Linear Space Logarithmic Stretch Path-Reporting Distance Oracles for General Graphs

Michael Elkin

Ben-Gurion University, Beer-Sheva

joint w/ S. Pettie

University of Michigan, Ann Arbor

Distance Oracles

Given an n-vertex undirected weighted $G = (V, E)$,
build a *compact* data structure (D.O.)
s.t. given $u, v \in V$,
it returns a path $\Pi(u, v)$ of length $\delta(u, v)$,

$$d_G(u, v) \leq \delta(u, v) \leq \alpha \cdot d_G(u, v)$$

for *small* α, *quickly*
(in time $O(|\Pi(u, v)| + something\_small)$).

Three parameters:

- Size,
- Stretch,
- Query time.

Thorup-Zwick'01 Distance Oracle

For any integer $1 \leq k \leq \log n$:

Size $= O(k \cdot n^{1+1/k})$,

Stretch $= 2k - 1$,

Query $= O(k + |\Pi(u, v)|)$.

(Query time was recently improved to $O(\log k)$ by Wulff-Nilsen'12 and to $O(1)$ by Chechik'14.)

Almost tight except for large k.

But the size is always $\Omega(n \cdot \log n)$.

Additional Oracles

Mendel-Naor's Oracle (2006):

Size $= O(n^{1+1/k})$, // No factor of k

Stretch $= O(k)$, // instead of $2k-1$

Query $= O(1)$

But *not path-reporting*!

Elkin-Neiman-Wulff-Nilsen's oracle (2014):

Size $= O(n \cdot t)$,

Stretch $= O(\sqrt{t} \cdot n^{2/\sqrt{t}})$ // Huge!

Query $= O(\log t \cdot \log_n \omega_{max} + |\Pi(u,v)|)$.

Size is $o(n \log n)$, but the stretch explodes!

The Trivial Oracle

1. Build a $(2k-1)$-spanner H
   w/ $O(n^{1+1/k})$ edges.

   [Althofer,Das,Dobkin,Joseph,Soares'91]

   $[d_H(u,v) \leq (2k-1)d_G(u,v) \quad \forall u,v \in V]$

2. Given $u, v \in V$ find shortest path in H
   between u and v (using Dijkstra).

Size $= O(n^{1+1/k})$,

Stretch $= 2k-1$,

Query $= O(|H|) = O(n^{1+1/k})$.

Query is too large!

Our Path-Reporting D.O.s for General Graphs

1) Size $= O(n^{1+1/k})$,
Stretch $= (2k-1) \cdot O\left((1/\epsilon)^{\log_{4/3} 7}\right)$,
Query $= O(n^{\epsilon} + |\Pi(u,v)|)$.

$\forall$ *constant* $\epsilon > 0$ we have:

size $= O(n^{1+1/k})$, stretch $= O(k)$, query $= n^{\epsilon}$.

In particular, for $k = \log n$:

size $= O(n)$, stretch $= O(\log n)$, query $= n^{\epsilon}$.

2) Size $= O(n \cdot \log\log n)$,
Stretch $= O(\log^{\log_{4/3} 7} n)$,
Query $= O(\log\log n + |\Pi(u,v)|)$.

Much smaller *query* time,
but *polylogarithmic stretch*.

Path-Reporting Almost Additive D.O.s for Unweighted Graphs

[Elkin,Peleg'01] (Almost additive spanners):

Def: $d_H(u,v) \leq (1+\epsilon) \cdot d_G(u,v) + \beta$, $\forall u, v \in V$.

$(1+\epsilon, \beta)$-*spanners* H w/ $O(\beta \cdot n^{1+1/k})$ edges, $\beta = \left(\frac{\log k}{\epsilon}\right)^{O(\log k)}$, are known.

Additional constructions of $(1+\epsilon, \beta)$-spanners were given in [Elkin'01],[Thorup-Zwick'06], [Pettie'08].

But no non-trivial D.O.s w/ analogous properties are known.

(The trivial D.O. has query time $= O(n^{1+1/k})$.)

