Distributed Computing
https://doi.org/10.1007/s00446-024-00472-6

Deterministic fault-tolerant connectivity labeling scheme

Taisuke Izumi[1] · Yuval Emek[2] · Tadashi Wadayama[3] · Toshimitsu Masuzawa[1]

Received: 16 November 2023 / Accepted: 9 October 2024

Abstract
The *f-fault-tolerant connectivity labeling* (f-FTC labeling) is a scheme of assigning each vertex and edge with a small-size label such that one can determine the connectivity of two vertices s and t under the presence of at most f faulty edges only from the labels of s, t, and the faulty edges. This paper presents a new deterministic f-FTC labeling scheme attaining $O(f^2\text{polylog}(n))$-bit label size and a polynomial construction time, which settles the open problem left by Dory and Parter (in: Proceedings of the 2021 ACM symposium on principles of distributed computing (PODC), pp 445–455, 2021). The key ingredient of our construction is to develop a deterministic counterpart of the graph sketch technique by Ahn et al. (in: Proceedings of the 31st ACM SIGMOD-SIGACT-SIGAI symposium on principles of database systems (PODS), pp 5–14, 2012), via some natural connection with the theory of error-correcting codes. This technique removes one major obstacle in de-randomizing the Dory–Parter scheme. The whole scheme is obtained by combining this technique with a new deterministic graph sparsification algorithm derived from the seminal *ϵ-net* theory, which is also of independent interest. As byproducts, our result deduces the first deterministic fault-tolerant approximate distance labeling scheme with a non-trivial performance guarantee and an improved deterministic fault-tolerant compact routing. The authors believe that our new technique is potentially useful in the future exploration of more efficient FTC labeling schemes and other related applications based on graph sketches.

Keywords Labeling scheme · Fault-tolerance · Graph sketch · Derandomization

1 Introduction

1.1 Motivation and background

Most message-passing distributed systems are modeled by graphs. By the nature of distributed computing, nodes in the network must cooperatively solve a given task without rich access to the whole topological information. In addition, the network is typically prone to faults, i.e., some of the vertices and/or links can be down by faults. Hence the distributed and compact representation of some property of the network (e.g., connectivity) adapting to topology modification is potentially useful for applications in distributed environments. The *f-fault-tolerant connectivity labeling* (f-FTC labeling) is a scheme of assigning each vertex and edge with a small-size label. For any two vertices s and t, and an edge set F of $|F| \leq f$, it determines the connectivity of two vertices s and t under the deletion of edges F only from the labels of s, t, and the edges in F. The concept of f-FTC labeling schemes (precisely, more general *fault-tolerant distance labeling schemes* returning the s-t distance rather than the

Yuval Emek, Tadashi Wadayama and Toshimitsu Masuzawa have contributed equally to this work.

✉ Taisuke Izumi
izumi.taisuke.ist@osaka-u.ac.jp

Yuval Emek
yemek@technion.ac.al

Tadashi Wadayama
wadayama@nitech.ac.jp

Toshimitsu Masuzawa
masuzawa@ist.osaka-u.ac.jp

[1] Graduate School of Information Science and Technology, Osaka Univerisity, 1-5, Yamadaoka, Suita, Osaka 565-0871, Japan

[2] Technion, Haifa 3200003, Israel

[3] Graduate School of Engineering, Nagoya Institute of Technology, Gokiso-cho, Nagoya, Aichi 466-8555, Japan

Published online: 04 November 2024

s-t connectivity) has been initiated explictly by Courcelle and Twigg [15], following an earlier work by Feigenbaum et al. [23]. The feature of FTC labeling schemes as a distributed data structure yields efficient structural algorithms for the *forbidden set routing* which routes packets avoiding a given set of faulty edges, and for more general *fault-tolerant compact routing* [12, 13, 15, 21, 50] where the faulty edge set is initially unknown.

1.2 Our result

While all of early results [1, 2, 6, 15, 16] mainly focus on the construction of small-sized labels for restricted graph classes, f-FTC labeling schemes for general graphs were recently proposed by Dory and Parter [21]. They propose two randomized f-FTC labeling schemes of $O(f + \log n)$-bit and $O(\log^3 n)$-bit label sizes respectively, which guarantee the weaker form of the correctness that the response to each *single* query is correct with high probability. In other words, they guarantee the correctness only for $1 - 1/O(\mathrm{poly}(n))$ fraction of all possible $n^{O(f)}$ queries. We refer to this type of correctness criteria as "whp query support", in contrast with the standard criteria of "full query support" ensuring correct answers for all possible queries with high probability. The authors of [21] also mention how the presented two schemes are converted to the ones with full query support, allowing the blow-up of their label sizes into $O(f \log n)$ bits and $O(f \log^3 n)$ bits respectively (see the footnote 4 of [21]). In total, the paper [21] presents the four randomized schemes, two of which attain full query support and the other two attain only whp query support. They leave as an open problem the polynomial-time *deterministic* construction of compact FTC labeling schemes for general graphs. The main contribution of this paper is to settle this open problem:

Theorem 1 *There exist two deterministic f-FTC labeling schemes for any graph G of n vertices, m edges, and diameter D, which respectively attain the following bounds:*

- *The label size is $O(\log n)$ bits per vertex, and $O(f^2(\log^2 n)\log\log n)$ bits per edge. The query processing time is $\tilde{O}(|F|^4)$,[1] where F is the set of queried edges satisfying $|F| \leq f$. The construction time is polynomial of m.*
- *The label size is $O(\log n)$ bits per vertex, and $O(f^2 \log^3 n)$ bits per edge. The query processing time is $\tilde{O}(|F|^4)$. The construction time is near linear, i.e., $\tilde{O}(mf^2)$. In addition, there exists a deterministic CONGEST distributed algorithm of constructing the labels in $\tilde{O}(\sqrt{mD} + f^2)$ rounds.*

[1] The $\tilde{O}(\cdot)$ notation hides polylog(n) factors.

Note that every deterministic scheme inherently achieves full query support. We emphasize that de-randomizing any of two original schemes of Dory and Parter is a highly non-trivial challenge. Those schemes are based on the other labeling schemes representing a cut structures, whose construction heavily depends on randomness. Our deterministic construction is based on the second scheme of Dory and Parter [21] utilzing the *graph sketch* technique by Ahn et al. [3] as the key structure. Informally, the graph sketch is a labeling scheme to vertices, admitting the detection of an outgoing edge for a given vertex set $S \subseteq V_G$ from the bitwise XOR of all labels of vertices in S. The technical highlight of our result is to develop a deterministic counterpart of the graph sketch technique via some natural connection with the theory of error-correcting codes. This technique is very simple, and completely removes one of two major obstacles in de-randomizing the outgoing edge detection by graph sketches. Yet another obstacle is the sparsification of the input graph. The sketch-based outgoing edge detection, including ours, works only when the input vertex set S has a small number of outgoing edges. To handle the case with many outgoing edges, the original approach prepares a collection of spanning subgraphs, where for each possible input S with a non-empty outgoing edge set, there exists at least one subgraph in the collection such that S has exactly one outgoing edge. The construction of such a collection follows random sampling of edges. Our second contribution is a novel de-randomization technique for this graph sparsification process based on the seminal ϵ-*net* theory in computational geometry [33]. On this part, we present two different algorithms respectively corresponding to the schemes presented in Theorem 1.

In addition to the key technical ideas above, the constructions in Theorem 1 have a few more notable features: First, our result is presented as a general framework with good modularity, and thus one can easily transform our deterministic scheme into an efficient randomized FTC labeling scheme with full query support, just by replacing the graph sparsification part with the conventional random edge sampling. The construction time and label size of this randomized scheme are competitive with the original sketch-based scheme in [21]. Second, we propose a new query optimization strategy. A drawback of our deterministic outgoing edge detection technique requires the decoding time roughly quadratic of the label size. Since the sketch-based f-FTC labeling scheme requires $|F|$ iterations of the outgoing edge detection for processing a single query, the straightforward implementations result in the $\tilde{O}(f^4|F|)$ time for the deterministic case, and $\tilde{O}(f^2|F|)$ time for the randomized case. Our query processing algorithm shaves off this additional $|F|$ factor, as well as getting rid of the dependency on f in the outgoing edge detection. Consequently, we obtain a slightly improved randomized f-FTC labeling scheme of $\tilde{O}(|F|^2)$ decoding time. While the improvement of replacing f by $|F|$ is very straight-

forward and easily applicable to any scheme not limited to ours, it is practically an intriguing feature because in typical scenarios the actual number of faults $|F|$ is substantially smaller than the upper bound f.

The detailed comparison between the schemes in [21] and our schemes are summarized in Table 1.

1.3 Applications

Our deterministic replacement of graph sketches provides a clearer insight to known outgoing edge detection techniques. It is simple, versatile, and potentially useful in the future exploration of other applications not limited to FTC-labeling schemes (e.g., [3, 24–27, 32, 34, 36–38, 40, 43]). Actually, we obtain several non-trivial de-randomization results in related topics. It has been shown in [21] that one can deduce the approximate distance labeling scheme, which provides an approximate s-t distance in $G - F$ given the labels of s, t and the edges in F, utilizing any f-FTC labeling scheme in the black-box manner. In addition, if the underlying FTC-labeling scheme ensures the additional property that its decoding process can report a s-t path avoiding F (in a succinct way), such an approximate distance labeling scheme further deduces an efficient fault-tolerant compact routing scheme. Since the deduction parts are deterministically implemented and our construction ensures this additional property, our deterministic f-FTC labeling schemes also de-randomize the construction of the schemes above. More precisely, we obtain the following applications as corollaries of Theorem 1.

Corollary 2 *Assume that the input graph is any weighted undirected graph with polynomially bounded edge weights. For any positive intergers $k > 0$ and $f > 0$, there exists a f-fault tolerant $O(|F|k)$-approximate distance labeling scheme which achieves $\tilde{O}(f^2n^{1/k})$-bit label size and $\tilde{O}(|F|^4)$ query time.*

Corollary 3 *For any positive integer $k > 0$ and $f > 0$, there exist two deterministic fault-tolerant compact routing schemes which achieve the stretch factor of $O(|F|^2k)$ and one of the following table-size bounds:*

- *$\tilde{O}(f^2n^{1+1/k})$-bit total table size and $\tilde{O}(f^2n^{1+1/k})$-bit maximum local table size.*
- *$\tilde{O}(f^5n^{1/k})$-bit maximum local table size.*

The result of Corollary 2 is the first deterministic scheme for general graphs achieving a non-trivial performance guarantee. On Corollary 3, the prior work by Chechik [12], which attains $O(|F|^2(|F| + \log^2 n)k)$ stretch factor with a smaller table size, is also implemented deterministically, and thus our result is not the first deterministic solution. However, our scheme takes an advantage with respect to stretch factors. Since the proofs completely follow the reduction techniques proposed in [21], this paper does not present the precise formalism on these corollaries. See [21] for details.

1.4 Related work

As mentioned in Sect. 1.1, FTC-labeling schemes, and more general fault-tolerant (approximate) distance labeling schemes are introduced in the literature of *forbidden set routing*, which is the routing scheme avoiding non-adaptive faulty edges/vertices (i.e., the set of faults is not specified at the construction of routing tables, but given at the beginning of packet routing). The first result by Courcelle and Twigg [15] presents a FTC-labeling scheme of $O(k^2 \log n)$-bit labels for graphs of treewidth k, as well as its application to forbidden-set routing. A few results following this line exist [1, 2, 6, 15, 16], but all of them are interested in the construction of compact labels for specific graph classes. FTC-labeling schemes for general graphs is not much addressed until the result by Dory and Parter [21]. In the context of deterministic construction, many of the results for restricted graphs stated above are deterministic, but the deterministic construction for general graphs is not known so far. In the paper of Dory and Parter, two randomized FTC labeling schemes relying on different techniques are presented. While the second scheme is based on graph sketches as we mentioned, the first one relies on the cut-verification labeling by Pritchard and Thurimella [49].

