Truly Subquadratic Exact Distance Oracles with Constant Query Time for Planar Graphs

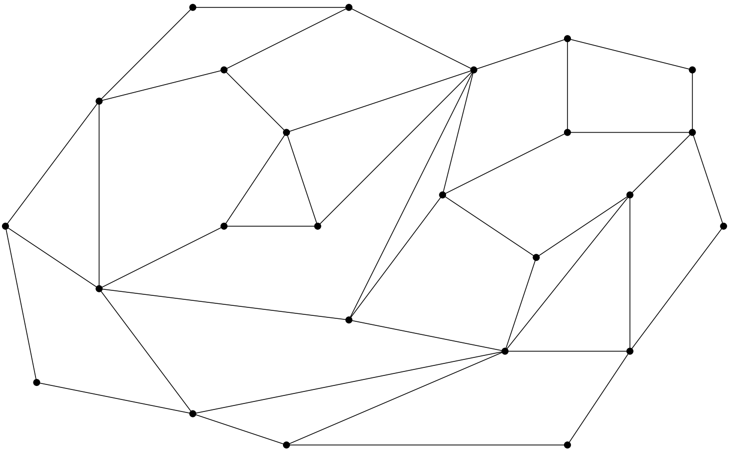
Viktor Fredslund-Hansen[1] Shay Mozes[2]
Christian Wulff-Nilsen[1]

[1]Univertisy of Copenhagen, Denmark

[2]Interdisciplinary Center Herzliya, Israel

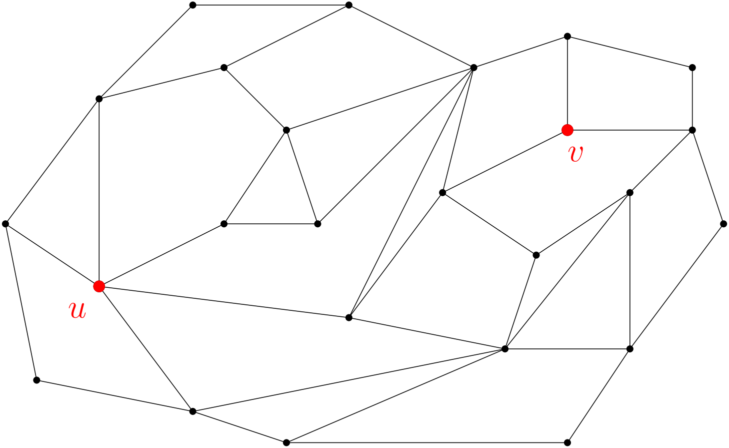
August 2020

Problem definition



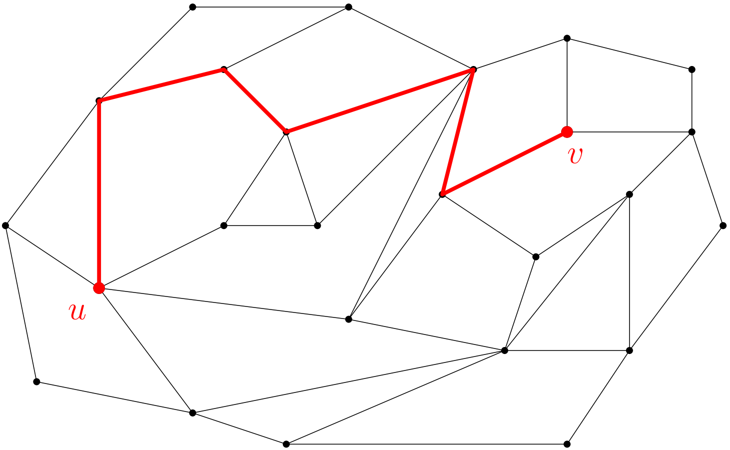
Preprocess an n-vertex planar graph $G = (V, E)$ into a data structure of truly subqudratic $O(n^{2-\delta})$ size, so that given any $u, v \in V$ we can compute $d(u, v)$ exactly in $O(1)$ time.

Problem definition



Preprocess an n-vertex planar graph $G = (V, E)$ into a data structure of truly subqudratic $O(n^{2-\delta})$ size, so that given any $u, v \in V$ we can compute $d(u, v)$ exactly in $O(1)$ time.

Problem definition



Preprocess an n-vertex planar graph $G = (V, E)$ into a data structure of truly subqudratic $O(n^{2-\delta})$ size, so that given any $u, v \in V$ we can compute $d(u, v)$ exactly in $O(1)$ time.