Our Path-Reporting D.O.s for Unweighted Graphs w/ Hybrid Stretch

Size $= O(k^{O(1)} \cdot n^{1+1/k})$,

Stretch $= (O(1), \beta(k))$, $\beta(k) = k^{O(1)}$.

Query $= O(n^{\epsilon} + |\Pi(u, v)|)$,
(for constant $\epsilon > 0$).

These are the first D.O.s w/ stretch-size tradeoff below the lower bound curve given by multiplicative stretch k versus size $O(n^{1+1/k})$.

Path-Reporting D.O.s for Sparse Graphs

For graphs w/ $O(n)$ edges:

1) Our path-reporting D.O. has

size $= O(n)$,

stretch $= O(k^{\log_{4/3} 7})$,

query $= O(n^{1/k})$.

2) Previous linear-size D.O.s for such graphs are *not path-reporting*,
but exhibit a better tradeoff.
[Agarwal,Godfrey,Har-Peled'11]

size $= O(n)$,

stretch $= 4k + 1$,

query $= O(n^{\frac{1}{k+1}})$.

Distance-Preserving Path-Reporting Oracle (DPPRO)

[Coppersmith,Elkin'05]

For an n-vertex (possibly directed weighted) graph $G = (V, E)$ and $\mathcal{P} = \{(s_i, t_i) \mid 1 \leq i \leq P\}$, $\exists$ subgraph $G' = (V, H)$ w/

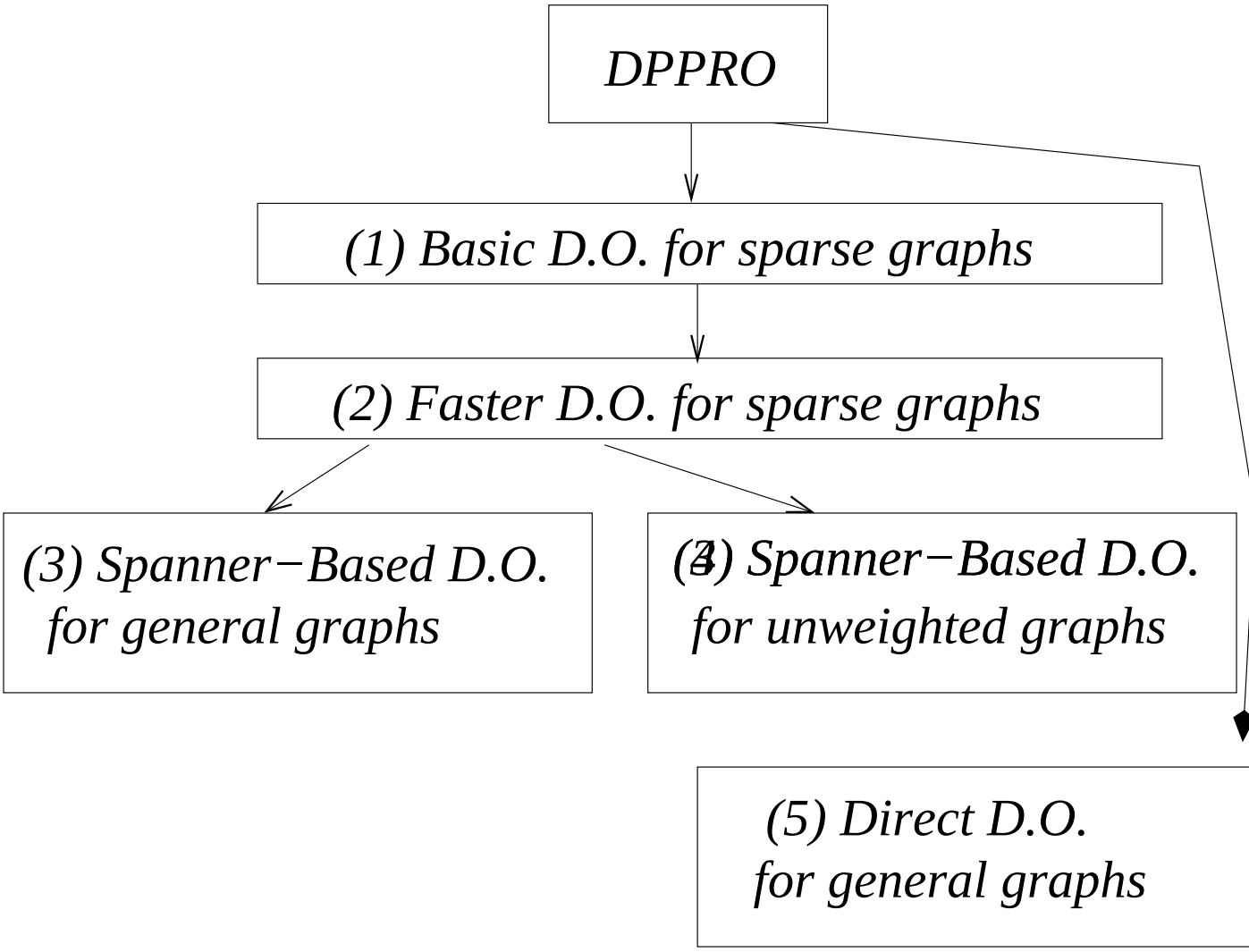
$$|H| = O(\max\{n + \sqrt{n} \cdot P, \sqrt{P} \cdot n\})$$