There are many works in the literature of the centralized version of connectivity oracles and (approximate) distance oracles supporting edge/vertex deletion [5, 7, 8, 10, 13, 18–20, 22, 28, 29]. One of the major settings on this line is the case of $f = 1$, which is known as the *replacement path problem* [8, 28, 29], or *distance sensitivity oracle* [5, 7, 8, 22]. Roughly, the replacement path problem computes all pair (approximate) shortest path distances for every possible single edge/vertex fault. The sensitivity oracle is further required to store the information of replacement paths into a compact data structure. Their single-source variants are also investigated [7, 9]. The sensitivity oracles for multiple faults are considered mainly in the context of *fault-tolerant compact routing*, which is a generalization of forbidden-set routing addressing adaptive faults (i.e., the set of faults is not explicitly given at the beginning of packet routing) [12, 13, 21, 50]. There are also a few results considering the oracles specific to connectivity [18–20, 48], which are seen as centralized counterparts of FTC labeling schemes. Obviously, any f-FTC labeling scheme is also usable as a centralized oracle with the space complexity of m times of label size. More general *dynamic connectivity* of undirected graphs aims to develop the data structure of supporting the operations of inserting/deleting edges as well as the connectivity query. By definition, such a data structure can be used as a fault-

Table 1 Comparison between the schemes in [21] and our results

	Label size	Query time	Det./Rand	Correctness	Construction
1st (whp) [21]	$O(f + \log n)$	$\tilde{O}(f^3)$	Rand	whp	$\tilde{O}(fm)$
2nd (whp) [21]	$O(\log^3 n)$	$\tilde{O}(\|F\|)$	Rand	whp	$\tilde{O}(fm)$
1st (full) [21]	$O(f \log n)$	$\tilde{O}(f^3)^\dagger$	Rand	full	$\tilde{O}(fm)$
2nd (full) [21]	$O(f \log^3 n)$	$\tilde{O}(f\|F\|)^\dagger$	Rand	full	$\tilde{O}(fm)$
This paper	$O(f^2 \log^3 n)$	$\tilde{O}(\|F\|^4)$	Det	full	$\tilde{O}(f^2 m)$
This paper	$O(f^2 \log^2 n \log\log n)$	$\tilde{O}(\|F\|^4)$	Det	full	poly(n)
This paper	$O(f \log^3 n)$	$\tilde{O}(\|F\|^2)$	Rand	full	$\tilde{O}(fm)$

The dagger mark † implies that the complexity is not explicitly stated in the original paper, and thus based on our analyses. For any scheme, the dependency on f in query processing time is easily replaced by $|F|$ utilizing the technique proposed in this paper

tolerant connectivity oracle with the query processing time of $O(|F| \cdot$ (operation cost)). While there is a long history of this problem [11, 25, 30, 31, 36, 44, 51, 52], all the known results with polylog(n)-time operation cost rely on amortized analyses or the correctness criteria of whp guarantee. Focusing on deterministic construction, the best known results are the algorithm by Chuzhoy, Gao, Li, Nanongkai, Peng and Saranurak for dynamic connectivity achieving $n^{o(1)}$ operation cost per one edge deletion [11], and the connectivity oracle by Pătraşcu and Thorup [48] achieving $\tilde{O}(|F|)$ query processing time. The deterministic algorithm for dynamic connectivity with worst-case polylog(n)-time operation is a major open problem in this research field.

While this paper focuses only on edge faults, it is also an interesting research direction to consider vertex faults. Despite its similarity, vertex fault-tolerance often exhibits a technical difficulty quite different from edge fault-tolerance. A trivial approach is to reduce the failure of a vertex v into the failure of all the edges incident to v, which results in a f-vertex fault tolerant connectivity labeling scheme of $\tilde{O}(\Delta f)$-bit label size (where Δ is the maximum degree of the input graph). Unfortunately, this approach does not provide a good worst-case bound because Δ could become $\Omega(n)$. The first non-trivial results for vertex fault tolerant labeling schemes are proposed by Parter and Petruschka [46], which provide two schemes respectively attaining a polylogarithmic label size for $f = 2$ and a sublinear size for $f = o(\log\log n)$. Very recently, Parter, Petruschka, and Pettie proposes poly$(f, \log n)$-bit schemes for vertex faults [47]. More precisely, a randomized scheme of $O(f^3 \log^5 n)$-bit labels, and a deterministic scheme of $O(f^7 \log^{13} n)$-bit labels are proposed. Their deterministic scheme also relies on the de-randomization techniques presented in this paper.

The graph sketch technique is first presented by Ahn et al. [3], aiming to develop space-efficient algorithms for graph stream [4, 38, 40]. There are a variety of applications not limited to graph stream, such as distributed computation [24, 26, 27, 32, 34, 37, 43] and dynamic algorithms [25, 36].

1.5 Roadmap

Following the introduction of necessary notations and definitions, we first explain the high-level idea of our framework in Sect. 3, including the explanation of the sub-components constituting our scheme. Section 4 explains the key technical ideas of our de-randomization technique. Following the summary of whole structure in Sect. 5, we present a further query optimization technique in Sect. 6. Section 7 explains the details of our construction. We consider the distributed construction of our labeling schemes in Sect. 8. Finally, we conclude this paper in Sect. 9, as well as a few promising future research directions.

2 Notations and terminologies

We denote the vertex set and edge set of a graph G respectively by V_G and E_G. We use the notation $H \subseteq G$ to represent that H is a subgraph of G. For any edge subset $E' \subseteq E_G$, we define $G - E'$ as the graph obtained from G by removing all the edges in E'. Given a vertex subset $S \subseteq V_G$, an *outgoing edge* of S is the edge having exactly one endpoint in S. We define $\partial_G(S)$ as the set of all outgoing edges of S in G. For any $E' \subseteq E_G$, we also define $\partial_{E'}(S) = \partial_G(S) \cap E'$.

Given a rooted tree T and vertex $v \in V_T$, we denote by $T(v)$ the subtree of T rooted by v. Given an edge $e \in E_T$, we also denote by $T(e)$ the subtree of T rooted by the lower vertex (i.e., the endpoint farther from the root than the other) of e.

An *f-FTC labeling scheme* for a given input graph G consists of a labeling function $L^{\mathsf{con}}_{G,f}$ and a universal decoding function D^{con}_f. The labeling function assigns each of vertices and edges $x \in V_G \cup E_G$ with a label $L^{\mathsf{con}}_{G,f}(x)$ (i.e., a binary string). Let $s, t \in V_G$ be any two vertices and $F \subseteq E$ be any edge subset of size at most f. The decoder function D^{con}_f correctly answers the connectivity between s and t in $G - F$ only with the information of $L^{\mathsf{con}}_{G,f}(s)$, $L^{\mathsf{con}}_{G,f}(t)$, and

$\{L^{\mathsf{con}}_{G,f}(e) \mid e \in F\}$. Note that the decoder function D^{con}_f is universal for all G, and cannot have any direct access to the information of G. The detailed formalism of f-FTC labeling scheme is given in Sect. 7.1.

3 Construction framework

We first introduce a general framework of constructing the f-FTC labeling scheme. The technical core of this framework relies on the scheme by Dory and Parter [21], but some additional techniques and abstractions are newly introduced. Let G be the undirected input graph of n vertices and m edges. Throughout this paper, we fix an arbitrary rooted spanning tree T of G. The framework consists of two technical components. The first one is a weaker variant of f-FTC labeling schemes which supports only the query (s, t, F) satisfying $F \subseteq E_T$ (i.e., only the edges in T can be faulty). We refer to this scheme as the *tree edge f-FTC labeling scheme*. Our tree edge f-FTC labeling scheme is implemented with two other labeling schemes, respectively referred to as the *ancestry labeling scheme* and the *$\mathcal{S}$-outdetect labeling scheme* (explained in the next section). The second component is the very simple transformer which deduces an f-FTC labeling scheme from any tree edge f-FTC labeling scheme with no blow up of label size. In the following sections we explain the outline of each component.

3.1 Tree edge *f*-FTC labeling scheme

First, we state the informal specifications of the two sub-schemes. The formal definitions of these sub-schemes are also presented in Sect. 7.1.

- **Ancestry Labeling Scheme**: Let T be any tree. This scheme assigns each vertex $v \in V_T$ with a label $L^{\mathsf{anc}}_T(v)$. Given two labels $L^{\mathsf{anc}}_T(u)$ and $L^{\mathsf{anc}}_T(v)$ of distinct vertices $u, v \in V_T$, one can determine if u is an ancestor of v, a descendant of v, or otherwise. There exists a linear-time deterministic algorithm which provides the ancestry labeling of $O(\log n)$ bits [39].
- **$\mathcal{S}$-Outdetect Labeling Scheme**: We assume that each edge $e \in E_G$ is assigned with a unique ID from some domain $\mathcal{E}$. Let $\mathcal{S} \subseteq 2^{V_G}$ be a collection of vertex subsets. An $\mathcal{S}$-outdetect labeling scheme assigns each vertex $v \in V_G$ with a label $L^{\mathsf{out}}_{G,\mathcal{S}}(v)$. For any vertex subset $S \in \mathcal{S}$ such that $\partial_G(S)$ is nonempty, one can compute an outgoing edge $e \in \partial_G(S)$ only from the bitwise XOR sum $\bigoplus_{v\in S} L^{\mathsf{out}}_{G,\mathcal{S}}(v)$ of all the labels assigned to vertices in S. If $\partial_G(S)$ is empty, the scheme also detects it. The *graph sketch* technique [1] provides a randomized $\mathcal{S}$-outdetect labeling scheme with $O((\log|\mathcal{S}|)\cdot \mathrm{polylog}(n))$-bit label size.

We define $\mathcal{S}_{f,T}$ as the collection of all vertex subsets S satisfying $\partial_T(S) \le f$. Note that S is not required to induce a connected subtree of T. In what follows, we mostly consider the construction of $\mathcal{S}$-outdetect labeling schemes for $\mathcal{S} = \mathcal{S}_{f,T}$, and thus often omit the subscript $\mathcal{S}_{f,T}$ of $L^{\mathsf{out}}_{G,\mathcal{S}_{f,T}}$ Roughly, our tree edge f-FTC labeling scheme is the combination of the ancestry labeling scheme of T and the $\mathcal{S}_{f,T}$-outdetect labeling scheme of $G - E_T$ for an appropriate edge ID domain $\mathcal{E}$ (explained later). Each vertex u is assigned with the ancestry label of u, and each tree edge $e = (u, v)$ in E_T is assigned with the concatenation of the ancestry labels of u and v, and the XOR sum of the $\mathcal{S}_{f,T}$-outdetect labels over all the descendant vertices of e. We do not have to assign any label to non-tree edges because we focus on the construction of the tree edge f-FTC labeling scheme.

Given a query (s, t, F) satisfying $F \subseteq E_T$ and $|F| \le f$, the spanning tree T is split into $|F| + 1$ subtrees by removing all the edges in F. We refer to the vertex set of each split subtree as a *fragment*. Let S be the fragment of containing s. The query processing algorithm iteratively grows S by detecting an outgoing edge $e \in \partial_{G-E_T}(S)$. If such an edge is found, the fragment that e reaches from S is merged into S. This process terminates until no outgoing edge is found or the fragment with t is merged. If no outgoing edge is found, one can conclude that s and t are not connected in $G - F$, or connected otherwise. Our framework detects an outgoing edge of S in $G - E_T$ by the $\mathcal{S}_{f,T}$-outdetect labeling scheme (recall $S \in \mathcal{S}_{f,T}$ by definition). With the support of the ancestry labeling scheme, one can detect the ancestor–descendant relationship between any entities in s, t, and F, which provides the information of the edge set $\partial_T(S')$ for all fragments S'. To compute $\bigoplus_{v\in S'} L^{\mathsf{out}}_{G-E_T}(v)$ for each fragment S', it suffices to compute the XOR sum of the $\mathcal{S}_{f,T}$-outdetect labels assigned to the edges in $\partial_T(S')$. Since the outdetect label of each edge in T is the XOR sum of all descendants' outdetect labels, it appropriately cancels out the labels assigned with the vertices not in S'. The fragment merging is simple but has a point to be careful. Let $e = (u, v)$ be an outgoing edge of S (assuming $u \in S$), and S' be the fragment containing v. Then the XOR sum over all the labels of $S \cup S'$ is easily calculated by the XOR sum of the two computed sums for S and S'. However, how can we identify the fragment S' containing v? This problem is resolved by embedding the ancestry labels of u and v into the edge ID of e. That is, as a preprocessing step, we assign each edge (u, v) with the pair of $(L^{\mathsf{anc}}_T(u), L^{\mathsf{anc}}_T(v))$ as the edge ID, and the outdetect labeling is constructed for the edge domain by this assignment. Then the decoding of an edge ID immediately yields the information of the fragments containing its endpoints. We formalize our framework explained above into the following lemma.

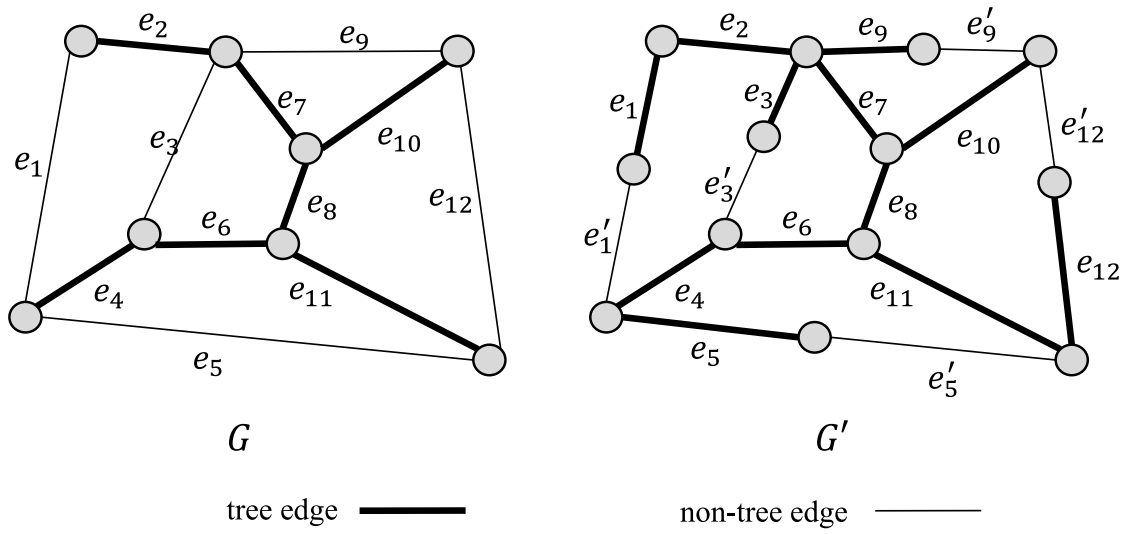

Fig. 1 Auxiliary graph G'

Lemma 4 *Assume any deterministic $\mathcal{S}_{f,T}$-outdetect labeling scheme $(L_G^{\mathsf{out}}, D^{\mathsf{out}})$ of label size α and decoding time β. Then there exists a deterministic tree edge f-FTC labeling scheme of $(\alpha + O(\log n))$-bit label size and $O(|F|(\beta + \log|F| + \alpha/\log n))$ decoding time, where F is a set of faulty edges given by a query.*

The proof details are given in Sect. 7.2.

