6.889 — Lecture 13: Approximate Distance Oracles

Christian Sommer csom@mit.edu

October 26, 2011

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

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

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 $\mathcal{O}(\sqrt{n})$, 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 ∞

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 ($\mathcal{O}(\log n)$ levels, 2 paths per level), *(ii)* each node stores distances to portals on $\mathcal{O}(\log n)$ paths

Query given (v, w), find best path through all the separator paths "shared" by v and w

Portals: ϵ–cover

Approximate representation of shortest paths crossing a *separator path* using few portals. prove that, per node v and separator path Q, $\mathcal{O}(1/\epsilon)$ 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: *ϵ–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_l 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_l, more precisely,

$$d(v, p_l) + d(p_l, q) > (1+\epsilon)d(v, q),$$

then insert q as a new portal. by definition, each node on Q^+ is *ϵ–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 $\mathcal{O}(1/\epsilon)$.*

Distance oracle based on this claim, the approximate distance oracle uses space $\mathcal{O}(\epsilon^{-1} \log n)$ per node ($\mathcal{O}(1/\epsilon)$ distances to portals on $\mathcal{O}(\log n)$ paths) and answers distance queries in time $\mathcal{O}(\epsilon^{-1} \log n)$ as follows: for any pair of nodes (v, w) at most $\mathcal{O}(\log n)$ separator paths need to be considered.[1] for a separator path Q we can efficiently "merge" portals in time $\mathcal{O}(1/\epsilon)$ 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*

$$\Phi(Q^+, v, \{p_j\}) := d(v, p_l) + d(p_l, s),$$

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:

$$\begin{array}{rcll}
\text{new } \Phi^{(j+1)} & = & d(v, p_{j+1}) \qquad\qquad + d(p_{j+1}, s), & \\
\text{old } \quad \Phi^{(j)} & = & d(v, p_j) + d(p_j, s), & \text{which is} \\
\Phi^{(j)} & = & d(v, p_j) + d(p_j, p_{j+1}) + d(p_{j+1}, s), & \text{portal } p_{j+1} \text{ is added since} \\
 & & d(v, p_j) + d(p_j, p_{j+1}) > (1+\epsilon)d(v, p_{j+1}) & \text{and thus} \\
\Phi^{(j)} & > & (1+\epsilon)d(v, p_{j+1}) \qquad + d(p_{j+1}, s), & \\
\hline
\text{diff. } \Phi^{(j)} - \Phi^{(j+1)} & > & \epsilon d(v, p_{j+1}) \geq \epsilon d(v, p_0) &
\end{array}$$

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 fist 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^- □

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

Efficient preprocessing

How to compute ϵ–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_∞

- 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

  $$\sigma(v) = (1+\epsilon)d_T(r,v) - (d(r,p_l(v)) + d(p_l(v),v)),$$

  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 α, we subtract $(1+\epsilon)\alpha$ from $\sigma(v')$

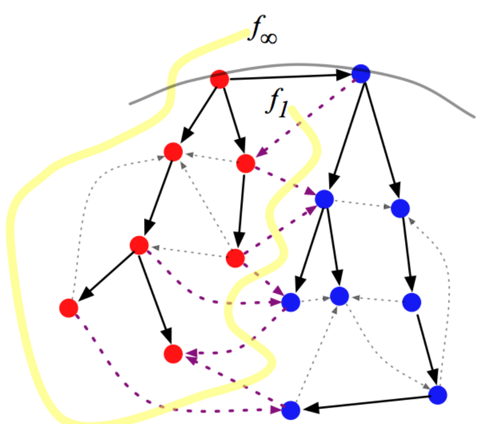

Figure 1: When moving the root from r_i to r_{i+1}, the tension of the edges on the fundamental cycle (yellow) change (see MSSP lecture for details). For such an edge uv, the MSSP algorithm deletes the edge previously pointing to v (CUT$(u'v)$) and adds uv to the tree (JOIN(uv)).

Extensions

Bounded-genus graphs

tree-cotree decomposition, $\mathcal{O}(g)$ 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 Π_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].

[AG06] Ittai Abraham and Cyril Gavoille. Object location using path separators. In *25th ACM Symposium on Principles of Distributed Computing (PODC)*, pages 188–197, 2006.

[KKS11] Ken-ichi Kawarabayashi, Philip Nathan Klein, and Christian Sommer. Linear-space approximate distance oracles for planar, bounded-genus and minor-free graphs. In *38th International Colloquium on Automata, Languages and Programming (ICALP)*, pages 135–146, 2011.

[Kle02] Philip Nathan Klein. Preprocessing an undirected planar network to enable fast approximate distance queries. In *13th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 820–827, 2002.

[Kle05] Philip Nathan Klein. Multiple-source shortest paths in planar graphs. In *16th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 146–155, 2005.

[KS98] Philip Nathan Klein and Sairam Subramanian. A fully dynamic approximation scheme for shortest paths in planar graphs. *Algorithmica*, 22(3):235–249, 1998. Announced at WADS 1993.

[MZ07] Laurent Flindt Muller and Martin Zachariasen. Fast and compact oracles for approximate distances in planar graphs. In *15th European Symposium on Algorithms (ESA)*, pages 657–668, 2007.

[Tho04] Mikkel Thorup. Compact oracles for reachability and approximate distances in planar digraphs. *Journal of the ACM*, 51(6):993–1024, 2004. Announced at FOCS 2001.