s.t. $\forall (u, v) \in \mathcal{P}$, $d_H(u, v) = d_G(u, v)$.

We devise a D.O.-counterpart of [CE05]:

size $= O(n + P^2)$.
Given a query $(s_i, t_i) \in \mathcal{P}$,
the D.O. produces shortest $\Pi(s_i, t_i)$
in time $O(|\Pi(s_i, t_i)|)$.

General Outline



(1) Basic D.O. for sparse graphs:
Size $O(n)$, stretch $6k-1$, Query $n^{1/2+O(1/k)}$.

(2) Faster D.O. for sparse graphs:
Size $O(n \cdot \log k)$, stretch $O(k^{\log_{4/3} 7})$,
Query $O(n^{1/k})$.

(3) Spanner-Based D.O. for general graphs:
Size $O(n^{1+1/k})$, stretch $O(k/\epsilon) = O(k)$,
Query $O(n^{\epsilon})$.

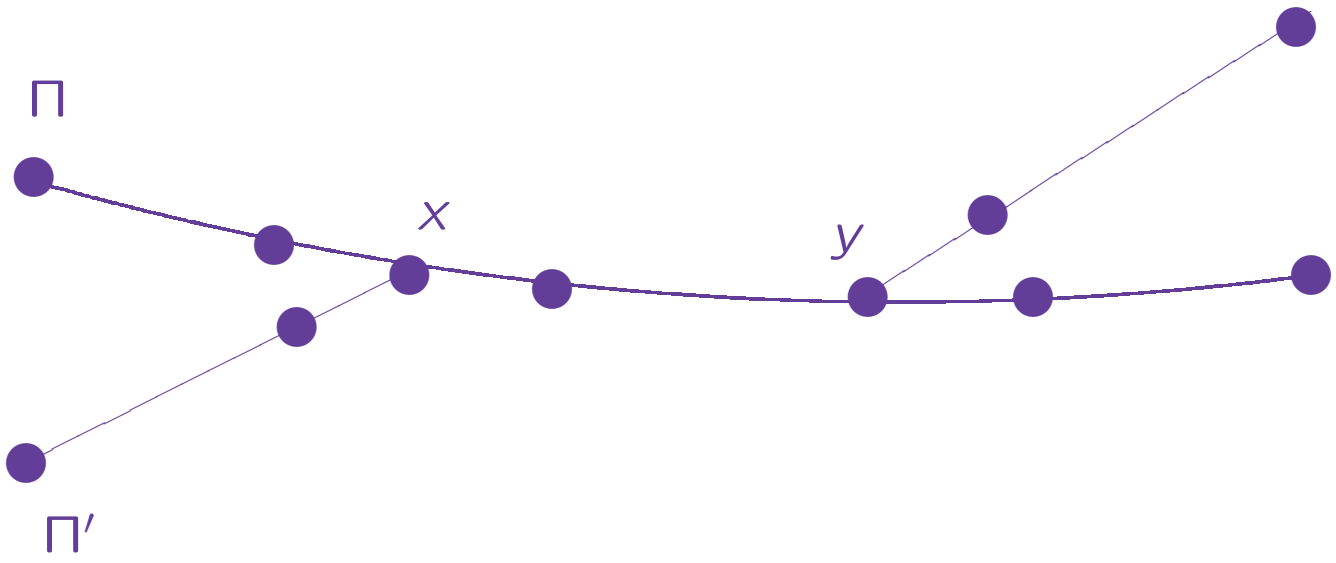
(4) Spanner-Based D.O. for unweighted graphs:
Size $O(n^{1+1/k})$, stretch $(O(1), \beta(k))$,
Query $O(n^{\epsilon})$.

