Graph & Geometry Problems in Data Streams

2009 Barbados Workshop on Computational Complexity

Andrew McGregor

Introduction

Models:

- Graph Streams: Stream of edges $E = \{e_1, e_2, \ldots, e_m\}$ describe a graph G on n nodes. Estimate properties of G.
- Geometric Streams: Stream of points $X = \{p_1, p_2, \ldots, p_m\}$ from some metric space $(\mathcal{X}, d)$. Estimate properties of X.

Introduction

Models:

- Graph Streams: Stream of edges $E = \{e_1, e_2, \ldots, e_m\}$ describe a graph G on n nodes. Estimate properties of G.
- Geometric Streams: Stream of points $X = \{p_1, p_2, \ldots, p_m\}$ from some metric space $(\mathcal{X}, d)$. Estimate properties of X.

Notes:

- $\tilde{O}$ is our friend: we'll hide dependence on polylog(m, n) terms.
- Assume that p_i can be stored in $\tilde{O}(1)$ space and $d(p_i, p_j)$ can be calculated if both p_i and p_j are stored in memory.
- Theory isn't as cohesive but we get to cherry-pick results...

Counting Triangles

Matching

Clustering

Graph Distances

Outline

Counting Triangles

Matching

Clustering

Graph Distances

Triangles

Problem

Given a stream of edges, estimate the number of triangles T_3 *up to a factor* $(1+\epsilon)$ *with probability* $1-\delta$ *given promise that* $T_3 > t$.

Triangles

Problem

Given a stream of edges, estimate the number of triangles T_3 *up to a factor* $(1+\epsilon)$ *with probability* $1-\delta$ *given promise that* $T_3 > t$.

Warm-Up

What's an algorithm using $O(\epsilon^{-2}(n^3/t)\log\delta^{-1})$ *space?*

Triangles

Problem

Given a stream of edges, estimate the number of triangles T_3 up to a factor $(1+\epsilon)$ with probability $1-\delta$ given promise that $T_3 > t$.

Warm-Up

What's an algorithm using $O(\epsilon^{-2}(n^3/t)\log \delta^{-1})$ space?

Theorem

$\Omega(n^2)$ *space required to determine if $t = 0$ (with $\delta = 1/3$).*

Triangles

Problem

Given a stream of edges, estimate the number of triangles T_3 up to a factor $(1+\epsilon)$ with probability $1-\delta$ given promise that $T_3 > t$.

Warm-Up

What's an algorithm using $O(\epsilon^{-2}(n^3/t)\log\delta^{-1})$ space?

Theorem

$\Omega(n^2)$ *space required to determine if $t = 0$ (with $\delta = 1/3$).*

Theorem (Sivakumar et al. 2002)

$\tilde{O}(\epsilon^{-2}(nm/t)^2\log\delta^{-1})$ *space is sufficient.*

Triangles

Problem

Given a stream of edges, estimate the number of triangles T_3 up to a factor $(1+\epsilon)$ with probability $1-\delta$ given promise that $T_3 > t$.

Warm-Up

What's an algorithm using $O(\epsilon^{-2}(n^3/t)\log\delta^{-1})$ space?

Theorem

$\Omega(n^2)$ *space required to determine if $t = 0$ (with $\delta = 1/3$).*

Theorem (Sivakumar et al. 2002)

$\tilde{O}(\epsilon^{-2}(nm/t)^2\log\delta^{-1})$ *space is sufficient.*

Theorem (Buriol et al. 2006)

$\tilde{O}(\epsilon^{-2}(nm/t)\log\delta^{-1})$ *space is sufficient.*

Lower Bound

Theorem

$\Omega(n^2)$ space required to determine if $T_3 \neq 0$ when $\delta = 1/3$.

Lower Bound

Theorem

$\Omega(n^2)$ space required to determine if $T_3 \neq 0$ when $\delta = 1/3$.

- Reduce from set-disjointness: Alice has $n \times n$ binary matrix A, Bob has $n \times n$ binary matrix B. Is $A_{ij} = B_{ij} = 1$ for some (i, j)? Needs $\Omega(n^2)$ bits of communication [Razborov 1992].

