On Gromov's Approximating Tree

Anders Cornect
Joint work with Dr. Eduardo Martínez-Pedroza

Memorial University of Newfoundland

August 2023

Approximating Trees

Definition

Let G be a graph. A *distance ϵ-approximating tree* of G is a tree with the same vertex set as G, such that for all $u, v \in G$,

$$|d_G(u, v) - d_T(u, v)| \leq \epsilon.$$

Approximating Trees

Definition

Let G be a graph. A *distance ϵ-approximating tree* of G is a tree with the same vertex set as G, such that for all $u, v \in G$,

$$|d_G(u, v) - d_T(u, v)| \leq \epsilon.$$

Many problems in graph theory are trivial on trees.

Gromov's Tree

Let G be a graph on n vertices with a base point w.

Gromov's Tree

Let G be a graph on n vertices with a base point w.

Theorem (Gromov, 1987)

Any graph G can be embedded by a function φ into a weighted tree T so that:

Gromov's Tree

Let G be a graph on n vertices with a base point w.

Theorem (Gromov, 1987)

Any graph G can be embedded by a function φ into a weighted tree T so that:

- $d_T(\varphi(u), w) = d_G(u, w)$ for all $u \in G$,

Gromov's Tree

Let G be a graph on n vertices with a base point w.

Theorem (Gromov, 1987)

Any graph G can be embedded by a function φ into a weighted tree T so that:

- $d_T(\varphi(u), w) = d_G(u, w)$ for all $u \in G$,
- T is non-distance-increasing, and

Gromov's Tree

Let G be a graph on n vertices with a base point w.

Theorem (Gromov, 1987)

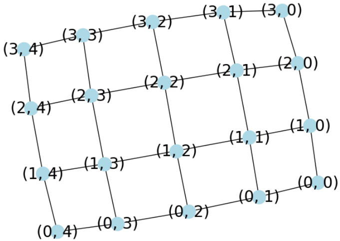
Any graph G can be embedded by a function φ into a weighted tree T so that:

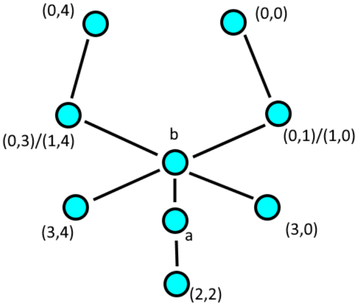
- $d_T(\varphi(u), w) = d_G(u, w)$ for all $u \in G$,
- T is non-distance-increasing, and
- T is a $2\delta \log(n)$-approximation of G, where δ is the *Gromov hyperbolicity* of G.

Combining these shows that, for all $u, v \in G$,

$$d_G(u, v) - 2\delta \log(n) \leq d_T(\varphi(u), \varphi(v)) \leq d_G(u, v).$$

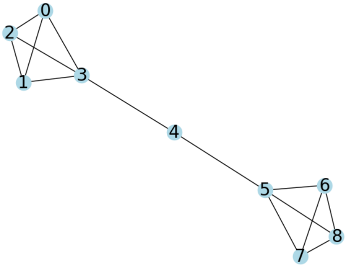
Gromov's Tree

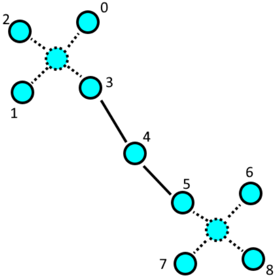




$$w = (2,2),\ \varphi^{-1}(a) = \{(1,2),(2,1),(2,3),(3,2)\},$$
$$\varphi^{-1}(b) = \{(0,2),(1,1),(1,3),(2,0),(2,4),(3,1),(3,3)\}$$

Gromov's Tree





$$w = 0$$

Gromov's Tree

The proof of this theorem gives a general method of its construction, but with no mention of time complexity.

However, many articles cite these two works with claims that it can be done in $O(n^2)$ time. This turns out to be the case, if we start with the distance matrix D of our graph G.

Gromov's Tree

In practical application, D is very rarely stored explicitly. Assume our graph G is stored as an adjacency matrix.

Gromov's Tree

In practical application, D is very rarely stored explicitly. Assume our graph G is stored as an adjacency matrix.

$$G \xrightarrow{O(n^{\omega} \log(n))} D \qquad \omega \approx 2.371552.$$

Our Contribution

Therefore, while this algorithm can theoretically find the distance matrix A of an approximating tree in $O(n^2)$ time, this is rarely true in practice.