Prior work

- exact oracles:
  - smallest oracle with $O(1)$ query time has $O(n^2 \log\log n / \log n)$ space (unweighted)
  - if polylog query time is acceptable, can have $O(n^{1+\delta})$ space [Charalampopoulos et al. STOC 19]
- for $(1+\varepsilon)$ approximate oracles one can get $O(1)$ query time and $O(n \log n)$ space [Thorup JACM 04, Chan and Skrepetos Algorithmica 18]

Prior work

- exact oracles:
  - smallest oracle with $O(1)$ query time has $O(n^2 \log\log n / \log n)$ space (unweighted)
  - if polylog query time is acceptable, can have $O(n^{1+\delta})$ space [Charalampopoulos et al. STOC 19]
- for $(1+\varepsilon)$ approximate oracles one can get $O(1)$ query time and $O(n \log n)$ space [Thorup JACM 04, Chan and Skrepetos Algorithmica 18]

Prior work

- exact oracles:
  - smallest oracle with $O(1)$ query time has $O(n^2 \log\log n / \log n)$ space (unweighted)
  - if polylog query time is acceptable, can have $O(n^{1+\delta})$ space [Charalampopoulos et al. STOC 19]
- for $(1+\varepsilon)$ approximate oracles one can get $O(1)$ query time and $O(n \log n)$ space [Thorup JACM 04, Chan and Skrepetos Algorithmica 18]

Main result

An exact distance oracle for unweighted undirected planar graphs with space $O(n^{5/3+\delta})$ and query time $O(\log(1/\delta))$ for every $\delta > 0$.

The oracle is based on a ideas used for compression of the shortest path metric of unweighted undirected planar graphs [Li and Parter, STOC 19].

The oracle is quite simple and natural, once the appropriate definitions are put forth.

Main result

An exact distance oracle for unweighted undirected planar graphs with space $O(n^{5/3+\delta})$ and query time $O(\log(1/\delta))$ for every $\delta > 0$.

The oracle is based on a ideas used for compression of the shortest path metric of unweighted undirected planar graphs [Li and Parter, STOC 19].

The oracle is quite simple and natural, once the appropriate definitions are put forth.

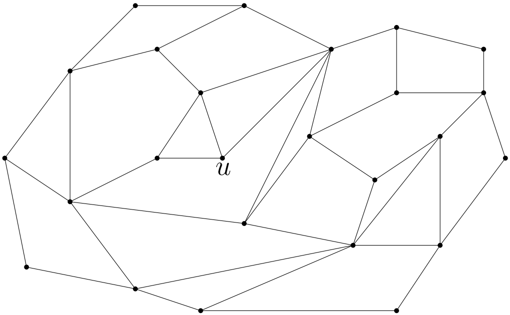
Main result

An exact distance oracle for unweighted undirected planar graphs with space $O(n^{5/3+\delta})$ and query time $O(\log(1/\delta))$ for every $\delta > 0$.

The oracle is based on a ideas used for compression of the shortest path metric of unweighted undirected planar graphs [Li and Parter, STOC 19].

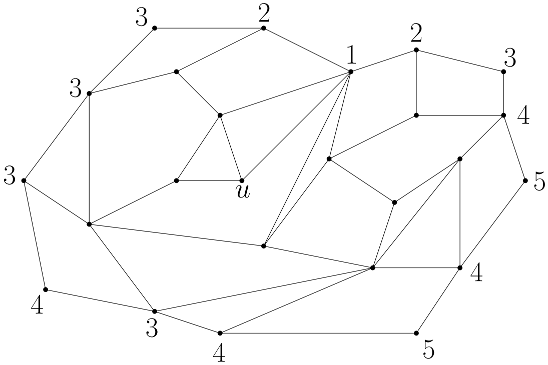
The oracle is quite simple and natural, once the appropriate definitions are put forth.

Key concept - distance pattern



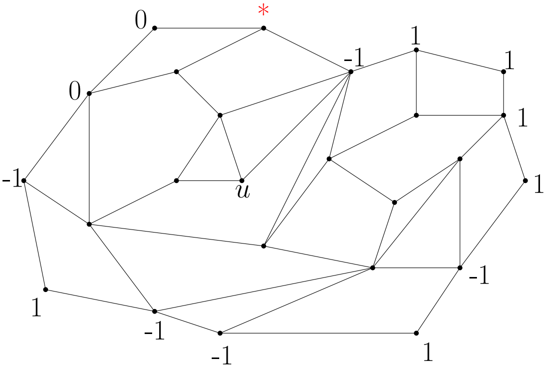
- consider some vertex *u*
- consider the sequence of distances from *u* to each of the *k* vertices of some face *f*.
- the pattern $\rho_u[\cdot]$ of *u* w.r.t. *f* is the vector of differences of each pair of consecutive distances along *f*.