Lower Bound

Theorem

$\Omega(n^2)$ *space required to determine if* $T_3 \neq 0$ *when* $\delta = 1/3$.

- Reduce from set-disjointness: Alice has $n \times n$ binary matrix A, Bob has $n \times n$ binary matrix B. Is $A_{ij} = B_{ij} = 1$ for some (i, j)? Needs $\Omega(n^2)$ bits of communication [Razborov 1992].
- Consider graph $G = (V, E)$ with

  $V = \{v_1, \ldots, v_n, u_1, \ldots, u_n, w_1, \ldots, w_n\}$ and $E = \{(v_i, u_i) : i \in [n]\}$

Lower Bound

Theorem

$\Omega(n^2)$ space required to determine if $T_3 \neq 0$ when $\delta = 1/3$.

- Reduce from set-disjointness: Alice has $n \times n$ binary matrix A, Bob has $n \times n$ binary matrix B. Is $A_{ij} = B_{ij} = 1$ for some (i,j)? Needs $\Omega(n^2)$ bits of communication [Razborov 1992].
- Consider graph $G = (V, E)$ with

  $$V = \{v_1, \ldots, v_n, u_1, \ldots, u_n, w_1, \ldots, w_n\} \text{ and } E = \{(v_i, u_i) : i \in [n]\}$$

- Alice runs algorithm on G and edges $\{(u_i, w_j) : A_{ij} = 1\}$.

Lower Bound

Theorem

$\Omega(n^2)$ *space required to determine if* $T_3 \neq 0$ *when* $\delta = 1/3$*.*

- Reduce from set-disjointness: Alice has $n \times n$ binary matrix A, Bob has $n \times n$ binary matrix B. Is $A_{ij} = B_{ij} = 1$ for some (i, j)? Needs $\Omega(n^2)$ bits of communication [Razborov 1992].
- Consider graph $G = (V, E)$ with

  $$V = \{v_1, \ldots, v_n, u_1, \ldots, u_n, w_1, \ldots, w_n\} \text{ and } E = \{(v_i, u_i) : i \in [n]\}$$

- Alice runs algorithm on G and edges $\{(u_i, w_j) : A_{ij} = 1\}$.
- Bob continues running algorithm on edges $\{(v_i, w_j) : B_{ij} = 1\}$.

Lower Bound

Theorem

$\Omega(n^2)$ space required to determine if $T_3 \neq 0$ when $\delta = 1/3$.

- Reduce from set-disjointness: Alice has $n \times n$ binary matrix A, Bob has $n \times n$ binary matrix B. Is $A_{ij} = B_{ij} = 1$ for some (i, j)? Needs $\Omega(n^2)$ bits of communication [Razborov 1992].
- Consider graph $G = (V, E)$ with

$$V = \{v_1, \ldots, v_n, u_1, \ldots, u_n, w_1, \ldots, w_n\} \text{ and } E = \{(v_i, u_i) : i \in [n]\}$$

- Alice runs algorithm on G and edges $\{(u_i, w_j) : A_{ij} = 1\}$.
- Bob continues running algorithm on edges $\{(v_i, w_j) : B_{ij} = 1\}$.
- $T_3 > 0$ iff $A_{ij} = B_{ij} = 1$ for some (i, j).

First Algorithm

Theorem (Sivakumar et al. 2002)

$\tilde{O}(\epsilon^{-2}(nm/T_3)^2 \log \delta^{-1})$ *space is sufficient.*

First Algorithm

Theorem (Sivakumar et al. 2002)
$\tilde{O}(\epsilon^{-2}(nm/T_3)^2 \log \delta^{-1})$ *space is sufficient.*

- Given stream of edges induce stream of node-triples:

$$\text{edge } (u, v) \text{ gives rise to } \{u, v, w\} \text{ for } w \in V \setminus \{u, v\}$$

First Algorithm

Theorem (Sivakumar et al. 2002)