(5) A direct D.O. for general graphs:
Size $O(n \cdot \log\log n)$, stretch $O(\log^{\log_{4/3} 7} n)$,
Query $O(\log\log n)$.

DPPRO - Definitions

Fix shortest paths $\Pi(u, v)$, for every $(u, v) \in \mathcal{P}$.

Def: *Branching event* $= (\Pi, \Pi', x)$



[CE05]: ≤ 2 branching events per pair (Π, Π') of paths.

$\mathcal{B}$ - set of branching events.

$$|\mathcal{B}| \leq P^2.$$

DPPRO - Construction

- $\forall\ v \in V$,
  store the identity of *home path* $\Pi(v)$:
  some path where v is an internal vertex.
  Store also *two edges* of $\Pi(v)$
  incident on v.

- $\forall$ branching event (Π, Π', x),
  keep two edges of Π and
  two edges of Π' incident on x.

- Keep hash table with all branching events.

- $\forall\ (u, v) \in \mathcal{P}$, store the first and the last edges of $\Pi(u, v)$.

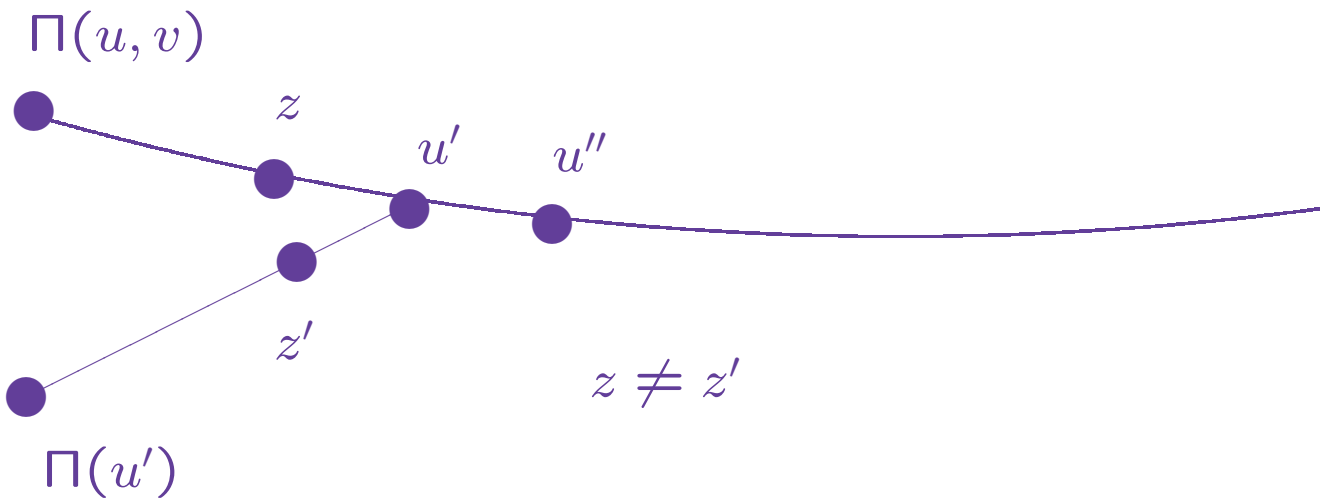
  Size $= O(n + |\mathcal{B}| + P) = O(n + P^2)$.

