HAL
open science

Local MST computation with short advice

Pierre Fraigniaud, Amos Korman, Emmanuelle Lebhar

To cite this version:

Pierre Fraigniaud, Amos Korman, Emmanuelle Lebhar. Local MST computation with short advice. ACM Symposium on Parallel Algorithms and Architectures, Jun 2007, San Diego, United States. pp.154. hal-00154849

HAL Id: hal-00154849

https://hal.science/hal-00154849

Submitted on 15 Jun 2007

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Local MST Computation with Short Advice

Pierre Fraigniaud
CNRS and University of Paris Sud
France

Amos Korman*
Technion
Israel

Emmanuelle Lebhar
CNRS and University of Paris 7
France

Abstract

We use the recently introduced *advising scheme* framework for measuring the difficulty of locally distributively computing a Minimum Spanning Tree (MST). An (m, t)-advising scheme for a distributed problem $\mathcal{P}$ is a way, for every possible input I of $\mathcal{P}$, to provide an "advice" (i.e., a bit string) about I to each node so that: (1) the maximum size of the advices is at most m bits, and (2) the problem $\mathcal{P}$ can be solved distributively in at most t rounds using the advices as inputs. In case of MST, the output returned by each node of a weighted graph G is the edge leading to its parent in some rooted MST T of G. Clearly, there is a trivial $(\lceil \log n \rceil, 0)$-advising scheme for MST (each node is given the local port number of the edge leading to the root of some MST T), and it is known that any $(0, t)$-advising scheme satisfies $t \geq \tilde{\Omega}(\sqrt{n})$. Our main result is the construction of an $(O(1), O(\log n))$-advising scheme for MST. That is, by only giving a constant number of bits of advice to each node, one can decrease exponentially the distributed computation time of MST in arbitrary graph, compared to algorithms dealing with the problem in absence of any a priori information. We also consider the average size of the advices. On the one hand, we show that any $(m, 0)$-advising scheme for MST gives advices of average size $\Omega(\log n)$. On the other hand we design an $(m, 1)$-advising scheme for MST with advices of constant average size, that is one round is enough to decrease the average size of the advices from $\log n$ to constant.

Keywords: Minimum spanning tree, distributed algorithm, local computation.

Submission SPAA 2007
Regular paper

*This work was partially done while Amos Korman was visiting LRI at University Paris Sud, supported by the COST Action 295 "DYNAMO".

1 Introduction

In their seminal paper, Naor and Stockmeyer [16] aimed at determining what can be computed locally, that is at determining what problem in distributed networks can be solved with each node exchanging information with only nodes at bounded distance from it. In particular, they proved that randomization does not help for locally checkable problem (LCL)[1]: if there is a local randomized algorithm in $O(t)$ rounds and bounded error probability, then there is a deterministic one in $O(t)$ rounds as well. Despite the limitations induced by the locality constraint, [16] proved that non-trivial LCL problems such as weak-coloring (a basis for a solution to a certain resource allocation problem) can be solved in constant time. This is in contrast with the standard coloring problem for which Linial [14] showed that, for any fixed $c \geq 1$, one cannot c-color properly an n-node ring in less than $\Omega(\log^\star n)$ rounds, i.e., nodes must consider their $\Omega(\log^\star n)$-neighborhood. More recently, Kuhn et al. [13] exhibited other problems that cannot be computed within a fixed neighborhood. For instance, minimum vertex cover, and minimum dominating set can basically not be well approximated locally. In time $O(k)$, the approximation factors are at least $\Omega(n^{c/k^2}/k)$, with $c > 1/4$ and $\Omega(\Delta^{1/k}/k)$, respectively, where Δ is the maximum degree of the graph.

Another framework was recently introduced in [9] for investigating tradeoffs between computation time and local knowledge: distributed computing *with advice*. In this framework, a priori knowledge about the instances of the problem is given to the nodes by an oracle. The oracle $\mathcal{O}$ looks at the entire network G and assigns to every node $v \in V(G)$ some information, coded as a string of bits $\mathcal{O}(G, v)$. The amount of knowledge given to a network G is measured either as the sum (or average) of the lengths of all the strings it assigns to nodes, or as the maximum length of all the strings. Solving a network problem $\mathcal{P}$ using oracle $\mathcal{O}$ consists in designing an algorithm that is *unaware* of the network G but solves the problem $\mathcal{P}$ for it, as long as every node v of the network G is provided with the string of bits $\mathcal{O}(G, v)$. In particular, [9] shows that broadcasting in n-node graphs can be performed with $O(n)$ message complexity by simply giving a constant average amount of advice to each node, whereas, to be performed with $O(n)$ message complexity, the wake-up task requires an average of $\Omega(\log n)$ bits of advice per node. This demonstrates that the framework of advising scheme enables to measure the amount of knowledge required to solve a specific task, and to compare tasks that look very similar at a first glance (in absence of advice, the broadcast and wake-up problems require both $\Theta(E)$ messages in E-edge graphs).

An (m, t)-advising scheme for a distributed problem $\mathcal{P}$ is a pair $(\mathcal{O}, \mathcal{A})$, where $\mathcal{O}$ is an oracle, and $\mathcal{A}$ is an algorithm using the advices of $\mathcal{O}$ for solving $\mathcal{P}$ such that the maximum size of the advices is at most m bits, and the problem $\mathcal{P}$ can be solved distributively in at most t rounds by $\mathcal{A}$ using the advices as inputs. The objective is to establish tradeoffs between the amount of knowledge given to the nodes, and the time required to solve the problem. In this paper, we investigate the local computation of a minimum spanning tree (MST) in an edge-weighted network, in the framework of distributed computing with advice.