$\tilde{O}(\epsilon^{-2}(nm/T_3)^2 \log \delta^{-1})$ *space is sufficient.*

- Given stream of edges induce stream of node-triples:

$$\text{edge } (u, v) \text{ gives rise to } \{u, v, w\} \text{ for } w \in V \setminus \{u, v\}$$

- Consider $F_k = \sum(\text{freq. of } \{\text{u,v,w}\})^k$ and note

$$\begin{pmatrix} F_0 \\ F_1 \\ F_2 \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 \\ 1 & 2 & 3 \\ 1 & 4 & 9 \end{pmatrix} \begin{pmatrix} T_1 \\ T_2 \\ T_3 \end{pmatrix}$$

  where T_i is the set of node-triples having exactly i edges in the induced subgraph.

First Algorithm

Theorem (Sivakumar et al. 2002)

$\tilde{O}(\epsilon^{-2}(nm/T_3)^2 \log \delta^{-1})$ *space is sufficient.*

- Given stream of edges induce stream of node-triples:

$$\text{edge } (u, v) \text{ gives rise to } \{u, v, w\} \text{ for } w \in V \setminus \{u, v\}$$

- Consider $F_k = \sum(\text{freq. of } \{\text{u,v,w}\})^k$ and note

$$\begin{pmatrix} F_0 \\ F_1 \\ F_2 \end{pmatrix} = \begin{pmatrix} 1 & 1 & 1 \\ 1 & 2 & 3 \\ 1 & 4 & 9 \end{pmatrix} \begin{pmatrix} T_1 \\ T_2 \\ T_3 \end{pmatrix}$$

  where T_i is the set of node-triples having exactly i edges in the induced subgraph.

- $T_3 = F_0 - 3F_1/2 + F_2/2$ so good approx. for F_0, F_1, F_2 suffice.

Second Algorithm

Theorem (Buriol et al. 2006)

$\tilde{O}(\epsilon^{-2}(nm/T_3)\log\delta^{-1})$ *space is sufficient.*

Second Algorithm

Theorem (Buriol et al. 2006)

$\tilde{O}(\epsilon^{-2}(nm/T_3)\log\delta^{-1})$ *space is sufficient.*

- Pick an edge $e_i = (u, v)$ uniformly at random from the stream.

Second Algorithm

Theorem (Buriol et al. 2006)
$\tilde{O}(\epsilon^{-2}(nm/T_3)\log\delta^{-1})$ *space is sufficient.*

- Pick an edge $e_i = (u, v)$ uniformly at random from the stream.
- Pick w uniformly at random from $V \setminus \{u, v\}$

Second Algorithm

Theorem (Buriol et al. 2006)

$\tilde{O}(\epsilon^{-2}(nm/T_3)\log\delta^{-1})$ *space is sufficient.*

- Pick an edge $e_i = (u, v)$ uniformly at random from the stream.
- Pick w uniformly at random from $V \setminus \{u, v\}$
- If $e_j = (u, w)$, $e_k = (v, w)$ for $j, k > i$ exist return 1; else 0.

Second Algorithm

Theorem (Buriol et al. 2006)

$\tilde{O}(\epsilon^{-2}(nm/T_3)\log\delta^{-1})$ *space is sufficient.*

- Pick an edge $e_i = (u, v)$ uniformly at random from the stream.
- Pick w uniformly at random from $V \setminus \{u, v\}$
- If $e_j = (u, w)$, $e_k = (v, w)$ for $j, k > i$ exist return 1; else 0.

Lemma

Expected outcome of algorithm is $\frac{T_3}{3m(n-2)}$.

Second Algorithm

Theorem (Buriol et al. 2006)

$\tilde{O}(\epsilon^{-2}(nm/T_3)\log\delta^{-1})$ *space is sufficient.*

- Pick an edge $e_i = (u, v)$ uniformly at random from the stream.
- Pick w uniformly at random from $V \setminus \{u, v\}$
- If $e_j = (u, w)$, $e_k = (v, w)$ for $j, k > i$ exist return 1; else 0.

