Streaming Algorithm For Graph Spanners

Satpute Aniket Tukaram : CS21BTECH11056
Vishal Vijay Devadiga : CS21BTECH11061
Harshit Pant : CS21BTECH11021
Mahin Bansal : CS21BTECH11034

Under the Guidance of
Prof(Dr).Rogers Mathew

भारतीय प्रौद्योगिकी संस्थान हैदराबाद
Indian Institute of Technology Hyderabad

April 14, 2024

Overview

Overview

- A spanner is a sparse sub-graph that preserves approximate distance between each pair of vertices.
- A t-spanner of a graph G $= (V, E)$, for any $t \in$ N, is a sub-graph (V, E_s), $E_s \subseteq E$ such that, for any u, v $\in$ V , their distance in the sub- graph is at most t times their distance in the original graph.
- The parameter t is called the stretch associated with the t-spanner

Overview

- Computing t-spanner of smallest size for a graph is NP-hard.
- The goal of this paper is to design an efficient algorithm that, for any weighted graph on n vertices, computes a $(2k-1)$-spanner of size $O(n^{1+\frac{1}{k}})$.

Preliminaries

- We assume that n, the number of vertices is known in advance and the vertices are numbered from 1 to n.
- The central idea of the algorithm is a suitable grouping of vertices called clustering.
- A cluster is a subset of vertices, and a clustering $\mathcal{C}$, is a union of disjoint clusters. Each cluster will have a unique vertex which will be called its center.
- $\mathcal{C}[v]$ will denote the center of the cluster containing v unless v does not belong to any cluster, in which case $\mathcal{C}[v] = 0$.
- A cluster c is said to be adjacent to a vertex u if there is some edge (u, v) in the graph for some $v \in c$.
- With respect to a clustering $\mathcal{C}$, v is said to be **clustered vertex** if it belongs to some cluster $c \in \mathcal{C}$.

Initializing Clusterings

$$
\begin{aligned}
& S_0 \leftarrow V;\ \ S_k \leftarrow \emptyset; \\
& \textbf{For } (0 < i < k) \\
& \quad S_i \text{ is formed by selecting each } v \in S_{i-1} \\
& \quad \text{independently with probability } n^{-1/k}; \\
& \textbf{For } (\text{each } v \in V \ \text{ and } 0 \leq i < k) \\
& \quad \textbf{if } (v \in S_i)\ \ \mathcal{C}_i[v] \leftarrow v \ \ \textbf{else } \mathcal{C}_i[v] \leftarrow 0.
\end{aligned}
$$

Figure: Forming the initial k clusterings

- We define $l_c(v)$ to be the highest level i $<$ k such that v appears as center of some cluster in $\mathcal{C}_i$.
- We will say that a cluster $c \in \mathcal{C}_i$ is a sampled cluster at level i if its center was selected to form a cluster center at $(i+1)^{th}$ level.

Working Example

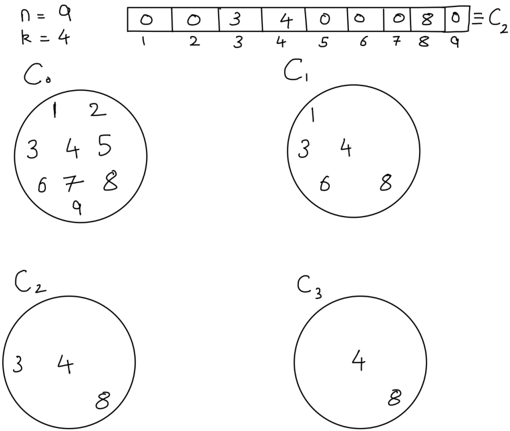


Figure: Example with n=9 and k=4

Assertion $\mathcal{A}$

Assertion

$\mathcal{A}$: For each cluster $c' \in \mathcal{C}_{i+1}$, there exists a unique sampled cluster c at level i such that $c \subseteq c'$, and vice versa.

