Principles of Distributed Computing

Sample Solutions: Low-Congestion Shortcuts in Planar Networks

ETH Eidgenössische Technische Hochschule Zürich | Swiss Federal Institute of Technology Zurich
Distributed Computing
FS 2026 | Dr. G. Zuzic

Task 1: The $O(D)$ Congestion, $O(D^2)$ Dilation Construction

Solution 1.1 (Congestion)

We want to bound the number of distinct shortcut subgraphs H_i containing a specific BFS tree edge $e = (v, u)$, where u is the parent of v. Edge e is included in H_i if S_i satisfies Condition 1 (Ancestor) or Condition 2 (Encapsulated).

1. **Congestion from Ancestors:** Because the BFS tree T has depth at most D, edge e has at most D tree ancestors. Since the sets $S_1, \dots, S_N$ are pairwise disjoint, at most D distinct parts can contain an ancestor of e. Thus, Condition 1 contributes at most D to the congestion.
2. **Congestion from Encapsulation:** Let k be the number of parts S_i that encapsulate e but do not contain an ancestor of e. For each such part S_i, consider its three encapsulation tree paths originating from the root r: the Left Path (to the leftmost node of S_i), the Right Path (to the rightmost node of S_i), and the Central Path (passing through e to a descendant $c \in S_i$).

In the planar embedding, the Left Path, Right Path, and S_i form a closed cycle in the plane. The Central Path splits this enclosed region into two closed pockets. For any other part S_j that also encapsulates e, if its encapsulation paths do not intersect S_i's encapsulation paths, S_j must lie entirely within one of these pockets. However, for S_j to encapsulate e, its encapsulation paths must originate outside the pocket at root r and pass through e. Thus, by the Jordan Curve Theorem, S_j's encapsulation paths must physically cross either S_i or S_i's encapsulation paths.

To formalize this, define an auxiliary intersection graph $\mathcal{G}_e$ where nodes represent the k parts encapsulating e, and a directed edge $S_i \to S_j$ exists if S_i touches any of the three encapsulation paths of S_j. Since S_j's encapsulation paths have depth $\leq D$, they contain at most $3D$ nodes. Because all parts S_i are vertex-disjoint, at most $3D$ distinct parts can touch S_j's encapsulation paths. Thus, the in-degree of every node in $\mathcal{G}_e$ is at most $3D$, meaning the average degree in the undirected version of $\mathcal{G}_e$ is at most $6D$.

By Turán's Theorem, any k-node graph with average degree $x \leq 6D$ has an independent set of size at least $\frac{k}{x+1} \geq \frac{k}{6D+1}$. An independent set represents a collection of parts where no part touches the encapsulation paths of any other. As shown by the Jordan Curve Theorem, in a planar graph ($g = 0$), we cannot have even 2 independent encapsulating parts without their paths crossing ($K_{3,3}$ minor obstruction). Setting $\frac{k}{6D+1} \leq 1$ yields $k \leq 6D + 1 = O(D)$.

Summing the contributions gives total congestion $D + (6D + 1) = O(D)$.

□

Solution 1.2 (Dilation)

Consider an arbitrary part S_i and let $\mathcal{P}$ be the union of its two boundary paths connecting the LCA to its leftmost and rightmost nodes. Since T has depth D, $\mathcal{P}$ contains at most $2D$ nodes.

Partition the nodes of S_i into equivalence classes such that nodes in the same class share the exact same lowest BFS tree ancestor on $\mathcal{P}$. Since $\mathcal{P}$ has at most $2D$ nodes, there are at most $2D$ classes.

For any node $w \in S_i$, consider the BFS tree path Q_w connecting w upwards to its lowest ancestor on $\mathcal{P}$. Every edge on Q_w is a BFS tree edge that has a descendant (w) in S_i but does not lie on $\mathcal{P}$. Therefore, every edge on Q_w is encapsulated by S_i and is explicitly included in H_i under Condition 2.