3.2 Transformation to general scheme

The second component reduces the construction of general f-FTC labeling schemes into that of tree edge f-FTC labeling schemes. For the input graph G and its rooted spanning tree T, our transformation constructs an auxiliary graph G' by subdividing all non-tree edges $e \in E_G \setminus E_T$ into two edges, respectively referred to as e, and e' (see Fig. 1). The spanning tree T' of G' is also obtained by adding e to the tree T. This input transformation naturally defines a bijective mapping $\sigma : E_G \to E_{T'}$, where every edge $e \in E_G$ is mapped to the corresponding edge in T' with the same name. A query (s, t, F) for G is also naturally interpreted to the query $(s, t, \{\sigma(e)|e \in F\})$ for G'. It is easy to see that s and t are connected in $G - F$ if and only if they are connected in $G' - \{\sigma(e)|e \in F\}$. Hence the following proposition obviously holds:

Proposition 5 *Let G and T be the input graph and its rooted spanning tree, and G', T', and σ be the graphs and the mapping as defined above. Assume any tree edge f-FTC labeling scheme $(L^{\mathsf{tree}}_{G',f}, D^{\mathsf{tree}}_f)$ for G' and T'. Then we define the labeling function $L^{\mathsf{con}}_{G,f}$ for G as follows:*

$$L^{\mathsf{con}}_{G,f}(x) = \begin{cases} L^{\mathsf{tree}}_{G',f}(x) & \text{if } x \in V_G \cup E_T \\ L^{\mathsf{tree}}_{G',f}(\sigma(x)) & \text{otherwise.} \end{cases}.$$

Then $(L^{\mathsf{con}}_{G,f}, D^{\mathsf{tree}}_f)$ is a f-FTC labeling scheme for G.

Combining Lemma 4 and Proposition 5, we obtain the following corollary:

Corollary 6 *Assume any deterministic $\mathcal{S}_{f,T}$-outdetect labeling scheme $(L_G^{\mathsf{out}}, D^{\mathsf{out}})$ of label size α whose decoding time is β. Then there exists a deterministic f-FTC labeling scheme of $(\alpha + O(\log n))$-bit label size and $O(|F|(\beta + \log|F| + \alpha/\log n))$ decoding time, where F is a set of faulty edges given by a query.*

4 Technical outline of our approach

4.1 Obstacles in de-randomization

Corollary 6 implies that the difficulty of de-randomization lies only at the implementation of the deterministic $\mathcal{S}_{f,T}$-outdetect labeling scheme. The known $\mathcal{S}$-outdetect labeling scheme based on the graph sketch includes two major points reling on random bits, which are summarized as follows:

- The first is at the computation of vertex labels. Let $I_G(v)$ be the set of incident edges of v in G. The graph sketch first prepares some function $g : \mathcal{E} \to \{0,1\}^k$, where $\mathcal{E}$ is the edge ID domain and k is the label length, and define the label $L_G^{\mathsf{out}}(v)$ of vertex v as the bitwise XOR sum of $g(e)$ for all the edges $e \in I_G(v)$. When computing $\bigoplus_{v \in S} L_G^{\mathsf{out}}(v)$ for a given subset $S \subseteq V_G$, the

value $g(e)$ for any e lying at the inside of S are canceled out because the term $g(e)$ appears exactly twice in the sum $\bigoplus_{v\in S} L_G^{\mathsf{out}}(v) = \bigoplus_{v\in S}(\bigoplus_{e\in I_G(v)} g(e))$. That is, $\bigoplus_{v\in S} L_G^{\mathsf{out}}(v) = \bigoplus_{e\in\partial_G(S)} g(e)$ holds. For clarifying the essence of the first point, we consider the simple case such that $|\partial_G(S)| = 1$ holds (the general case is addressed in the second point). In this case, $\bigoplus_{v\in S} L_G^{\mathsf{out}}(v) = g(e)$ obviously holds for the unique outgoing edge e of S. Hence one can extract the outgoing edge ID from $\bigoplus_{v\in S} L_G^{\mathsf{out}}(v)$, provided that there exists a way of computing the inverse g^{-1}. However, if g is not well-designed, some subset $S' \in \mathcal{S}_{f,T}$ which does not have e as an outgoing edge might accidentally satisfy $\bigoplus_{v\in S'} L_G^{\mathsf{out}}(v) = g(e)$. Then, e is wrongly detected as an outgoing edge of S'. To avoid it, the graph sketch needs to guarantee that $\bigoplus_{v\in S'} L_G^{\mathsf{out}}(v)$ becomes different from the value $g(e)$ of any edge $e \in E_G$ if $|\partial_G(S')| \neq 1$. The first point of utilizing random bits is to attain this condition by taking a random hash function g.

- As explained above, the strategy above provides the $\mathcal{S}$-outdetect labeling scheme working only for $S \in \mathcal{S}$ satisfying $|\partial_G(S)| = 1$. To cover the case of $|\partial_G(S)| > 1$, the original scheme prepares the collection $\mathcal{G}$ of spanning subgraphs of G such that for any $S \in \mathcal{S}$ there exists a corresponding $H \in \mathcal{G}$ which satisfies $|\partial_H(S)| = 1$. The label to vertex v is then obtained by the concatenation of $L_H^{\mathsf{out}}(v)$ for all $H \in \mathcal{G}$. Roughly, each spanning subgraph in $\mathcal{G}$ must be (almost everywhere) sparser than the original input G. The construction of each graph in that collection follows a stochastic sampling of edges, which is the second point relying on randomization.

The technical highlight of our deterministic scheme is twofold, which respectively resolve the two issues above. We explain the outline of each technique in the remainder of this section. The formal argument is provided in Sect. 7.3.

4.2 First technique: deterministic *k*-threshold outdetect labeling scheme

The first technique is a deterministic function g replacing the random function of the graph sketch, based on the theory of error-correcting codes. To explain it, we first present a very concise review of coding theory: A *linear code* W is a y-dimensional linear subspace of $\mathrm{GF}(2)^x$, where $\mathrm{GF}(2)$ is the finite field of two elements (i.e., the element set $\{0, 1\}$ and every calculation is done in modulo 2), y is the length of source data, and x is the length of codewords. The sum of two elements over $\mathrm{GF}(2)$ is described with the operator $\oplus$. We abuse W as the set of all codewords. The *minimum distance* of a linear code is the minimum Hamming distance over all pairs of the codewords in W. In principle, any linear code with minimum distance k can correct any error of less than $k/2$ symbols (but it does not necessarily imply that there exists an efficient algorithm of correcting errors). One of the standard approaches of correcting errors is the *syndrome decoding* based on *parity check matrices*. The parity check matrix C of W is the full-rank $x \times (x - y)$ matrix satisfying $w \cdot C = 0$ for any codeword $w \in W$. Since the parity check matrix of W is uniquely determined from W, linear codes are often defined by the corresponding parity check matrices. A key property of the parity check matrix is that given a codeword with noise $w \oplus \delta$, where $w \in W$ and δ is a noise vector, $(w \oplus \delta) \cdot C = \delta \cdot C$ holds. The *syndrome* of a received (noisy) codeword $w \oplus \delta$ is the vector $(w \oplus \delta) \cdot C$, and the syndrome decoding is the process of recovering δ from the syndrome $(w \oplus \delta) \cdot C = \delta \cdot C$. If δ is correctly recovered, the noiseless codeword w is also recovered by adding δ to the received codeword $w + \delta$.

The background idea of our first technique is as follows: We treat $g : \mathcal{E} \to \{0, 1\}^\ell$ as the mapping from $\mathcal{E}$ to ℓ-dimensional row vectors over $\mathrm{GF}(2)$ (where ℓ is the label size), and define the $|\mathcal{E}| \times \ell$ matrix $C = (c_{e,i})_{e\in\mathcal{E}, i\in[0,\ell-1]}$, where $c_{e,i}$ is the i-th bit of $g(e)$. Let $w(X) = (w_e)_{e\in\mathcal{E}}$ be the characteristic row vector for $X \subseteq \mathcal{E}$, i.e., $w_e = 1$ if $e \in X$, or zero otherwise. Then the following equality holds for any $S \subseteq V$:

$$w(\partial_G(S)) \cdot C = \bigoplus_{e\in\partial_G(S)} g(e) = \bigoplus_{v\in S} L_G^{\mathsf{out}}(v).$$

What we need is the recovery of one non-zero entry in $w(\partial_G(S))$ from the right-side sum. This task can be interpreted into the following scenario: Consider the linear code whose parity check matrix is C. Then recover the noise vector $w(\partial_G(S))$ from the syndrome $w(\partial_G(S))C = \bigoplus_{v\in S} L_G^{\mathsf{out}}(v)$. If the linear code defined by C has a minimum distance $k > 0$, one can obtain the complete recovery of $w(\partial_G(S))$ for any S satisfying $|\partial_G(S)| < k/2$. Since fixing C implies fixing g (and thus the labeling function L_G^{out}), one can obtain the deterministic function g from the parity check matrix of any linear code. We show that *Reed-Solomon code* nicely fits our objective[2], which provides the outdetect labeling of $O(k \log n)$ bits supporting the detection of *all* outgoing edges of a given subset $S \in 2^{V_G}$ in $O(k^2)$ time if $|\partial_G(S)| \leq k$ holds. We refer to such a scheme as the *k-threshold outdetect labeling scheme* hereafter. Let $(L_H^{\mathsf{RS}(k)}, D^{\mathsf{RS}(k)})$ be the k-threshold outdetect labeling scheme for $H \subseteq G$ defined by the $|\mathcal{E}| \times 2k$ parity check matrix of Reed-Solomon code. It satisfies the following properties:

[2] Precisely, Reed-Solomon code is a non-binary code. Hence we need to generalize the argument above slightly, from $\mathrm{GF}(2)$ to any general finite field of characteristic two. The detailed formalism is given in Sect. 7.1.

Proposition 7 *The k-threshold outdetect labeling scheme $(L_H^{\mathsf{RS}(k)}, D^{\mathsf{RS}(k)})$ satisfies the following conditions:*

- *The label size is $O(k \log n)$ bits.*
- *The time taken to assign the labels $L_H^{\mathsf{RS}(k)}(v)$ to all vertices $v \in V_H$ is $O(mk)$. The time of computing $D^{\mathsf{RS}(k)}(L_H^{\mathsf{RS}(k)}(S))$ for given $L_H^{\mathsf{RS}(k)}(S)$ is always bounded by $O(k^2)$.*
- *Given $L_H^{\mathsf{RS}(k)}(S)$, the output of the decoding function is the IDs of all edges in $\partial_H(S)$ if $|\partial_H(S)| \leq k$ holds. If $|\partial_H(S)| > k$, the returned value is unspecified. That is, an arbitrary value can be returned.*

The formal proof is given in Sect. 7.4. In contrast with the single edge detection capability of the original graph sketch, it is a great advantage that our technique admits the detection of at most k outgoing edges, which made the construction of the collection $\mathcal{G}$ much easier. It suffices to construct the sparsification hierarchy $E_{G-E_T} = E_0 \supseteq E_1 \supseteq E_2 \supseteq, \ldots, \supseteq E_h = \emptyset$ for $h = O(\log n)$ such that every $S \in \mathcal{S}_{f,T}$ satisfying $\partial_G(S) \neq \emptyset$ admits a graph $G_i = (V, E_i)$ satisfying $0 < |\partial_{G_i}(S)| \leq k$. We define such a hierarchy as a $(\mathcal{S}, k)$-*good hierarchy*:

Definition 1 Let $\mathcal{S} \subseteq 2^{V_G}$ and a k be a positive integer. A $(\mathcal{S}, k)$-*good hierarchy* of $E_G - E_T$ is the hierarchical edge set $E_G - E_T = E_0 \supseteq E_1 \supseteq E_2 \supseteq, \ldots, \supseteq E_h = \emptyset$ satisfying the following conditions

- The subset $E_{i+1} \subseteq E_i$ is a constant fraction size[3] for any $i \in [0, h-1]$. Note that this condition inherently deduces the property of $h = O(\log n)$.
- For any $S \in \mathcal{S}$ such that $|\partial_{E_i}(S)| > k$, $|\partial_{E_{i+1}}(S)| > 0$ holds.