Lemma

Expected outcome of algorithm is $\frac{T_3}{3m(n-2)}$.

- Repeat $O(\epsilon^{-2}(mn/t)\log\delta^{-1})$ times in parallel and scale average up by $3m(n-2)$.

Outline

Maximum Weight Matching

Problem

Stream of weighted edges (e, w_e)*: Find* $M \subset E$ *that maximizes* $\sum_{e \in M} w_e$ *such that no two edges in* M *share an endpoint.*

Maximum Weight Matching

Problem

Stream of weighted edges (e, w_e): Find $M \subset E$ that maximizes $\sum_{e \in M} w_e$ such that no two edges in M share an endpoint.

Warm-Up

An easy 2 approx. for unweighted case in $\tilde{O}(n)$ space?

Maximum Weight Matching

Problem

Stream of weighted edges (e, w_e): Find $M \subset E$ that maximizes $\sum_{e \in M} w_e$ such that no two edges in M share an endpoint.

Warm-Up

An easy 2 approx. for unweighted case in $\tilde{O}(n)$ space?

Theorem

$3 + 2\sqrt{2} = 5.83\ldots$ *approx. in $\tilde{O}(n)$ space.*

Maximum Weight Matching

Problem

Stream of weighted edges (e, w_e)*: Find* $M \subset E$ *that maximizes* $\sum_{e \in M} w_e$ *such that no two edges in* M *share an endpoint.*

Warm-Up

An easy 2 *approx. for unweighted case in* $\tilde{O}(n)$ *space?*

Theorem

$3 + 2\sqrt{2} = 5.83\ldots$ *approx. in* $\tilde{O}(n)$ *space.*

Improved to $5.59\ldots$ [Mariano 07] and $5.24\ldots$ [Sarma et al. 09].

Maximum Weight Matching

Problem

Stream of weighted edges (e, w_e)*: Find* $M \subset E$ *that maximizes* $\sum_{e \in M} w_e$ *such that no two edges in* M *share an endpoint.*

Warm-Up

An easy 2 *approx. for unweighted case in* $\tilde{O}(n)$ *space?*

Theorem

$3 + 2\sqrt{2} = 5.83\ldots$ *approx. in* $\tilde{O}(n)$ *space.*

Improved to $5.59\ldots$ [Mariano 07] and $5.24\ldots$ [Sarma et al. 09].

Open Problem

Prove a lower bound or a much better algorithm!

An Algorithm

- At all times maintain a matching M, initially $M = \emptyset$.

An Algorithm

- At all times maintain a matching M, initially $M = \emptyset$.
- On seeing $e = (u, v)$, suppose $e' = (u, u_1), e'' = (v, u_2) \in M$

An Algorithm

- At all times maintain a matching M, initially $M = \emptyset$.
- On seeing $e = (u, v)$, suppose $e' = (u, u_1), e'' = (v, u_2) \in M$
- If $w_e \geq (1 + \gamma)(w_{e'} + w_{e''})$, $M \leftarrow M \cup \{e\} \setminus \{e', e''\}$

An Algorithm

- At all times maintain a matching M, initially $M = \emptyset$.
- On seeing $e = (u, v)$, suppose $e' = (u, u_1), e'' = (v, u_2) \in M$
- If $w_e \geq (1 + \gamma)(w_{e'} + w_{e''})$, $M \leftarrow M \cup \{e\} \setminus \{e', e''\}$

For the analysis we use the following definitions to describe the execution of the algorithm:

An Algorithm

- At all times maintain a matching M, initially $M = \emptyset$.
- On seeing $e = (u, v)$, suppose $e' = (u, u_1), e'' = (v, u_2) \in M$
- If $w_e \geq (1 + \gamma)(w_{e'} + w_{e''})$, $M \leftarrow M \cup \{e\} \setminus \{e', e''\}$

For the analysis we use the following definitions to describe the execution of the algorithm:

An Algorithm

- At all times maintain a matching M, initially $M = \emptyset$.
- On seeing $e = (u, v)$, suppose $e' = (u, u_1), e'' = (v, u_2) \in M$
- If $w_e \geq (1 + \gamma)(w_{e'} + w_{e''})$, $M \leftarrow M \cup \{e\} \setminus \{e', e''\}$

For the analysis we use the following definitions to describe the execution of the algorithm:

- An edge e kills an edge e' if e' was removed when e arrives.

