Approximate Distance Oracle: given a graph $G = (V, E)$, preprocess it into a data structure such that we can compute approximate shortest-path distances efficiently (and output path if desired).

(same scenario as in Lecture 12 except that paths are allowed to be approximately shortest)

Assumption (all of Lecture 13) undirected planar $G$, non-negative edge lengths $\ell : E \to \mathbb{R}^+$

Stretch for any given $\epsilon > 0$, preprocessing algorithm constructs a data structure using which we can, queried for any pair of nodes $(v, w)$, output an estimate $\tilde{d}(v, w)$ satisfying

$$d_G(v, w) \leq \tilde{d}(v, w) \leq (1 + \epsilon)d_G(v, w)$$

Exact Distance Oracles if query time $poly(\log n)$ is desired, best methods use space $\tilde{\Omega}(n^2)$
inspect each and every portal on the separator between two query nodes. since separators have size $O(\sqrt{n})$, number of portals is “small”

Question can we safely reduce the number of portals? if yes, how?
negative: “neighbors” on cycle separator could be neighbors due to triangulation, edge length $\infty$

Cycle separators (Lecture 2): fundamental cycle is defined by a spanning tree $T$ and any non-tree edge

Lemma. For any planar graph $G = (V, E)$ and any spanning tree $T$ of radius $d$, we can partition $V$ into $A, B, S \subseteq V$ s.t.

- [balanced] $|A|, |B| \leq 3n/4$
- [separation] no edge between any $a \in A$ and $b \in B$ ($A \times B \cap E = \emptyset$)
- [separator size] $|S| \leq 2d + 1$
- [efficient] $A, B, S$ can be found in linear time.

Idea apply the lemma using a shortest-path tree $T$ rooted at an arbitrary node $r$. separator paths may contain many nodes (no bound on radius, increased number of potential portals) but they are shortest paths, which have good properties

Approximate Distance Oracle

Preprocessing (i) recursively separate $G$ using shortest-path separators ($O(\log n)$ levels, 2 paths per level), (ii) each node stores distances to portals on $O(\log n)$ paths

Query given $(v, w)$, find best path through all the separator paths “shared” by $v$ and $w$
Ports: \(\epsilon\)-cover

Approximate representation of shortest paths crossing a separator path using few portals. Prove that, per node \(v\) and separator path \(Q\), \(O(1/\epsilon)\) portals suffice to guarantee \((1+\epsilon)\)-approximation for any shortest path crossing \(Q\).

Given a node \(v\) and a separator path \(Q\), we compute a set of portals (also: \(\epsilon\)-cover) \(\{p_j\} = C(v, Q) \subseteq Q\) as follows. Let the first portal \(p_0\) be a node \(q \in Q\) minimizing \(d(v, q)\). For the analysis, split the path \(Q\) at \(p_0\) into path \(Q^+\) and \(Q^-\). Starting at \(p_0\), "walk" on \(Q^+\). Throughout the algorithm, let \(p_t\) denote the last portal selected. For each node \(q\), if the direct path from \(v\) to \(q\) is significantly shorter than the path with a detour through \(p_t\), more precisely,

\[
d(v, p_t) + d(p_t, q) > (1 + \epsilon)d(v, q),
\]

then insert \(q\) as a new portal. By definition, each node on \(Q^+\) is \(\epsilon\)-covered by some portal \(p_j\), meaning that

\[
\forall q \in Q \exists p_j \in C(v, Q) : d(v, p_j) + d(p_j, q) \leq (1 + \epsilon)d(v, q).
\]

Analogously, compute portals for \(Q^-\). Portals for \(Q\) is the union of the portals for \(Q^-\) and \(Q^+\).

Claim. The number of portals is at most \(O(1/\epsilon)\).

Distance oracle based on this claim, the approximate distance oracle uses space \(O(\epsilon^{-1} \log n)\) per node (from \(O(1/\epsilon)\) distances to portals on \(O(\log n)\) paths) and answers distance queries in time \(O(\epsilon^{-1} \log n)\) as follows: For any pair of nodes \((v, w)\) at most \(O(\log n)\) separator paths need to be considered. \(^1\) For a separator path \(Q\) we can efficiently "merge" portals in time \(O(1/\epsilon)\) to find the best \(p_v, p_w\) minimizing \(d(v, p_v) + d(p_v, p_w) + d(p_w, w)\) (the distance between portals is easily obtained from the relative position of \(p_v, p_w\) on \(Q\)).

Proof of Claim. To derive an upper bound on the number of portals, we define a potential function

\[
\Phi(Q^+, v, \{p_j\}) := d(v, p_1) + d(p_1, s),
\]

where \(p_l\) is the "last" portal chosen for \(v\) and \(s\) is the "last" node (sentinel) on \(Q^+\) (analogously for \(Q^-\)).