G D A

Our Contribution

Therefore, while this algorithm can theoretically find the distance matrix A of an approximating tree in $O(n^2)$ time, this is rarely true in practice.

$$G \xrightarrow{O(n^{\omega} \log(n))} D \xrightarrow{O(n^2)} A$$

Our Contribution

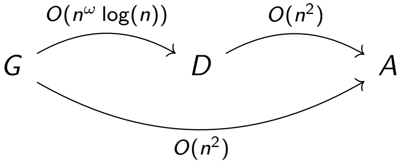
Therefore, while this algorithm can theoretically find the distance matrix A of an approximating tree in $O(n^2)$ time, this is rarely true in practice.

$$G \xrightarrow{O(n^{\omega} \log(n))} D \xrightarrow{O(n^2)} A$$

However, we have used the particular geometry of connected graphs to bypass finding D completely.

Our Contribution

Therefore, while this algorithm can theoretically find the distance matrix A of an approximating tree in $O(n^2)$ time, this is rarely true in practice.



However, we have used the particular geometry of connected graphs to bypass finding D completely.

What we have done is written an explicit algorithm that results in the following theorem.

Theorem (Cornect and Martinez-Pedroza, 2023)

There is an algorithm that takes as input the adjacency matrix of a graph G on n vertices, and outputs in time $O(n^2)$ the distance matrix A of an approximating tree, as described in Gromov's Theorem.

Gromov Hyperbolicity

Definition

In a metric space (X, d), the Gromov product of x and y with respect to z is given by

$$(x|y)_z = \frac{1}{2}\left(d(x,z) + d(y,z) - d(x,y)\right).$$

Gromov Hyperbolicity

Definition

In a metric space (X, d), the Gromov product of x and y with respect to z is given by

$$(x|y)_z = \frac{1}{2}\left(d(x,z) + d(y,z) - d(x,y)\right).$$

Definition (four-point condition)

A space is called δ-hyperbolic if, for all w, x, y, $z \in X$,

$$\delta \geq \min\{(x|y)_w, (y|z)_w\} - (x|z)_w.$$

Gromov Hyperbolicity

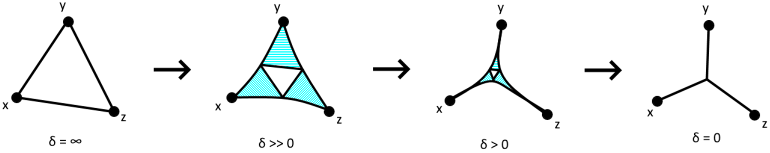
Definition (δ-slim triangle condition)

A space is called $\delta^* -$ *hyperbolic* if all triangles are δ^*-slim.

Gromov Hyperbolicity

Definition (δ-slim triangle condition)

A space is called $\delta^* - $ *hyperbolic* if all triangles are δ^*-slim.



Tree-Likeness

The δ-slim triangle condition is one way of seeing δ as a measure of how "tree-like" a graph is.

Tree-Likeness

The δ-slim triangle condition is one way of seeing δ as a measure of how "tree-like" a graph is.

Gromov's approximating tree gives another. The approximation is better for smaller δ. Small δ means a graph can be more accurately represented by a tree; it is more "tree-like".

Tree-Likeness

The δ-slim triangle condition is one way of seeing δ as a measure of how "tree-like" a graph is.

Gromov's approximating tree gives another. The approximation is better for smaller δ. Small δ means a graph can be more accurately represented by a tree; it is more "tree-like".

Fournier et al. (2015) detailed a way of approximating δ using Gromov's approximating tree in $O(n^2)$ from D. Our algorithm allows us to do this directly from G, while staying $O(n^2)$.

Simplified Explanation

An oversimplification, creating a tree T from graph G based at w:

Simplified Explanation

An oversimplification, creating a tree T from graph G based at w:

1. First, calculate the distance d_w from w to each other vertex ($O(n^2)$ by Dijkstra). Call the largest of these distances α.

Simplified Explanation

An oversimplification, creating a tree T from graph G based at w:

1. First, calculate the distance d_w from w to each other vertex ($O(n^2)$ by Dijkstra). Call the largest of these distances α.
2. Add all vertices with $d_w = \alpha$ to T. Keep edges between them.

Simplified Explanation

An oversimplification, creating a tree T from graph G based at w:

1. First, calculate the distance d_w from w to each other vertex ($O(n^2)$ by Dijkstra). Call the largest of these distances α.
2. Add all vertices with $d_w = \alpha$ to T. Keep edges between them.
3. Add all vertices with $d_w = \alpha - 1$ to T. Keep edges between them, or to existing vertices in T.

Simplified Explanation

An oversimplification, creating a tree T from graph G based at w:

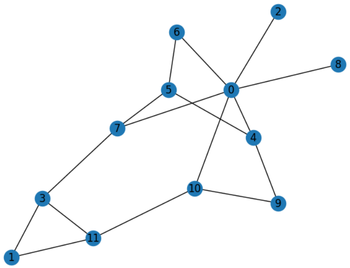
1. First, calculate the distance d_w from w to each other vertex ($O(n^2)$ by Dijkstra). Call the largest of these distances α.
2. Add all vertices with $d_w = \alpha$ to T. Keep edges between them.
3. Add all vertices with $d_w = \alpha - 1$ to T. Keep edges between them, or to existing vertices in T.
4. If two vertices with $d_w = \alpha - 1$ are connected to the same existing vertex in T, identify them.

