平面的グラフ上で距離ラベルを高速に計算する分散アルゴリズム

泉 泰介[1,a)]

概要：グラフ $G = (V, E, w)$ の距離ラベルとは，任意の 2 頂点 $u, v \in V$ 間の G における距離をそれら 2 頂点に割り当てられたラベル $L_G(u), L_G(v)$ の情報のみから計算可能であるような，V 中の各頂点へのラベル付け L_G である．本研究では，平面的グラフに対してラベルサイズ $\tilde{O}(D)$ ビットの距離ラベルを構成する，CONGEST モデル上での分散アルゴリズムを提案する (D はグラフ G のホップ直径を表す)．提案アルゴリズムは G が重みなしグラフの場合 $\tilde{O}(D)$ ラウンド，重み付きグラフの場合は $\tilde{O}(D^2)$ ラウンドで動作する．また，このアルゴリズムを用いて，平面的グラフにおける重み付き最短経路問題の厳密解，ならびに直径の $(1+\epsilon)$-近似解 (ϵ は定数) を同様の時間で与える分散アルゴリズムを併せて提示する．

キーワード：分散アルゴリズム，CONGEST モデル，距離ラベル，平面的グラフ

Distribted Construction of Distance Labels in Planar Graphs

Taisuke Izumi[1,a)]

Abstract: The distance labeling of a graph $G = (V, E)$ is the assignment L_G of a label to each node such that the distance between any two nodes $u, v \in V$ is computed only from their labels $L_G(u)$ and $L_G(v)$. In this paper, we newly propose the distributed algorithm which constructs the exact distance labels of $\tilde{O}(D)$ bits (where D is the hop diameter of G) in the CONGEST model. The construction time is $\tilde{O}(D)$ rounds if G is unweighted, and $\tilde{O}(D^2)$ rounds if G is weighted. Utilizing this algorithm as a building block, we also show two distributed algorithms for solving the exact weighted single-source shortest path and the $(1+\epsilon)$-approximate diameter for any constant ϵ. The running time of these algorithms have the same asymptotic bound as the distance-label construction.

Keywords: distributed algorithm, CONGEST model, distance labeling, planar graph

1. Introduction

1.1 Background and Motivation

There is absolutely no objection to claiming that the shortest path problem is one of the kings in fundamental graph problems, which has a huge number of applications in various areas, including distributed computing (e.g., distance-vector routing). The theory of distributed graph algorithms tackles how to solve fundamental graph problems in distributed

[1] Nagoya Institute of Technology, Gokiso-cho, Showa-ku, Nagoya, Aichi, 466-8555, Japan.
[a)] t-izumi@nitech.ac.jp.

ways. The distributed complexity of shortest-path problems in the CONGEST model receives much attention. The CONGEST model is one of the standard message-passing models often used in distribute graph algorithms, which runs following synchronous rounds, and each link can transfer only a "short" (i.e., $O(\log n)$-bit) message per one round (where n is the number of nodes). The inherent limitation of bandwidth precludes any centralized solution by aggregating the whole topological information of the network, and thus the computation must proceed in parallel at each node only using a partial information of the input graph, which is the central difficulty in the CONGEST model.

In this paper we consider various types of distance computation: The weighted single-source shortest path, distance labeling, and diameter. In the last decade, these problems are widely studied, and their distributed complexity in the CONGEST model attains much progress (mainly with respect to time). Since all the distance problems are categorized into the global problems, they exhibit the universal lower bound of $\Omega(D)$ rounds, where D is the hop diameter of the input graph, and the terminology "universal" means the lower bound holding for *any* instances. Conversely, attaining (near) $\tilde{O}(D)$-round upper bounds is the ultimate goal in the algorithm design. However, in the general case, almost all distance problems have much expensive lower bounds: The (approximate) shortest path problem in weighted graphs faces the typical $\tilde{\Omega}(\sqrt{n} + D)$-round lower bound [1], and the exact diameter and all-pair shortest paths (APSP) have more expensive lower bounds of $\tilde{\Omega}(n)$ rounds [2]. These bounds are obtained by showing some instances which have a small cut (communication bottleneck) but a large amount of information must be transferred through the cut, and thus those expensive lower bounds are a kind of "existential" bounds (i.e., holding only for specific hard-core instances). This observation brings us an interest of exploring reasonable graph classes which allow fast distance computation, and it is actually the primary motivation of this paper.