Our model is the same as the standard one in [13, 14], plus a predefined port labeling. We consider networks modeled as n-node edge-weighted connected simple graphs with no self-loops. Nodes of a network $G = (V, E)$ have (non-necessarily distinct) IDs. The $\deg(u)$ edges incident to node $u \in V(G)$ are locally labeled by $\deg(u)$ distinct labels, called *port numbers*. The port number

[1] LCL problems are problems on graphs of bounded degree whose solutions can be checked by each node via investigations within a fixed radius around it.

at u of the incident edge e is denoted by $\text{port}_u(e)$. Each node $u \in V(G)$ knows its ID, and the weight $\mathbf{w}(e)$ of each of its $\deg(u)$ incident edges e, identified by its port number. As in most previous works aiming at investigating local computation, we assume synchronous distributed computations proceeding in successive rounds in which the local computations times are negligible in front of the communication times, i.e., the complexity of a problem is computed in terms of number of rounds. At each round, every node u acts as follows: (1) it sends through each of its incident edge $e = \{u, v\}$ a message M_e, (2) it receives the message M'_e sent by the other extremity v of edge e, and (3) it performs local computations based on all data collected since the beginning of the execution of the algorithm. There is no limit on the volume of information that can be transmitted along an edge at each round, i.e., we assume the $\mathcal{LOCAL}$ model defined in [17]. However, all our algorithms send at most $O(\log n)$ bits through each edge at each round, and therefore all our upper bounds apply to the $\mathcal{CONGEST}$ model [17] as well.

In the model above, the MST problem requires to compute an upward tree representation of a minimum spanning tree T of the network, i.e. each node $u \in V(G)$ of the network $G = (V, E)$ must output the port number of the edge of T leading to its parent, excepted the root r itself that should simply output that it is the root. In addition to its own ID, and the weights of its incident edges, each node $u \in V(G)$ is given a bit-string $\mathcal{O}(G, u)$ corresponding to the advice given by the oracle $\mathcal{O}$ to node u about the network.

In this model, there is a straightforward $(\lceil \log n \rceil, 0)$-advising scheme $(\mathcal{O}, \mathcal{A})$ for MST in which each node is given by $\mathcal{O}$ the rank of the edge leading to the root of some MST T. Precisely, for an edge e incident to $u \in V(G)$, we define $\text{index}_u(e) = (x_u(e), y_u(e))$ where $x_u(e)$ is the rank of the weight $\mathbf{w}(e)$ of e among all the weights of the edges incident to u, and $y_u(e)$ is the rank of the port number of edge e among all the edges of weight $\mathbf{w}(e)$ incident to u. The straightforward $(\lceil \log n \rceil, 0)$-advising scheme $(\mathcal{O}, \mathcal{A})$ selects any MST T, and selects one node r as the root of T. $\mathcal{O}$ gives to every node $u \in V(G)$, $u \neq r$, the bit-string corresponding to the binary representation of the rank $r_u(e) \in \{1, \ldots, \deg(u)\}$ of $\text{index}_u(e)$, among all the indexes of the edges incident to u, where e is the edge incident to u that leads to the parent of u in T. Then $\mathcal{A}$ computes at each node u the port number or the edge having rank $r_u(e)$.

On the other hand, it is proved that, in the $\mathcal{CONGEST}$ model, $(0, t)$-advising scheme satisfies $t \geq \tilde{\Omega}(\sqrt{n})$ [18]. In the $\mathcal{LOCAL}$ model, of course there is a $(0, D+1)$-advising scheme for all graphs of diameter D, and having distinct node IDs[2].

1.1 Our results

Our main result is the construction of an $(O(1), O(\log n))$-advising scheme for MST in the $\mathcal{CONGEST}$ model. That is, by only giving a constant number of bits of advice to each node, one can decrease exponentially the distributed computation time of MST in arbitrary graph, compared to algorithms dealing with the problem in absence of any a priori information.

We also consider the average size of the advices. On one hand, we show that, for any $m \geq 0$, any $(m, 0)$-advising scheme for MST gives advices of average size $\Omega(\log n)$. On the other hand we design an $(O(\log^2 n), 1)$-advising scheme for MST with advices of constant average size, that is one

[2]The node IDs are used to break symmetry; In the anonymous ring with all edge-weights the same, there is no way to break symmetry.

round is enough to decrease the average size of the advices to constant.

1.2 Related works

One can rephrase many recent results of the literature in the framework of advising schemes. For instance, a $(\lceil \log n \rceil, 0)$-advising scheme, with average advice length $O(\log \log n)$ bits, is described in [3] for computing a spanning tree. It is also easy to extract a $(2, 1)$-advising scheme for spanning tree (with average advice length $\frac{4}{3}$) from the proof of the main result in [4]. [10] considers the competitive ratio of the exploration time of a robot unaware of the topology compared to a robot knowing the map of the graph. A (k, α)-advising scheme for exploration uses a robot that is given k bits of advice, and performs exploration in a number of moves at most α times the optimal number of moves. [10] describes a $(O(\log \log D), \alpha)$-advising scheme for exploration of diameter-D trees, with $\alpha < 2$, and prove that, for any $\epsilon > 0$, any $(k, 2 - \epsilon)$-advising scheme for exploration satisfies $k \geq \Omega(\log \log D)$.