The collection of k-threshold outdetect labeling schemes for all $G_i = (V_G, E_i)$ forms a $\mathcal{S}$-outdetect labeling scheme for $G - E_T$ with $O(k \log^2 n)$-bit label size and $O(k^2 \log n)$ decoding time. The decoding process tries to obtain the edge(s) in $\partial_{G_i}(S)$ in the decreasing order of i. For the largest i such that $\partial_{G_i}(S)$ is non-empty, the corresponding k-threshold outdetect labeling returns a subset of $\partial_G(S)$ correctly. Formally, the following lemma holds:

Lemma 8 *Assume that any k-threshold outdetect labeling scheme $(\hat{L}_H^{\mathsf{out}}, \hat{D}^{\mathsf{out}})$ of label size α and query processing time β is available. If there exists an algorithm of constructing a $(\mathcal{S}, k)$-good hierarchy for $\mathcal{S} \subseteq 2^{V_G}$ and $E_G - E_T$, there exists an $\mathcal{S}$-outdetect labeling scheme $(L^{\mathsf{out}}_{G-E_T,\mathcal{S}}, D^{\mathsf{out}})$ for $G - E_T$ whose label size is $O(\alpha \log n)$ bits and query processing time is $O(\beta \log n)$.*

[3] We use the statement "a subset $X' \subseteq X$ has a constant fraction size (of X)" to mean $|X'| \leq (1-c)|X|$ for some constant $c > 0$.

The proof is given in Sect. 7.3.

4.3 Second technique: deterministic construction of $(\mathcal{S}_{f,T}, k)$-good hierarchy

Our goal is to provide an $(\mathcal{S}_{f,T}, k)$-good hierarchy $E_0 \supseteq E_1 \supseteq \cdots \supseteq E_k$, for some specific choice of $k > 0$ (to be specified later). Reinterpreting Definition 1, we see that each $E_{i+1} \subseteq E_i$ must (1) have a constant fraction size, and (2) be a *hitting set* for the family of edge sets defined by $\mathcal{Z}_{i,f,k} = \{\partial_{E_i}(S) \mid S \in \mathcal{S}_{f,T}, |\partial_{E_i}(S)| > k\}$. Allowing randomization, it suffices to construct E_{i+1} by the independent edge sub-sampling from E_i with probability $1/2$. Such a construction satisfies the desired property for $k = O(f \log n)$ with high probability (see the Appendix A for details). While the standard greedy algorithm can deterministically construct the hitting set with the same guarantee, such an approach is not tractable because the size of $\mathcal{Z}_{i,f,k}$ could be super-polynomial, i.e., $|\mathcal{Z}_{i,f,k}| = \Theta(|\mathcal{S}_{f,T}|) = \Theta(n^f)$ can hold. The second key ingredient of our construction is to provide a polynomial-time deterministic algorithm of constructing the hitting set for $k = \tilde{O}(f^2)$ through the geometric representation based on the *Euler-tour structure* by Duan and Pettie [19]. In this structure, each undirected edge e in T is replaced by two directed edges with opposite orientations. We refer to the tree T after the replacement as $\vec{T}$, and extend the definition of $\partial_T(S)$ into the directed case $\partial_{\vec{T}}(S)$, which consists of all the directed edges obtained by the replacement of an edge in $\partial_T(S)$. All the edges in $\vec{T}$ are ordered by any Euler tour ET of $\vec{T}$ starting from the root r, and each vertex in the tree is assigned with the smallest order of the incident in-edge (i.e. the edge coming from its parent) as its one-dimensional coordinate in the range $[1, 2n-2]$. We denote the one-dimensional coordinate of a vertex $v \in V_{\vec{T}}$ by $c(v)$. Then, one can map each non-tree edge $e = (u, v)$ into the 2D-point $(c(u), c(v))$ in the range $[1, 2n-2] \times [1, 2n-2]$ (assuming the x-coordinate is always smaller than the y-coordinate to make the mapping well-defined). An example of the geometric representation for the instance of Fig. 1 is presented in Fig. 2. For any integer $a \in [1, 2n-2]$ and axis $z \in \{x, y\}$, let $hs(z, a)$ be the axis-aligned halfspace consisting of all points in the plane whose z-coordinate is at least a. Then it is observed that the point set $\partial_{E_i}(S)$ for any $S \in \mathcal{S}_{f,T}$ lies in the symmetric difference of at most $4f$ axis-aligned halfspaces. More precisely, the following lemma holds:

Lemma 9 *For each directed tree edge $e = (u, v) \in E_{\vec{T}}$, we define $c(e) = c(v)$. Given any vertex subset $S \subseteq V_G$ and edge subset $E' \subseteq E_{G^*}$, the following equality holds.*

$$\partial_{E'}(S) = E' \cap \left(\triangle_{e \in \partial_{\vec{T}}(S), z \in \{x,y\}} hs(z, c(e)) \right),$$

where $\triangle$ represents the symmetric difference of sets.

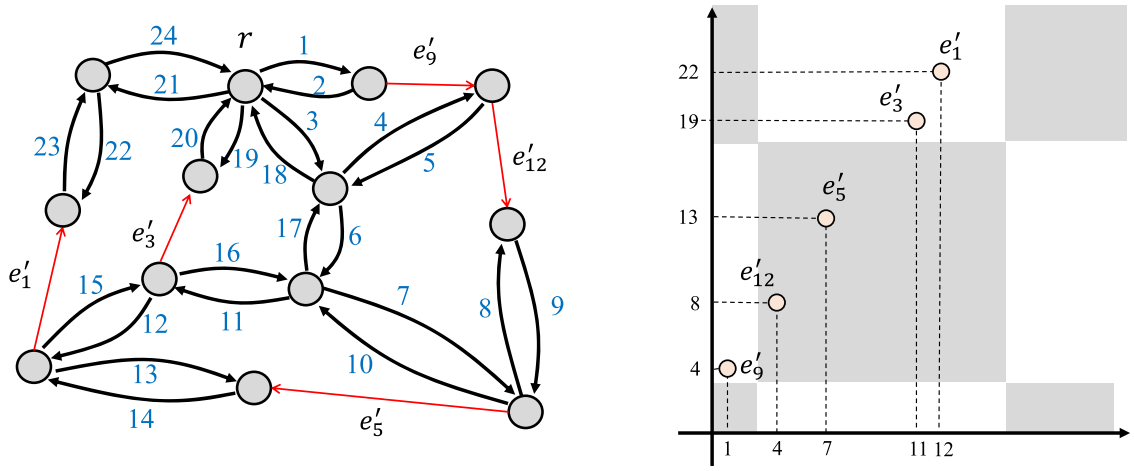

Fig. 2 The geometric interpretation of cutsets. The blue number is the ordering of the directed tree edges by an Euler tour. The non-tree edges e'_1, e'_3, e'_5, e'_9 and e'_{12} are respectively mapped into the points shown in the right-side coordinate system

The proof of the lemma is given in Sect. 7.5. This lemma implies that every $\partial_{E_i}(S)$ for $S \in \mathcal{S}_{f,T}$ is associated with a "checkered shape" in the plane with at most $2f$ vertical (or horizontal) alternations. In Fig. 2, the region colored by white corresponds to the outgoing edges of S such that $\partial_{\vec{T}}(S)$ consists of two directed edges with numbers 3 and 18. Then the problem of constructing the hitting set is seen as the construction of ϵ-*nets* [33].

Definition 2 (ϵ-nets) Let $\mathcal{X}$ be a class of geometric shapes (e.g., rectangles or disks) in some space, and P be a set of points in the space. An ϵ-*net* for $(P, \mathcal{X})$ is a subset $P' \subseteq P$ such that for any $X \in \mathcal{X}$, $X \cap P' \neq \emptyset$ holds if $|X \cap P| \geq \epsilon |P|$.

Let us define the class $\mathcal{H}_q$ which consists of all the shapes formed by the symmetric difference of at most q horizontal halfspaces $hs(y, a_0), hs(y, a_1), \ldots, hs(y, a_{q'-1})$ $(q' \leq q$, $a_i \in [1, 2n-2])$ and the corresponding vertical halfspaces $hs(x, a_0), hs(x, a_1), \ldots, hs(x, a_{q'-1})$. Recalling that the construction of E_{i+1} is equivalent to the computation of a hitting set of a constant fraction size for the family $\mathcal{Z}_{i,f,k} = \{\partial_{E_i}(S) \mid S \in \mathcal{S}_{f,T}, |\partial_{E_i}(S)| > k\}$, our goal is to construct the hitting set of a constant fraction size for $\{Z \cap E_i \mid Z \in \mathcal{H}_{2f}, |Z \cap E_i| \geq k\}$, i.e., to construct an ϵ-net of a constant fraction size for $(E_i, \mathcal{H}_{2f})$ and $\epsilon = k/|E_i|$.

While there are a few deterministic polynomial-time algorithms of constructing nearly-optimal ϵ-nets for a given class of shapes [14, 41], their running times exponentially depend on the VC dimension of the given class. The VC dimension of $\mathcal{H}_{2f}$ is $\Omega(f)$, and thus those algorithms cannot be applied.[4] To circumvent this issue, we regard any shape in $\mathcal{H}_{2f}$ as the union of $(2f+1)^2/2$ disjoint axis-aligned rectangles. For any $H \in \mathcal{H}_{2f}$ containing at least $\gamma(2f+1)^2/2$ points ($\gamma \geq 1$), there exists at least one axis-aligned rectangle as a subset of H which contains at least γ points. Hence the construction of an $O(\gamma/|E_i|)$-net of a constant fraction size for all axis-aligned rectangles deduces an $O(\gamma f^2/|E_i|)$-net of a constant fraction size for $\mathcal{H}_{2f}$. In other words, we can obtain a deterministic polynomial-time algorithm of constructing $(\mathcal{S}_{f,T}, O(\gamma f^2))$-good hierarchy from any deterministic polynomial-time algorithm of constructing a (γ/N)-net of a constant fraction size for N points and all axis-aligned rectangles.

Since the VC-dimension of axis-aligned rectangles is a constant, it is possible to use the general de-randomization technique as stated above. The optimal ϵ-net for N points and axis-aligned rectangles is of size $O(\log\log N/\epsilon)$ (i.e., $O(\log\log N/N)$-net of a constant fraction size), which is known to be deterministically constructed [42]. However, the construction time takes a high-exponent polynomial. Hence we also present a simpler alternative construction which provides a $O(\log N/N)$-net of a constant fraction size for axis-aligned rectangles in a near linear time. In summary, we obtain the following lemma:

Lemma 10 (Partly by Moustafa et al. [42]) *There exist two deterministic algorithms of constructing an $O(\gamma/N)$-net of a constant fraction size for any N points and all axis-aligned rectangles, each of which attains the following performance guarantee.*

- *$\gamma = \log\log N$ and the construction time is $\mathrm{poly}(N)$.*
- *$\gamma = \log N$ and the construction time is $\tilde{O}(N)$.*

By the argument above, we obtain the following lemma.

[4] More precisely, they takes $\Omega(1/\epsilon)^d$ time for the class with VC dimension d. This would efficiently work if $1/\epsilon$ is small, but in our use $1/\epsilon$ is roughly close to m/f, and thus they are not tractable for $f = \omega(1)$.

Lemma 11 *There exist two deterministic algorithms respectively constructing a $(\mathcal{S}_{f,T}, k)$-good hierarchy with the following performance guarantees:*

- $k = O(f^2 \log n)$ *and the construction time is* $\tilde{O}(m)$.
- $k = O(f^2 \log \log n)$ *and the construction time is* $\mathrm{poly}(m)$.

The formal proofs of the lemmas above are given 7.5.

5 Wrap-up

We summarize how all the components are combined into the f-FTC labeling scheme. We present below the case of the f-FTC labeling scheme of label size $O(f^2 \log^3 n)$. Yet another scheme of label size $O(f^2(\log^2 n) \log \log n)$ is constructed in the same way. Consider any input graph G of n vertices and m edges. The whole construction algorithm works as the following steps:

1. Construct any spanning tree T of G, and transform G and T into the auxiliary graph G' and its spanning tree T' explained in Sect. 3.2. Note that the graph G' satisfies $|V_{G'}| = O(m)$ and $|E_{G'}| = O(m)$.
2. Utilizing the algorithm of Lemma 11, construct a $(\mathcal{S}_{f,T}, cf^2 \log n)$-good hierarchy $E_0 \supseteq E_1 \supseteq E_2 \supseteq , \ldots, \supseteq E_h$ for $E_{G'} - E_{T'}$, where c is a hidden constant. The construction time is $\tilde{O}(|E_{G'}|) = O(m)$.
3. Let $G_i = (V_{G'}, E_i)$ $(0 \leq i \leq h)$. Construct $L_{G_i}^{\mathsf{RS}(cf^2 \log n)}$ for all i. By the construction of step 2 and Lemma 8, we obtain the $\mathcal{S}_{f,T}$-outdetect labeling scheme of $O(f^2 \log^3 n)$-bit label size. The construction time is in $\tilde{O}(mkh) = \tilde{O}(mf^2)$, and the decoding time is $\tilde{O}(f^4)$ due to Proposition 7 and Lemma 8.
4. Construct the ancestry labeling scheme for T' by the algorithm of [39], which takes $O(m)$ time.
5. By Lemma 4, the labels constructed in the steps 3 and 4 form a tree edge f-FTC labeling scheme for G' and T'. Then we also obtain the f-FTC labeling scheme for G by Corollary 6.

Finally we have the following theorem.