Key concept - distance pattern



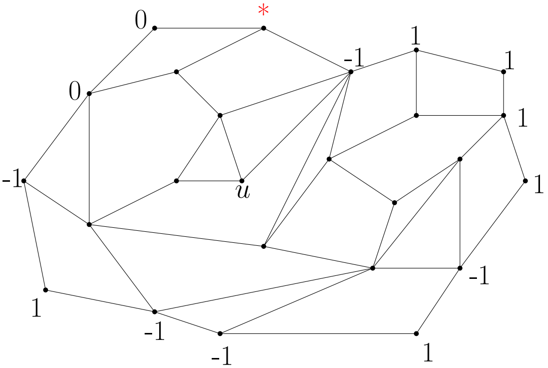
- consider some vertex u
- consider the sequence of distances from u to each of the k vertices of some face f.
- the pattern $\rho_u[\cdot]$ of u w.r.t. f is the vector of differences of each pair of consecutive distances along f.

Key concept - distance pattern



- consider some vertex *u*
- consider the sequence of distances from *u* to each of the *k* vertices of some face *f*.
- the pattern $\rho_u[\cdot]$ of *u* w.r.t. *f* is the vector of differences of each pair of consecutive distances along *f*.

Few patterns



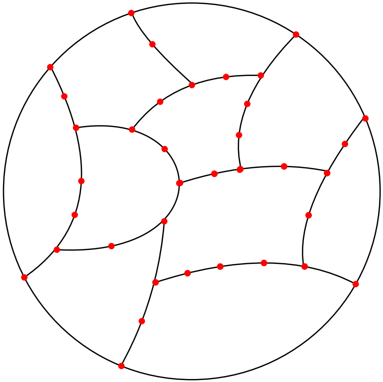
Few Patterns [Li and Parter STOC 2019]

Let G be a graph with n nodes. Let f be a face with k vertices. Then the number of distinct patterns w.r.t. f (over all possible n nodes of G) is $O(k^3)$.

r-divisions

For $r \in [1, n]$, a decomposition of a graph into:

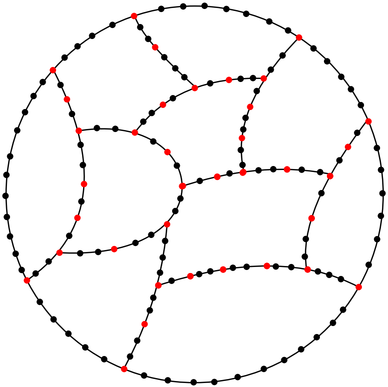
- $O(n/r)$ pieces;
- each piece is connected;
- each piece has $O(r)$ vertices;
- each piece has $O(\sqrt{r})$ boundary vertices (vertices incident to edges in other pieces).
- in each piece, all boundary nodes lie on a constant number of faces.



r-divisions

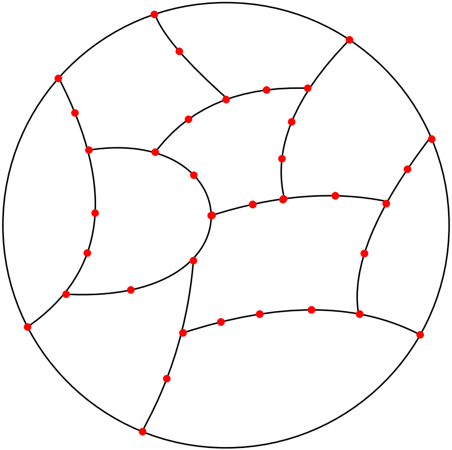
For $r \in [1, n]$, a decomposition of a graph into:

- $O(n/r)$ pieces;
- each piece is connected;
- each piece has $O(r)$ vertices;
- each piece has $O(\sqrt{r})$ boundary vertices (vertices incident to edges in other pieces).
- in each piece, all boundary nodes lie on a constant number of faces.



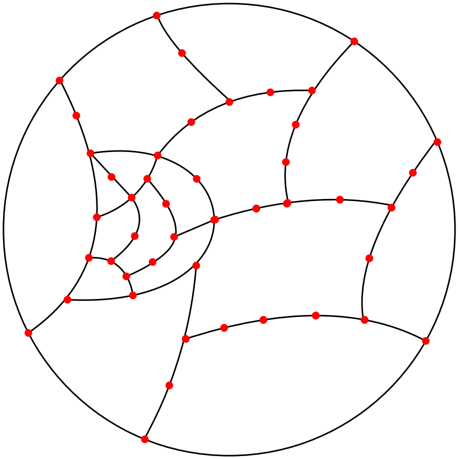
Recursive r-divisions

For $r_1 < r_2 < \cdots < r_m \in [1, n]$, we can efficiently compute r_i-divisions, such that each r_i-division respects the r_{i+1}-division.