Regarding the MST problem, the classical algorithm by Gallager, Humblet, and Spira [12] proceeds in $O(n \log n)$ rounds with constant size messages. [5] improves this result with an algorithm proceeding in $O(n \log \log n)$ rounds. Yet another improvement was achieved in [11], with an algorithm in $O(n \log^\star n)$ rounds. [7] gives an algorithm in $O(\mu(G) \log^3 n + \log n \cdot \sqrt{n \log^\star n / B})$ rounds in the $\mathcal{CONGEST}(B)$ model, where $\mu(G)$ is the MST-radius of G. [8] considers approximation of MST. It is shown that approximating MST within a factor k takes at least $\Omega(\sqrt{\frac{n}{kB}})$ rounds, and at most $O(D + \frac{W_{max}}{k-1} \cdot \log^\star n)$ rounds, where D is the diameter and W_{max} is the maximum weight. [18] presents several lower bounds. In particular, $\Omega(\frac{\sqrt{n}}{B})$ rounds are required for computing MST in graphs of unweighted diameter $O(n^\delta)$, $0 < \delta < 1/2$, and at least $\Omega(\frac{\sqrt{n}}{B \log n})$ rounds are required for graphs of unweighted diameter $O(\log n)$. These results were improved in [15] where it is shown that at least $\Omega(n^{1/3}/B)$ rounds are required for graphs of diameter at most 4, and at least $\Omega(n^{1/4}/B)$ rounds for graphs of diameter at most 3.

2 Advising schemes with small average size

In this section, we first prove that the aforementioned straightforward $(\lceil \log n \rceil, 0)$-advising scheme $(\mathcal{O}, \mathcal{A})$ for MST is actually optimal even if one considers not the maximum size of the advices $\max_{v \in V(G)} |\mathcal{O}(G, v)|$, but instead the average size of the advices $\frac{1}{n} \sum_{v \in V(G)} |\mathcal{O}(G, v)|$. On the other hand, we present an $(O(\log^2 n), 1)$-advising scheme for MST with advices of constant average size.

2.1 Lower Bound on the Average Size

Theorem 1 *For any n-node graph G, and any $m \geq 0$, any $(m, 0)$-advising scheme for computing an MST of G gives advices of average size $\Omega(\log n)$. This result holds even if all edge-weights are pairwise distinct.*

Proof. Let G_n be the network with $2n$ nodes $u_1, \ldots, u_n, v_1, \ldots, v_n$, $n \geq 1$, defined as follows. Let K_n be the complete graph of n nodes. Let $(x_1, \ldots, x_n)$ be an hamiltonian path of K_n. This path is

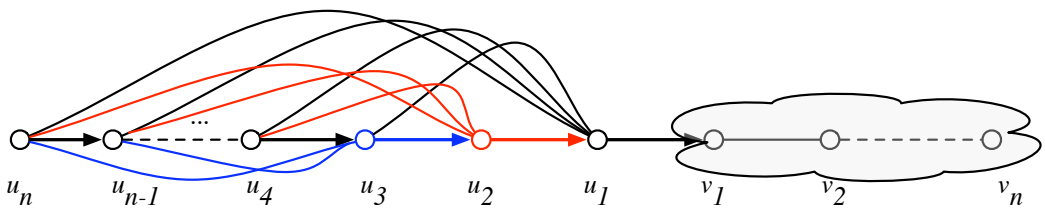

Figure 1: The graph G_n in the proof of Theorem 1.

called the *spine* of K_n. Let A_n and B_n be two copies of K_n, with respective spines $u_1, \ldots, u_n$ and $v_1, \ldots, v_n$. G_n consists of the graphs A_n and B_n connected by the edge $\{u_1, v_1\}$, see Figure 2.1.

The edge-weights in G_n are assigned as follows. We set $\mathbf{w}(\{u_1, v_1\}) = 0$. Let ω be any positive integer. For every i, $1 \leq i \leq n$, let $a_i = \omega^2 - (i+1)\omega + 1$ and $b_i = \omega^2 - i\omega$. For all $1 < i \leq n$, $\mathbf{w}(\{u_i, u_{i-1}\})$ and $\mathbf{w}(\{v_i, v_{i-1}\})$ can be any integer in $[a_i, b_i]$. Similarly, for all $1 \leq i \leq n-2$, and for all $j \in \{i+2, \ldots n\}$, the weights $\mathbf{w}(\{u_i, u_j\})$ and $\mathbf{w}(\{v_i, v_j\})$ can be any integer in $[a_i, b_i]$. Note that these weights are assigned arbitrarily from the ω values in the range $a_i, \ldots, b_i$. In particular, a node can have incident edges with equal weights. Conversely, if $\omega \geq n$, then all the incident edges can be assigned different weights.