1.2 Our Contribution

Given a graph G, if it has an embedding into the 2D-Euclidean plane without crossing edges, it is called a *planar graph*. In the context of centralized graph algorithms, the planar graph enables efficient solutions for many computationally hard problems. Recently, with the development of several technical tools, the study on the distributed complexity for planar graphs (and related classes under the umbrella of minor-closed family) is emerging [5], [6], [7], [8], [9]. One of the results initiating this research direction is an $\tilde{O}(D)$-round MST algorithm for planar graphs by Haeupler and Ghaffari [5], [6]. Similar with the weighted shortest path, MST also has $\tilde{\Omega}(\sqrt{n} + D)$ rounds in general cases, and thus the feasibility of such a fast (universally optimal) MST algorithm raises up the question of what problems can be accelerated like as MST, and can achieve the universal optimality. The main contribution of this paper is to answer this question positively for several distance problems. The precise statement of our result is summarized as follows:

- For any planar graphs, it is possible to solve the weighted single-source shortest path problem exactly in $\tilde{O}(D^2)$ rounds.
- For any weighted planar graphs, the $(1 + \epsilon)$-approximate diameter can be computed in $\tilde{O}(D^2)$ rounds. If the input graph is unweighted, the running time is improved to $\tilde{O}(D)$ rounds.

The key building block of the two algorithms above is a distributed algorithm of constructing *exact distance labeling* for weighted/unweighted planar graphs, which assign each node with the label such that the distance between two nodes u and v can be computed from the two labels assigned to u and v. The round complexity of the construction is $\tilde{O}(D^2)$ for weighted graphs and $\tilde{O}(D)$ for unweighted graphs. While it is known that the centralized construction of distance labeling for planar graphs [3], its efficient distributed implementation is not addressed so far. The author believes that this part may be of independent interest and potentially has many other applications.

2. Preliminaries

2.1 CONGEST model

A distributed system consists of n nodes interconnected with m communication links. We model it by an undirected weighted or unweighted graph $G = (V_G, E_G, w_G)$, where $V_G = [0, n-1]$ is the set of nodes (i.e., each node is identified by an integer value), $E_G \subseteq V_G \times V_G$ is the set of links (edges), and $w_G \colon E(G) \to \mathbb{N}$ is a function assigning each edge with some integer weight. When we treat unweighted graphs, we define $w_G(e) = 1$ for any $e \in E_G$, and usually represents G as $G = (V_G, E_G)$. Executions of the system proceed with a sequence of consecutive rounds. In each round, each process sends a (possibly different) message to each neighbor, and within the round, receives all messages from the neighbors. After receiving the messages, it performs local computation. The number of bits transmittable through any communication link per one round is restricted to $O(\log n)$ bits. Note that in weighted networks the weight of each edge does not imply the delay of communication. It is guaranteed that messages transferred through weighted edges reach their destinations within one round.

2.2 Notations

In the following argument, we denote the vertex and edge sets of any graph H by V_H and E_H respectively. The (weighted) distance bewteen $u, v \in V_G$ with respect to a subgraph H (i.e., the length of the shortest path between u and v in H) is denoted by $d_H(u, v)$. For any node $v \in V_G$ and a subset $X \subseteq V_G$, we also define the *distance set* $d_G(u, X)$ as $d_G(u, X) = \{(v, d_G(u, v)) \mid v \in X\}$. Given a weighted graph $G = (V_G, E_G, w_G)$, its *hop diameter* (the diameter of the unweighted graph (V_G, E_G)) and *weighted diameter* (the diameter of G in the standard sense) are respectively denoted by D and D_w. If G is an unweighted graph, $D = D_w$ obviously holds. Given two graphs G_1 and G_2, we define their union as $G_1 \cup G_2 = (V_{G_1} \cup V_{G_2}, E_{G_1} \cup E_{G_2})$. The intersection of G_1 and G_2 is defined as the subgraph of $G_1 \cup G_2$ induced by $V_{G_1} \cap V_{G_2}$. We use notation $H \subseteq G$ to state that H is a subgraph of G.

Letting $\boldsymbol{x} = x_0 x_1 \ldots x_{k-1}$ and $\boldsymbol{y} = y_0 y_1 \ldots y_{j-1}$ be any two strings, we denote its concatenation $x_0 x_1 \ldots x_{k-1} y_0 y_1 \ldots y_{j-1})$ by $\boldsymbol{x} \circ \boldsymbol{y} =$. We also use the notation $\boldsymbol{x} \sqsubseteq \boldsymbol{y}$ if $\boldsymbol{x}$ is a prefix of $\boldsymbol{y}$, and $\boldsymbol{x} \parallel \boldsymbol{y}$ if neither $\boldsymbol{x} \sqsubseteq \boldsymbol{y}$ nor $\boldsymbol{y} \sqsubseteq \boldsymbol{x}$ holds. Given a collection X of strings, $\boldsymbol{x} \in X$ is called *maximal* with respect to X if no $\boldsymbol{y} \in X \setminus \{\boldsymbol{x}\}$ satisfies $\boldsymbol{x} \sqsubseteq \boldsymbol{y}$.