Simplified Explanation

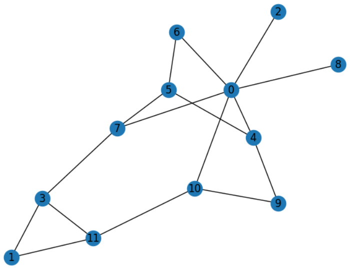
An oversimplification, creating a tree T from graph G based at w:

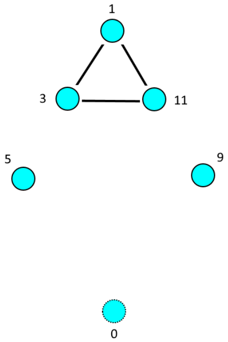
1. First, calculate the distance d_w from w to each other vertex ($O(n^2)$ by Dijkstra). Call the largest of these distances α.
2. Add all vertices with $d_w = \alpha$ to T. Keep edges between them.
3. Add all vertices with $d_w = \alpha - 1$ to T. Keep edges between them, or to existing vertices in T.
4. If two vertices with $d_w = \alpha - 1$ are connected to the same existing vertex in T, identify them.
5. Repeat steps 3 and 4 for $d_w = \alpha - 2, \alpha - 3, \ldots, 0$.

An Example

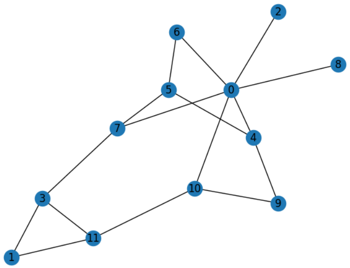


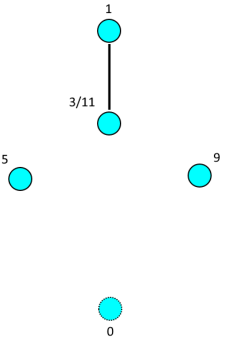
An Example



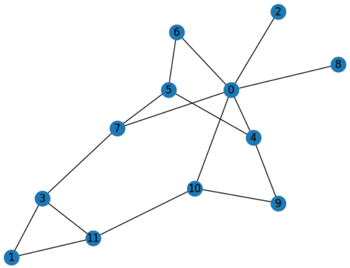


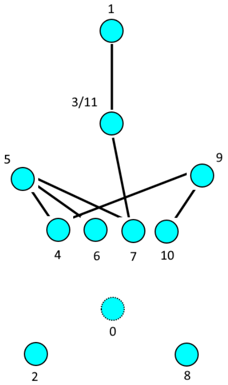
An Example



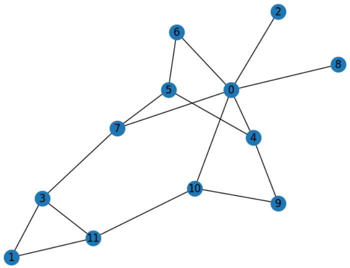


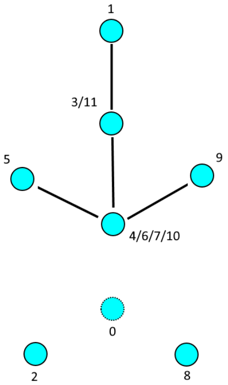
An Example



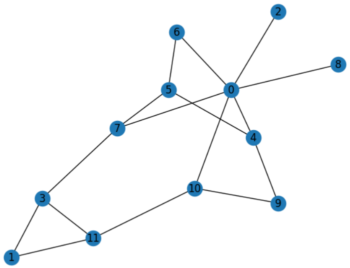


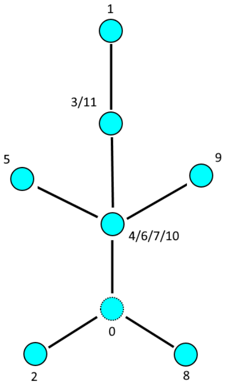
An Example





An Example





Future Questions

- Can the bound of $O(n^{\omega}\log(n))$ be improved in the general case, or for other types of metric spaces?

Future Questions

- Can the bound of $O(n^{\omega} \log(n))$ be improved in the general case, or for other types of metric spaces?
- Is there a similar algorithm for strongly connected directed graphs (non-commutative metrics)?

Future Questions

- Can the bound of $O(n^{\omega} \log(n))$ be improved in the general case, or for other types of metric spaces?
- Is there a similar algorithm for strongly connected directed graphs (non-commutative metrics)?
- What is the connection between the All Pairs Bottleneck Problem (APBP) and Gromov's approximating tree?

Acknowledgements

Thank you!

NSERC
CRSNG