

Distance Oracle, Routing Schemes

Shruti Rastogi

Visiting Student, IIT Delhi

CS Theory Winter School 2023

Table of Contents

Introduction

Distance Oracle

Tools and Algorithm

Routing Scheme

Introduction

Application of Distance Approximation

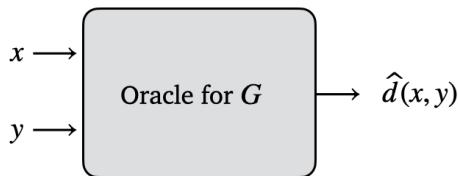
- ▶ network routing
- ▶ traffic engineering
- ▶ distributed computing

Objectives

1. To compute **compact** data-structure for **quickly** reporting approximate distances.
2. Designing scheme/procedure for routing packets along approximate shortest-paths in a network.

Distance Oracle

An oracle is a **compact** data-structure (typically of size $\ll n^2$) to **quickly** report exact/approximate graph distances.



Stretch of Oracle:

An oracle is said to have stretch (α, β) if for every $x, y \in V$:

$$\hat{d}(x, y) \leq \alpha \cdot \text{dist}_G(x, y) + \beta$$

Some Trivial Distance Oracles

1. Store Distance Matrix

- ▶ Space: $O(n^2)$
- ▶ Query time: $O(1)$

2. Store graph G

- ▶ Space: $O(m + n)$
- ▶ Query time: $O(m + n)$

In above oracles $(\alpha, \beta) = (1, 0)$

Approximation

Why Approximation?

Any exact solution requires n^2 bits

($\text{dist}_G(v_i, v_j) = 1$ iff (v_i, v_j) is an edge).

So, using the oracle one can output all entries of adjacency matrix.

	v_1	v_2	$\dots$	v_n
v_1	0	1	$\dots$	1
v_2	1	0	$\dots$	0
$\vdots$	$\vdots$	$\vdots$		$\vdots$
v_n	1	0	$\dots$	0

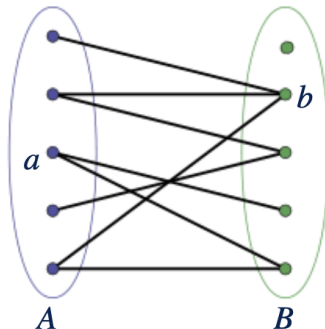
Natural Question:

Obtain efficient space and query time at the cost of approximation.

Approximation

Can we have $\alpha < 3$ and $\beta = 0$?

Hint:



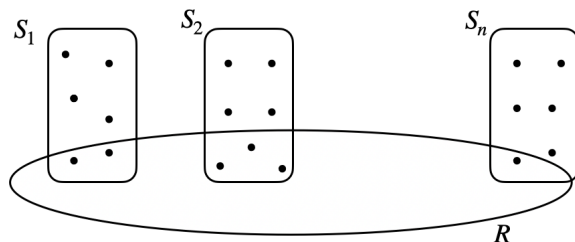
This talk

Theorem [BCC⁺23]

For any undirected graph $G = (V, E)$, any $\epsilon > 0$ such that $\frac{1}{\epsilon} \in \mathbb{Z}$, and any $L \leq n$, we can compute a distance oracle satisfying:

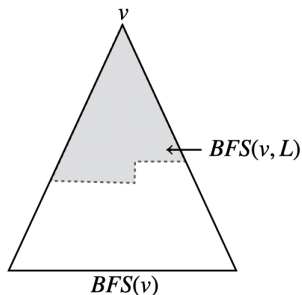
- ▶ Stretch: $(1 + 2\epsilon, 2)$
- ▶ Size: $o(n^2)$
- ▶ Query Time: $o(n)$

Tool 1: Hitting Set



- If $S_1, \dots, S_n \subseteq V$ are of size L , then we can compute a set R of size $O(\frac{n}{L} \log n)$ satisfying $R \cap S_i \neq \emptyset$ for each i .
- Set R is said to be Hitting Set for $(S_1, \dots, S_n)$.

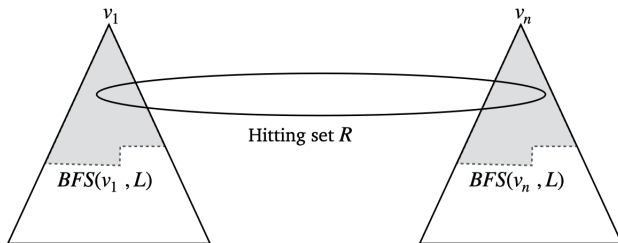
Tool 2: Partial BFS (up to first L vertices)



$BFS(v, L)$ stores closest L vertices to v

Oracle stores:

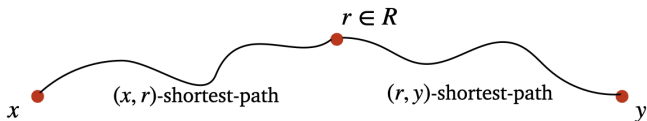
1. $\text{BFS}(v_i, L)$, and the corresponding distances from “ v_i ”
2. Hitting set R for $\text{BFS}(v_i, L)$, and distance matrix: $R \times V$.