2.3 Assumptions

The node with the minimum identitier is treated as the special node called the *root* of G, and referred as r. We define T_{BFS} as an arbitrary (unweighted) BFS tree rooted by r, where "unweighted" means that it is a shortest-path tree constructed with omitting edge weights. For simplicity of presentation, we assume that the height of T_{BFS} is equal to D. Since the actual height of T_{BFS} is $\Theta(D)$, this assumption does not affect the correctness of any asymptotic analysis hereafter. We define $A_{\mathrm{BFS}}(v)$ as the path to the root from v in T_{BFS}. Given a subgraph H of G, $T_{\mathrm{BFS}}(H)$ represents the subgraph of T_{BFS} which spans H (that is, $T_{\mathrm{BFS}}(H)$ is a spanning forest of H). We regard a subgraph of T_{BFS} as a set of rooted trees. That is, for any connected component $C \subseteq T_{\mathrm{BFS}}$, the node closest to r (with respect to T_{BFS}) is the root of C The set of the root nodes for all connected components in $T_{\mathrm{BFS}}(H)$ is denoted by $R_{\mathrm{BFS}}(H)$.

Throughout this paper, we assume that each node v knows the following information: The value of n and D, the identifier of r, and $A_{\mathrm{BFS}}(v)$ (as the sequence of nodes from v to r). This assumption is not essential because all of the information becomes available by a trivial $O(D)$-round preprocessing.

2.4 Distance labeling Scheme

A *distance labeling scheme* for a graph family $\mathcal{G}$ is defined as the pair (L_G, f_G) of two functions determined by the input graph $G \in \mathcal{G}$, which are respectively called a *node-labeling function* and a *distance-decoder function* respectively. The node labeling function $L_G : V(G) \to \{0,1\}^*$ assigns a label to each node, and the decoder function $f_G : \{0,1\}^* \times$

$\{0,1\}^* \to \mathbb{N}$ recovers the distance between any two nodes u and v from $L_G(v)$ and $L_G(u)$, i.e., for any $u, v \in V(G)$, $f_G(L_G(u), L_G(v)) = d_G(u, v)$ holds. The size of a distance labeling scheme is defined as the maximum label size $\max_{v \in V(G), G \in \mathcal{G}} |L(v)|$.

3. Construction of Distace Labels

3.1 A Brief Overview of the Construction by Gavoille et al.

Our algorithm is based on the distance labeling scheme by Gavoille et al. [3]. It constructs the label of each node following the recursive partition of the input graph by vertex separators. The idea is roughly as follows: First, the algorithm partitions the graph into three disjoint subgraphs G^+, S, G^- using a balanced vertex separator S. The nodes in G^a $(a \in \{+, -\})$ recursively construct the distance label for G^a. In addition, each node $v \in V_G$ also maintains the distance set $d_G(v, V_S)$. Since S is a vertex separator, the distance between two nodes $u \in V(G^+)$ and $v \in V(G^-)$ can be computed as $d_G(u, v) = \min_{w \in V_S}(d_G(u, w) + d_G(w, v))$. In the case of $u, v \in V_{G^a}$ ($(a \in \{+, -\})$, we can compute the distance as $d_G(u, v) = \min\{\min_{w \in V_S}(d_G(u, w) + d_G(w, v)), f_{G^a}(L_{G^a}(u), L_{G^a}(v))\}$. The depth of this recursive construction is at most $O(\log n)$ because of the balancing property of S. Hence provided that the upper bound s of the separator size, we can obtain a distance labeling scheme with $O(s \log^2 n)$-bit size.

Following the framework by Gavoille et al., we need to find the separator of $\tilde{O}(D)$ nodes for attaining our goal of $\tilde{O}(D)$-bit labels. While it is known that any planar graph has a balanced separator S of $O(D)$ nodes, the separated subgraphs G^+ and G^- might have the diameters larger than D, which prevents us from applying the same separation strategy to G^+ and G^- recursively. An approach to address this matter is to decompose G into several subgraphs which can overlap each other. For example, it is easy to show that the diameters of the subgraphs $G^+ \cup S$ and $G^- \cup S$ are both $O(D)$, and thus we can apply the recursive construction to them. Unfortunately, this approach causes two other problems: First, if decomposed subgraphs overlap, some edge e can be shared among many separated subgraphs. Then, when we want to run some algorithm concurrently in each decomposed subgraph (which is crucial for the recursive construction), e may suffer high congestion. Similarly, if a node v is shared among many decomposed subgraphs, v must maintain the distance labels for all subgraphs containing v. It implies that the size of v's label for the whole graph G can become huge, which is the second problem we point out.

The technical ingredient of our algorithm for resolving the problems above is twofold: The first idea is to introduce a new graph decomposition scheme called (l, c, d, s)*-decomposition*, which characterizes the properties so that we can construct a small-size distance labeling efficiently, as well as a fast distributed algorithm for obtaining that decomposition in planar graphs. The second idea is to propose a new scheme of label construction not relying on recursive calls, which is very close to the original scheme but substantially saves the size of labels. In the remainder of this section, we introduce the concepts above, and present our distributed algorithm for constructing the labels, provided that a decomposition is given. The decomposition algorithm itself is explained in Section 4.

