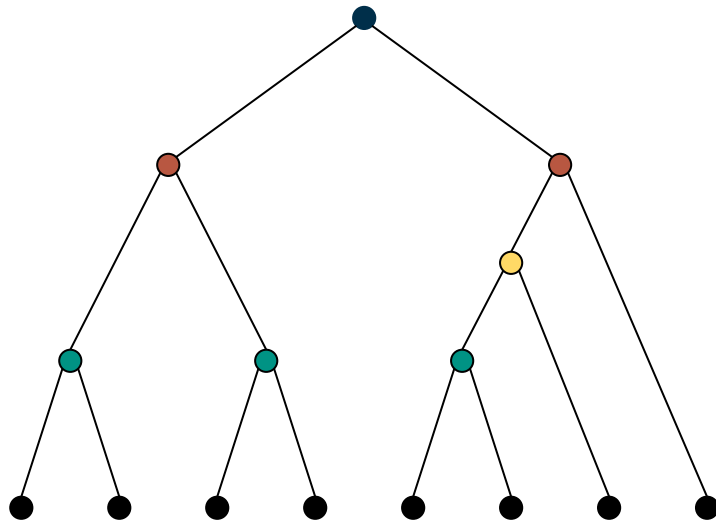


- Cluster analysis
- Features for machine learning
- Data compression
- etc.

Hierarchical clustering



- Recursive partitioning of the data
- n points $\rightarrow 2n-1$ nested clusters with different granularities



Ultrametric

Ultrametric

Metric where:

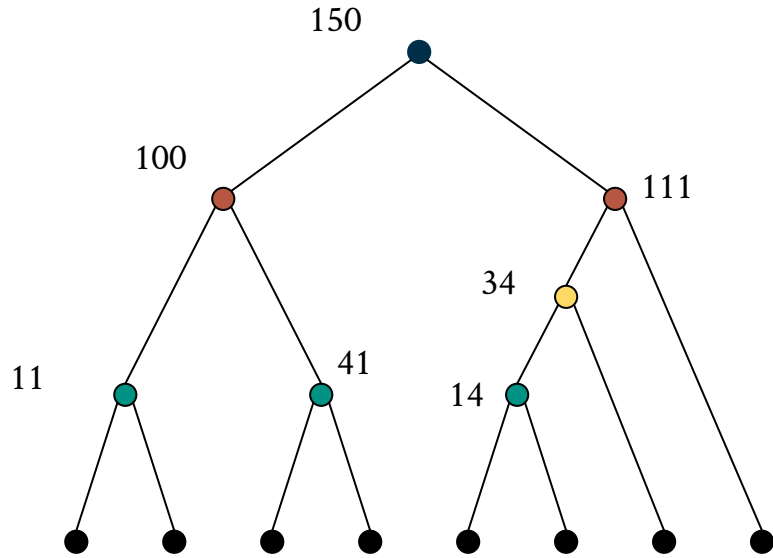
~~Triangle inequality
 $d(x,z) \leq d(x,y) + d(y,z)$~~

is strengthened to

Ultrametric inequality

$$d(x,z) \leq \max(d(x,y), d(y,z))$$

- Recursive partitioning of the data
- n points $\rightarrow 2n-1$ nested clusters with different granularities



Ultrametric

Ultrametric

Metric where:

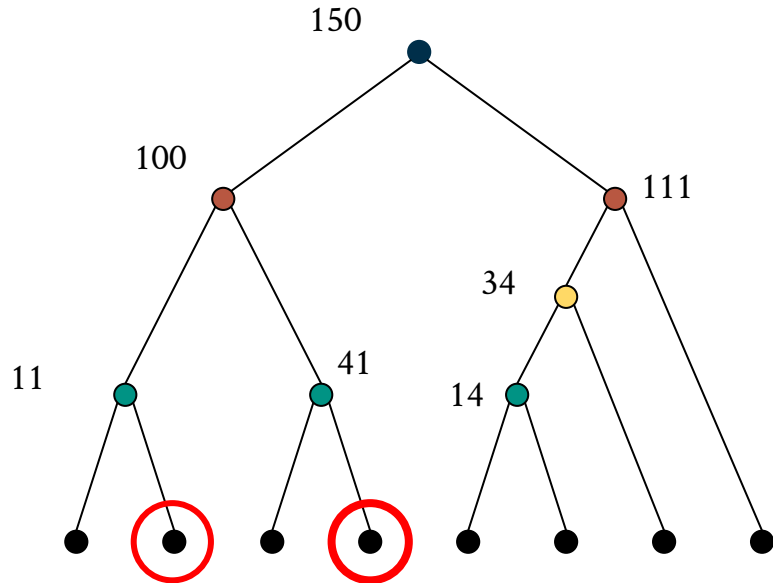
~~Triangle inequality
 $d(x,z) \leq d(x,y) + d(y,z)$~~

