6.889 — Lecture 11: Multiple-Source Shortest Paths

Christian Sommer csom@mit.edu
(figures by Philip Klein)

October 19, 2011

Single-Source Shortest Path (SSSP) Problem: given a graph $G = (V, E)$ and a *source* vertex $s \in V$, compute shortest-path distance $d_G(s, v)$ for each $v \in V$ (and encode shortest-path tree)

Multiple-Source Shortest Path (MSSP) Problem: given a graph $G = (V, E)$ and a *source* **set** $S \subseteq V$, compute shortest-path distance $d_G(s, v)$ for *some* $(s, v) \in S \times V$ (and encode shortest-path trees rooted at each $s \in S$)

Assumption (all of Lecture 11) *planar* G (extends to *bdd. genus*), non-negative edge lengths $\ell : E \to \mathbb{R}^+$

Straightforward SSSP for each source $s \in S$, time and encoding size $\mathcal{O}(|S| \cdot n)$

This Lecture if all $s \in S$ on *single face* f, time and encoding size $\mathcal{O}(n \log n)$ (*independent* of $|S|$ / face size!)

Why? one important application: all-pairs shortest paths between boundary nodes of a piece in r–division. requires only time $\mathcal{O}(r \log r)$ (instead of $\mathcal{O}(r^{3/2})$)

How? Main Idea compute one *explicit* shortest-path tree rooted at a root $r_i \in f$, then *modify* tree to obtain shortest-path tree rooted at neighbor $r_{i+1} \in f$. tree changes: some edges not in tree anymore, some new edges join.
modify using *dynamic trees*, each modification can be done in time $\mathcal{O}(\log n)$

- how many modifications?
- which edges to modify?

How many modifications?

Claim. *Going around the face f, for each edge $e \in E$:*

- *e joins the tree at most once and*
- *e leaves the tree at most once.*

total of at most $2|E|$ modifications, each takes time $\mathcal{O}(\log n) \rightsquigarrow$ overall running time $\mathcal{O}(n \log n)$

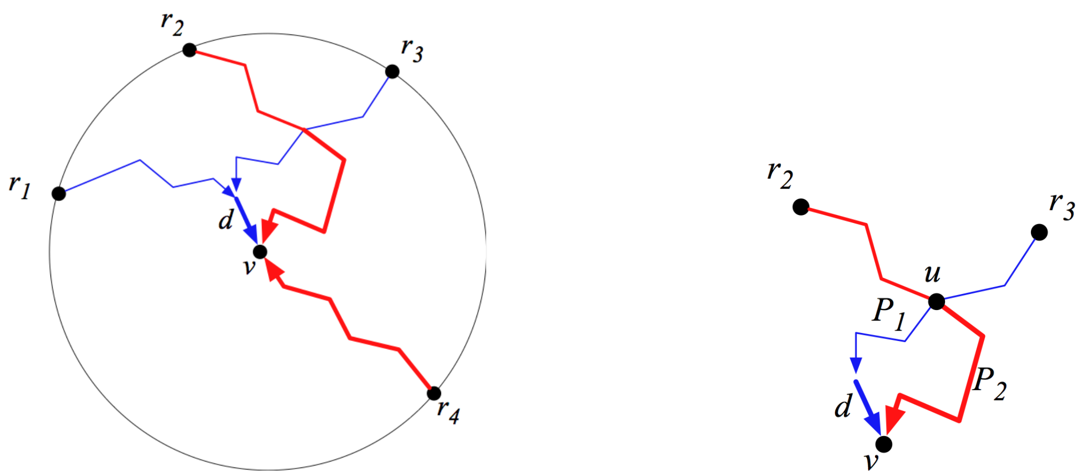

Figure 1: Roots whose shortest-path trees contain dart d form an interval. Pf. by contradiction. Assume *unique* shortest paths. Suppose d is in r_1– and r_3–rooted SP trees but neither in r_2– nor r_4–rooted SP tree. If shortest r_1–to–v and r_3–to–v paths **do use** d but shortest r_2–to–v and r_4–to–v paths **do not use** d, one of the latter paths must cross one of the former (planarity). Let u denote a vertex where they cross. Since $P_1 \circ d$ is shortest u–to–v path, its length is shorter than that of P_2, which implies that d must be in r_2–rooted SP tree, a contradiction.

Recall: Dynamic Trees

the following operations can be implemented in $\mathcal{O}(\log n)$ amortized time

primal: need four operations of Euler-Tour trees

- Cut(e) removes edge e from forest
- Join(e) adds edge e, joins two trees
- GetValue(v) returns root distance $d_T(r, v)$
- AddSubtree(Δ, x) increases distances in subtree of x by Δ

dual: need four operations of link-cut trees / top trees

- Cut(e) removes edge e from forest
- Join(e) adds edge e, joins two trees
- MaxPath(π) finds edge with maximum *tension* (to be defined)
- AddPath(Δ, π) adds $\pm\Delta$ to tension of edges in π

Which edges enter the tree?

Tension for an edge uv define its *tension* $t(uv) = d_T(r, v) - \ell(uv) - d_T(r, u)$, where T is a tree rooted at r. edge is *tense* if $t(uv) > 0$ (shorter path to v via node u). if no tense edge $uv \in T$ then T is shortest-path tree.