Clearly, the edge $\{u_1, v_1\}$ has to belong to any spanning tree of G_n because it is a cut-edge. In fact, any MST for G_n contains all the edges $\{u_i, u_{i-1}\}$, and $\{v_i, v_{i-1}\}$ for $i \in \{2, \ldots n\}$. Indeed, $a_k > b_i$ for $k \leq i-1$, and therefore $\mathbf{w}(\{u_i, u_{i-1}\}) < \mathbf{w}(\{u_j, u_k\})$, and $\mathbf{w}(\{v_i, v_{i-1}\}) < \mathbf{w}(\{v_j, v_k\})$ for any $j \geq i$ and any $k \leq i-1$ such that $(j,k) \neq (i, i-1)$. Therefore, a simple induction on $i = 2, \ldots, n$ enables to prove that all the edges $\{u_i, u_{i-1}\}$, and $\{v_i, v_{i-1}\}$ belong to any MST for G_n. In other words, G_n has a unique MST that is the path $P = (u_n, \ldots, u_1, v_1, \ldots, v_n)$.

Consider an $(m, 0)$-advising scheme $(\mathcal{O}, \mathcal{A})$, and assume, w.l.o.g., that the root of P chosen by the scheme is in $\{v_1, \ldots, v_n\}$. Hence $\mathcal{A}$ has to return $\{u_i, u_{i-1}\}$ at node u_i. Suppose that the average size of the advices is less than $\frac{1}{2n}\sum_{i=2}^{n-1} \log(n-i)$ bits. Therefore, there exists $i \in \{2, \ldots, n-1\}$ such that $|\mathcal{O}(G_n, u_i)| < \log(n-i)$. Let $w_0, w_1, \ldots, w_{n-i-1}$ be $n-i$ integers in $[a_i, b_i]$, and let $e_0, e_1, \ldots, e_{n-i-1}$ be the $n-i$ edges incident to u_i with weights in $[a_i, b_i]$. We consider $n-i$ possible settings $S_1, \ldots, S_{n-i}$ of the weights for the e_js. For $k = 1, \ldots, n-i$, let $\pi_k \in \Sigma_{n-i}$ be such that $\pi_k(j) = j + k \pmod{n-i}$. In S_k, we have $\mathbf{w}(e_j) = w_{\pi_k(j)}$ for $j = 0, \ldots, n-i+1$. Since u_i is given less than $\log(n-i)$ bits of advice by $\mathcal{O}$, there are two different configurations S_k and $S_{k'}$ for which $\mathcal{A}$ will return the same port number, and thus $\mathcal{A}$ will not return $\{u_i, u_{i-1}\}$ at u_i for at least one of these two configurations. Therefore, the average size of the advices is at least $\frac{1}{2n}\sum_{i=2}^{n-1} \log(n-i)$ bits, i.e., at least $\Omega(\log |V(G_n)|)$ bits. $\square$

2.2 Upper Bound on the Average Size

In this section, we present an advising scheme for MST, performing in one round with advices of constant average size. Hence, compared to Theorem 1, this demonstrates that only one round of computation enables to drastically decrease the average advice size. Our scheme is based on Boruvka's algorithm, which provides a decomposition of the MST construction into exponentially

growing sets. It is basically a variant of Kruskal's algorithm (cf. [6]). Precisely, our MST construction proceeds in at most $\lceil \log n \rceil$ phases, as follows.

MST Construction. Before phase 1, each node is a *fragment* reduced to a single node. At each phase, fragments are merged to produce larger fragments. Eventually, there remains only one fragment. To perform phase $i \geq 1$, one considers only fragment F satisfying $|F| < 2^i$. These fragments are said *active* at phase i, and the others are said *passive*. Every fragment F that is active at phase i selects an incident edge e leading out of F, and of minimum weight. Ties are broken using the port numbers. If ties remain, then they are broken arbitrarilly. The edge e is called the *selected* edge of F at phase i. The node of F that is incident to e is called the *choosing* node of F at phase i. All fragments connected together by selected edges at phase i become one new fragment for phase $i+1$.

We state the following straightforward lemma for further references.

Lemma 1 *After phase i, the size of every fragments is at least 2^i. Thus a fragment F that is active at phase i satisfies $2^{i-1} \leq |F| < 2^i$. Hence there are at most $n/2^{i-1}$ active fragments at phase i.*

Since the graph has no self-loop and no double edge, a node u of a fragment F cannot have more than $|F|-1$ incident edges whose other extremities are in F. Moreover, since the edges are consumed in order of increasing weights and port numbers, we get the following result:

Lemma 2 *If e has been selected by fragment F and $u \in F$ is the choosing node of e, then* $\text{index}_u(e) = (x_u(e), y_u(e))$ *satisfies $x_u(e) + y_u(e) \leq |F|$.*

In the following, we choose an arbitrary node r of the MST T returned by Boruvska's algorithm to be the root of T. The choice for the root induces an orientation on the edges. We say that a T-edge incident to a node v is *up at v* if it is the first edge on the shortest path from v to r in T, and *down at v* otherwise. Figure 2.2 illustrates one phase of Boruvska's algorithm. In this figure, there are three fragments F_1, F_2, F_3. Dashed lines represents edges that are not selected, and the black nodes are the choosing nodes. Labels up and down give the orientation of the selected edges from the point of view of their choosing node. Note that an edge can be up for one of its extremities, and down for the other.

While from Theorem 1 it is not possible to improve the naive advising scheme when no step of computation is allowed, interestingly enough the next theorem shows that a *single* round of computation enables to lower the *average* advice length down to a constant.