This guarantees that any two nodes u, v in the same boundary class can reach their common boundary ancestor in at most $2D$ hops using H_i edges. To traverse between different classes, imagine contracting each boundary class (along with its tree paths to $\mathcal{P}$) into a single 'big-node'. Because S_i is connected, there is a path traversing $G[S_i]$ edges connecting any two big-nodes. This contracted path traverses at most $2D$ big-nodes (since there are at most $2D$ classes in total).

Expanding the contractions back to $G[S_i]+H_i$, crossing each big-node requires traversing up to the boundary ancestor and back down, costing $\leq 2D$ hops per class. Across all $2D$ classes, the total hop distance between any two nodes in S_i is at most $2D \times 2D = 4D^2 = O(D^2)$.

□

Task 2: The $O(D \log D)$ Congestion and Dilation Construction

Solution 2.1 (Congestion)

We examine how many distinct shortcut subgraphs H_j can include a specific candidate edge e as a boundary edge under Version 2.

Walking upwards from e along the BFS tree path towards the root: 1. Let w_1 be the lowest LCA of any part S_j that uses e as a boundary edge. The path from e to w_1 defines **Segment 1**. 2. Within Segment 1, planar topological constraints (similar to Solution 1.1) guarantee that at most $O(D)$ parts can have their first junction in this segment without their encapsulation paths crossing. 3. Now consider parts whose LCA lies strictly above w_1. Let w_2 be the lowest LCA among these remaining parts. The path between w_1 and w_2 defines **Segment 2**.

For a part with LCA at w_2, its first junction u' must lie on or above w_1 (otherwise it would have been included in Segment 1). Therefore, the distance ℓ'_1 from e to junction u' is at least $\text{dist}(e, w_1)$. The threshold rule requires that the distance ℓ'_2 from junction u' to LCA w_2 must be at least $\frac{\ell'_1}{2}$.

This implies the total tree distance from e to w_2 satisfies:

$$\text{dist}(e, w_2) = \ell'_1 + \ell'_2 \geq \ell'_1 + \frac{\ell'_1}{2} = \frac{3}{2}\ell'_1 \geq \frac{3}{2}\text{dist}(e, w_1)$$

Because the distance from e to the lowest LCA grows by at least a factor of $3/2$ at each segment, and the maximum tree depth is D, there can be at most $O(\log D)$ distinct geometric segments along the path.

Since each segment can contribute at most $O(D)$ parts (via planar topology), the total congestion on edge e is exactly:

$$O(\log D) \times O(D) = O(D \log D)$$

□

Solution 2.2 (Dilation)

To bound dilation, we analyze how many boundary edges are skipped (omitted from H_i) for a specific part S_i.

Walking down the boundary path from the LCA towards the leftmost node, the junction nodes divide the boundary path into consecutive intervals called **pieces**.

Suppose a piece j is skipped (meaning its lowest edge e_j fails the $\ell_2 \geq \frac{\ell_1}{2}$ test). If e_j fails the test, it means $\ell_2 < \frac{\ell_1}{2}$, or equivalently $\ell_1 > 2\ell_2$. Here, ℓ_1 is the length of piece j, and ℓ_2 is the tree distance from the junction strictly above piece j to the LCA. Note that ℓ_2 is exactly the sum of the lengths of all preceding pieces 1 to $j-1$.

Therefore, for piece j to be skipped, its length must satisfy:

$$\text{Length}(\text{Piece } j) > 2 \times \sum_{k=1}^{j-1} \text{Length}(\text{Piece } k)$$

This implies that every time we encounter a skipped piece, the total tree distance from the LCA to the bottom of that piece more than triples. Because the maximum depth of the tree is D, the distance from LCA can triple at most $O(\log D)$ times. Therefore, there are at most $O(\log D)$ skipped boundary pieces.

Because at most $O(\log D)$ boundary pieces are skipped, the nodes of S_i are partitioned into at most $O(\log D)$ connected boundary classes (instead of $2D$ classes in Version 1). Within each class, nodes reach their common boundary ancestor in at most $2D$ hops using encapsulated edges. Traversing across all $O(\log D)$ classes takes at most $O(\log D) \times 2D = O(D \log D)$ hops.

□