3.2 (l, c, d, s)-Decomposition: Definition

Each subgraph generated in the (l, c, d, s)-decomposition scheme is indexed by some string $\boldsymbol{x}$ over alphabet $[0, n-1]$. The initial graph is defined as $G_\phi = G$, where ϕ is the string of length zero. The length $|\boldsymbol{x}|$ of $\boldsymbol{x}$ implies the depth of the recursion, and the progress of the recursion by depth one decomposes $G_{\boldsymbol{x}}$ into several subgraphs $G_{\boldsymbol{x} \circ 0}, G_{\boldsymbol{x} \circ 1}, \ldots, G_{\boldsymbol{x} \circ k}$, which are called the *children* of $G_{\boldsymbol{x}}$. The decomposition is recursively applied to each child until its size (w.r.t. the number of nodes) becomes sufficiently small. Let $C(\boldsymbol{x})$ be the set of indices $i \in [0, n-1]$ such that $G_{\boldsymbol{x} \circ i}$ is defined. In the decomposition process, a separator $S_{\boldsymbol{x}} \subseteq G_{\boldsymbol{x}}$ is associated with each subgraph $G_{\boldsymbol{x}}$. We formally specify the properties that (l, c, d, s)-decomposition must satisfy:

Definition 1 A (l, c, d, s)-decomposition of G is an index set $\mathcal{B} \subseteq [0, n-1]^*$ and two collections $\mathcal{G} = \{G_{\boldsymbol{x}}\}_{\boldsymbol{x} \in \mathcal{B}}$ and $\mathcal{S} = \{S_{\boldsymbol{x}}\}_{\boldsymbol{x} \in \mathcal{B}}$ of subgraphs of G satisfying the following conditions:

(1) (**Hierarchical Decomposition**) Any child of $G_{\boldsymbol{x}}$ is a subgraph of $G_{\boldsymbol{x}}$, and their union $\cup_{i \in C(\boldsymbol{x})} G_{\boldsymbol{x} \circ i}$ is equal to $G_{\boldsymbol{x}}$.

(2) (**Small-Size Separation**) The number of vertices in $S_{\boldsymbol{x}}$ is bounded by s, and for any two different children $G_{\boldsymbol{x} \circ i}$ and $G_{\boldsymbol{x} \circ j}$ $(i, j \in C(\boldsymbol{x}))$ of $G_{\boldsymbol{x}}$, their intersection $G_{\boldsymbol{x} \circ i} \cap G_{\boldsymbol{x} \circ j}$ is a subgraph of $S_{\boldsymbol{x}}$. In addition, if $\boldsymbol{x}$ is maximal with respect to $\mathcal{B}$, $G_{\boldsymbol{x}} = S_{\boldsymbol{x}}$ holds.

(3) (**Low-Congestion**) For any $e \in E_G$ and $h \in [0, l]$, there exists at most c strings $\boldsymbol{x} \in \mathcal{B}$ with length $|\boldsymbol{x}| = h$ such that $G_{\boldsymbol{x}}$ contains e.

(4) (**Low-Dilation**) The diameter of $G_{\boldsymbol{x}}$ is bounded by d.

3.3 Constructing Distance Labels

The distance label $L_G(v)$ for each node v is represented by a triple $(v, \boldsymbol{s}(v), \lambda(v))$, where $\boldsymbol{s}(v)$ is the shortest string $\boldsymbol{x}$ satisfying $v \in S_{\boldsymbol{x}}$. Note that $\boldsymbol{s}(v)$ is well-defined because $S_{\boldsymbol{x}} = G_{\boldsymbol{x}}$ holds for any maximal string $\boldsymbol{x}$. The key idea of our scheme is that $\lambda(v)$ contains the distance set $d_{G_{\boldsymbol{y}}}(v, V_{S_{\boldsymbol{y}}})$ only for any $\boldsymbol{y} \sqsubseteq \boldsymbol{s}(v)$. Since the total number of entries in $\lambda(G, v)$ is at most ls, the label size of this scheme is bounded by $O(ls \log n)$ bits. Let $V_\lambda(v) = \{u \in V \mid (u, \cdot) \in \lambda(v)\}$, and $d_\lambda(v, u) = \min_{(u,x) \in \lambda(v)} x$. If $u \notin V_\lambda(v)$ holds, we define $d_\lambda(v, u) = \infty$. Note that $d_\lambda(v, u)$ is locally computed from $L_G(v)$. The distance computation function f_G is defined as follows:

$$f_G(L_G(v_1), L_G(v_2)) = \min_{u \in V_\lambda(v_1) \cap V_\lambda(v_2)} d_\lambda(v_1, u) + d_\lambda(u, v_2).$$

We can show that this function correctly computes the exact distance between two nodes.

Lemma 1 $f_G(L_G(v_1), L_G(v_2)) = d_G(v_1, v_2)$.