DPPRO - Query Algorithm

Input: $(u, v) \in \mathcal{P}$.

1) Find the first edge (u, u') of $\Pi_{u,v}$;
move to u'.

2) *If* $(\Pi_{u,v}, \Pi(u'), u')$ is a branching event *then*
fetch the next edge (u', u'') of $\Pi_{u,v}$
and move to u''.
else
fetch the next edge (u', u'') of $\Pi(u')$
and move to u''.



// In this case this is also
// the next edge of $\Pi(u, v)$.

Query time $= O(|\Pi(u, v)|)$.

A Basic D.O. for Sparse Graphs

Combination of [Agarwal,Godfrey,Har-Peled'11] w/ our DPPRO.

Achieves size $= O(n)$, stretch $= O(k)$, query $= n^{1/2+O(1/k)}$.

- Store G.
  Assume G has constant arboricity λ.
  Hence $|E(G)| = O(n)$.

- Sample $L \subseteq V$, $|L| = n^{1/4}$ landmarks.

- $\forall\ v \in V$, keep $\Pi(v, \ell(v))$:
  the shortest path between v and
  its closest landmark $\ell(v)$.
  // forest - $O(n)$ edges.

- Build a DPPRO for $\binom{L}{2}$.
  // size $O(n + P^2) = O(n + L^4) = O(n)$.

Query Algorithm of the Basic D.O.

Given (u, v):

- Conduct Dijkstra search from u until either v or $\ell(u)$ is found.

  // If v is found, stretch is 1.
  // Requires $\tilde{O}(n^{3/4})$ time,
  // because G has constant arboricity.

- Conduct Dijkstra search from v until either u or $\ell(v)$ is found.

  // Requires $\tilde{O}(n^{3/4})$ time.

- Given $\ell(u), \ell(v)$, invoke DPPRO for $\binom{L}{2}$
  // Requires $O(|\Pi(\ell(u), \ell(v))|)$ time.
  Return $(u - \ell(u) - \ell(v) - v)$.

Stretch Analysis

$$d_G(u, \ell(u)) \leq d_G(u, v),$$

because Dijkstra search from u
did not find v.

Symmetrically, $d_G(v, \ell(v)) \leq d_G(u, v)$.

The algorithm returns a path of length:

$$\begin{aligned} & d_G(u, \ell(u)) + d_G(\ell(u), \ell(v)) + d_G(\ell(v), v) \leq \\ & 2 \cdot d_G(u, \ell(u)) + d_G(u, v) + 2 \cdot d_G(\ell(v), v) \leq \\ & 5 \cdot d_G(u, v) \end{aligned}$$

Query $= \tilde{O}(n^{3/4})$, stretch $= 5$.

Extension of the Basic D.O.

- Sample L, $|L| = n^{1/2-1/k}$ landmarks.
  DPPRO for $\binom{L}{2}$ requires L^4 space.
  Too much!

- $\mathcal{L} = (L, \binom{L}{2}, d_G|L)$ - landmark metric.

- Invoke TZ (or Mandel-Naor) oracle
  on $\mathcal{L}$ with a parameter k.

  $H = \bigcup\{\Pi_{u,u'} \mid u, u' \in L\}$ is a $(2k-1)$-spanner
  w/ $O(k \cdot L^{1+1/k}) = O(n^{1/2})$ edges.

- Invoke a DPPRO on the set H of pairs.

Analysis of the Extension

The query time becomes $O\left(\frac{n}{L}\right) = O(n^{1/2+1/k})$.
(Dijkstra explorations dominate it.)

The size of the DPPRO is
$O(n + P^2) = O(n + (L^{1+1/k})^2) = O(n)$.

Note: TZ oracle $\mathcal{L}$ returns paths
which may contain edges *not* in G,
but the DPPRO fills in gaps in them.

The stretch becomes $O(k)$.

Multi-level sampling scheme takes the query time down to $O(n^{1/k})$, at the expense of greater stretch $O(k^{\log_{4/3} 7})$.