is strengthened to

Ultrametric inequality

$$d(x,z) \leq \max(d(x,y), d(y,z))$$

- Recursive partitioning of the data
- n points $\rightarrow 2n-1$ nested clusters with different granularities



$d(x,y)$ = value of the lowest common ancestor

Ultrametric

Ultrametric

Metric where:

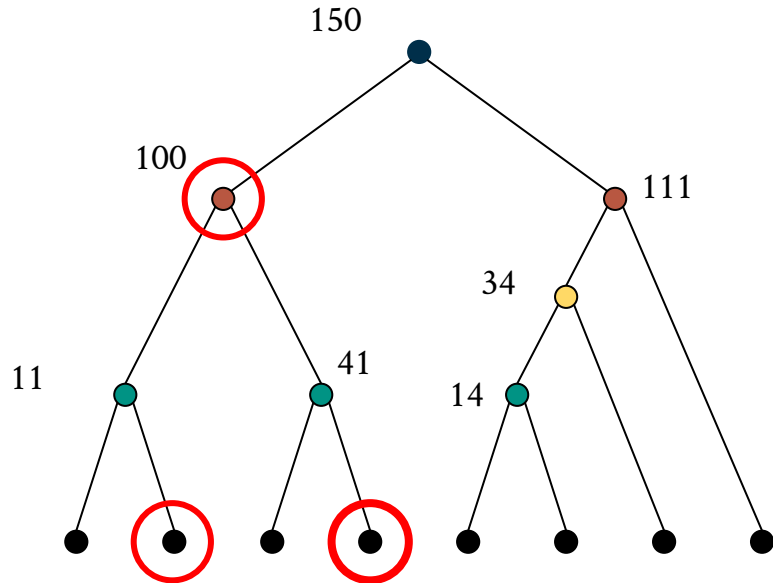
~~Triangle inequality
 $d(x,z) \leq d(x,y) + d(y,z)$~~

is strengthened to

Ultrametric inequality

$$d(x,z) \leq \max(d(x,y), d(y,z))$$

- Recursive partitioning of the data
- n points $\rightarrow 2n-1$ nested clusters with different granularities



$d(x,y)$ = value of the lowest common ancestor

Agglomerative algorithms

- average-linkage, single-linkage, Ward's method, complete-linkage, ...
- Produce an embedding of a metric into an ultrametric
- Bottom-up: proceed by agglomerating the pair of clusters of minimum dissimilarity

Agglomerative algorithms

- average-linkage, single-linkage, Ward's method, complete-linkage, ...
- Produce an embedding of a metric into an ultrametric
- Bottom-up: proceed by agglomerating the pair of clusters of minimum dissimilarity

Major drawback: quadratic running time

Ultrametric

Goal: given some dataset, find *efficiently*
its *best* ultrametric representation

Ultrametric

Goal: given some dataset, find *efficiently*
its *best* ultrametric representation

wait... the *best* ?

Problem statement

BEST ULTRAMETRIC FIT (BUF_∞)

INPUT:

- a set V of n elements v_1, v_2, \dots, v_n
- a weight function $w : V \times V \rightarrow \mathbb{R}$

OUTPUT:

- an *ultrametric* Δ such that

$$w(v_i, v_j) \leq \Delta(v_i, v_j) \leq \alpha \cdot w(v_i, v_j)$$

for the *minimal value* α .