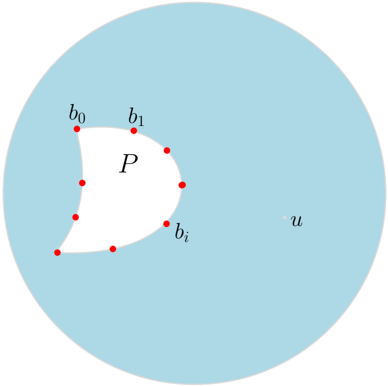
Recursive r-divisions

For $r_1 < r_2 < \cdots < r_m \in [1, n]$, we can efficiently compute r_i-divisions, such that each r_i-division respects the r_{i+1}-division.



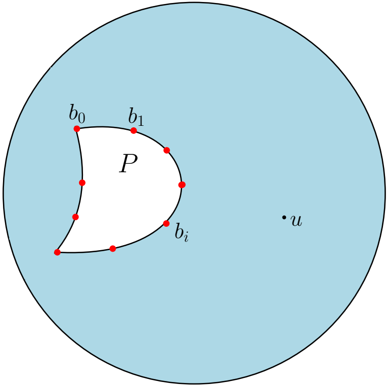
Working with patterns

- pattern can be defined w.r.t. distances to a cycle with *k* vertices, not just a face.
- number of patterns is still $O(k^3)$.



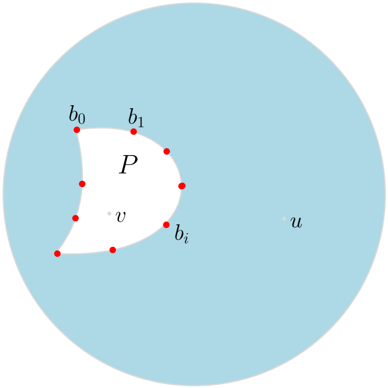
Working with patterns

- pattern can be defined w.r.t. distances to a cycle with k vertices, not just a face.
- number of patterns is still $O(k^3)$.



Working with patterns

- the distance between u and b_i is: $d(u, b_i) = d(u, b_0) + \sum_{j=1}^{i} \rho_u[j]$
- the distance between u and v is: $\min_i d(u, b_0) + \sum_{j=1}^{i} \rho_u[j] + d(b_i, v)$



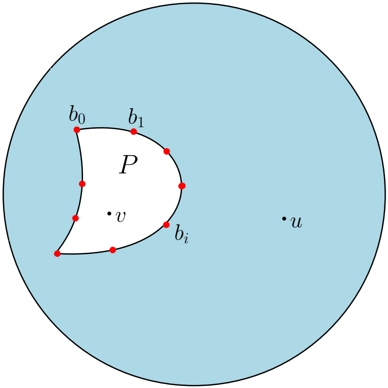
distance from pattern to node

define the distance from a pattern ρ to node v as: $d(\rho, v) \equiv \min_i \sum_{j=1}^{i} \rho[j] + d(b_i, v)$.

Now $d(u, v) = d(u, b_0) + d(\rho_u, v)$.

Working with patterns

- the distance between u and b_i is: $d(u, b_i) = d(u, b_0) + \sum_{j=1}^{i} \rho_u[j]$
- the distance between u and v is: $\min_i d(u, b_0) + \sum_{j=1}^{i} \rho_u[j] + d(b_i, v)$



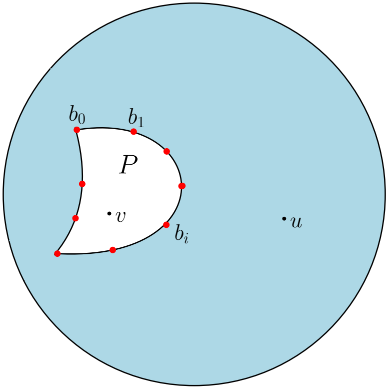
distance from pattern to node

define the distance from a pattern ρ to node v as: $d(\rho, v) \equiv \min_i \sum_{j=1}^{i} \rho[j] + d(b_i, v)$.

Now $d(u, v) = d(u, b_0) + d(\rho_u, v)$.