Theorem 12 *There exist two deterministic f-FTC labeling schemes for any graph G of n vertices and m edges which respectively attain the following bounds:*

- *The label size is $O(\log n)$ bits per vertex, and $O(f^2(\log^2 n) \log \log n)$ bits per edge. The query processing time is $\tilde{O}(|F| f^4)$, where F is the set of queried edges satisfying $|F| \leq f$. The construction time is polynomial of m.*
- *The label size is $O(\log n)$ bits per vertex, and $O(f^2 \log^3 n)$ bits per edge. The query processing time is $\tilde{O}(|F| f^4)$. The construction time is near linear, i.e., $\tilde{O}(mf^2)$.*

This theorem is a weaker form of Theorem 1. In the next section we present how one can improve this to Theorem 1 claiming faster query processing time.

6 Improving query processing time

The algorithmic idea of this improvement is twofold: The first idea attains the *adaptiveness* of $\mathcal{S}_{f,T}$-outdetect labeling scheme, i.e., to get rid of the dependency on f in the decoding time. There is a simple technique of transforming any $\mathcal{S}_{f,T}$-outdetect labeling scheme with $\tilde{O}(f^c)$ decoding time into the one with $\tilde{O}(|\partial_T(S)|^c)$ decoding time for any given query $S \in \mathcal{S}_{f,T}$: Instead of single labeling, we prepare the multiple instances of the (non-adaptive) $\mathcal{S}_{f',T}$-outdetect labeling scheme for $f' = 2, 4, \ldots, f$. If the original labeling scheme has a $\Omega(f)$-bit label size, this transformation does not cause any asymptotic blow-up of label size. Assume that a query $S \subseteq \mathcal{S}_{f,T}$ is given. Since $S \in \mathcal{S}_{|\partial_T(S)|,T}$ necessarily holds, the adaptive scheme can find the outgoing edge of S by utilizing the $\mathcal{S}_{f',T}$-outdetect labeling scheme for f' such that $f'/2 < |\partial_T(S)| \leq f'$ holds, which runs in $\tilde{O}(|\partial_T(S)|^c)$ time.[5]

The second idea is to utilize the adaptive scheme for further acceleration of the decoding time. In processing the query of (s, t, F), every query $S \subseteq V_G$ issued to the $\mathcal{S}_{f,T}$-outdetect labeling scheme necessarily belongs to $\mathcal{S}_{|F|,T}$. Hence The adaptive decoding of the $\mathcal{S}_{f,T}$-outdetect labeling scheme always runs in $\tilde{O}(|F|^4)$ time for the deterministic cases, and in $\tilde{O}(|F|^2)$ time for the randomized case. Since the decoding time of the tree edge f-FTC labeling scheme is dominated by $|F|$ queries to the $\mathcal{S}_{f,T}$-outdetect labeling scheme, the straightforward implementation respectively results in the decoding time of $\tilde{O}(|F|^5)$ and $\tilde{O}(|F|^3)$. We shave off this extra $|F|$ factor by a simple refinement of the decoding process of the tree edge f-FTC labeling scheme: In the refined process, the outgoing edge detection for merging fragments is always applied to the fragment S such that $|\partial_T(S)|$ is the smallest of all the fragments currently managed, while the original process always applies it to the fragment with s. By a careful analysis, we obtain the following lemma:

[5] In reality, this transformation is not necessary if we utilize our deterministic $\mathcal{S}_{f,T}$-outdetect labeling scheme based on the Reed-Solomon code. More precisely, it inherently admits the adaptive decoding without any modification of the label construction. See the Appendix B for details.

Lemma 13 *Assume that there exists a $\mathcal{S}_{f,T}$-outdetect labeling scheme of label size $\alpha = \tilde{O}(f^b)$ and decoding time $\beta = \tilde{O}(f^c)$. Then there exists a f-FTC labeling scheme of $O(\alpha + \log n)$-bit label size and $\tilde{O}(|F|^{b+1} + |F|^c)$ decoding time. The resultant f-FTC labeling scheme is deterministic if the corresponding $\mathcal{S}_{f,T}$-outdetect labeling scheme is deterministic.*

The three $\mathcal{S}_{f,T}$-outdetect labeling schemes we presented in this paper (including the randomized case) attain $(\alpha, \beta) = (O(f^2(\log^2 n)\log\log n), \tilde{O}(f^4))$, $(O(f^2\log^3 n), \tilde{O}(f^4))$, and $O(f\log^3 n), \tilde{O}(f^2))$. By this lemma, they respectively deduce the schemes as claimed in Table 1.

7 Technical details

7.1 Formal specification of labeling schemes

Fault-Tolerant Connectivity Labeling A *f-fault-tolerant connectivity labeling scheme* (f-FTC labeling scheme) for a given input graph G consists of a labeling function $L^{\mathsf{con}}_{G,f} : V_G \cup E_G \to \{0,1\}^*$ and a universal decoding function $D^{\mathsf{con}}_f : \{0,1\}^* \times \{0,1\}^* \times \{0,1\}^* \to \{0,1\}$. For an edge subset $F = \{e_1, e_2, \ldots, e_{|F|}\} \subseteq E_G$, let $L^{\mathsf{con}}_{G,f}(F)$ be the concatenation of the labels $L^{\mathsf{con}}_{G,f}(e_1) \circ L^{\mathsf{con}}_{G,f}(e_2) \circ \cdots \circ L^{\mathsf{con}}_{G,f}(e_{|F|})$ in an arbitrary order. For a query (s,t,F) of $s,t \in V_G$ and an edge subset $F \subseteq E_G$ of cardinality at most f, the decoder function returns the s-t connectivity in $G - F$ when given as input the labels of s,t and all the edges in F, i.e., D^{con} satisfies that $D^{\mathsf{con}}(L^{\mathsf{con}}_G(s), L^{\mathsf{con}}_G(t), L^{\mathsf{con}}_G(F)) = 1$ if and only if s and t are connected in $G - F$. The *label size* of the scheme is defined as the maximum length of the labels assigned to vertices and edges.

Ancestry labeling Given any rooted tree T, this labeling scheme assigns all vertices with the labels such that the ancestor–descendant relationship between any two vertices is determined only from their labels. More precisely, the ancestry labeling scheme for T consists of a labeling function $L^{\mathsf{anc}}_T : V_T \to \{0,1\}^*$ and a universal decoding function $D^{\mathsf{anc}} : \{0,1\}^* \times \{0,1\}^* \to \{-1,0,1\}$ (not dependent on T). It determines if two given vertices $x, y \in V_T$ have the ancestor–descendant relationship or not from their labels, i.e., $D^{\mathsf{anc}}(L^{\mathsf{anc}}_T(x), L^{\mathsf{anc}}_T(y)) = 1$ if x is an ancestor of y, -1 if y is an ancestor of x, or 0 otherwise (including the case of $x = y$). The following lemma is well-known:

Lemma 14 (Kannan et al. [39]) *Let T be a rooted tree of n vertices. There exists a deterministic ancestry labeling scheme with label size of $O(\log n)$ bits. Computing $L^{\mathsf{anc}}_T(x)$ for all $x \in V_T$ takes $O(n)$ time, and $D^{\mathsf{anc}}(L^{\mathsf{anc}}_T(x), L^{\mathsf{anc}}_T(y))$ for each $x, y \in V_T$ takes $O(1)$ time. The labeling function L^{anc}_T is injective for any T, i.e., a unique label assignment.*

$\mathcal{S}$-Outdetect Labeling for $\mathcal{S} \subseteq 2^{V_G}$

Let $\mathbb{F}$ be any finite field of characteristic two whose addition and multiplication operators are respectively denoted by "$\oplus$" and "$\cdot$",[6] and $\mathcal{E}$ be the domain of unique edge IDs not depending on G.[7] An $\mathcal{S}$-outdetect labeling scheme for $\mathcal{S} \subseteq 2^{V_G}$ consists of a vertex labeling function $L^{\mathsf{out}}_{G,\mathcal{S}} : V_G \to \mathbb{F}^\ell$ and a universal decoding function $D^{\mathsf{out}} : \mathbb{F}^\ell \to \mathcal{E}$, where ℓ is a positive integer representing the size of labels. It must satisfy the following two conditions:

- For any $S \in \mathcal{S}$, $D^{\mathsf{out}}(\sum_{v\in S} L^{\mathsf{out}}_{G,\mathcal{S}}(v))$ returns the ID of an outgoing edge of S in G if $\partial_G(S)$ is nonempty.
- If $\partial_G(S) = \emptyset$, $D^{\mathsf{out}}(\sum_{v\in S} L^{\mathsf{out}}_{G,\mathcal{S}}(v))$ returns formal zero, which is a special value in $\mathcal{E}$ never assigned to actual edges.
- If $S \notin \mathcal{S}$, the returned value is undefined, i.e., an arbitrary value is returned.

For short, we use the notation $L^{\mathsf{out}}_{G,\mathcal{S}}(S)$ to represent $\sum_{v\in S} L^{\mathsf{out}}_{G,\mathcal{S}}(v)$ in the following argument. By definition, $D^{\mathsf{out}}(L^{\mathsf{out}}_{G,\mathcal{S}}(V_G)) = 0$ must hold. The *k-threshold outdetect labeling scheme* for G is a special case of $\mathcal{S}$-outdetect labeling schemes such that $\mathcal{S}$ consists of all $S \subseteq V_G$ satisfying $|\partial_G(S)| \le k$.

7.2 Proof of Lemma 4

Let $G^* = G - E_T$ for short. The tree edge f-FTC labeling scheme is implemented by any $\mathcal{S}_{f,T}$-outdetect labeling scheme for G^* and any ancestry labeling scheme for T. Let $(L^{\mathsf{anc}}_T, D^{\mathsf{anc}})$ be the ancestry labeling scheme. Without loss of generality, we assume that $L^{\mathsf{anc}}_T(v)$ for all v has a fixed bit length $p = O(\log n)$. We assign the edge ID $L^{\mathsf{anc}}_T(u) \circ L^{\mathsf{anc}}_T(v) \in \{0,1\}^{2p}$ to each edge $(u,v) \in E_{G^*}$. The uniqueness of the edge ID follows the uniqueness of the ancestry labeling guaranteed in Lemma 14. For the edge ID domain $\mathcal{E} = \{0,1\}^{2p}$, we construct the $\mathcal{S}_{f,T}$-outdetect labeling scheme $(L^{\mathsf{out}}_{G^*}, D^{\mathsf{out}})$. The labeling function $L^{\mathsf{con}}_G(v)$ of our tree edge f-FTC labeling scheme is defined as follows:

- $L^{\mathsf{con}}_G(v) = L^{\mathsf{anc}}_T(v)$.
- For any $e = (u,v) \in E_T$, $L^{\mathsf{con}}_G(e) = L^{\mathsf{anc}}_T(u) \circ L^{\mathsf{anc}}_T(v) \circ L^{\mathsf{out}}_{G^*}(V_{T(e)})$. Recall that any tree edge f-FTC labeling scheme does not have to assign labels to non-tree edges.

[6] A field $\mathbb{F}$ has characteristic two if and only if any element $x \in \mathbb{F}$ satisfies $x \oplus x = 0$ (where 0 is the unit element)

[7] Since the size of this domain inherently depends on n, it should be defined precisely as $\mathcal{E}_n$, which is the universal domain valid for all the graphs with at most n vertices. But we intentionally omit such a dependency for avoiding non-essential complication.

We see how to implement the decoding function. Assume that a query (s, t, F) of $|F| \leq f$ is given. Let $\mathcal{C}(F) = \{C_0, C_1, \ldots, C_{|F|}\}$ be the collection of the fragments (i.e., the vertex subset inducing a connected component of $T - F$). Let $V(F)$ be the set of the endpoints of the edges in F. By introducing any total order over $\{0, 1\}^p$, we define the ID of $C_i \in \mathcal{C}(F)$ as the maximum ancestry label in $V_{C_i} \cap V(F)$, and abuse C_i itself as the ID of C_i. The *component graph* $H/\mathcal{C}(F)$ of a spanning subgraph $H \subseteq G$ is the multigraph obtained from H by contracting each vertex set C_i $(0 \leq i \leq |F|)$ into a single vertex and then removing self-loops. The following proposition is known:

Proposition 15 (Claim 3.14 in [21]) *Let (s, t, F) be any given query, and $|F| \leq f$. Then $T/\mathcal{C}(F)$ is computed deterministically in $O(|F| \log |F|)$ time. In addition, there exists a deterministic algorithm which is given $L_T^{\mathsf{anc}}(v)$ and returns in $O(\log |F|)$ time the ID of $C_i \in \mathcal{C}(F)$ containing v.*

The proposition above allows the detection of the two connected components in $T - F$ respectively containing s and t. We assume $s \in C_0$ and $t \in C_1$ without loss of generality. Starting from C_0, the decoding procedure grows the component containing s iteratively by finding its outgoing edge: Initially, let $S = C_0$. The procedure detects an outgoing edge (u, v) of S in G^*, where u is the vertex in S and $v \in C_j$ for some $j \in [1, |F|]$, until no outgoing edge is found. When the edge (u, v) is found, S is updated with $S \cup \{C_j\}$. Obviously, s and t are connected in $G - F$ if and only if this process merges C_1 with S. Throughout this process, $\partial_T(S) \subseteq F$ obviously holds and thus $S \in \mathcal{S}_{f,T}$ always holds. Hence one can use the $\mathcal{S}_{f,T}$-outdetect labeling scheme to find an outgoing edge (u, v). The primary matter is how to manage $L^{\mathsf{out}}_{G^*}(S)$. We resolve it with a technique similar to [21] (Claim 3.15). The following proposition holds.