An Algorithm

- At all times maintain a matching M, initially $M = \emptyset$.
- On seeing $e = (u, v)$, suppose $e' = (u, u_1), e'' = (v, u_2) \in M$
- If $w_e \geq (1 + \gamma)(w_{e'} + w_{e''})$, $M \leftarrow M \cup \{e\} \setminus \{e', e''\}$

For the analysis we use the following definitions to describe the execution of the algorithm:

- An edge e kills an edge e' if e' was removed when e arrives.
- We say an edge is a survivor if it's in the final matching.

An Algorithm

- At all times maintain a matching M, initially $M = \emptyset$.
- On seeing $e = (u, v)$, suppose $e' = (u, u_1), e'' = (v, u_2) \in M$
- If $w_e \geq (1 + \gamma)(w_{e'} + w_{e''})$, $M \leftarrow M \cup \{e\} \setminus \{e', e''\}$

For the analysis we use the following definitions to describe the execution of the algorithm:

- An edge e kills an edge e' if e' was removed when e arrives.
- We say an edge is a survivor if it's in the final matching.
- For survivor e, the trail of the dead is $T(e) = C_1 \cup C_2 \cup \ldots$, where $C_0 = \{e\}$ and

$$C_i = \cup_{e' \in C_{i-1}} \{\text{edges killed by } e'\}$$

Analysis

Lemma

Let S be set of survivors and $w(S)$ be weight of final matching.

1. $w(T(S)) \leq w(S)/\gamma$
2. $\textsc{opt} \leq (1+\gamma)(w(T(S)) + 2w(S))$

Approximation factor is $1/\gamma + 3 + 2\gamma$ *and* $\gamma = 1/\sqrt{2}$ *gives result.*

Analysis

Lemma

Let S be set of survivors and $w(S)$ be weight of final matching.

1. $w(T(S)) \leq w(S)/\gamma$
2. $\textsc{opt} \leq (1+\gamma)\,(w(T(S)) + 2w(S))$

Approximation factor is $1/\gamma + 3 + 2\gamma$ *and* $\gamma = 1/\sqrt{2}$ *gives result.*

Proof.

$\square$

Analysis

Lemma

Let S be set of survivors and $w(S)$ be weight of final matching.

1. $w(T(S)) \leq w(S)/\gamma$
2. $\textsc{opt} \leq (1+\gamma)\,(w(T(S)) + 2w(S))$

Approximation factor is $1/\gamma + 3 + 2\gamma$ *and* $\gamma = 1/\sqrt{2}$ *gives result.*

Proof.

1. Consider $e \in S$:

$$(1+\gamma)w(T(e)) = \sum_{i\geq 1}(1+\gamma)w(C_i) \leq \sum_{i\geq 0} w(C_i) = w(T(e)) + w(e)$$

□

Analysis

Lemma

Let S be set of survivors and $w(S)$ be weight of final matching.

1. $w(T(S)) \leq w(S)/\gamma$
2. $\textsc{opt} \leq (1+\gamma)\,(w(T(S)) + 2w(S))$

Approximation factor is $1/\gamma + 3 + 2\gamma$ and $\gamma = 1/\sqrt{2}$ gives result.

Proof.

1. Consider $e \in S$:

$$(1{+}\gamma)w(T(e)) = \sum_{i\geq 1}(1{+}\gamma)w(C_i) \leq \sum_{i\geq 0} w(C_i) = w(T(e)){+}w(e)$$

2. Can charge the weights of edges in $\textsc{opt}$ to the $S \cup T(S)$ such that each edge $e \in T(S)$ is charged at most $(1+\gamma)w(e)$ and each edge $e \in S$ is charged at most $2(1+\gamma)w(e)$.