Working with patterns

- the distance between u and b_i is: $d(u, b_i) = d(u, b_0) + \sum_{j=1}^{i} \rho_u[j]$
- the distance between u and v is: $\min_i d(u, b_0) + \sum_{j=1}^{i} \rho_u[j] + d(b_i, v)$



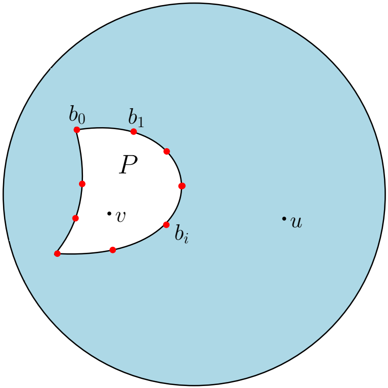
distance from pattern to node

define the distance from a pattern ρ to node v as: $d(\rho, v) \equiv \min_i \sum_{j=1}^{i} \rho[j] + d(b_i, v)$.

Now $d(u, v) = d(u, b_0) + d(\rho_u, v)$.

Working with patterns

- the distance between u and b_i is: $d(u, b_i) = d(u, b_0) + \sum_{j=1}^{i} \rho_u[j]$
- the distance between u and v is: $\min_i d(u, b_0) + \sum_{j=1}^{i} \rho_u[j] + d(b_i, v)$



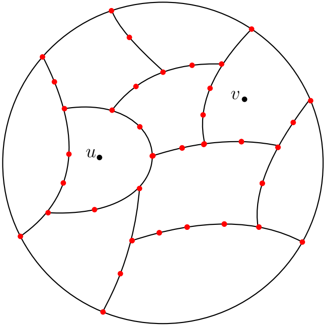
distance from pattern to node

define the distance from a pattern ρ to node v as:
$d(\rho, v) \equiv \min_i \sum_{j=1}^{i} \rho[j] + d(b_i, v)$.

Now $d(u, v) = d(u, b_0) + d(\rho_u, v)$.

A $O(n^{7/4})$-space oracle

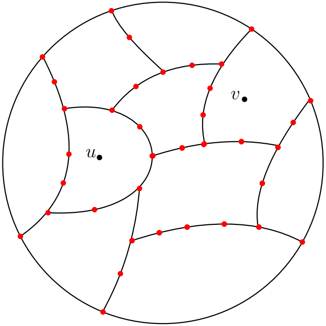
- for each vertex $u \in G$ and each piece P:
  1. a pointer to the pattern of u w.r.t. P
  2. the distance from u to the cannonical vertex of P.
- store for every piece P:
  3. pairwise distances between all nodes in P.
  4. distance from each possible pattern of P to each node of P.



- query(u, v): if u, v in same piece, distance is precomputed. Otherwise, let P be a piece containing v. Let b_0 be the canonical vertex of P. Let ρ be the pattern of u w.r.t. P. Return $d(u, b_0) + d(\rho, v)$.
- balance is at $r = n^{1/4}$, so space is $O(n^{7/4})$.

A $O(n^{7/4})$-space oracle

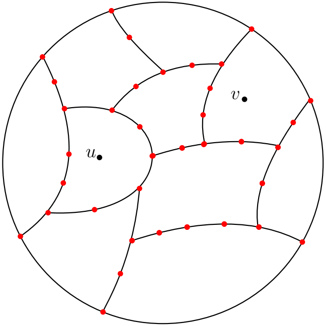
- for each vertex $u \in G$ and each piece P:
  1. a pointer to the pattern of u w.r.t. P
  2. the distance from u to the cannonical vertex of P.
- store for every piece P:
  3. pairwise distances between all nodes in P.
  4. distance from each possible pattern of P to each node of P.



- query(u, v): if u, v in same piece, distance is precomputed. Otherwise, let P be a piece containing v. Let b_0 be the canonical vertex of P. Let ρ be the pattern of u w.r.t. P. Return $d(u, b_0) + d(\rho, v)$.
- balance is at $r = n^{1/4}$, so space is $O(n^{7/4})$.

A $O(n^{7/4})$-space oracle

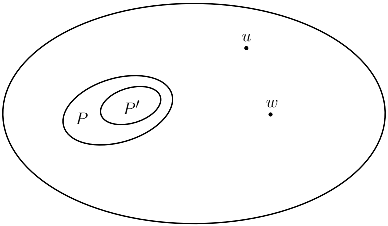
- for each vertex $u \in G$ and each piece P:
  1. a pointer to the pattern of u w.r.t. P
  2. the distance from u to the cannonical vertex of P.
- store for every piece P:
  3. pairwise distances between all nodes in P.
  4. distance from each possible pattern of P to each node of P.



- query(u, v): if u, v in same piece, distance is precomputed. Otherwise, let P be a piece containing v. Let b_0 be the canonical vertex of P. Let ρ be the pattern of u w.r.t. P. Return $d(u, b_0) + d(\rho, v)$.
- balance is at $r = n^{1/4}$, so space is $O(n^{7/4})$.