Proposition 16 *For any $X \subseteq V_T$, $L^{\mathsf{out}}_{G^*}(X) = \bigoplus_{e \in \partial_T(X)} L^{\mathsf{out}}_{G^*}(V_{T(e)})$ holds.*

Proof Let $k = |\partial_T(X)|$. The proof is based on the induction on k. (Basis) $k = 0$: Then $X = V_G$ or $X = \emptyset$ holds. Since we have $L^{\mathsf{out}}_{G^*}(V_T) = 0$ by the definition of the $\mathcal{S}_{f,T}$-outdetect labeling scheme, the proposition obviously holds. (Inductive step) Suppose that the proposition holds for any X' such that $|\partial_T(X')| = k$, and consider X such that $|\partial_T(X)| = k + 1$ holds. Let $\partial_T(X) = \{e_1, e_2, \ldots, e_{k+1}\}$. Without loss of generality, we assume that there is no descendant of e_{k+1} in $\partial_T(X)$, i.e., $T(e_{k+1})$ does not contain any edge in $\partial_T(X)$. Let $e_{k+1} = (u, v)$ where v is the lower vertex of e_{k+1}. We consider the case of $v \in X$. In this case, we have $V(T(e_{k+1})) \subseteq X$ because no edge in $\partial_T(X)$ is a descendant of e_{k+1}. Then, for $W = X \backslash V_{T(e_{k+1})}$, we have $\partial_T(W) = \partial_T(X) \backslash \{e_{k+1}\}$, and thus $|\partial_T(W)| = k$ holds. By the induction hypothesis, we obtain $L^{\mathsf{out}}_{G^*}(W) = \bigoplus_{e \in \partial_T(W)} L^{\mathsf{out}}_{G^*}(V_{T(e)})$. Since W and $V_{T(v)}$ are disjoint by definition, we have

$$\begin{aligned} L^{\mathsf{out}}_{G^*}(X) &= L^{\mathsf{out}}_{G^*}(W) + L^{\mathsf{out}}_{G^*}(V_{T(e_{k+1})}) \\ &= \bigoplus_{e \in \partial_T(W)} L^{\mathsf{out}}_{G^*}(V_{T(e)}) + L^{\mathsf{out}}_{G^*}(V_{T(e_{k+1})}) \\ &= \bigoplus_{e \in \partial_T(X) \backslash \{e_{k+1}\}} L^{\mathsf{out}}_{G^*}(V_{T(e)}) + L^{\mathsf{out}}_{G^*}(V_{T(e_{k+1})}) \\ &= \bigoplus_{e \in \partial_T(X)} L^{\mathsf{out}}_{G^*}(V_{T(e)}). \end{aligned}$$

In the case of $v \notin X$, one can utilize the fact of $L^{\mathsf{out}}_{G^*}(X) = L^{\mathsf{out}}_{G^*}(V_T - X)$, which is easily deduced from the fact of $L^{\mathsf{out}}_{G^*}(V_T) = 0$ (and the property of characteristic two). Due to this equality, it suffices to show $L^{\mathsf{out}}_{G^*}(V_T - X) = \bigoplus_{e \in \partial_T(V_T - X)} L^{\mathsf{out}}_{G^*}(V_{T(e)})$. Since $v \in V_T - X$ holds, it is proved in the same way as the case of $v \in X$. □

We prove Lemma 4 by the proposition above.

Lemma 4 *Assume any deterministic $\mathcal{S}_{f,T}$-outdetect labeling scheme $(L^{\mathsf{out}}_G, D^{\mathsf{out}})$ of label size α and decoding time β. Then there exists a deterministic tree edge f-FTC labeling scheme of $(\alpha + O(\log n))$-bit label size and $O(|F|(\beta + \log |F| + \alpha/\log n))$ decoding time, where F is a set of faulty edges given by a query.*

Proof By Proposition 16, one can compute the $L^{\mathsf{out}}_{G^*}(C_i)$ for all $i \in [0, |F|]$ from the labels of the edges in F. Assume an outgoing edge $e = (u, v)$ of S ($v \notin S$) is detected by the $\mathcal{S}_{f,T}$-outdetect labeling. Since we can obtain the ID $L_T^{\mathsf{anc}}(u) \circ L_T^{\mathsf{anc}}(v)$ of e, the ancestry label $L_T^{\mathsf{anc}}(v)$ is available. By Proposition 15, it also gives the ID of the component in $\mathcal{C}(F)$ which contains v. Let us assume $v \in C_j$. When merging C_j into S, we have known both $L^{\mathsf{out}}_{G^*}(S)$ and $L^{\mathsf{out}}_{G^*}(C_j)$. Thus the label $L^{\mathsf{out}}_{G^*}(S)$ is updated by adding $L^{\mathsf{out}}_{G^*}(C_j)$. The query processing time is spent for $|F|$ times of querying the $\mathcal{S}_{f,T}$-outdetect labeling scheme, which takes $O(|F|(\beta + \log |F|))$ time in total. The initial set-up takes $O(|F| \log |F| + (|F|\alpha)/\log n)$ time, where the term $|F| \log |F|$ is for computing the component graph, and $(|F|\alpha)/\log n$ is for computing the $\mathcal{S}_{f,T}$-outdetect labels of all fragments (recall that we assume the standard word-RAM model, and thus a single XOR sum operation of α-bit labels takes $O(\alpha/\log n)$ time). The label size is obviously $O(\alpha + \log n)$. □

7.3 Construction of deterministic $\mathcal{S}_{f,T}$-outdetect labeling scheme

Lemma 8 *Assume that any k-threshold outdetect labeling scheme $(\hat{L}^{\mathsf{out}}_H, \hat{D}^{\mathsf{out}})$ of label size α and query processing time β is available. If there exists an algorithm of constructing a $(\mathcal{S}, k)$-good hierarchy for $\mathcal{S} \subseteq 2^{V_G}$ and $E_G - E_T$, there exists an $\mathcal{S}$-outdetect labeling scheme $(L^{\mathsf{out}}_{G-E_T,\mathcal{S}}, D^{\mathsf{out}})$ for $G - E_T$ whose label size is $O(\alpha \log n)$ bits and query processing time is $O(\beta \log n)$.*

Proof Assume that a $(\mathcal{S}, k)$-good hierarchy $E_0 \supseteq E_1 \supseteq E_2 \supseteq, \ldots, \supseteq E_h$ is obtained. Let $G_i = (V_G, E_i)$. The labeling function $L^{\mathsf{out}}_{G-E_T,\mathcal{S}}$ is defined as the concatenation of the labels by $\hat{L}^{\mathsf{out}}_{G_i}$ for all $i \in [0, h]$, i.e., $L^{\mathsf{out}}_{G-E_T}(v) = \hat{L}^{\mathsf{out}}_{G_0}(v) \circ \hat{L}^{\mathsf{out}}_{G_1}(v) \circ \cdots \circ \hat{L}^{\mathsf{out}}_{G_h}(v)$ (where $\circ$ is the binary operator of concatenating two strings). To detect an outgoing edge of $S \in \mathcal{S}$, it suffices to compute the value of $\hat{D}^{\mathsf{out}}(\bigoplus_{v\in S} \hat{L}^{\mathsf{out}}_{G_i}(v))$ such that it returns a non-zero value and $\hat{D}^{\mathsf{out}}(\bigoplus_{v\in S} \hat{L}^{\mathsf{out}}_{G_j}(v))$ for any $j > i$ returns zero. The condition of the hierarchy implies $0 < |E_i \cap \partial_G(S)| \leq k$, and thus $\hat{D^{\mathsf{out}}}(\bigoplus_{v\in S} \hat{L}^{\mathsf{out}}_{G_i}(v))$ correctly returns all the outgoing edges of S in $E_i \subseteq E_G - E_T$. The bounds for the label size and the query processing time are obvious. □

7.4 *k*-threshold outdetect labeling: choice of codes

Following the construction presented in Sect. 4.2, any linear code of minimum distance $2k$ naturally induces a k-threshold outdetect labeling scheme for any graph G. The label size is determined by the number of columns of the parity check matrix C of the code. Obtaining both a smaller number of columns and a larger minimum distance is roughly equivalent to achieving a good code rate. Hence, as a general principle, any high-rate code would provide a good scheme. In addition, we need to care other additional criteria for efficient implementation of the outdetect labeling scheme, which are stated as follows:

- We use a parity check matrix which has $|\mathcal{E}|$ rows, i.e., the codeword length is $|\mathcal{E}|$. Thus it is not appropriate to use the error-correcting codes whose decoding time depends on the codeword length. Ideally, the decoding time should depend only on the length of the syndrome (i.e., the length of labels).
- The construction of the label for an edge e corresponds to the computation of the row vector of C corresponding to e. Since the number of rows $|\mathcal{E}|$ could be much larger than the actual number of edges $|E_G|$, computing the whole matrix C can result in slower computation of edge labels to all $e \in E_G$. To complete the label assignment in time dependent only on the actual number $|E_G|$ of edges but not on $|\mathcal{E}|$, C must admit efficient "local" computation of a specified row.

One of the error-correcting codes addressing the issues above is Reed-Solomon code. Reed-Solomon code is a non-binary code whose alphabet is a finite field $\mathbb{F}$ of order $|\mathcal{E}| + 1$, and the number of rows is chosen arbitrarily. Since $|\mathcal{E}|$ is a polynomial of n in our application, each code symbol is encoded with $O(\log n)$ bits, and the addition and multiplication over $\mathbb{F}$ takes $O(1)$ time in the standard word-RAM model. Let C_{2k} be the $|\mathcal{E}| \times 2k$ parity check matrix of Reed-Solomon code. We have the following nice features:

- The minimum distance of the code defined by C_{2k} is equal to $2k$. That is, a k-threshold outdetect labeling scheme is deduced from C_{2k}. Since each row vector of C_{2k} is encoded by $O(k \log |\mathcal{E}|)$ bits, the label size is $O(k \log n)$ bits.
- Let w be any $|\mathcal{E}|$-dimensional vector over $\mathbb{F}$ which contains at most k nonzero elements. There exists a deterministic algorithm of computing all non-zero elements in w (in the form of the pairs of value and position) from $w \cdot C_{2k}$, which runs in $O(k^2)$ time in the standard word-RAM model [17]. The recovery of w necessarily succeeds if w contains at most k non-zero elements, but the result can become arbitrary if w contains more than k non-zero elements.
- Given any $e \in \mathcal{E}$, the row vector of C_{2k} corresponding to e is deterministically computed in $O(k)$ time in the standard word-RAM model.

Let $(L^{\mathsf{RS}(k)}_H, D^{\mathsf{RS}(k)})$ be the k-threshold outdetect labeling scheme for $H \subseteq G$ defined by the $|\mathcal{E}| \times 2k$ parity check matrix of Reed-Solomon code. The features above obviously deduces Proposition 7.

7.5 Deterministic construction of good hierarchy

We first focus on the proof of Lemma 9. Since $\partial_{G'}(S) = \partial_{G'}(V_G \setminus S)$ holds for any $S \subseteq V_G$ and any spanning subgraph G' of G, we assume S always contains the root r wlog. We use notations $\vec{T}$, ET, $c(v)$, $hs(z, a)$, and $\mathcal{H}_{2f}$ as defined in Sect. 4.3. In addition, we introduce a few additional notations. Let $ET = e_1, e_2, \ldots, e_i, \ldots, e_{2n-2}$. The prefix $e_1, e_2, \ldots, e_i$ of ET up to the i-th element is denoted by $ET(i)$. For the proof, we introduce an auxiliary lemma.

Lemma 17 *Assume that $S \subseteq V_T$ contains the root r of T. For any vertex $v \in V_T$, $|ET(c(v)) \cap \partial_{\vec{T}}(S)|$ is even if $v \in S$, or odd otherwise.*

Proof Suppose we walk from r to v along $ET(c(v))$. This walk moves between different sides of the cut $(S, V_T \setminus S)$ precisely when traversing edges from $\partial_{\vec{T}}(S)$. Thus, $|ET(c(v)) \cap \partial_{\vec{T}}(S)|$ is the total number of side changes. As we started from $r \in S$, this number is even if and only if we end up in the same side S, i.e., if and only if $v \in S$. □

Now we are ready to prove Lemma 9.

Lemma 9 *For each directed tree edge $e = (u, v) \in E_{\vec{T}}$, we define $c(e) = c(v)$. Given any vertex subset $S \subseteq V_G$ and edge subset $E' \subseteq E_{G^*}$, the following equality holds.*

$$\partial_{E'}(S) = E' \cap \left(\triangle_{e\in\partial_{\vec{T}}(S), z\in\{x,y\}} hs(z, c(e)) \right),$$

where $\triangle$ represents the symmetric difference of sets.