Main results

$$V = \mathbb{R}^d$$

$$w(\mathbf{v}_i, \mathbf{v}_j) = \|\mathbf{v}_i - \mathbf{v}_j\|_2$$

Theorem 1 (upper bound)

There are algorithms that produce, for Euclidean instances of BUF_∞

- For any $\gamma > 1$, a **5γ -approximation** in time $O(nd + n^{1+O(1/\gamma^2)})$
- a **$\sqrt{\log n}$ -approximation** in time $O(nd + n \log^2 n)$

Main results

$$V = \mathbb{R}^d$$

$$w(v_i, v_j) = \|v_i - v_j\|_2$$

SAT can't be
solved in
 $2^{n(1-o(1))}$

SETH

$$w(v_i, v_j) = \|v_i - v_j\|_\infty$$

Theorem 1 (upper bound)

There are algorithms that produce, for Euclidean instances of BUF_∞

- For any $\gamma > 1$, a **5γ -approximation** in time $O(nd + n^{1+O(1/\gamma^2)})$
- a **$\sqrt{\log n}$ -approximation** in time $O(nd + n \log^2 n)$

Theorem 2 (lower bounds) -- informal statement

- Assuming the **Strong Exponential Time Hypothesis (SETH)**, there is no algorithm running in subquadratic time that can approximate BUF_∞ within a factor $3/2 - o(1)$ for the L_∞ norm
- + another lower bound for Euclidean metric under a “**Colinearity Hypothesis**”.

Related work

[**CM10**] (Carlsson and Mémoli)

→ study of linkage algorithms

[**Das15**] (Dasgupta)

→ what is a good hierarchical clustering? (cost functions)

[**MW17**] (Moseley and Wang)

[**CAKMTM18**] (Cohen-Addad, Kanade, Mallmann-Trenn, Mathieu)

→ good approximation guarantees for average-linkage for the (dual of) Dasgupta's cost function
& new algorithms 'beyond-worst-case' scenario

[**CM15**] (Cochez and Mou)

[**ACH19**] (Abboud, Cohen-Addad, and Houdrouge)

→ subquadratic running time implementation of average-linkage and Ward's method

+ many others [**RP16**, **CC17**, **CAKMT17**, **CCN19**, **CCNY18**, ...]

A Robust Model for Finding Optimal Evolutionary Trees

M. Farach,¹ S. Kannan,² and T. Warnow³

Abstract. Constructing evolutionary trees for species sets is a fundamental problem in computational biology. One of the standard models assumes the ability to compute distances between every pair of species, and seeks to find an edge-weighted tree T in which the distance d_{ij}^T in the tree between the leaves of T corresponding to the species i and j exactly equals the observed distance, d_{ij} . When such a tree exists, this is expressed in the biological literature by saying that the distance function or matrix is *additive*, and trees can be constructed from additive distance matrices in $O(n^2)$ time. Real distance data is hardly ever additive, and we therefore need ways of modeling the problem of finding the best-fit tree as an optimization problem.

In this paper we present several natural and realistic ways of modeling the inaccuracies in the distance data. In one model we assume that we have upper and lower bounds for the distances between pairs of species and try to find an additive distance matrix between these bounds. In a second model we are given a partial matrix and asked to find if we can fill in the unspecified entries in order to make the entire matrix additive. For both of these models we also consider a more restrictive problem of finding a matrix that fits a tree which is not only additive but also *ultrametric*. Ultrametric matrices correspond to trees which can be rooted so that the distance from the root to any leaf is the same. Ultrametric matrices are desirable in biology since the edge weights then indicate evolutionary time. We give polynomial-time algorithms for some of the problems while showing others to be NP-complete. We also consider various ways of “fitting” a given distance matrix (or a pair of upper- and lower-bound matrices) to a tree in order to minimize various criteria of error in the fit. For most criteria this optimization problem turns out to be NP-hard, while we do get polynomial-time algorithms for some.

Starting point

Algorithmica (1995) 13: 155–179

Algorithmica
© 1995 Springer-Verlag New York Inc.

A Robust Model for Finding Optimal Evolutionary Trees

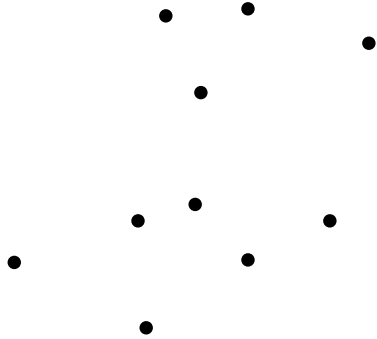
M. Farach,¹ S. Kannan,² and T. Warnow³

Abstract. Constructing evolutionary trees for species sets is a fundamental problem in computational

- Solves a slightly more general problem
- Provides an algorithm that **runs in $O(n^2)$** (given queries to w are done in constant time)
- This algorithm is **optimal**