Idea maintain tension for every non-tree edge (recall: non-tree edges form *interdigitating tree*). **gradually** move from old root r_i to new root r_{i+1}. blue nodes: already "under" r_{i+1}, no change; red nodes: not yet. changing tension: edges in fundamental cycle in $(G \setminus T_i)^*$ defined by $(r_i r_{i+1})^*$

primal	*dual*
let $D = d_{T_i}(r_i, r_{i+1})$ (which is $\ell(r_i, r_{i+1})$ if $r_i r_{i+1} \in T_i$; shorter otherwise — assume $r_i r_{i+1} \in T_i$).	
let $T = T_i$.	let $\pi^* = \pi(f_1 f_\infty)$ (path in fundamental cycle).
• edge uv with max. tension (found in dual), let $t(uv) = \Delta$	• $(uv)^*$:=MaxPath(π^*)
If $\Delta \leqslant D$: move root from r_i to r_{i+1} by Δ (decrease D by Δ)	
• AddSubtree(r_i, Δ) • AddSubtree$(r_{i+1}, -\Delta)$	• AddPath$(\pi^*, \pm 2\Delta)$
delete $u'v \in T$ from T. add uv to T.	
• Cut$(u'v)$ • Join(uv)	• Cut$((uv)^*)$ (cannot change anymore) • Join$((u'v)^*)$
(some red nodes are now blue)	(fundamental cycle and π^* changed)

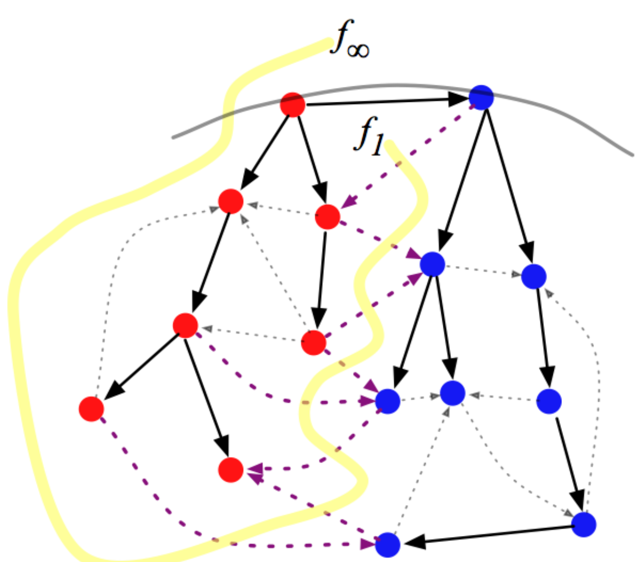

Figure 2: Tension of the edges in the fundamental cycle in $(G \setminus T_i)^*$ defined by $f_\infty f_1 = (r_i r_{i+1})^*$ changes.

Query data structure

Multiple-Source Shortest Path (MSSP) Data Structure: given planar G and face f_∞, preprocess into data structure of size $\mathcal{O}(n \log n)$ such that queries $d_G(r, v)$ for r on f_∞ and $v \in V(G)$ can be answered in $\mathcal{O}(\log n)$

Main Idea GETVALUE(v) of Euler Tour tree returns root distance $d_T(r, v)$. we did compute T during preprocessing. need to efficiently *recover* the right T at query time $\rightsquigarrow$ can be done using *persistent* data structure, "remember" all changes made to dynamic tree, recover *any* state of the data structure

r–division with $\mathcal{O}(1)$ "holes" per piece

Application all-pairs shortest paths between boundary nodes of a piece in r–division. using MSSP: requires only time $\mathcal{O}(r \log r)$ (instead of $\mathcal{O}(r^{3/2})$)

Need boundary nodes on $\mathcal{O}(1)$ faces! let *holes* of a piece be internal faces containing boundary nodes

Lemma. *For planar G, an r–division with $\mathcal{O}(1)$ holes per piece can be computed in time $\mathcal{O}(n \log r + nr^{-1/2} \log n)$.*

Idea interleave separator steps: in odd steps, separate nodes, in even steps, separate holes

References

The *Multiple-Source Shortest Paths* problem for planar graphs and the corresponding data structure was first considered by Klein [Kle05] (earlier work considered grid graphs [Sch98]). Cabello and Chambers [CC07] simplified and extended it to graphs of genus g, obtaining an algorithm that runs in time $\mathcal{O}(g^2 n \log n)$. For planar graphs, an MSSP algorithm has been implemented and evaluated experimentally [EK11].

Both algorithms and data structures rely on efficient *dynamic tree* representations such as [HK99, TW05]. Dynamic data structures can be made persistent using minimal overhead [DSST89].

[CC07] Sergio Cabello and Erin W. Chambers. Multiple source shortest paths in a genus g graph. In *18th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 89–97, 2007.

[DSST89] James R. Driscoll, Neil Sarnak, Daniel Dominic Sleator, and Robert Endre Tarjan. Making data structures persistent. *Journal of Computer and System Sciences*, 38(1):86–124, 1989. Announced at STOC 1986.

[EK11] David Eisenstat and Philip N. Klein. A dynamic-tree product line, 2011.

[HK99] Monika Rauch Henzinger and Valerie King. Randomized fully dynamic graph algorithms with polylogarithmic time per operation. *Journal of the ACM*, 46(4):502–516, 1999. Announced at STOC 1995.

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

[Sch98] Jeanette P. Schmidt. All highest scoring paths in weighted grid graphs and their application to finding all approximate repeats in strings. *SIAM Journal on Computing*, 27(4):972–992, 1998. Announced at ISTCS 1995.

[TW05] Robert Endre Tarjan and Renato Fonseca F. Werneck. Self-adjusting top trees. In *16th ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 813–822, 2005.