Some Definitions

Let $G(V, E)$ be the graph under consideration with n vertices
Let there be k clusterings $\{\mathcal{C}_i | 0 \leq i < k\}$ of the vertices of G formed by the preprocessing steps.
For the below definitions, consider v to be a vertex $\in V$ and $\mathcal{C}$ is the collection of the k clusterings

- $\mathcal{C}_i[v]$ denotes the center of the cluster $c \in \mathcal{C}_i$ such that $v \in c$, if v doesn't belong to any cluster in $\mathcal{C}_i$ then $\mathcal{C}_i[v] = 0$
- A cluster c is said to **adjacent** to v if $\exists(u, v) \in G(E)$ for some $u \in c$.
- With respect to a clustering $\mathcal{C}$, v is said to be **clustered vertex** if it belongs to some cluster $c \in \mathcal{C}$.

Some Definitions (Contd. . .)

- $l_C(v)$: highest $i < k$ such that v appears as center of some cluster in $\mathcal{C}_i$
- $l(v)$: highest $i < k$ such that v appears as a member of some cluster in $\mathcal{C}_i$. (Initially $l(v) = l_C(v)$)s
- A cluster $c \in \mathcal{C}_i$ is a **sampled cluster** at level i if its center is selected to a form a cluster center at level $(i+1)$
- For each $v \in V, \varepsilon(v)$: stores one edge per unsampled cluster at level $l(v)$ which is adjacent to v.
- For each $v \in V$, $Temp(v)$ is a buffer to store the edges.
- ε_s stores the partially constructed spanner.

Algorithm - Processing an edge in the stream

If $(\ell(u) > \ell(v))$ swap (u, v) **Endif**
$i \leftarrow \ell(u)$; $x \leftarrow \mathcal{C}_i[v]$; $h \leftarrow \ell_c(x)$;
If $(h > i)$ *// opportunity for u to move up*
  For $j = i + 1$ to h **do** $\mathcal{C}_j[u] \leftarrow x$;
  $\ell(u) \leftarrow h$;
  $\mathcal{E}_S \leftarrow \mathcal{E}_S \cup Temp(u) \cup \mathcal{E}(u)$;
  $Temp(u) \leftarrow \emptyset$; $\mathcal{E}(u) \leftarrow \{(u, v)\}$;
Else
  $Temp(u) \leftarrow Temp(u) \cup \{(u, v)\}$;
  If $(|Temp(u)| = |\mathcal{E}(u)|)$ $\textit{Prune}(u, i)$ **Endif**
Endif

Figure: Processing an edge (u, v) of the stream

Algorithm - *Prune*(u, i)

1. **For** each $(u, w) \in \mathcal{E}(u)$ **do**
   $A[\mathcal{C}_i[w]] \leftarrow 1.$

2. **For** each $(u, v) \in Temp(u)$ **do**
   If $(A[\mathcal{C}_i[v]] = 0)$
   $A[\mathcal{C}_i[v]] \leftarrow 1;$
   $\mathcal{E}(u) \leftarrow \mathcal{E}(u) \cup \{(u, v)\}$ **Endif**
   $Temp(u) \leftarrow Temp(u) \backslash (u, v).$

3. **For** each $(u, w) \in \mathcal{E}(u)$ **do** $A[\mathcal{C}_i[w]] \leftarrow 0.$

Figure: The procedure *Prune* (u, i)

Working Example

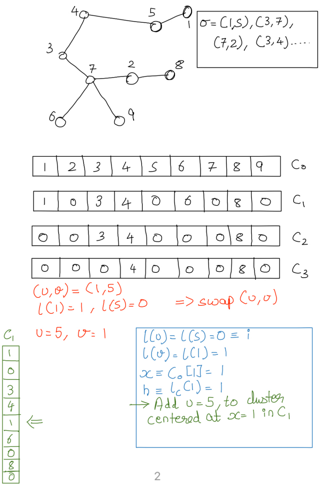


Figure: Example with n=9 and k=4 and for the first edge

Explanation: Algorithm

The algorithm is explained informally below:

- Let $(u, v) \in \sigma$ be an edge $\in G(E)$. When this edge appears we process on u if $l(u) \leq l(v)$, otherwise we process on v.
- A vertex $u \in V$ waits in the $l(u)$ clustering until it appears in one of the vertex in an edge of the stream, where it might get a chance to move to a level higher than $l(u)$
- WLOG, consider $l(v) > l(u)$, two cases arise here