Proof Let $\mathcal{Q}_x = \{hs(x, c(e)) \mid e \in \partial_{\vec{T}}(S)\}$, $\mathcal{Q}_y = \{hs(y, c(e)) \mid e \in \partial_{\vec{T}}(S)\}$, and $Q = \triangle_{Q' \in \mathcal{Q}_x \cup \mathcal{Q}_y} Q'$. For any $(u, v) \in \partial_{E'}(S)$, exactly one of u and v belongs to S and the other one belongs to $V_G \setminus S$. By symmetry, we assume $u \in S$ and $c(u) < c(v)$ wlog. Lemma 17 implies $|ET(c(u)) \cap \partial_{\vec{T}}(S)|$ is even, and $|ET(c(v)) \cap \partial_{\vec{T}}(S)|$ is odd. It implies that (u, v) lies in the two regions respectively defined as the intersection of an even number of halfspaces in $\mathcal{Q}_y$ and as the intersection of an odd number of halfspaces in $\mathcal{Q}_x$. Since $\mathcal{Q}_x$ and $\mathcal{Q}_y$ are disjoint, (u, v) lies in the region defined as the intersection of an odd number of halfspaces in $\mathcal{Q}_x \cup \mathcal{Q}_y$, i.e., it is contained in Q. Similarly, if (u, v) is not an edge in $\partial_{E'}(S)$, the parities of $|ET(c(u)) \cap \partial_{\vec{T}}(S)|$ and $|ET(c(v)) \cap \partial_{\vec{T}}(S)|$ becomes the same, and thus e lies in the region defined as the intersection of an even number of halfspaces in $\mathcal{Q}_x \cup \mathcal{Q}_y$, and thus not contained in Q. The lemma is proved. □

Next, we focus on the deterministic ϵ-net construction. We first quote a known deterministic construction for axis-aligned rectangles.

Lemma 18 (Mustafa et al. [42]) *Let $\epsilon > 0$. There exists a deterministic polynomial-time algorithm of constructing an ϵ-net of size $O(\log\log N/\epsilon)$ for any N point set and all axis-aligned rectangles.*

As we mentioned in Sect. 4.3, the polynomial of the construction time has a high exponent, and thus we consider a slightly weaker but much faster solution. Let P be any set of points in a 2D-range $R = [a_1, a_2] \times [b_1, b_2]$. For simplicity, we assume that $|P|$ is a power of two in the lemma below, but it is not essential.

Lemma 19 *Let P be any point set in $R = [a_1, a_2] \times [b_1, b_2]$, $0 < \epsilon < 1$, and, $M \in [a_1, a_2]$. There exists a $O(|P| \log |P|)$-time algorithm of computing a subset $P^* \subseteq P$ of size at most $\epsilon|P|$ such that any axis-aligned rectangle X crossing the vertical line $x = M$ and satisfying $|X \cap P| \geq 6/\epsilon$ necessarily contains at least one point in P^*.*

Proof The construction follows the technique by Kulkarni and Govindarajan [35]. For simplicity, we assume that any two points in P have different y-coordinates, $2/\epsilon$ is an integer, and $|P|$ is divisible by $2/\epsilon$. Let Y_P be the sequence of points in P sorted by their y-coordinates. We split Y_P into $\epsilon|P|/2$ subsequences $Y_P^1, Y_P^2, \ldots$ of the length $2/\epsilon$. For each Y_P^i, we define $p_i^- \in Y_P^i$ as the point with the maximum x-coordinate not exceeding M, and $p_i^+ \in Y_P^i$ as the one with the minimum x-coordinate not lower than or equal to M. We construct P^* as the union of $\{p_i^+, p_i^-\}$ for all i.

It is easy to check that the constructed point set P^* satisfies the condition of the lemma: The size of P^* is obviously bounded by $\epsilon|P|$. We define the y-range $[b_1^i, b_2^i]$ of Y_P^i as the minimal interval containing the y-coordinates of all the points in Y_P^i. Consider any rectangle $X = [a_1, a_2] \times [b_1, b_2]$ such that $|X \cap P| \geq 6/\epsilon$. Then there exists at least one subsequence Y_p^i such that at least one point in Y_P^i is contained in $X \cap P$ and the y-range of Y_P^i is covered by $[b_1, b_2]$. For such an i, either p_i^- or p_i^+ must be contained in X. □

We obtain the deterministic algorithm of constructing a $O(\log |P|/|P|)$-net of a constant fraction size for any point set P and axis-aligned rectangles in near linear time.

Lemma 20 *Let P be any set of points in $[a_1, a_2] \times [b_1, b_2]$, and N be any upper bound of $|P|$. There exists a deterministic algorithm* NetFind(N, P) *which constructs a $(12 \log N/|P|)$-net of size at most $|P| \log |P|/(2 \log N)$ for P and all axis-aligned rectangles in $O(|P| \log |P| \log N)$ time.*

Proof We first present the algorithm. It is based on the divide and conquer approach as follows:

1. If $|P| \geq 12 \log N$: find the vertical line $x = M$ bisecting P into two equal-size subsets P_0 and P_1 (with an arbitrary tie-breaking rule for the points on $x = M$). Let $R_0 = [a_1, M] \times [b_1, b_2]$ and $R_1 = [M, a_2] \times [b_1, b_2]$. Then NetFind$(N, P)$ outputs the union of the following four subsets of P:
   - The outputs of NetFind(N, P_0) and NetFind(N, P_1).
   - The point set P^* obtained from P by Lemma 19 for $\epsilon = 1/(2 \log N)$ and $x = M$.
2. If $|P| < 12 \log N$: output the empty set.

Let P' be the output in the run of NetFind(N, P). The proof is based on the induction on the size of P.
(Basis) $|P| < 12 \log N$ holds: Then the case 2 of the algorithm applies. The output size is trivially bounded by $|P| \log |P|/(2 \log N) \geq 0$. Since $|P| < 12 \log N$ holds, there is no axis-aligned rectangle containing more than or equal to $12 \log N$ points. Hence the constructed output (i.e., the empty set) is a $(12 \log N/|P|)$-net.
(Inductive Step): Let P be the set of points such that $|P| \geq 12 \log N$ holds. Consider any axis-aligned rectangle X such that $|X \cap P| \geq 12 \log N$ holds. We show that X necessarily contains a point in P'. By the induction hypothesis, both NetFind(N, P_0) and NetFind(N, P_1) correctly computes a $(12 \log N/|P_0|)$-net and a $(12 \log N/|P_1|)$-net. Hence if X is contained either R_0 or R_1, X necessarily contains a point in P'. If X intersects $x = M$, it also intersects P' by Lemma 19. Hence the constructed P' is a $(12 \log N/|P_1|)$-net. We bound the output size $|P'|$. By the induction hypothsis, the output sizes of NetFind(N, P_0) and NetFind(N, P_1) are respectively bounded by $|P| \log(|P|/2)/(4 \log N)$. The size of P^*

is bounded by $|P|/(2\log N)$. Summing up them, we obtain

$$\begin{aligned}|P'| &\le 2\cdot\frac{|P|\log\frac{|P|}{2}}{4\log N}+\frac{|P|}{2\log N}\\ &\le \frac{|P|(\log|P|-1)}{2\log N}+\frac{|P|}{2\log N}\\ &\le \frac{|P|\log|P|}{2\log N}.\end{aligned}$$

It is easy to bound the running time because the total running time of all recursive calls at the same depth is bounded by $O(|P|\log N)$. The lemma is proved. □

Lemma 10 is obviously obtained from Lemmas 18 and 20. We finally show the main lemma of this section.

Lemma 11 *There exists two deterministic algorithms respectively constructing a $(\mathcal{S}_{f,T},k)$-good hierarchy with the following performance guarantees:*

- $k=O(f^2\log n)$ *and the construction time is* $\tilde{O}(m)$.
- $k=O(f^2\log\log n)$ *and the construction time is* $\mathrm{poly}(m)$.

Proof We only show that first construction, but the second construction is proved in the same way. Consider the construction of E_{i+1} from E_i. The algorithm first maps all the edges in E_i into the space $[1,2n-2]^2$. Applying the algorithm of Lemma 20 to E_i with $P=E_i$ and $N=|E_i|$, we obtain an $(6\log|E_i|/|E_i|)$-net E_{i+1} of a constant fraction size for E_i and all axis-aligned rectangles. As mentioned in Sect. 4.3, E_{i+1} works as a $(6(2f+1)^2\log n/|E_i|)$-net for $\mathcal{H}_{2f}$ and thus it satisfies the condition of $(\mathcal{S}_{f,T},6(2f+1)^2\log n)$-good edge hierarchy. Since E_{i+1} is a subset of E_i with a constant fraction size, the depth h of hierarchy is bounded by $O(\log n)$. The construction time of E_{i+1} from E_i follows the running time of the algorithm of Lemma 20, i.e., it takes $\tilde{O}(m)$ time. Hence the total running time is $\tilde{O}(m)$. □

7.6 Fast query processing

We construct a refined query processing algorithm for our tree edge f-FTC labeling scheme. Let $G^*=G-E_T$ for short. Throughout this section, we assume that the tree edge f-FTC labeling scheme is implemented with any adaptive $\mathcal{S}_{f,T}$-outdetect labeling scheme $(L^{\mathsf{out}}_{G^*},D^{\mathsf{out}})$ of $\tilde{O}(f^b)$-bit label size which admits $\tilde{O}(|\partial_T(S)|^c)$ decoding time for a given query $S\subseteq V_{G^*}$, as explained in Sect. 6. Similarly as the original one, the refined algorithm also iteratively merges the vertices of the component graph $T/\mathcal{C}(F)$. It manages a collection $\mathcal{X}$ of disjoint subsets of $\mathcal{C}(F)$ throughout the procedure. A subset $S\subseteq\mathcal{C}(F)$ in $\mathcal{X}$ is called a *component fragment* (note that S is not a subset of V_G). For any component fragment $S\subseteq\mathcal{C}(F)$, we define $V(S)\subseteq V_G$ as $V(S)=\bigcup_{C\in S}C$. As a loop invariant, the algorithm guarantees that $V(S)$ for any component fragment $S\in\mathcal{X}$ induces a connected subgraph of $G-F$. Each component fragment $S\subseteq\mathcal{C}(F)$ is maintained as the triple $(S,\partial_T(V(S)),L^{\mathsf{out}}_{G^*}(V(S)))$. We denote this triple associated with $S\subseteq\mathcal{C}(F)$ by $\tau(S)$, and abuse the notation $\mathcal{X}$ as the set of associated triples. The whole structure of the algorithm is stated below:

1. Initially, we set $\mathcal{X}=\{\tau(\{C\})\mid C\in\mathcal{C}(F)\}$.
2. In each iteration, we pick up $\tau(S)$ such that $|\partial_T(V(S))|$ is the smallest, and find an outgoing edge of S by decoding $L^{\mathsf{out}}_{G^*}(V(S))$, which is obtained from $L^{\mathsf{out}}_{G^*}(V(S))$ stored in $\tau(S)$.
3. If no outgoing edge of S is found, we remove $\tau(S)$ from $\mathcal{X}$ and go to the next iteration. Otherwise, let S' be the component fragment the outgoing edge from S reaches.
4. If S and S' contains s and t respectively, the procedure terminates with returning true. Otherwise, the algorithm deletes $\tau(S)$ and $\tau(S')$ from $\mathcal{X}$, and newly insert the entry of $\tau(S'')$ for $S''=S\cup S'$. The entry $\tau(S'')$ is computed as $\tau(S'')=(S'',(\partial_T(V(S))\cup\partial_T(V(S')))\backslash(\partial_T(V(S))\cap\partial_T(V(S'))),L^{\mathsf{out}}_{G^*}(V(S))+L^{\mathsf{out}}_{G^*}(V(S')))$. After the insertion, the algorithm proceeds to the next iteration unless $|\mathcal{X}|=1$ holds. If $|\mathcal{X}|=1$ holds, the algorithm terminates with returning false (this case occurs only when the component fragment containing s or t is discarded).

To implement the algorithm above efficiently, we manage $\mathcal{X}$ by the heap which supports $O(\log|\mathcal{X}|)$-time insert, delete, and search of the element having the minimum cutset. Each cutset associated with an element in $\mathcal{X}$ is stored as the bit vector of length $|F|$ and the additional integer value representing the size of the stored cutset. This data structure obviously supports union and intersection in $O(|F|)$ time, as well as getting the cutset size in $O(1)$ time. Each fragment S associated with a triple in $\mathcal{X}$ is managed by any disjoint-set data structure (e.g., union-find) over $\mathcal{C}(F)$. Combining this structure with Proposition 15, one can determine the fragment $S\in\mathcal{X}$ containing a given vertex $u\in V_G$ from $L^{\mathsf{anc}}_T(u)$ in $O(\log|F|)$ time (i.e., identify $C\in\mathcal{C}(F)$ containing u first, and then identify $S\in\mathcal{X}$ containing C). With support of all the data structures above, we can implement one iteration of the refined procedure in $\tilde{O}(|F|^b+|\partial_T(V(S))|^c)$ time. The initialization of $\mathcal{X}$ is implemented in $\tilde{O}(|F|^{b+1})$ time. We show that the refined algorithm runs in $\tilde{O}(|F|^c)$ time in total.

Lemma 13 *Assume that there exists a $\mathcal{S}_{f,T}$-outdetect labeling scheme of label size $\alpha=\tilde{O}(f^b)$ and decoding time $\beta=\tilde{O}(f^c)$. Then there exists a f-FTC labeling scheme of $O(\alpha+\log n)$-bit label size and $\tilde{O}(|F|^{b+1}+|F|^c)$ decoding time. The resultant f-FTC labeling scheme is determinis-*

tic if the corresponding $\mathcal{S}_{f,T}$-outdetect labeling scheme is deterministic.