Theorem 2 *For any n-node graph G, there exists an $(O(\log^2 n), 1)$-advising scheme for computing an MST of G, that gives advices of average size $c = \sum_{i=1}^{\lceil \log n \rceil} \frac{i+1}{2^{i-2}} = O(1)$.*

Proof. The scheme picks an arbitrary node of the MST returned by Boruvska's algorithm to be the root. This induces a direction (*up* or *down*) on every selected edge from the point of view of the choosing node. The construction of the advices follows from the phases of Boruvska's algorithm. For each phase $1 \leq i \leq \lceil \log n \rceil$, each choosing node u at phase i stores two items of advice:

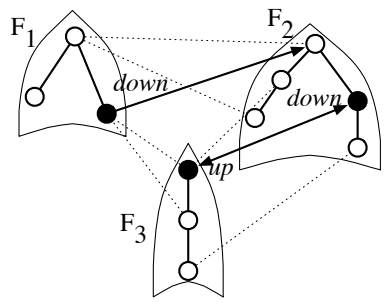

Figure 2: One phase of Boruvska's algorithm

- $\text{index}_u(e)$ where e is the selected edge at phase i corresponding to u; and
- a boolean b stating whether e is *up*.

The size of this advice is at most $i+1$ bits. Indeed, active fragments at phase i have size less than 2^i, and thus from Lemma 2, $\text{index}_u(e) < 2^i$. Since there is exactly one choosing node per fragment, and at most $n/2^{i-1}$ active fragments at phase i (cf. Lemma 1), this consumes a total of at most $(i+1) \cdot n/2^{i-1}$ bits of advice per phase. The advices possibly received at different phases are concatenated. Every node that was choosing at some phase stores a bit-map indicating the separation between the advices corresponding to different phases. This doubles the size of the advices. Thus, the scheme uses at most $2\sum_{i=1}^{\lceil \log n \rceil}(i+1) \cdot n/2^{i-1} = O(n)$ bits in total, that is a constant number of bits per node on average.

Given these advices, the rooted MST can be computed as follows: every choosing node u sends a message on each incident edge e for which the corresponding index belongs to its advice. If the bit b associated to the index of e is 1, then u directly learns the index of the port number of its parent, and if $b = 0$ then u informs the other extremity of e that it is its parent. This algorithm thus runs in at most one round.

Since a node can be a choosing node in all $\lceil \log n \rceil$ phases, the maximum length of an advice can be up to $\sum_{i=1}^{\lceil \log n \rceil}(i+1) = O(\log^2 n)$ bits. □

3 An $(O(1), O(\log n))$-Advising Scheme for MST

This section presents our main result, that is, by assigning just a constant amount of advices to each node, it is possible to compute an MST in any n-node graph in at most $O(\log n)$ rounds. Precisely, we prove the following.

Theorem 3 *There exists a (m,t)-advising scheme to compute an MST in any n-node graph, with $m = 12$, and $t \leq 9\lceil \log n \rceil$.*

Proof. Let $G = (V, E)$ be an n-node graph, and let T be the MST of G returned by the variant of Boruvska's algorithm described in Section 2.2, simply called Boruvska's algorithm in the following.

The scheme picks an arbitrary node r to be the root of T. The setting of the advices is done by induction, in $\lceil \log\log n \rceil + 1$ phases of Boruvska's algorithm. For the purpose of the scheme, we define the *level* of a fragment, for each phase, as a value in $\{0,1\}$, as follows. Before any given phase $i \geq 1$, each fragment F induces a subtree T_F of T. These subtrees are connected by edges of T between fragments. Contracting each subtree T_F into a singe node x_F, and connecting two nodes x_F and $x_{F'}$ by an edge if and only if there is an edge of T between the two fragments F and F', results in a "tree of fragments" T_i. The node x_F corresponding to the fragment F which contains the root r of T becomes the root of T_i, and F is given level 0. More generally, a fragment F at phase i is given level 0 if x_F occupies an even level in the rooted tree T_i, and level 1 if x_F occupies an odd level.

The construction of the advices at each node follows the phases of Boruvska's algorithm up to phase $\lceil \log\log n \rceil + 1$. For any given phase i, $1 \leq i \leq \lceil \log\log n \rceil$, our scheme encodes three items of advice inside each active fragment F of phase i: the choosing node of F, the orientation (up or down) of the edge selected by F, and the level of F at phase i. At phase $\lceil \log\log n \rceil + 1$, the scheme encodes inside each fragment F the index of the edge of T leading from x_F to its parent in T_i. We will show how to distribute these bits of advice inside the fragments so that no node has to store more than 12 bits, and so that these bits can be easily extracted by the choosing node of each fragment at each phase.

For every phase i of Boruvska's algorithm, $1 \leq i \leq \lceil \log\log n \rceil$, and for every active fragment F at phase i, we construct a bit-string $A(F)$ as follows. Let $v_1, \ldots v_\ell$ be the BFS ordering of the ℓ vertices of F, $2^{i-1} \leq \ell < 2^i$, in the subtree T_F, starting from the root r_F of T_F, i.e. the node of F that is the closest in T to the root r of the tree T. Precisely, the BFS is guided by the indexes of the edges in T_F according to the rule: lower index first. Let v_j be the choosing node of F, $1 \leq j \leq \ell$. Let $\mathrm{bin}(j)$ be the binary encoding of j on i bits, let b_{up} be the boolean indicating whether the edge selected by v_j in F is up or down, and let b_{level} be a boolean indicating whether the level of F is odd or even. We define $A(F) = b_{up}|b_{level}|\mathrm{bin}(j)$ where "$|$" denote the concatenation of strings. Hence $A(F) \in \{0,1\}^{i+2}$. The advice given to the fragment F at phase i is $A(F)$.