Case 1 The clustering $c \in \mathcal{C}_{l(u)}$ containing v is a sampled cluster[10] at the level $l(u)$

Case 2 The clustering $c \in \mathcal{C}_{l(u)}$ containing v is an unsampled cluster at the level $l(u)$

Explanation (Contd. . .): Case 1

We handle the two cases separately

Case 1:

- Follows from the assertion, $\exists c' \in \mathcal{C}_{l(u)+1}$ such that $c \subseteq c'$ and hence we add u to the cluster c'.
- We keep on adding u to the cluster in the next higher level which is guaranteed to exist by the lemma, until the level $l_C(\mathcal{C}_{l(u)}[v])$
- We update the new $l(u)$ which is now $l_C(C_{l(u)}[v])$
- We add the previous values of Temp(u) and $\varepsilon(u)$ to ε_s
- Since $l(u)$ has been updated, we also need to update Temp(u) and $\varepsilon(u)$
- $Temp(u)$ is set to ϕ and $\varepsilon(u)$ becomes $\{(u, v)\}$

Explanation (Contd...): Case 2

Case 2:

- c is an unsampled cluster.
- In this case we add the edge (u, v) only if we have not seen any edge incident from any other vertex $\in$ c.
- In order to achieve this we use the $Temp(u)$ and $\varepsilon(u)$ lists.
- We simply add the edge to Temp(u) as it is a probable edge which can be added to the spanner.
- The list $Temp(u)$ will be pruned when its size becomes significant (equal to $\varepsilon(u)$)

Explanation (Contd. . .): Prune

$Prune(u, i)$

- We need to decide whether the edges $(u, w) \in Temp(u)$, which is essentially an edge from an unsampled cluster incident on u, should or not be added to the spanner, since only one edge per unsampled cluster incident on u needs to be added to the spanner.
- In order to achieve this, we use A$[0 \ldots n]$ is used which is initially set to 0 $\forall i \in \{1 \ldots n\}$
- Next using the centers of the clusters, we mark all clusters which already have an edge incident on u in $\varepsilon(u)$ to 1
- Finally $\forall (u, v) \in Temp(u)$, we check if the cluster is marked and if not then no edge from this cluster is included in $\varepsilon(u)$ and hence we mark the cluster and include this edge in $\varepsilon(u)$
- Finally we again mark all the entries in A to 0

Some Observations

Observation

For each vertex $v \in V, |Temp(v)\,| < |\varepsilon(v)|$ always except just before the invocation of $Prune(v, i)$ when $|Temp(v)| = |\varepsilon(v)|$

Final Output

- After executing the algorithm at any stage we define two sets of edges

$$\varepsilon^+ = \bigcup_{u \in V} \varepsilon(u) \cup \varepsilon_S \tag{1}$$

$$Temp = \bigcup_{u \in V} Temp(u) \tag{2}$$

- The obtained $\varepsilon^+ \bigcup Temp$ is our $(2k-1)$-spanner till that stage.

Analyzing the running time

- The time complexity of algorithm depends on two processes:
  - Processing an edge (u, v) of the stream:(11)
  - Procedure of $Prune(u, i)$:(12
- In case of Processing an edge only the for loop that increases the level of vertex is to be considered. All other steps happen in $O(1)$ time
- As we can see in function $Prune(u, i)$, its time complexity depends on $|\varepsilon(u)| + |Temp(u)|$
- The time for pre-processing i.e. creating the initial clusters, clusterings, defining $l(u)$ and $l_c(u)$ is in order of $O(nk)$

Time for $Prune(u, i)$

- As we can see from (18) order of $|\varepsilon(u)| + |Temp(u)|$ is same as $O(|Temp(u)|)$
- In the procedure $Prune(u, i)$ it can be seen that any edge from $Temp(u)$ is accessed only once, after that the edge is either discarded or becomes member of $\varepsilon(u)$
- This means each edge is processed for $O(1)$ time in $Prune(u, i)$ which leads to $O(m)$ time complexity for this step (m is number of edges in stream)