$\square$

Outline

k-center

Problem

Given a stream of distinct points $X = \{p_1, \ldots, p_n\}$ from a metric space $(\mathcal{X}, d)$, find the set of k points $Y \subset X$ that minimizes:

$$\max_i \min_{y \in Y} d(p_i, y)$$

k-center

Problem

Given a stream of distinct points $X = \{p_1, \ldots, p_n\}$ from a metric space $(\mathcal{X}, d)$, find the set of k points $Y \subset X$ that minimizes:

$$\max_i \min_{y \in Y} d(p_i, y)$$

Warm-Up

- *Find 2-approx. if you're given* $\textsc{opt}$.
- *Find $(2 + \epsilon)$-approx. if you're given that $a \leq \textsc{opt} \leq b$*

k-center

Problem

Given a stream of distinct points $X = \{p_1, \ldots, p_n\}$ from a metric space $(\mathcal{X}, d)$, find the set of k points $Y \subset X$ that minimizes:

$$\max_i \min_{y \in Y} d(p_i, y)$$

Warm-Up

- *Find 2-approx. if you're given* $\textsc{opt}$.
- *Find $(2+\epsilon)$-approx. if you're given that $a \leq \textsc{opt} \leq b$*

Theorem (Khuller and McCutchen 2009, Guha 2009)

$(2+\epsilon)$ *approx. for metric k-center in $\tilde{O}(k\epsilon^{-1} \log \epsilon^{-1})$ space.*

k-center: Algorithm and Analysis

- Consider first $k+1$ points: this gives a lower bound a on $\textsc{opt}$.

k-center: Algorithm and Analysis

- Consider first $k+1$ points: this gives a lower bound a on $\textsc{opt}$.
- Instantiate basic algorithm with guesses

$$\ell_1 = a,\ \ell_2 = (1+\epsilon)a,\ \ell_3 = (1+\epsilon)^2 a, \ldots\ \ell_{1+t} = O(\epsilon^{-1})a$$

k-center: Algorithm and Analysis

- Consider first $k+1$ points: this gives a lower bound a on $\textsc{opt}$.
- Instantiate basic algorithm with guesses

$$\ell_1 = a,\ \ell_2 = (1+\epsilon)a,\ \ell_3 = (1+\epsilon)^2 a, \ldots\ \ell_{1+t} = O(\epsilon^{-1})a$$

- Say instantiation goes bad if it tries to open $(k+1)$-th center

k-center: Algorithm and Analysis

- Consider first $k+1$ points: this gives a lower bound a on $\textsc{opt}$.
- Instantiate basic algorithm with guesses

$$\ell_1 = a,\ \ell_2 = (1+\epsilon)a,\ \ell_3 = (1+\epsilon)^2 a, \ldots\ \ell_{1+t} = O(\epsilon^{-1})a$$

- Say instantiation goes bad if it tries to open $(k+1)$-th center
- Suppose instantiation with guess ℓ goes bad when processing $(j+1)$-th point

k-center: Algorithm and Analysis

- Consider first $k+1$ points: this gives a lower bound a on OPT.
- Instantiate basic algorithm with guesses

$$\ell_1 = a,\ \ell_2 = (1+\epsilon)a,\ \ell_3 = (1+\epsilon)^2 a, \ldots\ \ell_{1+t} = O(\epsilon^{-1})a$$

- Say instantiation goes bad if it tries to open $(k+1)$-th center
- Suppose instantiation with guess ℓ goes bad when processing $(j+1)$-th point
  - Let $q_1, \ldots, q_k$ be centers chosen so far.

k-center: Algorithm and Analysis

- Consider first $k+1$ points: this gives a lower bound a on OPT.
- Instantiate basic algorithm with guesses

$$\ell_1 = a,\ \ell_2 = (1+\epsilon)a,\ \ell_3 = (1+\epsilon)^2 a, \ldots\ \ell_{1+t} = O(\epsilon^{-1})a$$