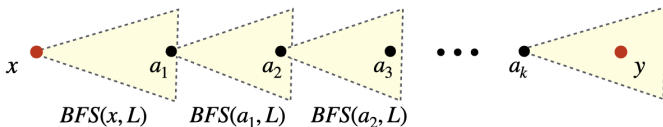
$$\text{Total space} = O(nL) + O\left(n \cdot \frac{n}{L} \log n\right)$$

Main Idea

- Either consider concatenation of (x, r) -shortest-path and (r, y) -shortest-path, for some $r \in R$.



- Or perform traversal using partial BFS trees up to some k hops.



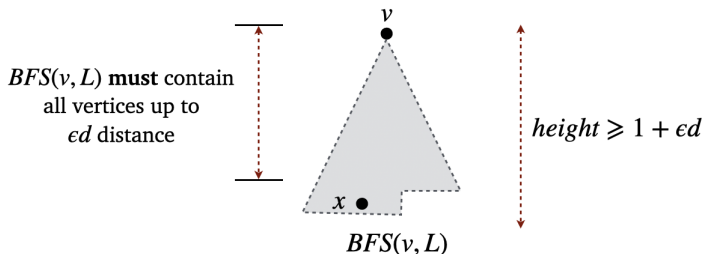
k will be small number

Observation

Observation

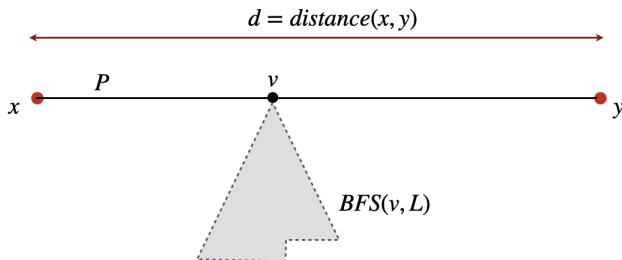
If $\text{HEIGHT}(\text{BFS}(v, L)) \geq 1 + \epsilon d$, then $\text{BFS}(v, L)$ contains ALL vertices up to “ ϵd ” distance from v .

Proof



Two Cases

Consider $x, y \in V$. Let P be an x - y shortest path.

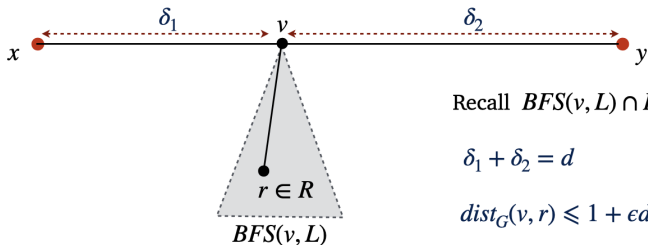


For any $v \in P$:

1. $\text{HEIGHT}(\text{BFS}(v, L)) \leq 1 + \epsilon d$, or
2. $\text{HEIGHT}(\text{BFS}(v, L)) \geq 1 + \epsilon d$

Case 1: $\text{HEIGHT}(\text{BFS}(v, L)) \leq 1 + \epsilon d$, for some $v \in P$

- Report: $\min_{r \in R} (\text{dist}_G(x, r) + \text{dist}_G(r, y))$



Recall $\text{BFS}(v, L) \cap R \neq \emptyset$

$$\delta_1 + \delta_2 = d$$

$$\text{dist}_G(v, r) \leq 1 + \epsilon d$$

$$\Rightarrow \text{dist}_G(x, r) \leq \delta_1 + 1 + \epsilon d$$

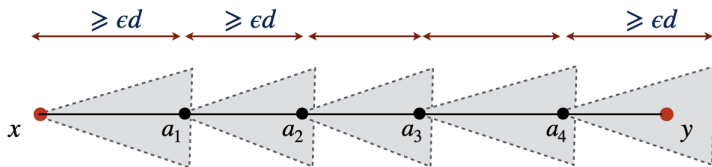
$$\Rightarrow \text{dist}_G(r, y) \leq \delta_2 + 1 + \epsilon d$$

- Stretch: $(1 + 2\epsilon, 2)$
- Time: $O(|R|) = O(\frac{n}{L} \log n)$

Case 2: $\text{HEIGHT}(\text{BFS}(v, L)) \geq 1 + \epsilon d$, for all $v \in P$

Observation

If $\text{HEIGHT}(\text{BFS}(v, L)) \geq 1 + \epsilon d$, then $\text{BFS}(v, L)$ contains ALL vertices up to " ϵd " distance from v .



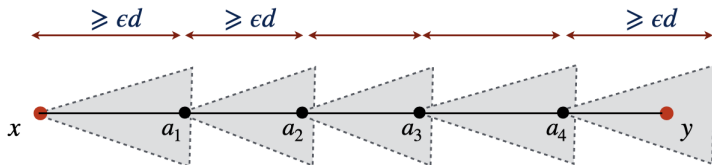
Hops needed to cover the shortest path: $\leq \frac{1}{\epsilon}$.