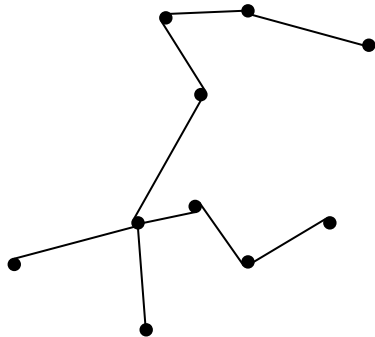
make the entire matrix additive. For both of these models we also consider a more restrictive problem of finding a matrix that fits a tree which is not only additive but also *ultrametric*. Ultrametric matrices correspond to trees which can be rooted so that the distance from the root to any leaf is the same. Ultrametric matrices are desirable in biology since the edge weights then indicate evolutionary time. We give polynomial-time algorithms for some of the problems while showing others to be NP-complete. We also consider various ways of “fitting” a given distance matrix (or a pair of upper- and lower-bound matrices) to a tree in order to minimize various criteria of error in the fit. For most criteria this optimization problem turns out to be NP-hard, while we do get polynomial-time algorithms for some.

APPROX-BUF: an approximation algorithm for BUF_∞



APPROX-BUF

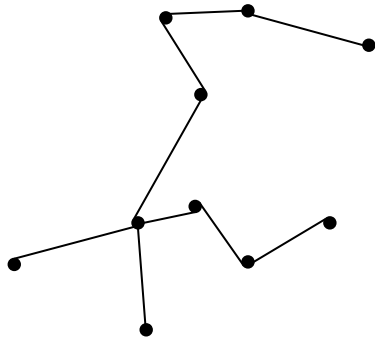
APPROX-BUF: an approximation algorithm for BUF_∞



APPROX-BUF

1. Compute a γ -approximate MST T over the complete graph G

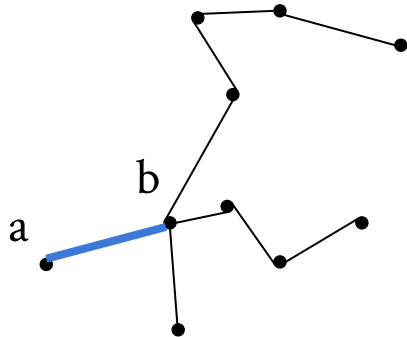
APPROX-BUF: an approximation algorithm for BUF_∞



APPROX-BUF

1. Compute a γ -approximate MST T over the complete graph G
2. Compute a \square -estimate of the cut weights of the edges in T

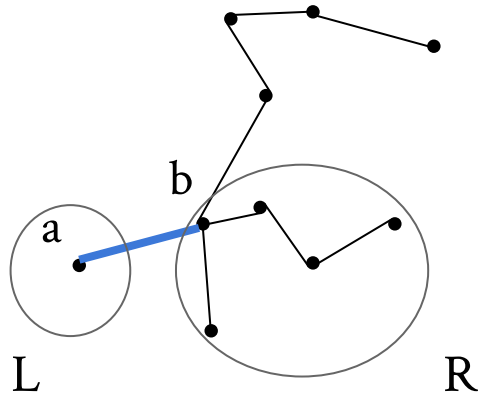
APPROX-BUF: an approximation algorithm for BUF_∞



APPROX-BUF

1. Compute a γ -approximate MST T over the complete graph G
2. Compute a \square -estimate of the cut weights of the edges in T

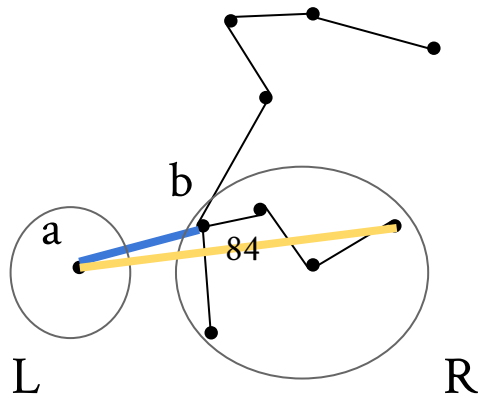
APPROX-BUF: an approximation algorithm for BUF_∞



APPROX-BUF

1. Compute a γ -approximate MST T over the complete graph G
2. Compute a \square -estimate of the cut weights of the edges in T