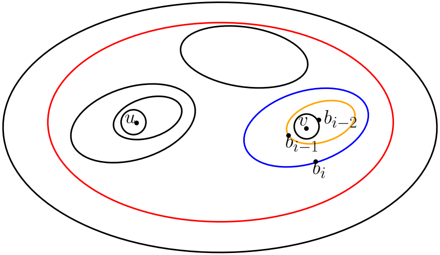
Working with patterns II

- Consider a Piece P with a subpice P' of P.
- If $u, w \notin P$ have the same pattern w.r.t. P then they have the same pattern w.r.t. P'.
- so a pattern of P induces a pattern on P'.



A $O(n^{5/3+\delta})$-space oracle

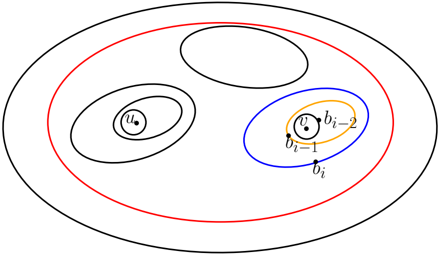
Compute a recursive r-division with $r_0 = 1 < r_1 < \cdots < r_m < r_{m+1} = n$



1. For each $u \in G$ store a "cone" of pieces $P_0 \supset P_1 \supset \cdots \supset P_{m+1}$ containing u.
2. For each $u \in G$, for each P_i containing u, for each subpiece P' of P_i at level $i-1$, store the pattern of u w.r.t. P' and the distance from u to canonical node of P'.
3. for each level i, for each piece P at level i, for each possible pattern ρ of P, for each subpiece P' of P_i at level $i-1$, store a pointer to the pattern ρ' induced by ρ on P', and the distance from the pattern ρ to the canonical vertex of P'.

A $O(n^{5/3+\delta})$-space oracle

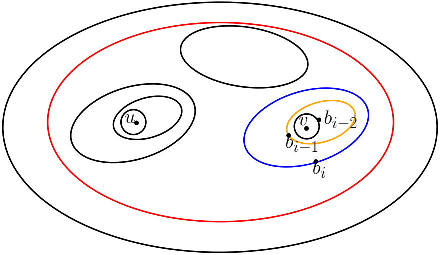
Query(u, v):



- Locate smallest piece P containing both u and v.
- Let P_i be the child piece of P that contains v
- retrieve the distance from u to the canonical vertex b_i of P_i, and the pattern ρ_i of u w.r.t. P_i.
- Let P_{i-1} be the child piece of P_i that contains v
- retrieve the distance from ρ_i to the canonical vertex b_{i-1} of P_{i-1}, and the pattern ρ_{i-1} induced by ρ_i on P_{i-1}.
- compute the distance from u to b_{i-1} by $d(u, b_i) + dist(\rho_i, b_{i-1})$
- continue with smaller and smaller pieces until $b_0 = v$.

A $O(n^{5/3+\delta})$-space oracle

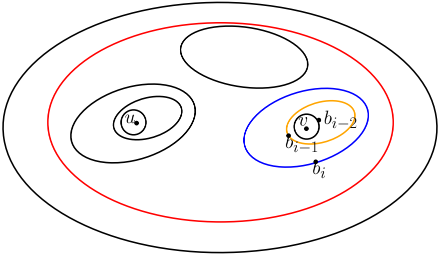
Query(u, v):



- Locate smallest piece P containing both u and v.
- Let P_i be the child piece of P that contains v
- retrieve the distance from u to the canonical vertex b_i of P_i, and the pattern ρ_i of u w.r.t. P_i.
- Let P_{i-1} be the child piece of P_i that contains v
- retrieve the distance from ρ_i to the canonical vertex b_{i-1} of P_{i-1}, and the pattern ρ_{i-1} induced by ρ_i on P_{i-1}.
- compute the distance from u to b_{i-1} by $d(u, b_i) + dist(\rho_i, b_{i-1})$
- continue with smaller and smaller pieces until $b_0 = v$.

A $O(n^{5/3+\delta})$-space oracle

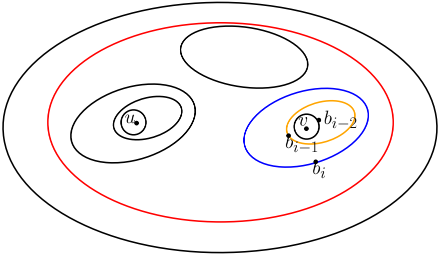
Query(u, v):



- Locate smallest piece P containing both u and v.
- Let P_i be the child piece of P that contains v
- retrieve the distance from u to the canonical vertex b_i of P_i, and the pattern ρ_i of u w.r.t. P_i.
- Let P_{i-1} be the child piece of P_i that contains v
- retrieve the distance from ρ_i to the canonical vertex b_{i-1} of P_{i-1}, and the pattern ρ_{i-1} induced by ρ_i on P_{i-1}.
- compute the distance from u to b_{i-1} by $d(u, b_i) + dist(\rho_i, b_{i-1})$
- continue with smaller and smaller pieces until $b_0 = v$.