Potential change by adding portal \(p_{j+1}\), the potential function decreases by at least \(\epsilon d(v, p_0)\) as follows:

<table>
<thead>
<tr>
<th>new (\Phi^{(j+1)})</th>
<th>old (\Phi^{(j)})</th>
<th>which is</th>
<th>portal (p_{j+1}) is added since</th>
</tr>
</thead>
<tbody>
<tr>
<td>(d(v, p_{j+1}) + d(p_{j+1}, s))</td>
<td>(d(v, p_j) + d(p_j, s))</td>
<td>(d(v, p_j) + d(p_j, p_{j+1}) + d(p_{j+1}, s))</td>
<td>(d(v, p_j) + d(p_j, p_{j+1}) &gt; (1 + \epsilon)d(v, p_{j+1})) and thus</td>
</tr>
</tbody>
</table>

\[
\Phi^{(j)} - \Phi^{(j+1)} > \epsilon d(v, p_{j+1}) \geq \epsilon d(v, p_0)
\]

Total change (here we use the property that \(Q\) is a shortest path) After choosing the first portal \(p_0\), we have \(\Phi(Q, v, \{p_0\}) = d(v, p_0) + d(p_0, s)\), which, by triangle inequality is at least \(d(v, s)\). Again, using the triangle inequality, we have \(d(v, s) \geq d(p_0, s) - d(p_0, v)\). The first value of the potential function is \(d(v, p_0) + d(p_0, s)\) and the last value is at least \(d(p_0, s) - d(p_0, v)\). In total, the value of the potential function may decrease by at most \(2d(v, p_0)\) (\(G\) is undirected and thus \(d(p_0, v) = d(v, p_0)\)).

Number of portals bounded by \(\lceil 2d(v, p_0)/\epsilon d(v, p_0) \rceil = \lceil 2/\epsilon \rceil\) per path \(Q^+, Q^-\). \(\square\)

\(^1\)Note: there is a variant of the oracle where, at query time, only \(O(1)\) path separators need to be considered. Space does not increase but preprocessing is more complicated and time-consuming.
Efficient preprocessing

How to compute $\epsilon$-cover? using (a variant of) MSSP for each path $Q$. split each node $q \in Q$ into $q_l, q_r$, split path to create one face $f_{\infty}$

- compute first portal $p_0$ for each node $v$ (one SSSP from dummy node)
- run MSSP once in each direction
  - forward: moving roots from the first node of $Q$ towards to the last, only add connection $q \in Q$ for a node $v$ with first portal $p_0(v)$ if $q$ comes after $p_0(v)$
  - backward: moving roots from the last node of $Q$ towards the first, only add connection $q \in Q$ for a node $v$ with first portal $p_0(v)$ if $q$ comes before $p_0(v)$
- let the tree $T$ currently be rooted at $r$. for each node $v$, maintain the value
  \[
  \sigma(v) = (1 + \epsilon) d_T(r, v) - \left( d(r, p_l(v)) + d(p_l(v), v) \right),
  \]
  where $p_l(v)$ denotes the last portal created for $v$. whenever $\sigma(v) < 0$, the node $v$ needs a portal. dynamic tree operations can efficiently find nodes $v$ with $\sigma(v) < 0$.

moving the root from $r_i$ to $r_{i+1}$, update $\sigma(v)$ as follows:

- if $r$ is the first portal for $v$, set $\sigma(v) := \epsilon d_T(r, v)$. also, whenever we add a portal $r$, we reset $\sigma(v) := \epsilon d_T(r, v)$ to that value
- in the standard MSSP algorithm, we add $\ell(r_i r_{i+1})$ (using AddSubtree) to maintain $d_T(r, v)$. analogously, we add $(1 + \epsilon) \ell(r_i r_{i+1})$ to maintain the first summand of $\sigma(v)$.
- furthermore, for an edge $uv$ being inserted (Cut($u'$v) and Join($uv$)), we update $\sigma(v')$ for all $v'$ in the subtree rooted at $v$. if the distance to $r_{i+1}$ decreased by $\alpha$, we subtract $(1 + \epsilon) \alpha$ from $\sigma(v')$

![Figure 1: When moving the root from $r_i$ to $r_{i+1}$, the tension of the edges on the fundamental cycle (yellow) change (see MSSP lecture for details). For such an edge $uv$, the MSSP algorithm deletes the edge previously pointing to $v$ (Cut($u'v$)) and adds $uv$ to the tree (Join($uv$)).](image)
Extensions

Bounded-genus graphs

tree-cotree decomposition, $O(g)$ paths at highest level of separation; thereafter, remaining graphs are planar

Minor-free graphs

small shortest-path separators do not exist. counterexample: grid with one apex (no $K_6$): all shortest paths use apex. need $\Omega(\sqrt{n})$ “paths”!

$k$–path separator sequence of sets of paths $\Pi_i$ with the property that all $Q \in \Pi_i$ are shortest in $G \setminus \bigcup_{j<i} \Pi_j$. $k$–path separator if total number of paths is at most $k$, i.e. $|\{Q : \exists i. Q \in \Pi_i\}| \leq k$, and strong $k$–path separator if $|\{\Pi_i\}| = 1$ (one set of paths, exists for planar graphs)

$H$–minor-free graphs have $k$–path separator for $k \leq k(H)$ (pf. using Robertson-Seymour decomposition)

Approximate distance oracle same as for planar graphs, only difference: need to compute portals in the right order (and cannot use MSSP)

References

Approximate distance oracles and shortest-path queries for planar graphs have been investigated by Thorup [Tho04] (both directed and undirected) and Klein [Kle02] (undirected graphs only). Earlier work includes dynamic data structures as well [KS98]. The preprocessing algorithm discussed is due to [Kle05] (see also [KKS11]). Extensions are known for bounded-genus [KKS11] and minor-free graphs [AG06]. Thorup’s oracle for undirected planar graphs has been implemented and evaluated experimentally [MZ07].