- Say instantiation goes bad if it tries to open $(k+1)$-th center
- Suppose instantiation with guess ℓ goes bad when processing $(j+1)$-th point
  - Let $q_1, \ldots, q_k$ be centers chosen so far.
  - Then $p_1, \ldots, p_j$ are all at most 2ℓ from a q_i.

k-center: Algorithm and Analysis

- Consider first $k+1$ points: this gives a lower bound a on $\textsc{opt}$.
- Instantiate basic algorithm with guesses

$$\ell_1 = a,\ \ell_2 = (1+\epsilon)a,\ \ell_3 = (1+\epsilon)^2 a, \ldots\ \ell_{1+t} = O(\epsilon^{-1})a$$

- Say instantiation goes bad if it tries to open $(k+1)$-th center
- Suppose instantiation with guess ℓ goes bad when processing $(j+1)$-th point
  - Let $q_1, \ldots, q_k$ be centers chosen so far.
  - Then $p_1, \ldots, p_j$ are all at most 2ℓ from a q_i.
  - Optimum for $\{q_1, \ldots, q_k, p_{j+1}, \ldots, p_n\}$ is at most $\textsc{opt} + 2\ell$.

k-center: Algorithm and Analysis

- Consider first $k+1$ points: this gives a lower bound a on OPT.
- Instantiate basic algorithm with guesses

$$\ell_1 = a,\ \ell_2 = (1+\epsilon)a,\ \ell_3 = (1+\epsilon)^2 a, \ldots\ \ell_{1+t} = O(\epsilon^{-1})a$$

- Say instantiation goes bad if it tries to open $(k+1)$-th center
- Suppose instantiation with guess ℓ goes bad when processing $(j+1)$-th point
  - Let $q_1, \ldots, q_k$ be centers chosen so far.
  - Then $p_1, \ldots, p_j$ are all at most 2ℓ from a q_i.
  - Optimum for $\{q_1, \ldots, q_k, p_{j+1}, \ldots, p_n\}$ is at most $\text{OPT} + 2\ell$.
- Hence, for an instantiation with guess $2\ell/\epsilon$ only incurs a small if we use $\{q_1, \ldots, q_k, p_{j+1}, \ldots, p_n\}$ rather than $\{p_1, \ldots, p_n\}$.

Outline

Distance Estimation

Problem

Stream of unweighted edges E defines a shortest path graph metric $d_G : V \times V \to \mathbb{N}$. For $u, v \in V$, estimate $d_G(u, v)$.

Distance Estimation

Problem

Stream of unweighted edges E defines a shortest path graph metric $d_G : V \times V \to \mathbb{N}$. For $u, v \in V$, estimate $d_G(u, v)$.

Definition

An α-spanner of a graph $G = (V, E)$ is a subgraph $H = (V, E')$ such that for all u, v,

$$d_G(u, v) \leq d_H(u, v) \leq \alpha d_G(u, v)$$

Distance Estimation

Problem

Stream of unweighted edges E defines a shortest path graph metric $d_G : V \times V \to \mathbb{N}$. For $u, v \in V$, estimate $d_G(u, v)$.

Definition

An α-spanner of a graph $G = (V, E)$ is a subgraph $H = (V, E')$ such that for all u, v,

$$d_G(u, v) \leq d_H(u, v) \leq \alpha d_G(u, v)$$

Warm-Up

$2t - 1$ *spanner using* $\tilde{O}(n^{1+1/t})$ *space.*

Distance Estimation

Problem

Stream of unweighted edges E defines a shortest path graph metric $d_G : V \times V \to \mathbb{N}$. For $u, v \in V$, estimate $d_G(u, v)$.

Definition

An α-spanner of a graph $G = (V, E)$ is a subgraph $H = (V, E')$ such that for all u, v,

$$d_G(u, v) \leq d_H(u, v) \leq \alpha d_G(u, v)$$

Warm-Up

$2t - 1$ *spanner using* $\tilde{O}(n^{1+1/t})$ *space.*

Theorem (Elkin 2007)

$2t - 1$ *stretch spanner using* $\tilde{O}(n^{1+1/t})$ *space with constant update time.*

Towards better results if you're allowed mulitple passes...