More precisely, the advising scheme assigns bits of $A(F)$ to nodes of F as follows. Each of the ℓ nodes $v_1, \ldots v_\ell$ of F is given at most c bits of $A(F)$, where $c = 11$. The oracle assigns a c-bit string advice(u) to each node u of G as follows. In order to bound the number of bits per node, we use a variable used(u,i) that indicates the number of bits that node u had to store in its advice(u) until phase i (included). We set used$(u,0) = 0$ for every $u \in V(G)$, and assign the bits of $A(F)$ as follows:

While there remains $k > 0$ bits of $A(F)$ unassigned **do**
  Let j be the minimum index in $\{1, \ldots, \ell\}$ such that $c - \mathrm{used}(v_j, i-1) > 0$;
  Concatenate the $\min\{k,\ c - \mathrm{used}(v_j, i-1)\}$ next unassigned bits of $A(F)$ to advice(v_j);
  Update the value of used(v_j, i) according to the number of bits added to advice(v_j).
End

The following claim guarantees that each string $A(F)$ can actually be assigned to F during this process. In other words, we prove that it is possible to pack all strings $A(F)$ for all phases i, $1 \leq i \leq \lceil \log\log n \rceil$, using at most c bits at each node. Recall that $A(F) \in \{0,1\}^{i+2}$.

Claim 1 *For any i, $1 \leq i \leq \lceil \log\log n \rceil$, for any active fragment F at phase i, we have $c \cdot |F| -$*

$\sum_{v \in F} \text{used}(v, i-1) \geq i+2$.

To establish the claim, we note first that the result is clear for $i = 1$, because $|F| = 1$, $\text{used}(v, 0) = 0$, and indeed, $c \geq 3$. Consider an active fragment F at phase $i > 1$. F is composed of at most $|F|/2^{j-1}$ fragments of phase j for any $1 \leq j \leq i-1$. By construction, an active fragment F' of phase j satisfies $|F'| < 2^j$, and thus consumes at most $j+2$ bits. Therefore, the total number of bits that have been consumed in F until phase $i-1$ is the sum of the $j+2$ bits consumed by the active fragments F' at phase j, for all $F' \subseteq F$ and $j < i$. Thus:

$$\sum_{v \in F} \text{used}(v, i-1) \leq \sum_{j=1}^{i-1} \frac{|F|}{2^{j-1}} \cdot (j+2) \leq |F| \sum_{j=1}^{\infty} \frac{j+2}{2^{j-1}} = 8|F|.$$

Therefore,

$$c \cdot |F| - \sum_{v \in F} \text{used}(v, i-1) \geq 3|F| \geq 3 \cdot 2^{i-1} \geq i+2$$

because $i \geq 1$ and $c \geq 3 + \sum_{j=1}^{\infty} (j+2)/2^{j-1}$. This complete the proof of Claim 1.

At phase $\lceil \log \log n \rceil + 1$ of the advices construction, we consider all fragments, not only the active ones. The bit string $A(F)$ of fragment F consists of the binary representation of the index of the edge of T leading from the node r_F to its parent in T, where we recall that r_F is the node of F that is the closest in T to the root r of the tree T. Since this index is at most $n-1$, $A(F)$ is a string of at most $\lceil \log n \rceil$ bits. Since F is a fragment of phase $\lceil \log \log n \rceil + 1$, we get

$$|F| \geq 2^{\lceil \log \log n \rceil} \geq \lceil \log n \rceil$$

and therefore $A(F)$ can be distributed in F, one bit per node. Precisely, we can distribute the bits of $A(F)$ using again the BFS ordering $v_1, \ldots v_\ell$ of the vertices in F: v_i stores the ith bit of $A(F)$.

To sum up, the oracle $\mathcal{O}$ assigns the bit-string $\mathcal{O}(G, v) = \text{advice}(v)|b(v)$ to node v where $\text{advice}(v)$ are advices given in phases 1 to $\lceil \log \log n \rceil$, and $b(v)$ is the 1-bit advice given at phase $\lceil \log \log n \rceil + 1$. Since Claim 1 guarantees that c bits per node suffices to construct the advices up to phase $\lceil \log \log n \rceil$ and one bit per node is enough to store the advices of phase $\lceil \log \log n \rceil + 1$, we conclude that all advices are of length at most $c+1$ at the end of the setting of the advices. Since $m = c+1$, our scheme does use advices of maximum size m.

The decoding of the advices follows Boruvska's algorithm up to phase $\lceil \log \log n \rceil + 1$ for reconstructing the MST T returned by the algorithm. We will insure that, during all phases $i = 1, \ldots, \lceil \log \log n \rceil$ of the decoding process, every node u is permanently aware of the number of bits of advice that has already been consumed. This can be done using a pointer $\text{cons}(u, i)$ that specifies how many bits of $\text{advice}(u)$ have been decoded until phase i (included).