3.4 Distributed Implementation

Assume that a (l, c, d, s)-decomposition is given and each node v knows all $\boldsymbol{x}$ such that $v \in S_{\boldsymbol{x}}$ holds. Let $\mathcal{G}_h$ be the set of subgraphs $G_{\boldsymbol{x}}$ satisfying $|\boldsymbol{x}| = h$, which we call *level-h graphs*. By the definition of the (l, c, d, s)-decomposition scheme, all the level-h subgraphs $G_{\boldsymbol{x}} \in \mathcal{G}_h$ can run their own tasks highly in parallel. Since some edge e can belong to several subgraphs in $\mathcal{G}_h$, the parallel runs of those tasks cannot be completely independent due to the edge congestion. Fortunately, our decomposition scheme guarantees that any edge e is contained in at most c subgraphs in $\mathcal{G}_h$, we can complete any independent r-round tasks executed in level-h subgraphs within rc rounds [1]. For the construction of distance labels, we take two different approaches according to the existence of edge weights:

Unweighted Graphs: If G is unweighted, each node $v \in G_{\boldsymbol{x}}$ has to compute the distance set $d_{G_{\boldsymbol{x}}}(v, V_{S_{\boldsymbol{x}}})$. To compute it, we can use the distributed bellman-ford algorithm [10], which provides the construction of s BFS trees with different sources in $O(d+s)$ rounds. Since we can run the algorithms for the graphs with the same level in parallel (with extra c factor), the total running time is $O(cl(d+s))$ rounds.

Weighted Graphs: If G is weighted, the following task is executed in the bottom-up way (i.e., solving the collection of tasks for level-h subgraphs in the order of $h = l, l-1, \cdots 0$): In each task for $G_{\boldsymbol{x}}$, the distance labels of $G_{\boldsymbol{x}}$ is constructed, provided that the distance labels for all children of $G_{\boldsymbol{x}}$ are given. Let $L_h(v) = \{L_{G_{\boldsymbol{x}}}(v) \mid G_{\boldsymbol{x}} \in \mathcal{G}_h \wedge v \in V_{G_{\boldsymbol{x}}}\}$, and $C(\boldsymbol{x}, v)$ be the subset of $C(\boldsymbol{x})$ such that $v \in G_{\boldsymbol{x} \circ i}$ holds for any $i \in C(\boldsymbol{x}, v)$. The detailed construction for $G_{\boldsymbol{x}}$ is stated below:

(1) If $\boldsymbol{x}$ is maximal with respect to $\mathcal{B}$, the nodes in $G_{\boldsymbol{x}}$ solve the all-pair shortest path problem in the centralized way. That is, a leader node (elected on demand) aggregates the whole infor-

[1] While a more sophisticated algorithm achieves a better bound of $\tilde{O}(r+c)$ rounds [4], the bound rc is enough for our application because we propose the decomposition scheme with $c = 2$.

mation of $G_{\boldsymbol{x}}$. After computing the APSP, the leader broadcasts the result to all the nodes. If $\boldsymbol{x}$ is not maximal, the following steps are applied

(2) For any $v \in V_{S_{\boldsymbol{x}}}$ and $i \in C(\boldsymbol{x}, v)$, v broadcasts $L_{G_{\boldsymbol{x} \circ i}}(v)$ to all nodes in $G_{\boldsymbol{x} \circ i}$.

(3) Following the broadcast information, each node $v \in V_{S_{\boldsymbol{x}}}$ computes $\hat{d}_{G_{\boldsymbol{x}}}(v, u) = \min_{i \in C(\boldsymbol{x}, v)} d_{G_{\boldsymbol{x} \circ i}}(v, u)$ for all $u \in V_{S_{\boldsymbol{x}}}$. Note that $d_{G_{\boldsymbol{x} \circ i}}(v, u) = f_{G_{\boldsymbol{x} \circ i}}(L_{G_{\boldsymbol{x} \circ i}}(v), L_{G_{\boldsymbol{x} \circ i}}(u))$ holds and thus v can compute it using the label $L_{G_{\boldsymbol{x} \circ i}}(u)$ broadcast by u. All the computed results are broadcast to all the nodes in $G_{\boldsymbol{x}}$.

(4) Each node $v \in V_{G_{\boldsymbol{x}}}$ computes the all-pair shortest paths among nodes in $V_{S_{\boldsymbol{x}}}$ with respect to $G_{\boldsymbol{x}}$. More precisely, v locally constructs a weighted complete graph $K_{\boldsymbol{x}} = (V_{S_{\boldsymbol{x}}}, V_{S_{\boldsymbol{x}}} \times V_{S_{\boldsymbol{x}}}, w_{K_{\boldsymbol{x}}})$ such that $w_{K_{\boldsymbol{x}}}(v, u) = \hat{d}_{G_{\boldsymbol{x}}}(v, u)$ holds for any $(u, v) \in E_{K_{\boldsymbol{x}}}$, and solves the APSP problem for $K_{\boldsymbol{x}}$.