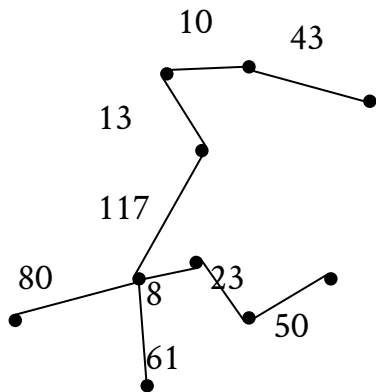
APPROX-BUF: an approximation algorithm for BUF_∞



APPROX-BUF

1. Compute a γ -approximate MST T over the complete graph G
2. Compute a \square -estimate of the cut weights of the edges in T

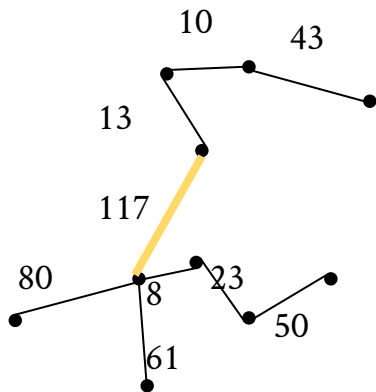
APPROX-BUF: an approximation algorithm for BUF_∞



APPROX-BUF

1. Compute a γ -approximate MST T over the complete graph G
2. Compute a \square -estimate of the cut weights of the edges in T

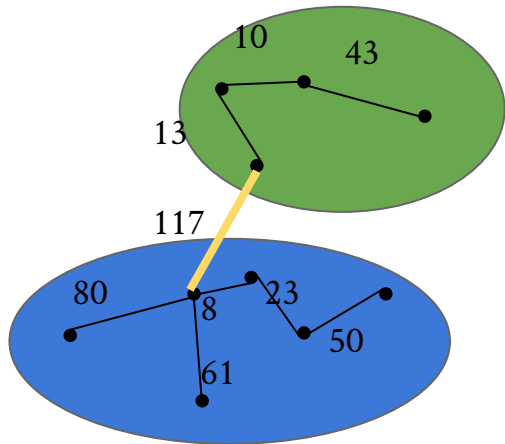
APPROX-BUF: an approximation algorithm for BUF_∞



APPROX-BUF

1. Compute a γ -approximate MST T over the complete graph G
2. Compute a \square -estimate of the cut weights of the edges in T
3. Compute the cartesian tree

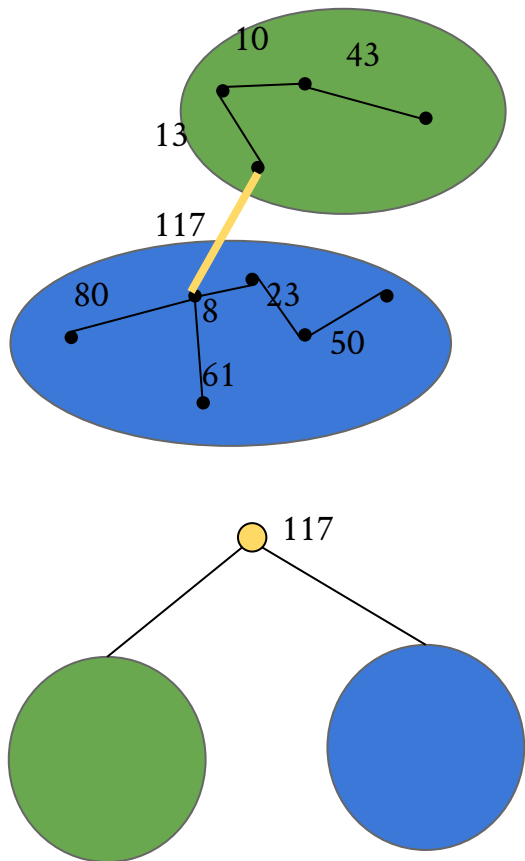
APPROX-BUF: an approximation algorithm for BUF_∞



APPROX-BUF

1. Compute a γ -approximate MST T over the complete graph G
2. Compute a \square -estimate of the cut weights of the edges in T
3. Compute the cartesian tree