Problem

Can we get better approximation for $d_G(u, v)$ with multiple passes?

Towards better results if you're allowed mulitple passes...

Problem

Can we get better approximation for $d_G(u, v)$ with multiple passes?

Warm-Up

Find $d_G(u, v)$ exactly in $\tilde{O}(n^{1+\gamma})$ space and $\tilde{O}(n^{1-\gamma})$ passes.

Towards better results if you're allowed mulitple passes. . .

Problem

Can we get better approximation for $d_G(u, v)$ with multiple passes?

Warm-Up

Find $d_G(u, v)$ exactly in $\tilde{O}(n^{1+\gamma})$ space and $\tilde{O}(n^{1-\gamma})$ passes.

Theorem

$O(k)$ approx in $\tilde{O}(n)$ space with $O(n^{1/k})$ passes.

Towards better results if you're allowed mulitple passes...

Problem

Can we get better approximation for $d_G(u, v)$ with multiple passes?

Warm-Up

Find $d_G(u, v)$ exactly in $\tilde{O}(n^{1+\gamma})$ space and $\tilde{O}(n^{1-\gamma})$ passes.

Theorem

$O(k)$ approx in $\tilde{O}(n)$ space with $O(n^{1/k})$ passes.

Theorem (via Thorup, Zwick 2006)

$(1+\epsilon)$ approx in $\tilde{O}(n)$ space with $n^{O(\log \epsilon^{-1})/\log\log n}$ passes.

Ramsey Partition Approach

Definition (Mendel, Naor 2006)

Ramsey Partition $\mathcal{P}_\Delta$ is a random partion of metric space. Each cluster has diameter at most Δ and for $t \leq \Delta/8$,

$$\Pr(B_X(x,t) \in \mathcal{P}_\Delta) \geq \left(\frac{|B_X(x,\Delta/8)|}{|B_X(x,\Delta)|}\right)^{16t/\Delta} \geq \left(\frac{1}{n}\right)^{16t/\Delta} .$$

Can construct in stream model in $\tilde{O}(n)$ space and $O(\Delta)$ passes.

Ramsey Partition Approach

Definition (Mendel, Naor 2006)

Ramsey Partition $\mathcal{P}_\Delta$ is a random partion of metric space. Each cluster has diameter at most Δ and for $t \leq \Delta/8$,

$$\Pr(B_X(x,t) \in \mathcal{P}_\Delta) \geq \left(\frac{|B_X(x,\Delta/8)|}{|B_X(x,\Delta)|}\right)^{16t/\Delta} \geq \left(\frac{1}{n}\right)^{16t/\Delta} .$$

Can construct in stream model in $\tilde{O}(n)$ space and $O(\Delta)$ passes.

Algorithm

1. *Sample "beacons" $b_1, \ldots, b_{n^{1-1/k}}$ including s and t from V*

Ramsey Partition Approach

Definition (Mendel, Naor 2006)

Ramsey Partition $\mathcal{P}_\Delta$ is a random partion of metric space. Each cluster has diameter at most Δ and for $t \leq \Delta/8$,

$$\Pr(B_X(x,t) \in \mathcal{P}_\Delta) \geq \left(\frac{|B_X(x,\Delta/8)|}{|B_X(x,\Delta)|}\right)^{16t/\Delta} \geq \left(\frac{1}{n}\right)^{16t/\Delta} .$$

Can construct in stream model in $\tilde{O}(n)$ space and $O(\Delta)$ passes.

Algorithm

1. *Sample "beacons" $b_1, \ldots, b_{n^{1-1/k}}$ including s and t from V*
2. *Repeat $O(n^{1/k} \log n)$ times:*
    2.1 *Create RP with diameter $\Delta \approx kn^{1/k}$ and consider $t \approx n^{1/k}$.*
    2.2 *For each beacon, add Δ-weighted edge to center of its cluster.*

Summary: We looked at some nice problems, our curiousity is piqued, and now we want to start finding more problems to solve.

Thanks!