(5) Finally, for any $i \in C(\boldsymbol{x}, v)$ and $u \in S_{\boldsymbol{x}}$, each node $v \in V_{G_{\boldsymbol{x} \circ i}}$ determines $\min_{w \in V_{S_{\boldsymbol{x}}}} d_{G_{\boldsymbol{x} \circ i}}(v, w) + d_{G_{\boldsymbol{x}}}(w, u)$ as the distance between v and u (with respect to $G_{\boldsymbol{x}}$). Note that $d_{G_{\boldsymbol{x} \circ i}}(v, w)$ can be locally computed by v from the label $L_{G_{\boldsymbol{x} \circ i}}(w)$ broadcast by w in the first step.

The running time of this task is dominated by the steps 1, 2, and 3. The first steps obviously finishes in $O(s^2$ rounds because $|V_{G_{\boldsymbol{x}}}| \leq s$ holds. The second step (over all $G_{\boldsymbol{x}} \in \mathcal{G}_h$) can be seen as a collection of the tasks for all level-$(h+1)$ subgraphs where s source nodes broadcast a message of $O(ls \log n)$ bits in $G_{\boldsymbol{y}}$. Using a pipelined scheduling it can be completed within $O(c(ls^2 + d))$ rounds. The third step consists of the same tasks as the second step, except for the level and message size. The running time is $O(c(ls + d))$ rounds. Thus, the total construction time is $O(Ocl(ls^2 + d))$ rounds.

Theorem 1 Let G be any graph, and assume that a (l, c, d, s)-decomposition of G is given. Then there exists a distributed algorithm constructing the distance labels for all nodes in G. The label size is $O(ls \log n)$ bits,and the construction time is $O(cl(s + d))$ rounds if G is unweighted, and $O(cl(ls^2 + d))$ rounds if G is weighted.

4. Distributed Algorithm for (l, c, d, s)-Decomposition

4.1 Path Separator in Planar Graphs

The key technical tool of our decomposition algorithm is the use of *path separators*, which is known as a variant of the seminal separator theorem by Lipton and Tarjan [11]. The main statement of the existence of the path separator is stated below.

Theorem 2 ([11]) Let G be any n-vertex planar graph with hop diameter D ($n \geq 4$). Then the graph G can be separated into the three disjoint components G^+, G^-, and S such that (1) both G^+ and G^- respectively contain at most $3n/4$ nodes, and (2) no edge connecting G^+ and G^-, and (3) $|S| \leq 2D$.

Since our algorithm utilizes several additional properties of path separators, we concisely explain the detailed structure of this theorem. Consider any embedding of G. We take a pair of nodes (u_1, u_2) such that both u_1 and u_2 are on the boundary of a common face X. The unique path between u_1 and u_2 in T_{BFS} and a "fictional" edge between u_1 and u_2 drawn in X induces a geometric cycle in the embedded plane. Since this cycle does not cross any edge in G, the path between u_1 and u_2 separates G into the two components which are placed at the inside and outside of the cycle. Theorem 2 states that an appropriate choice of (u_1, u_2) achieves a balanced separation with respect to the number of vertices in G^+ and G^-.

4.2 Decomposition via Path Separators

The decomposition scheme follows the recursive application of Theorem 2. As we stated, a main difficulty in such a recursion is that the separated subgraph can have a diameter larger than D. As a first step for avoiding this matter, we consider the decomposition of G into two subgraphs $H^+ = G^+ \cup S$ and $H^- = G^- \cup S$ for a separation $\{G^+, S, G^-\}$, instead of G^+ and G^-. It guarantees the connectivity (and thus the height D) of $T_{\mathrm{BFS}}(H^+)$ and $T_{\mathrm{BFS}}(H^-)$, but edges in S can suffer high congestion. Our algorithm in reality avoids high congestion by carefully assigning each edge in S to G^+ or G^- for achieving both low-congestion and low-dilation. The intuition of the assignment strategy is very simple. Letting $\mathcal{S}$ be a set of path separators, the set of (geometric) cycles associated with each $S \in \mathcal{S}$ separates the embedded plane into several regions, and all the separated subgraphs are placed in a single region. An edge e in a separator is added to the separated subgraphs which are located in the regions e touches. Since any edge in the separator can touch at most two regions, this assignment guarantees the congestion at most two.

For the detailed analysis, We introduce our decomposition scheme in more precise way.

(1) We first constructs an auxiliary graph $G'_{\boldsymbol{x}}$. Letting $T'_{\boldsymbol{x}}$ be the minimal connected subgraph of T_{BFS} containing all nodes in $G_{\boldsymbol{x}}$, we define $G'_{\boldsymbol{x}} = G_{\boldsymbol{x}} \cup T'_{\boldsymbol{x}}$.

(2) We apply Theorem 2 to $G'_{\boldsymbol{x}}$, and find a balanced separator $S'_{\boldsymbol{x}}$ of $2D$ nodes (with respect to $T'_{\boldsymbol{x}}$). Let $S_{\boldsymbol{x}} = S'_{\boldsymbol{x}} \cap G_{\boldsymbol{x}}$. It is obvious that $S_{\boldsymbol{x}}$ separates $G_{\boldsymbol{x}}$ into two subgraphs G^+ and G^-.