A Faster D.O. for Sparse Graphs

An h level-hierarchy.

$\forall\ i \in [h]$,
L_i - set of i-level landmarks.

L_i - sampled wp $\frac{\rho_i}{n}$.

$$n \geq \rho_1 > \rho_2 > \ldots > \rho_h \geq 1.$$

$L_0 = V$.

$\forall v \in V,\ \forall i \in [h]$,
$\ell_i(v)$ - the closest i-landmark to v.

$r_i(v) = d_G(v, \ell_i(v))$.

A Faster D.O (Cont.)

The ith ball of v:

$Ball_i(v) = \{u \mid d_G(v,u) < r_i(v)\}$.

The ith *one-third-ball* of v:

$\mathcal{B}_i^{1/3}(v) = \{u \mid d_G(v,u) < \frac{1}{3} r_i(v)\}$.

$\forall u \in V = L_0$, keep a shortest path between $u = u^{(0)}$ and $\ell_1(u) = u^{(1)}$.

(This is a collection of vertex-disjoint SPTs.)

$\forall i \in [h-1],\ \forall u^{(i)} \in L_i$,
keep a shortest path between $u^{(i)}$ and $u^{(i+1)} = \ell_{i+1}(u^{(i)})$.

The h forests require $O(n \cdot h)$ space.

G itself is also stored.
(Assumed to have constant arboricity λ.)

A Faster D.O (Cont. II)

For L_h we build a DPPRO $\mathcal{L}_h$.

Given a pair $u^{(h)}, v^{(h)} \in L_h$,
it returns $\Pi(u^{(h)}, v^{(h)})$ in time $|\Pi(u^{(h)}, v^{(h)})|$.

It requires $O(n + |L_h|^4)$ space.

So ρ_h will be set $\rho_h = n^{1/4}$,
to ensure $|L_h|^4 = O(n)$.

$\forall i \in [h-1]$,
$\mathcal{P}_i = \{(u^{(i)}, v^{(i)}) \in L_i \mid v^{(i)} \in \mathcal{B}^{1/3}_{i+1}(u^{(i)})$
$or\ u^{(i)} \in \mathcal{B}^{1/3}_{i+1}(v^{(i)})\}$.

$\mathcal{D}_i$ - a DPPRO for $\mathcal{P}_i$.

The size of $\mathcal{D}_i$ is $O(n + |Branch_i|)$.

We will ensure that $|Branch_i| = O(n)$.

Query Algorithm

Conduct Dijkstra search from $u^{(0)}$ to $u^{(1)}$.
If v is not found, then
$d_G(u^{(0)}, u^{(1)}) \leq d_G(u, v)$.