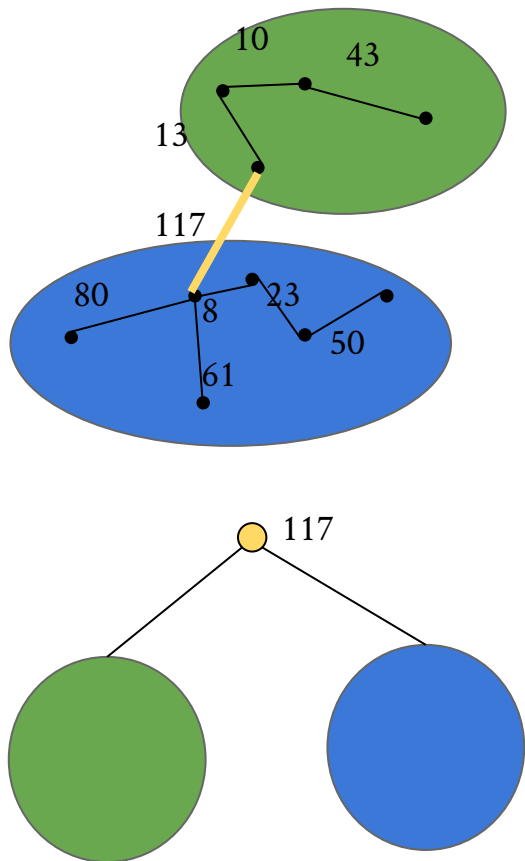
APPROX-BUF: an approximation algorithm for BUF_∞



APPROX-BUF

1. Compute a γ -approximate MST T over the complete graph G
2. Compute a \square -estimate of the cut weights of the edges in T
3. Compute the cartesian tree

APPROX-BUF: an approximation algorithm for BUF_∞



APPROX-BUF

1. Compute a γ -approximate MST T over the complete graph G
2. Compute a \square -estimate of the cut weights of the edges in T
3. Compute the cartesian tree

→ This gives a $\gamma \cdot \square$ -approximation to BUF_∞

APPROX-BUF: an approximation algorithm for BUF_∞

Fast implementation in Euclidean space of dimension d

Based on γ -spanner constructions using Har-Peled, Indyk, Sidiropoulos

Any $\gamma > 1$	$\gamma = \sqrt{(\log n)}$
<i>Locality sensitive hash family</i> (Andoni and Indyk)	<i>Lipschitz partitions</i> (Charikar et al.)
$O(nd + n^{1+O(1/\gamma^2)})$	$O(nd + n \log^2 n)$

APPROX-BUF

1. Compute a γ -approximate MST T over the complete graph G
2. Compute a \square -estimate of the cut weights of the edges in T
3. Compute the cartesian tree