(3) Let $I_{\boldsymbol{x}}(\boldsymbol{y}) = S_{\boldsymbol{x}} \cap S_{\boldsymbol{y}}$ for $\boldsymbol{x}, \boldsymbol{y}$ satisfying $\boldsymbol{y} \sqsubset \boldsymbol{x}$. Note that any $I_{\boldsymbol{x}}(\boldsymbol{y})$ is a path in T_{BFS}. We define I^+ (resp. I^-) as the subgraph consisting of all $I_{\boldsymbol{x}}(\boldsymbol{y})$ where a node except for the two endpoints is adjacent to a node in G^+ (resp. G^-).

(4) Each connected component in $(G^+ \cup S_{\boldsymbol{x}}) \setminus I^-$ or $(G^- \cup S_{\boldsymbol{x}}) \setminus I^+$ becomes a chird of $G_{\boldsymbol{x}}$.

The correctness of the algorithm is derived from the two lemmas below:

Lemma 2 Let $(\mathcal{B}, \mathcal{G}, \mathcal{S})$ be the decomposition of G outputted by our algorithm. For any $\boldsymbol{x} \in \mathcal{B}$ and $i \in C(\boldsymbol{x})$, $|R_{\mathrm{BFS}}(\boldsymbol{x} \circ i)| \leq |R_{\mathrm{BFS}}(\boldsymbol{x})| + 2|\boldsymbol{x}|$ holds.

Lemma 3 Let $(\mathcal{B}, \mathcal{G}, \mathcal{S})$ be the decomposition of G outputted by our algorithm. For any $\boldsymbol{x} \in \mathcal{B}$, if an edge $e \in E_{G_{\boldsymbol{x}}}$ is contained in both I^+ and I^-, there is no $\boldsymbol{y} \sqsubset \boldsymbol{x}$ such that $e \in E_{S_{\boldsymbol{y}}}$.

Informally, Lemma 2 and Lemma 3 respectively correspond to the low-dilation and low-congestion properties of (l, c, d, s)-decomposition. Then we obtain the following main theorem.

Theorem 3 The algorithm above outputs the $(3 \log n, 2, 9D \log^2 n, 8D)$-decomposition.

4.3 Distributed Implementation

The decomposition proceeds in the top-down manner. Thus, we can assume that when we consider the decomposition of $G_{\boldsymbol{x}}$ for some $\boldsymbol{x} \in \mathcal{B}$, each node $v \in V_{G_{\boldsymbol{x}}}$ knows if it belongs to $S_{\boldsymbol{y}}$ or not for any $\boldsymbol{y} \sqsubset \boldsymbol{x}$. Since the steps 3 and 4 are easily implemented in the CONGEST model, the main challenge of the distributed implementation is to compute $S'_{\boldsymbol{x}}$. For the computation of the balanced separator $S'_{\boldsymbol{x}}$, we can utilize the distributed algorithm of finding a balanced path separator by Ghaffari and Parter [7], which runs in $\tilde{O}(D)$ rounds. It should be noted that we cannot straightforwardly apply that algorithm for implementing the first and second steps because we have to run it in graph $G'_{\boldsymbol{x}}$, not in $G_{\boldsymbol{x}}$. Our main idea of filling this gap is to simulate the execution of the Ghaffari-Parter algorithm in $G'_{\boldsymbol{x}}$ on the top of $G_{\boldsymbol{x}}$. Let $\hat{T}_{\boldsymbol{x}}$ be the subgraph of $T'_{\boldsymbol{x}}$ consisting of the edges in $E_{G'_{\boldsymbol{x}}} \setminus E_{G_{\boldsymbol{x}}}$. Our algorithm uses the following strategy to simulate one-round execution in $G'_{\boldsymbol{x}}$.

Communication through the edges in $G_{\boldsymbol{x}}$: The message is simply transmitted through the real edge.

Communication through the edges in $\hat{T}_{\boldsymbol{x}}$: Let $\hat{V} = V_{G'_{\boldsymbol{x}}} \setminus V_{G_{\boldsymbol{x}}}$. We first elect a leader $v^L_{\boldsymbol{x}}$ (with the minium ID) from $V_{G_{\boldsymbol{x}}}$, which takes the role of simulating all the nodes in $\hat{V}$. For $v^L_{\boldsymbol{x}}$ to know the topology of $\hat{T}_{\boldsymbol{x}}$, each node $v \in R_{\mathrm{BFS}}(G_{\boldsymbol{x}})$ broadcasts $A_{\mathrm{BFS}}(v)$ to all the nodes in $G_{\boldsymbol{x}}$. It also enables each node $u \in G_{\boldsymbol{x}}$ to detect its incident edges reaching $\hat{T}_{\boldsymbol{x}}$, i.e., (u, u') such that $u' \in \hat{T}_{\boldsymbol{x}}$. We call those edges *unusable edges*. For the local one-round simulation of $\hat{T}_{\boldsymbol{x}}$ by $v^L_{\boldsymbol{x}}$,$v^L_{\boldsymbol{x}}$ must receive the messages transmitted through unusable edges. Similarly, any node to which an unusable edge is incident must receive the message through it. All of those communication is substituted by the broadcast operation in $G_{\boldsymbol{x}}$.