Case 2: $\text{HEIGHT}(\text{BFS}(v, L)) \geq 1 + \epsilon d$, for all $v \in P$

Implicit graph H :

For each $a \in V$ and $b \in \text{BFS}(a, L)$:

H contains a directed ($a \rightarrow b$) edge of weight $\text{dist}_G(a, b)$.



Algorithm: Compute distances from x in H up to $\frac{1}{\epsilon}$ hops.

Time: $O(\max_{outdeg}(H)^{\frac{1}{\epsilon}}) = O(L^{\frac{1}{\epsilon}})$

Algorithm

Preprocessing

1. Compute random set R of $O(\frac{n}{L} \log n)$ size, and distance matrix: $R \times V$.
2. Store $\text{BFS}(v_i, L)$, and the corresponding distances from v_i .
(Or equivalently, we store H).

Query(x, y)

1. $d_1 = \min_{r \in R} (\text{dist}_G(x, r) + \text{dist}_G(r, y))$
2. d_2 = Distance from x to y obtained by performing traversal in H up to $\frac{1}{\epsilon}$ depth.
3. Return $\min(d_1, d_2)$.

Result

- ▶ Stretch: $(1 + 2\epsilon, 2)$
- ▶ Query time: $O(\frac{n}{L} \log n + L^{\frac{1}{\epsilon}})$
- ▶ Space: $O(nL + \frac{n^2}{L})$

What if G has integer edge weights in range $[1, W]$?

- ▶ Stretch: $O(1 + 2\epsilon, 2W)$
- ▶ Query time: $O\left(\left(\frac{n}{L} \log n + L^{\frac{1}{\epsilon}}\right) \cdot \left(1 + \frac{\log W}{\log n}\right)\right)$
- ▶ Space: $O(nL + \frac{n^2}{L} \log n)$

Remark

Algorithm remains same, but analysis changes.

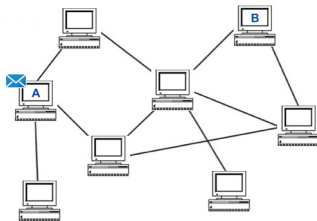
Can we get constant query time?

Stretch	Query Time	Space
3	$O(3)$	$O(n^{1.5})$
5	$O(5)$	$O(n^{\frac{4}{3}})$
$\vdots$		
$2k - 1$	$O(k)$	$O(n^{1+\frac{1}{k}})$

Table: Approximate distance oracles with multiplicative stretch ($\beta = 0$).
Thorup and Zwick [TZ01a]

Routing

Routing scheme: Distributed algorithm to deliver messages from any node to any other node of the network.



Header: Each message appends information such as destination, routing information, etc. in its header.

Routing Table: Local table at each node processes information from header to decide where to pass the message.

Routing Scheme

We achieve:

- ▶ $(1 + 2\epsilon, 2)$ stretch
- ▶ $O(L + \frac{n}{L} \log n)$ sized routing table
- ▶ $O(\epsilon^{-1} \log n)$ -bits header
- ▶ Routing decisions in *constant* time.

Key Ideas I

Routing on Trees

We use Routing scheme for following trees:

1. $\text{BFS}(r)$, for each $r \in R$.
2. $\text{BFS}(v, L)$, for each $v \in V$.

Size of Routing Table of w :

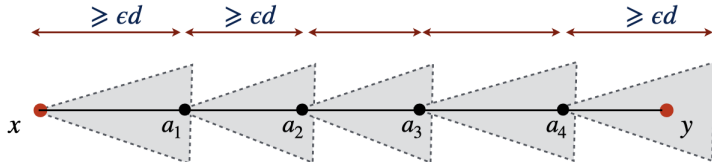
$$O(|R| + \text{Number of } \text{BFS}(v, L) \text{ that contains } w)$$

To ensure a worst case $O(\frac{n}{L} \log n + L)$ size bound on routing table, we use a modified algorithm to compute R given by [TZ01b].

Key Ideas II

Message Headers

The header stores information of $\frac{1}{\epsilon}$ hop vertices (used in Case 2), or the node $r \in R$ (used in Case 1):



Round I: Use broadcasting to compute Header information.

Subsequent Rounds: Use header information for an efficient routing.

References I



Davide Bilò, Shiri Chechik, Keerti Choudhary, Sarel Cohen, Tobias Friedrich, and Martin Schirneck.

Improved approximate distance oracles: Bypassing the thorup-zwick bound in dense graphs.

CoRR, abs/2307.11677, 2023.



Mikkel Thorup and Uri Zwick.

Approximate distance oracles.

In Jeffrey Scott Vitter, Paul G. Spirakis, and Mihalis Yannakakis, editors, *Proceedings on 33rd Annual ACM Symposium on Theory of Computing, July 6-8, 2001, Heraklion, Crete, Greece*, pages 183–192. ACM, 2001.

References II



Mikkel Thorup and Uri Zwick.

Compact routing schemes.

In Arnold L. Rosenberg, editor, *Proceedings of the Thirteenth Annual ACM Symposium on Parallel Algorithms and Architectures, SPAA 2001, Heraklion, Crete Island, Greece, July 4-6, 2001*, pages 1–10. ACM, 2001.