→ This gives a $\gamma \cdot \square$ -approximation to BUF_∞

APPROX-BUF: an approximation algorithm for BUF_∞

Fast implementation in Euclidean space of dimension d

Tweak a union-find data structure and compute bottom-up the cut weights

$\gamma=5$

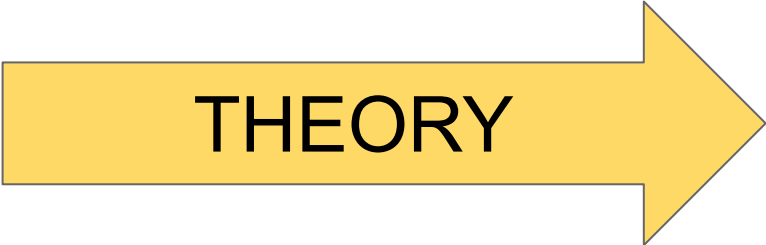
Triangular inequality

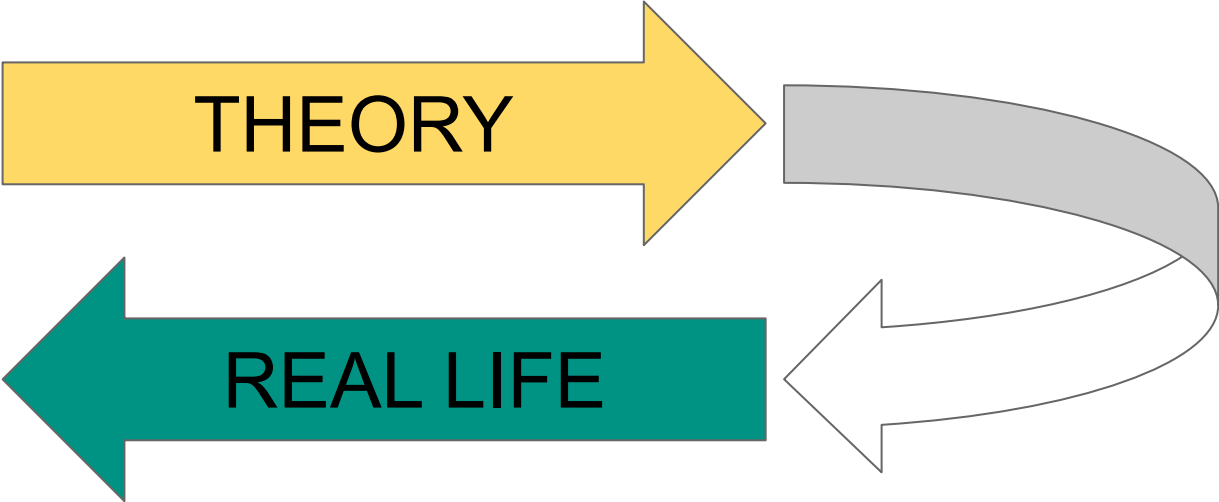
$O(nd+n \log n)$

APPROX-BUF

1. Compute a γ -approximate MST T over the complete graph G
2. Compute a γ -estimate of the cut weights of the edges in T
3. Compute the cartesian tree


→ This gives a $\gamma \cdot 5$ -approximation to BUF_∞





Experiments: maximum distortion

- DIABETES -- 768 samples, 8 features
- MICE -- 1080 samples, 77 features
- PENDIGITS -- 10992 samples, 16 features

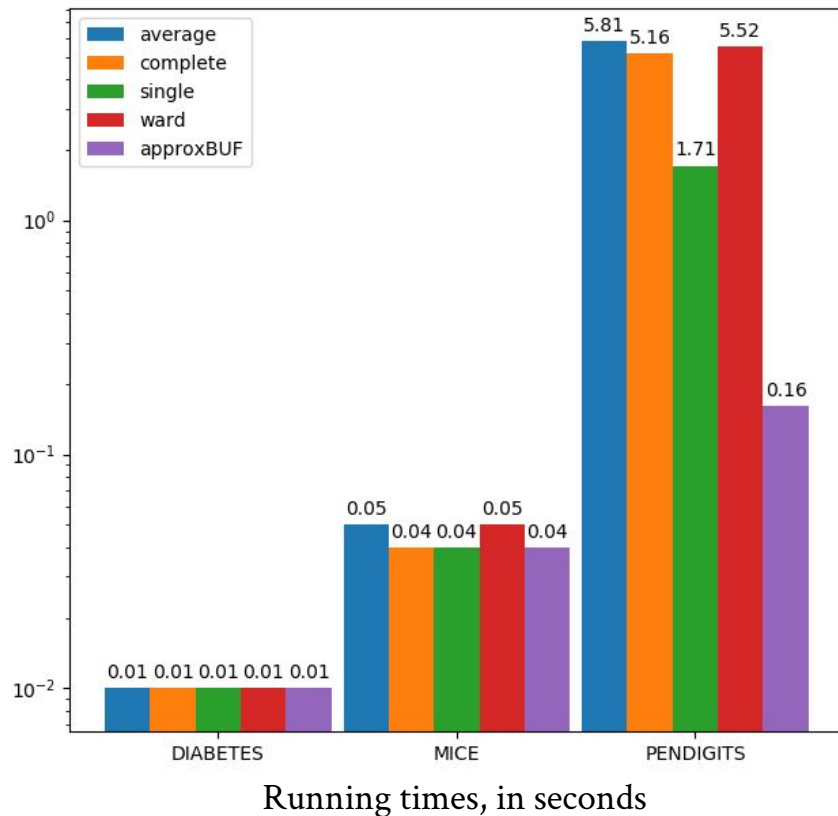

$$\alpha = \max_{v_i, v_j} \Delta(v_i, v_j) / w(v_i, v_j)$$

	DIABETES	MICE	PENDIGITS
Average	11.1	9.7	27.5
Complete	18.5	11.8	33.8
Single	6.0	4.9	14
Ward	61.0	59.3	433.8
Approx-BUF	41.0	51.2	109.8
Approx-BUF2	9.6	9.4	37.2
Farach et al.	6.0	4.9	13.9

Approx-BUF: approx MST + approx cut weights

Approx-BUF2: exact MST + approx cut weights

Experiments: running time



Conclusion

- Seems promising
- A good MST is crucial → can we compute a better one efficiently?
- Cut weights suffer from an approximation of $\square=5$ → can we do better?

Thanks!