Since the number of leaves in $\hat{T}_{\boldsymbol{x}}$ is bounded by $R_{\mathrm{BFS}}(G_{\boldsymbol{x}}) = O(\log^2 n)$, we can also bound the number of unusable edges by $O(\log^2 n)$. Thus the cost of the one-round simulation above is $d + D\log^2 n = O(D\log^4 n)$ rounds (the broadcast of multiple messages is optimized by the pipelined scheduling). While the direct application of this strategy yields a $\tilde{O}(D^2)$-round algorithm for finding $S_{\boldsymbol{x}}$ with using only the communication links in $G_{\boldsymbol{x}}$, we can improve it to $\tilde{O}(D)$ rounds by a few more optimization (we omit the details for lack of space). Finally, we obtain the following theorem.

Theorem 4 There exists a distributed algorithm in the CONGEST model which computes $(3\log n, 2, 9D\log^2 n, 8D)$-decomposition in $\tilde{O}(D)$ rounds.

5. Applications

Our distance labeling scheme and its distributed implementation can derives efficient solutions for other distance problem in planar graphs. For lack of space, we present only the results here.

Theorem 5 Let $G = (V, E, w)$ be any weighted planar graph, and $X \subset V$ be the set of source nodes. There exists a distributed algorithm constructing the exact weighted shortest path trees for all sources $x \in X$ in $\tilde{O}(D^2 + D|X|)$ rounds.

Theorem 6 Let G be any (weighted) planar graph. For any constant $\epsilon > 0$, t here exists a distributed algorithm computing the $(1+\epsilon)$-approximate diameter. The running time of the algorithm is $\tilde{O}(D^2)$ rounds for weighted graphs, and $\tilde{O}(D)$ rounds for unweighted graphs.

Acknowledgement

This research was supported by Japan Science and Technology Agency(JST) SICORP, and JSPS KAKENHI Grant Number 16H02878.

参考文献

[1] Das Sarma, A., Holzer, S., Kor, L., Korman, A., Nanongkai, D., Pandurangan, G., Peleg, D. and Wattenhofer, R.: Distributed Verification and Hardness of Distributed Approximation, *Proc. of the 43rd Annual ACM Symposium on Theory of Computing*, pp. 363–372 (2011).

[2] Frischknecht, S., Holzer, S. and Wattenhofer, R.: Networks Cannot Compute Their Diameter in Sublinear Time, *Proc. of the Twenty-Third Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pp. 1150–1162 (2012).

[3] Gavoille, C., Peleg, D., P 辿 rennes, S. and Raz, R.: Distance labeling in graphs, *Journal of Algorithms*, Vol. 53, No. 1, pp. 85 – 112 (2004).

[4] Ghaffari, M.: Near-optimal scheduling of distributed algorithms, *Proceedings of the 2015 ACM Symposium on Principles of Distributed Computing*, ACM, pp. 3–12 (2015).

[5] Ghaffari, M. and Haeupler, B.: Distributed Algorithms for Planar Networks I: Planar Embedding, *Proc. of the 48th ACM Symposium on Principles of Distributed Computing (PODC)*, pp. 29–38 (2016).

[6] Ghaffari, M. and Haeupler, B.: Distributed Algorithms for Planar Networks II: Low-congestion Shortcuts, MST, and Min-Cut, *Proc. of the 27th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pp. 202–219 (2016).

[7] Ghaffari, M. and Parter, M.: Near-Optimal Distributed DFS in Planar Graphs, *Proc. of 31st International Symposium on Distributed Computing (DISC)*, Leibniz International Proceedings in Informatics (LIPIcs), Vol. 91, pp. 21:1–21:16 (2017).

[8] Haeupler, B., Izumi, T. and Zuzic, G.: Near-Optimal Low-Congestion Shortcuts on Bounded Parameter Graphs, *Proc. of 30th International Symposium on Distributed Computing (DISC)*, pp. 158–172 (2016).

[9] Haeupler, B., Li, J. and Zuzic, G.: Minor Excluded Network Families Admit Fast Distributed Algorithms, *Proc. of the 2018 ACM Symposium on Principles of Distributed Computing, (PODC)*, pp. 465–474 (2018).

[10] Lenzen, C. and Peleg, D.: Efficient Distributed Source Detection with Limited Bandwidth, *Proc. of the 2013 ACM Symposium on Principles of Distributed Computing (PODC)*, pp. 375–382 (2013).

[11] Lipton, R. and Tarjan, R.: A Separator Theorem for Planar Graphs, *SIAM Journal on Applied Mathematics*, Vol. 36, No. 2, pp. 177–189 (1979).