Initially, $\text{cons}(u, 0) = 0$ for all nodes u. At phase i, every node u sends to the root r_F of its fragment F all its unconsumed bits, i.e. those from $\text{cons}(u, i-1)+1$ to c. In return, u will receive from r_F the largest BFS index k such that v_k stores some bits of $A(F)$. Moreover, r_F sends the number of bits of $\text{advice}(v_k)$ it has consumed. These two informations are stored in a pair of integers denoted by $\text{end}(F)$. Every node u of F receiving $\text{end}(F)$ is able to update its variable $\text{cons}(u, i)$.

The decoding process $\mathcal{A}$ is precisely described below.

Decoding process $\mathcal{A}$
Input: The distributed network G, with each node v provided with $\mathcal{O}(G,v) = \text{advice}(v)|b(v)$;
Output: Each node v computes the port number of the edge leading to its parent in the MST T.
Begin
  For $i = 1$ to $\lceil \log\log n \rceil$ **do**
    /* The objective is to determine each node's parent inside active fragments after phase i */
    For all active fragments F at phase i **do** (in parallel)
(1) Every node $u \in F$, $u \neq r_F$, sends all the bits of advice(u) between position $\text{cons}(u, i-1)+1$ and position c, to its parent in T_F;
(2) Messages are forwarded up to the root r_F of T_F in BFS order;
    /* At that point, nodes of F know their children in T_F */
(3) r_F collects and concatenates the several parts of the $i+2$ bits of $A(F)$, and r_F broadcast $(\text{end}(F), A(F))$, where $A(F) = b_{up}|b_{level}|\text{bin}(j)$ in T_F;
(4) Each node u updates $\text{cons}(u,i)$ according to $\text{end}(F)$;
(5) The choosing node v_j of F selects its incident edge e of minimum weight among all its incident edges having its other extremity at level $\overline{b}_{level}$.
    If b_{up} is true
(6) **then** v_j is now aware of the port number of e, leading to its parent;
(7) **else** v_j sends a message across e to inform the other extremity of e that v_j is its parent.
    Endfor
  Endfor
  $i := \lceil \log\log n \rceil + 1$;
  Do (in parallel)
(8) The root r_F of each fragment F at phase i collects the bits of $A(F)$ in BFS order in T_F;
(9) $A(F) = \text{index}_{r_F}(e)$ allows r_F to compute the port number of edge e leading to its parent in T.
End

Before proving that Algorithm $\mathcal{A}$ is correct, we first prove the following.

Claim 2 *The edge e selected by the node v_j of F in Algorithm $\mathcal{A}$ is the edge selected by Boruvska's algorithm for the fragment F.*

To establish the claim, we first note that, from the encoding of the advices, node v_j is the choosing node of fragment F in Boruvska's algorithm. Its incident edge of minimum weight going out of F is the edge selected by Boruvska's algorithm. By definition of the levels, the selected edge has its extremity in a fragment at level $\overline{b}_{level}$. Therefore, selecting the edge incident to v_j of minimum weight among all incident edge leading to a fragment at level $\overline{b}_{level}$ enables to discard edges going inside F while preventing from missing the selected edge.

The following claim proves the correctness of the decoding algorithm $\mathcal{A}$.

Claim 3 *After phase i, $1 \leq i \leq \lceil \log\log n \rceil$, for every fragment F resulting from phase i, each node $u \neq r_F$ in F knows the port number of its incident edge leading to its parent in T_F, and the root r_F of T_F knows that it is the root.*

The proof is by induction on i. At phase 1 of the decoding process, every fragment F is active, and reduced to the unique node r_F, and every node knows that it is the root of its fragment.

Assume by induction that the claim holds for phases up to phase $i-1 \geq 1$, and let us prove that it holds for phase i. At phase i, fragments of phase i are merged into larger fragments. Let F be an active fragment at phase i. By induction, every node $u \neq r_F$ of F knows its parent in T_F. Therefore, step (1) of $\mathcal{A}$ can be executed. From Claim 2, the edge e selected by v_j at step (5) is the edge selected by Boruvska's algorithm for fragment F. If b_{up} is true, i.e., e is up, then v_j is actually the root r_F of F, and e is the edge leading to its parent in the new fragment (cf. Step (6)). If b_{up} is false, i.e., e is down, then, by Step (6), v_j informs the other extremity of e, say w_j, that e is the edge leading to its parent in the new fragment. Node w_j is actually the root $r_{F'}$ of a fragment F' one level below F in T_i. Receiving a message from v_j, node w_j learns its parent in the new fragment. A root of a fragment that did not acquire a parent at phase i remains root, possibly of a larger fragment resulting from merges at phase i. This proves the claim.

From the previous claim, algorithm $\mathcal{A}$ computes the MST T. Indeed, by this claim we get that after phase $\lceil \log \log n \rceil$, every node but the root of its fragment knows its parent. Steps (8) and (9) of Algorithm $\mathcal{A}$ enables each of these roots to learn its parent.