Time for Processing an edge

- We know that each iteration of the for loop increases level of a vertex and the level of vertex doesn't exceed $k - 1$.
- Therefore maximum of $O(nk)$ iteration of this loop will be executed (as number of vertex is n)
- This gives total time complexity as $O(nk + m) = O(m)$
- To ensure $O(m)$ time complexity we can start algorithm after nk edges. If stream is less than nk we can output those nk edges as it is as spanner edges

A vertex becoming a member of cluster c'

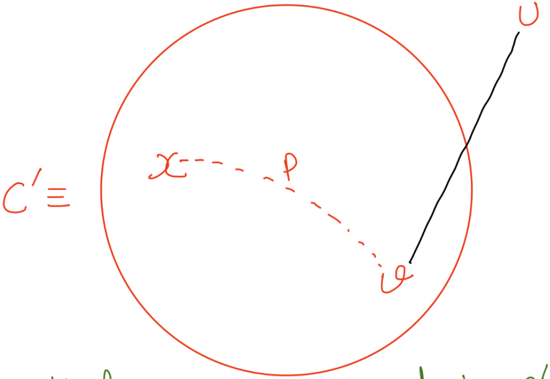
A vertex u becomes a member of c' only when:

- Edge (u, v) appears in the stream
- $l(u) < l(v)$, where $l(u) = j$, and $j < i$
- Vertex v is a member of some sampled cluster c in $\mathcal{C}_j$
- v belongs to c' in $\mathcal{C}_i$

By assertion $\mathcal{A}$, c is a subset of c'

Current edge $\equiv (u, v)$, Cluster $C' \in C_i$

Cluster center of $C' \equiv x$



P is a path from v to x in cluster C' such that,

$|P| = j < i$

Lemma

Lemma

Let c' be any cluster in $\mathcal{C}_i$.
Each vertex $u \in c'$ is connected to its center x by a path of length at most i edges from ε^+

Proof by Induction

The lemma is to be proved by induction on i.
Let:

- $x \leftarrow$ Center of cluster c'
- $(u, v) \leftarrow$ Edge appearing in stream

Considering that if c' is a singleton cluster, there is nothing to prove.

Assuming otherwise, let $u \neq x$ be a vertex belonging to c'.

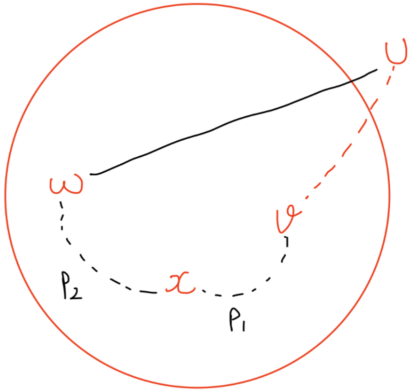
- By induction hypothesis, there exists a path $\subseteq \varepsilon^{+}$ between v and x of length at most j
- Since vertex u adds edge (u, v) to $\varepsilon(u)$ while joining c', there exists a path $\subseteq \varepsilon^{+}$ of length at most $j + 1 \leq i$ between u and center of cluster c'

Stretch of Spanner

An edge (u, v) that appears in the stream, either belongs to $\varepsilon^+ \cup Temp$ or it is discarded by the prune function.

- If it belongs to $\varepsilon^+ \cup Temp$, that is, the spanner, then is the distance between u and v is the same in both the spanner and the original graph.
- If it is discarded, then:
  - There exists an edge (u, w) in $\varepsilon(u)$ incident from the same cluster, say in $\mathcal{C}_i$ that v belongs to.
  - v and w is connected by a path through the center of the cluster with length at most $2i$.
  - Thus, distance between u and v is at most $2i + 1$.
  - Since, $i < k$, distance between u and v is at most $2k - 1$.

C'



$|P_1| \le i$

$|P_2| \le i$

$$u \to v \le 1 + i + i$$
$$= 2i + 1$$

Stretch of Spanner

This implies any shortest path in the original graph is stretched by a factor of at most $2k - 1$.

Thus, the set $\varepsilon^{+} \cup \mathit{Temp}$ is a $2k - 1$ spanner for the stream of edges seen so far.