A $O(n^{5/3+\delta})$-space oracle

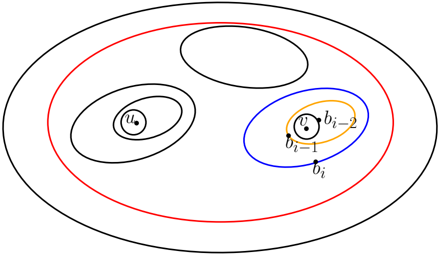
Query(u, v):



- Locate smallest piece P containing both u and v.
- Let P_i be the child piece of P that contains v
- retrieve the distance from u to the canonical vertex b_i of P_i, and the pattern ρ_i of u w.r.t. P_i.
- Let P_{i-1} be the child piece of P_i that contains v
- retrieve the distance from ρ_i to the canonical vertex b_{i-1} of P_{i-1}, and the pattern ρ_{i-1} induced by ρ_i on P_{i-1}.
- compute the distance from u to b_{i-1} by $d(u, b_i) + dist(\rho_i, b_{i-1})$
- continue with smaller and smaller pieces until $b_0 = v$.

A $O(n^{5/3+\delta})$-space oracle

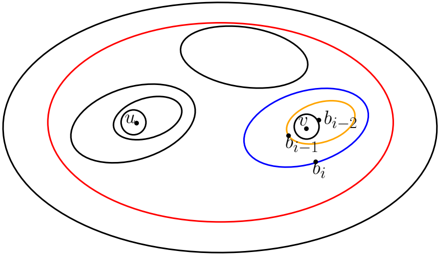
Query(u, v):



- Locate smallest piece P containing both u and v.
- Let P_i be the child piece of P that contains v
- retrieve the distance from u to the canonical vertex b_i of P_i, and the pattern ρ_i of u w.r.t. P_i.
- Let P_{i-1} be the child piece of P_i that contains v
- retrieve the distance from ρ_i to the canonical vertex b_{i-1} of P_{i-1}, and the pattern ρ_{i-1} induced by ρ_i on P_{i-1}.
- compute the distance from u to b_{i-1} by $d(u, b_i) + dist(\rho_i, b_{i-1})$
- continue with smaller and smaller pieces until $b_0 = v$.

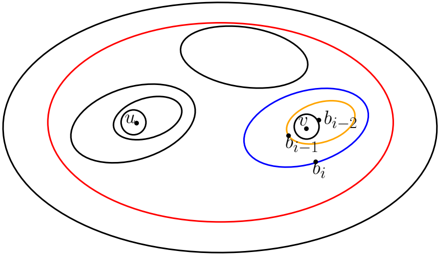
A $O(n^{5/3+\delta})$-space oracle

Query(u, v):



- Locate smallest piece P containing both u and v.
- Let P_i be the child piece of P that contains v
- retrieve the distance from u to the canonical vertex b_i of P_i, and the pattern ρ_i of u w.r.t. P_i.
- Let P_{i-1} be the child piece of P_i that contains v
- retrieve the distance from ρ_i to the canonical vertex b_{i-1} of P_{i-1}, and the pattern ρ_{i-1} induced by ρ_i on P_{i-1}.
- compute the distance from u to b_{i-1} by $d(u, b_i) + dist(\rho_i, b_{i-1})$
- continue with smaller and smaller pieces until $b_0 = v$.

A $O(n^{5/3+\delta})$-space oracle

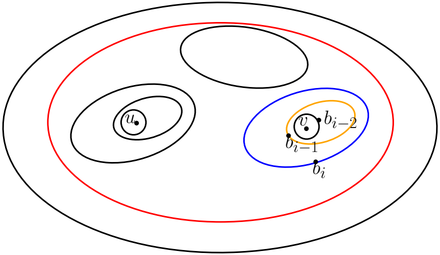


Query(u, v):

- Locate smallest piece P containing both u and v.
- Let P_i be the child piece of P that contains v
- retrieve the distance from u to the canonical vertex b_i of P_i, and the pattern ρ_i of u w.r.t. P_i.
- Let P_{i-1} be the child piece of P_i that contains v
- retrieve the distance from ρ_i to the canonical vertex b_{i-1} of P_{i-1}, and the pattern ρ_{i-1} induced by ρ_i on P_{i-1}.
- compute the distance from u to b_{i-1} by $d(u, b_i) + dist(\rho_i, b_{i-1})$
- continue with smaller and smaller pieces until $b_0 = v$.