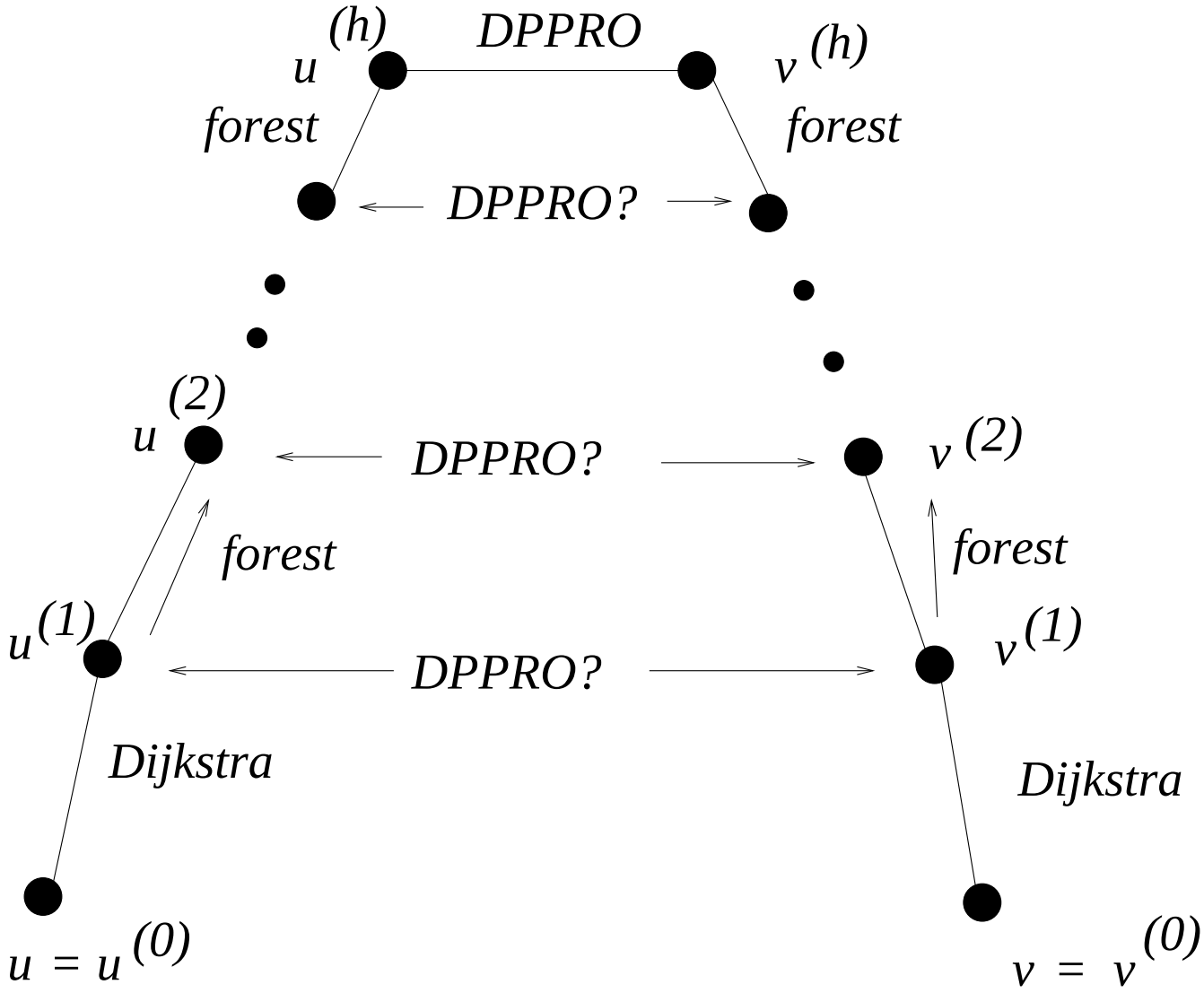
Analogously, either or a shortest $u - v$ path is found, or $d_G(v^{(0)}, v^{(1)}) \leq d_G(u, v)$.

Check if $(u^{(1)}, v^{(1)}) \in \mathcal{P}_1$.
If true, employ $\mathcal{D}_1$ to retrieve $\Pi(u^{(1)}, v^{(1)})$.

Otherwise

$$d_G(u^{(1)}, v^{(1)}) \geq \frac{1}{3} d_G(u^{(1)}, u^{(2)}), \frac{1}{3} d_G(v^{(1)}, v^{(2)}).$$

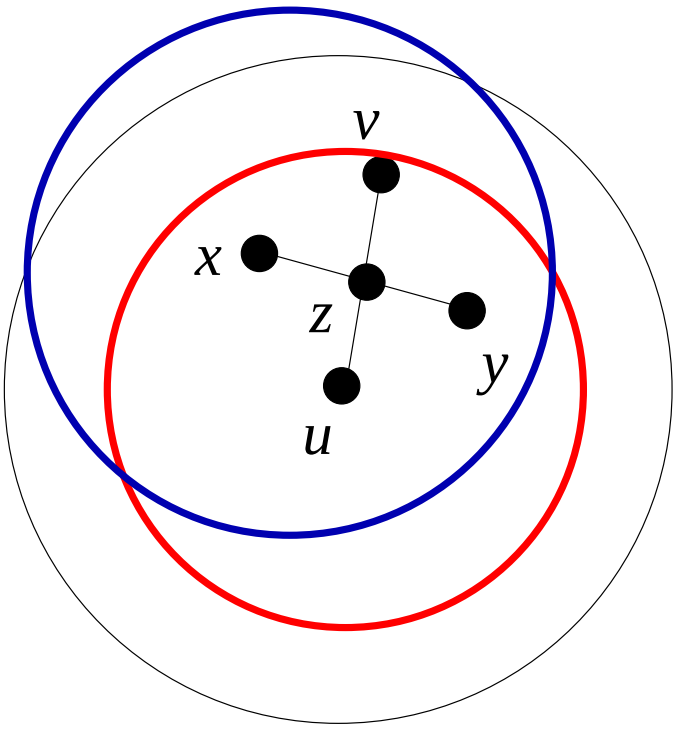
Hence
$d_G(u^{(1)}, u^{(2)}), d_G(v^{(1)}, v^{(2)}) \leq 3 d_G(u^{(1)}, v^{(1)})$.