We complete the proof by showing that algorithm $\mathcal{A}$ runs in at most $9\lceil \log n \rceil$ rounds. For every phase i, $1 \leq i \leq \lceil \log \log n \rceil$, the decoding process runs in a time corresponding to one convergecast and one broadcast in every tree T_F, where F is an active fragment. Therefore, phase i runs in at most twice the maximum size of an active fragment, hence, from Claim 1, in at most 2^{i+1} rounds. The last phase of $\mathcal{A}$, i.e., the one corresponding to phase $\lceil \log \log n \rceil + 1$ of Boruvska's algorithm, runs in the time required to collect $\lceil \log n \rceil$ bits from $\lceil \log n \rceil$ different nodes in every tree T_F for each fragment F (not only the active ones). This takes at most $\lceil \log n \rceil$ rounds. Thus in total the decoding process runs in at most

$$\lceil \log n \rceil + \sum_{i=1}^{\lceil \log \log n \rceil} 2^{i+1} \leq \lceil \log n \rceil + 4(2^{\lceil \log \log n \rceil} - 1) \leq 9\lceil \log n \rceil.$$

This completes the proof of Theorem 3. □

4 Conclusion

Our result suggests a tradeoff between the computation time and the amount of advice for constructing a MST distributively. In particular we show that computing a MST in time 0 requires an average of $\Omega(\log n)$ bits of advice whereas computing a MST in time 1 can be achieved by an algorithm using an average of $O(1)$ bits of advice. As far as the maximum size of the advices is concerned, we have designed an $(O(1), O(\log n))$-advising scheme for MST. It would be interesting to establish whether the tradeoff between computation and knowledge does exist or not for the maximum size of the advices. In particular, it would be of interest to prove or disprove the existence of an $(O(1), O(1))$-advising scheme for MST.

References

[1] B. Awerbuch. Optimal distributed algorithms for minimum weight spanning tre, counting, leader election and related problems. In *Proc. Annual ACM Symp. on Theory of Computing (STOC)*, pp. 230-240, 1987.

[2] Y. Afek, S. Kuten, and M. Yung. Local detection for global self stabilization. In *Proc. 4th Workshop on Distributed Algorithms*, pp. 15-28, 1991.

[3] R. Cohen, P. Fraigniaud, D. Ilcinkas, A. Korman, and D. Peleg. Labeling schemes for tree representation. In *Proc. 7th International Workshop on Distributed Computing (IWDC)*, pp. 13-24, 2005.

[4] R. Cohen, P. Fraigniaud, D. Ilcinkas, A. Korman, and D. Peleg. Label-Guided Graph Exploration by a Finite Automaton. In 32nd Int. Colloquium on Automata, Languages and Programming (ICALP), LNCS 3580, Springer, pp. 335-346, 2005.

[5] F. Chin and H. F. Ting. An almost linear time and $O(n \log n + e)$ messages distributed algorithm for minimum-weight spanning trees. In *Proc 26th IEEE Symp. on Foundations of Computer Science (FOCS)*, pp. 257-266, 1985.

[6] T.H. Cormen, T. Leiserson, and R.L. Rivest. *Introduction to Algorithms.* MIT Press/McGraw-Hill, 1990.

[7] M. Elkin. A Faster Distributed Protocol for Constructing a Minimum Spanning Tree. In *Proc. ACM-SIAM on Discrete Algorithms (SODA)*, pp. 352-361, 2004.

[8] M. Elkin. An Unconditional Lower Bound on the Hardness of Approximation of Distributed Minimum Spanning Tree Problem. In *Proc. 36th Annual ACM Symp. on Theory of Computing (STOC)*, pp. 331-340, 2004.

[9] P. Fraigniaud, D. Ilcinkas, and A. Pelc. Oracle size: a new measure of difficulty for communication tasks. Proc. 25th Annual ACM Symposium on Principles of Distributed Computing (PODC), pp. 179-187, 2006.

[10] P. Fraigniaud, D. Ilcinkas, and A. Pelc. Tree Exploration with an Oracle. 31st Int. Symp. on Mathematical Foundations of Computer Science (MFCS), LNCS 4162, Springer, pp. 24-37, 2006

[11] E. Gafni. Improvements in the time complexity of two message-optimal election algorithms. In *Proc 4th ACM Symp. on Principles of Distributed Computing (PODC)*, pp. 175-185, 1985.

[12] R. G. Gallager, P. A. Humblet, and P. M. Spira. A distributed algorithm for minimum-weight spanning trees. *ACM Trans. on Programming Lang. and Syst.*, 5:66-67, 1983.

[13] F. Kuhn, T. Moscibroda, and R. Wattenhofer. What cannot be compute Locally! In *Proc. 23th ACM Symp. on Principles of Distributed Computing, (PODC)*, pp. 300-309, 2004.

[14] N. Linial. Locality in distributed graph algorithms. SIAM J. on Computing 21(1):193-201, 1992.

[15] Z. Lotker, B. Patt-Shamir and D. Peleg. Distributed MST for Constant Diameter Graphs In *Proc. 20th ACM Symp. on Principles of Distributed Computing, (PODC)*, pp. 63-72, 2001.

[16] M. Naor and L. Stockmeyer. What can be computed locally? In *25th ACM Symposium on Theory of Computing (STOC)*, pp. 184–193, 1993.

[17] D. Peleg. Distributed Computing: A Locality-Sensitive Approach. SIAM Monographs on Discrete Mathematics, 2000.

[18] D. Peleg and R. Rubinovich. A near-tight lower bound on the time complexity of distributed MST construction. In *40th IEEE SYmp. on Foundations of Computer Science (FOCS)*, pp. 253-261, 1999.