A $O(n^{5/3+\delta})$-space oracle

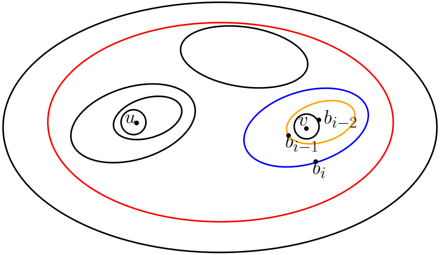
Query(u, v):



- Locate smallest piece P containing both u and v.
- Let P_i be the child piece of P that contains v
- retrieve the distance from u to the canonical vertex b_i of P_i, and the pattern ρ_i of u w.r.t. P_i.
- Let P_{i-1} be the child piece of P_i that contains v
- retrieve the distance from ρ_i to the canonical vertex b_{i-1} of P_{i-1}, and the pattern ρ_{i-1} induced by ρ_i on P_{i-1}.
- compute the distance from u to b_{i-1} by $d(u, b_i) + dist(\rho_i, b_{i-1})$
- continue with smaller and smaller pieces until $b_0 = v$.

A $O(n^{5/3+\delta})$-space oracle

Compute a recursive r-division with $r_0 = 1 < r_1 < \cdots < r_m < r_{m+1} = n$



2. For each $u \in G$, for each P_i containing u, for each subpiece P' of P_i at level $i-1$, store the pattern of u w.r.t. P' and the distance from u to canonical node of P'.
   $O(n \cdot \frac{n}{r_m} + n \sum_{i=1}^{m} \frac{r_i}{r_{i-1}})$
3. for each level i, for each piece P at level i, for each possible pattern ρ of P, for each subpiece P' of P_i at level $i-1$, store the pattern ρ' induced by ρ on P', and the distance from the pattern ρ to the canonical vertex of P'.
   $O(\sum_{i=1}^{m} \frac{n}{r_i} \cdot r_i^3 \cdot \frac{r_i}{r_{i-1}}) = O(n \sum_{i=1}^{m} \frac{r_i^3}{r_{i-1}})$

Overall $O(\frac{n^2}{r_m} + n \sum_{i=1}^{m} \frac{r_i^3}{r_{i-1}})$ space.

Analysis

- Overall $O(\frac{n^2}{r_m} + n\sum_{i=1}^{m} \frac{r_i^3}{r_{i-1}})$ space.
- Setting $m = \frac{1}{3\delta}$ and $r_i = n^{i\delta}$ yields $O(\frac{n^2}{r_m} + nr_m^2 \sum_{i=1}^{m} \frac{r_i}{r_{i-1}}) = O(\frac{1}{3\delta} n^{5/3+\delta})$ space and $O(\frac{1}{\delta})$ query time.
- We can be more aggressive in decreasing the sizes of pieces. Setting $r_m = n^{1/3}$ and $\frac{r_i^3}{r_{i-1}} = n^{2/3+\delta}$ gives $r_1 = O(1)$ with $m = O(\log(1/\delta))$.
This yields space $O(\frac{1}{\delta} n^{5/3+\delta})$ and query time $O(\log \frac{1}{\delta})$.

Analysis

- Overall $O(\frac{n^2}{r_m} + n\sum_{i=1}^{m} \frac{r_i^3}{r_{i-1}})$ space.
- Setting $m = \frac{1}{3\delta}$ and $r_i = n^{i\delta}$ yields $O(\frac{n^2}{r_m} + nr_m^2 \sum_{i=1}^{m} \frac{r_i}{r_{i-1}}) = O(\frac{1}{3\delta} n^{5/3+\delta})$ space and $O(\frac{1}{\delta})$ query time.
- We can be more aggressive in decreasing the sizes of pieces. Setting $r_m = n^{1/3}$ and $\frac{r_i^3}{r_{i-1}} = n^{2/3+\delta}$ gives $r_1 = O(1)$ with $m = O(\log(1/\delta)$.
  This yields space $O(\frac{1}{\delta} n^{5/3+\delta})$ and query time $O(\log \frac{1}{\delta})$.

Summary and Discussion

An exact distance oracle for unweighted undirected planar graphs with space $O(n^{5/3+\delta})$ and query time $O(\log(1/\delta))$ for every $\delta > 0$.

- First truly subquadratic-space oracle with constant query time.
- Unlike all other oracles we are aware of (not just exact and not just for planar graphs), the query algorithm does not necessarily revoer the distance to any internal node on the path from u to v.
- This makes it more difficult to recover the shortest path, but just reporting the distances is actually easier than also reporting the path.