u (h)
DPPRO
v (h)
forest
forest
DPPRO?
u (2)
DPPRO?
v (2)
forest
forest
u (1)
DPPRO?
v (1)
Dijkstra
Dijkstra
u = u (0)
v = v (0)

Sketch of Analysis

Stretch $= O(7^h)$.

Lemma: [Pettie'08]
Suppose $v \in \mathcal{B}_{i+1}^{1/3}(u)$.
Then if $(x, y) \in \mathcal{P}_{i+1}$ *and*
there is a branching event between
(u, v) and (x, y),
then $v, x, y \in Ball_{i+1}(u)$.



Lemma: $|Branch_i| = O(\frac{\rho_i^4}{\rho_{i+1}^3})$.

Proof:

Count the number of quadruples $u \in L_i$, $v, x, y \in Ball_{i+1}(u) \cap L_i$.

There are ρ_i i-landmarks u.

For a fixed u,

$$\mathbb{E}(|Ball_{i+1}(u)|) = O(n/\rho_{i+1})$$

(A geometric random variable with parameter $p = \rho_{i+1}/n$.)

Each of the points in $Ball_{i+1}(u)$ is an i-landmark wp ρ_i/n.

So the total number of i-landmarks in $Ball_{i+1}(u)$ is

$$\frac{n}{\rho_{i+1}} \cdot \frac{\rho_i}{n} = \frac{\rho_i}{\rho_{i+1}}.$$

The total number of quadruples is

$$\rho_i \cdot (\frac{\rho_i}{\rho_{i+1}})^3 = \frac{\rho_i^4}{\rho_{i+1}^3}.$$

QED

Setting Parameters

$\rho_i = n^{\alpha_i}$, $\alpha_i = 1 - (3/4)^{h-i+1}$.

So $\rho_h = n^{1/4}$.

The query time is $\tilde{O}(\frac{n}{\rho_1}\lambda) = O(n^{(3/4)^h} \cdot \lambda)$.

Set $t = (4/3)^h$, i.e., $h = \Theta(\log t)$. Then

Stretch $= O(t^{\log_{4/3} 7}) \approx O(t^{6.46})$,

Query $= \tilde{O}(n^{1/t} \cdot \lambda)$,

Space $= O(n \cdot \log t)$.

For general graphs we either use this oracle on top of a *spanner*, or use it with $h = \Theta(\log \log n)$.

In the latter case all points become 1-landmarks, and so *no Dijkstra search* is required to reach the closest 1-landmarks.
The query time becomes $O(h) = O(\log \log n)$.

Open Questions

1. Size $O(n)$, *Query* $O(1)$, stretch $O(\log n)$?

   We have either
   Size $O(n)$, query n^{ϵ}, stretch $O(\log n)$
   or
   size $O(n \log\log n)$, query $O(\log\log n)$,
   stretch $O(\log^{\log_{4/3} 7} n)$.

2. Improve the exponent of stretch
   in the second result.

3. D.O. w/ stretch $(1+\epsilon, \beta)$, size $O(n^{1+1/k})$,
   and small query time for *unweighted* graphs.

   We have stretch $(O(1), \beta)$.