Proof Consider the refined query processing algorithm above. We denote by $\mathcal{X}_i$ the set $\mathcal{X}$ at the beginning of the i-th iteration, and let $\mathcal{Y}_i = \{S \mid \tau(S) \in \mathcal{X}_i\}$. Since $\mathcal{Y}_i$ is a disjoint collection of component fragments and each $S \in \mathcal{Y}_i$ satisfies $\partial_T(V(S)) \subseteq F$, we have $\sum_{S\in\mathcal{Y}_i} |\partial_T(S)| \leq 2|F|$. Let $S_1, S_2 \ldots, S_x$ be the component fragments chosen in each iteration ($x \leq |F|$). Since the algorithm chooses S_i minimizing $|\partial_T(V(S_i))|$, we have $|\partial_T(V(S_i))| \leq 2|F|/|\mathcal{Y}_i|$. As discussed in this section, the detection of an outgoing edge of S_i takes $\tilde{O}(|\partial_T(V(S_i))|^c)$ time. Since exactly one component in $\mathcal{Y}_i$ is merged or discarded in the i-th iteration, we have $|\mathcal{Y}_{i+1}| = |\mathcal{Y}_i| - 1 = |F| + 1 - i$. The computation time excluding that for the outgoing edge detection is $O(|F|^b \log n)$ per one iteration. The total running time is bounded as follows:

$$\begin{aligned}\sum_{1\leq i\leq |F|} \tilde{O}\left(\left(\frac{|F|}{|F|+1-i}\right)^c + |F|^b\right) &\leq \tilde{O}(|F|^c) \cdot \sum_{1\leq i\leq |F|} \frac{1}{i^c} + \tilde{O}(|F|^{b+1}) \\ &\leq \tilde{O}(|F|^c + |F|^{b+1}).\end{aligned}$$

□

8 Distributed construction

In this section, we explain how our deterministic f-FCT labeling scheme is constructed in the standard CONGEST model, i.e., the round-based synchronous system with the $O(\log n)$-bit message size bound. For the input graph G, we fix T as its BFS tree for an arbitrary chosen root. Since the corresponding auxiliary graph G' is easily simulated on the top of the original graph, the behavior of the proposed algorithm is described as the message passing on G'. The spanning tree of G' transformed from T is denoted by T'.

8.1 Construction of ancestry labels

The construction by Kannan, Naor, and Rudich is to assign each node and edge with the pair of its pre-order and post-order in the Euler-tour traversal of T' starting from the root. For any two labels (a, b) and (c, d) assigned to u and v, u is an ancestor of v if and only if the interval $[a, b]$ contains $[c, d]$. In the following argument, we focus on the computation of the pre-orders and post-orders of edges in T'. The computation of vertex orders is processed similarly. For every edge e, the algorithm computes the number of edges in the subtree $T'(e)$, which is implemented by the subtree-sum aggregation over T', taking $O(D)$ rounds. The twice of the computed value, denoted by $\mathsf{gap}(e)$, is equal to the gap between the pre-order and the post-order of e. Then the algorithm determines the pre-orders and post-orders of all tree edges from the root side. Assume that we have fixed the orders (a, b) of an edge e, and let $e_1, e_2, \ldots, e_j$ be the set of children edges of e. Then the pre-order of e_1 is obviously $a+1$, and its post-order is $a+2+\mathsf{gap}(e_1)$. The orders of $e_2, e_3, e_4, \ldots, e_j$ are decided similarly.

8.2 Construction of outdetect labels

Suppose that an $(\mathcal{S}_{f,T'}, O(f^2 \log n))$-good hierarchy has already been distributedly computed, meaning that each vertex v knows all of its incident edges in E_i, for every $0 \leq i \leq h$. Then, it is easy to verify that v can *locally* compute its $\mathcal{S}_{f,T'}$-outdetect label $L^{out}_{G'-E_{T'}}(v)$, which consists of $\tilde{O}(f^2)$ bits. This is done using the $O(f^2 \log n)$-threshold outdetect labeling scheme of Proposition 7 and Lemma 8. To compute the tree edge f-FTC labels of the edges in T', we execute subtree-sum aggregation of the $\mathcal{S}_{f,T'}$-outdetect labels of the vertices, which takes $\tilde{O}(D + f^2)$ rounds using standard pipeline techniques. So, for each edge e of T', the root of $T'(e)$ holds $L^{out}_{G'-E_{T'}}(T'(e))$. Concatenating the ancestry labels of e's to this information yields the tree edge f-FTC label of e. We therefore focus henceforth on the distributed construction of a good hierarchy.

8.3 Construction of a $(\mathcal{S}_{f,T'}, O(f^2 \log n))$-good hierarchy

To construct a good hierarchy, it suffices to implement NetFind in the CONGEST model. As a preprocessing, the algorithm computes the coordinates of all non-tree edges. They are computed in the same way as the construction of ancestry labels. We assume that both of the endpoints of any non-tree edge e know the x- and y-coordinates of e. We refer to the vertex corresponding to the x-coordinate (resp. y-coordinate) of a non-tree edge e as the x-side (resp. y-side) endpoint of e. Let $\mathsf{seq}_x(P)$ (resp. $\mathsf{seq}_y(P)$) be the minimal consecutive subsequence of the Euler tour of T' containing all x-side (resp. y-side) endpoints of edges in P. In the following argument, we often abuse $\mathsf{seq}_x(P)$ and $\mathsf{seq}_y(P)$ as the sets of vertices in the sequences. The CONGEST version of $\mathsf{NetFind}(N, P)$ is executed in the subgraph induced by $\mathsf{seq}_x(P)$.

Let $m' = |E_{G'}| - |E_{T'}|$, and j^* be the smallest integer such that $2^{j^*} > \sqrt{m'/D}$ holds. Our implementation processes the invocations of NetFind at the recursion level j^* in parallel. It is easy to check that the following two conditions are satisfied:

- For any call of $\mathsf{NetFind}(N, P)$ at the recursion level $j > j^*$, we have $|P| = O(\sqrt{m'D})$. In addition, since the diameter of T' is $O(D)$, the diameter of the subtree of T' induced by $\mathsf{seq}_x(P)$ is also $O(D)$.
- For any two invocations of $\mathsf{NetFind}(N, P_1)$ and $\mathsf{NetFind}(N, P_2)$ at the same recursion level, $\mathsf{seq}_x(P_1)$ and $\mathsf{seq}_x(P_2)$ induces two edge-disjoint consecutive subtours of the Euler tour of T' under the treatment of two edges (u, v) and (v, u) as distinct ones.

For the invocation of $\mathsf{NetFind}(N, P)$, every node in $\mathsf{seq}_x(P)$ can aggregate whole information of P in $O(\sqrt{m'D} + D) = O(\sqrt{m'D})$ rounds. Hence each node can execute the centralized version of NetFind locally. Due to the second condition above, the aggregation tasks for all invocations at the recursion level j^* are efficiently processed in parallel: each edge $e \in E_{T'}$ is contained at most two induced subtrees, and thus the total running time is still bounded by $O(\sqrt{m'D})$. Note that we do not have to handle the recursion level more than j^* distributedly because $\mathsf{NetFind}(N, P)$ at the recursion level j^* is processed in the centralized manner. Consequently, the total running time for the recursion level $j > j^*$ is $O(\sqrt{m'D})$.

For the recursion level $j \leq j^*$, the total number of invocations is $O(\sqrt{m'/D})$, and thus the algorithm sequentially processes each invocation. The main body of $\mathsf{NetFind}(N, P)$ is the construction of the point set P^* shown in Lemma 19. To identify the sets $Y_P^1, Y_P^2, \ldots$, it suffices to compute the order of each non-tree edge in the sequence Y_P, which is also computed in $O(D)$ rounds similarly to the construction of ancestry labels. Only the difference is that for every tree edge $e \in E_{T'}$, the algorithm computes the number of non-tree edges in P whose y-side endpoints lie in $T'(e)$. Next, the algorithm finds two non-tree edges p_i^- and p_i^+ for each Y_P^i. Each subgraph induced by $\mathsf{seq}_y(Y_P^i)$ independently finds the non-tree edge in Y_P^i with the maximum x-coordinate not exceeding M, and that with the minimum x-coordinate not lower than M. Each of them is found with a single-shot aggregation in the subgraph, whose running time is obviously $O(D)$. Hence the total running time of $\mathsf{NetFind}(N, P)$ for the recursion level $j \leq j^*$ is $O(\sqrt{m'/D}D) = \tilde{O}(\sqrt{m'D})$. It concludes that our distributed implementation of NetFind takes $O(\sqrt{m'D})$ rounds. To construct whole $(\mathcal{S}_{f,T'}, cf^2 \log n)$-good hierarchy, $O(\log n)$ repetition of invoking NetFind suffices. Consequently, we obtain the following lemma

Lemma 21 *Let T' be any BFS tree of the auxiliary graph G' of the input graph G. There exists a deterministic CONGEST algorithm of constructing a $(\mathcal{S}_{f,T'}, O(f^2 \log n))$- good hierarchy in $\tilde{O}(\sqrt{mD})$ rounds.*

This lemma obviously deduces the theorem below:

Theorem 22 *There exists a deterministic CONGEST algorithm of constructing f-FTC labels for all vertices and edges in $\tilde{O}(\sqrt{mD} + f^2)$ rounds.*

9 Concluding remarks

This paper presented a new deterministic f-FTC labeling scheme which attains $O(f^2\text{polylog}(n))$-bit label size, polynomial-time construction, and $\tilde{O}(\text{poly}(|F|))$-time query processing time for a given faulty edge set F. This is the first deterministic and polynomial-time f-FTC labeling scheme with a non-trivial label size. The scheme is developed on the top of a general framework, and only by the modification of graph sparsification, we can also obtain a randomized f-FTC labeling scheme which is competitive to the original Dory-Parter scheme and attains an adaptive query processing time. The key technical ingredient is a new deterministic $\mathcal{S}$-outdetect labeling scheme based on error-correcting codes. From the authors' perspective, our results pose a few promising future research directions. We conclude this paper with summarizing them.

- Is it possible to develop a deterministic algorithm yielding better edge hierarchies, i.e., the hierarchy such that for any S there exists i satisfying $0 < |\partial_{E_i}(S)| = o(f^2 \log n)$? Our framework automatically deduces a deterministic f-FTC labeling scheme with an improved label size if such an algorithm is found.
- With respect to the construction time in the CONGEST model, our deterministic scheme still has a large gap with the known randomized construction, only taking $\tilde{O}(f+D)$ rounds. Is it possible to obtain the deterministic f-FTC labeling scheme of $O(\text{poly}(f, \log n))$-bit label size which is implemented in the CONGEST model with $\tilde{O}(\text{poly}(f) + D)$ or $\tilde{O}(\text{poly}(f) \cdot D)$ rounds?
- Can we obtain any non-trivial lower bound for the label size of f-FTC labeling schemes with full query support? It seems plausible that the $\Omega(f)$-bit lower bound holds, but no promising way of proving this is found so far.
- Can our technique be exported to other applications of $\mathcal{S}$-outdetect labeling schemes, such as centralized fault-tolerant connectivity oracles [20], distributed computation of sparse spanning subgraphs [24, 27, 37, 43] or small cut detection [45, 49], and dynamic algorithms [25, 36], for obtaining any improved result?

A Randomized construction of edge set hierarchy

Comparing the quality of labeling schemes with the label size, decoding time, and construction time, our deterministic construction is competitive but certainly worse than the known randomized scheme. However, most of high costs incurred by our construction is derived from the construction of edge set hierarchies. As mentioned in Sect. 4.3, a simple edge sub-sampling strategy suffices to construct good hierarchies.

Proposition 23 *Let $E_{G-E_T} = E_0 \supseteq E_1 \supseteq E_2 \supseteq, \ldots, \supseteq E_h = \emptyset$ be the edge set hierarchy such that E_{i+1} is constructed by sampling each edge in E_i independently with probability $1/2$ if $|E_i| > 5f \log n$, or $E_{i+1} = \emptyset$ otherwise. Then with probability $1 - 1/n^{O(1)}$,this hierarchy is $(\mathcal{S}_{f,T}, 5f \log n)$-good.*

Proof Consider the construction of E_{i+1} from E_i. If $|E_i| \leq 5f \log n$, E_{i+1} obviously satisfies the two conditions of Definition 1, and thus consider only the case of $|E_i| > 5f \log n$. We show that with high probability, $|\partial_{E_{i+1}}(S)| > 0$ holds for any $S \in \mathcal{S}_{f,T}$ satisfying $|\partial_{E_i}(S)| > 5f \log n$. The probability that no edge in $\partial_{E_i}(S)$ is added to E_{i+1} is at most $(1/2)^{5f \log n} = 1/n^{5f}$. Since the cardinality of $\mathcal{S}_{f,T}$ is bounded by $\sum_{i \leq f} \binom{|E_T|}{i} = O(n^f)$, by the union bound argument, we conclude that $|\partial_{E_{i+1}}(S)| > 0$ holds for any $S \in \mathcal{S}_{f,T}$ satisfying $|\partial_{E_i}(S)| > 5f \log n$ with probability $1 - O(1/n^5)$. That is, the probability that E_{i+1} does not satisfies the second condition of Definition 1 is $O(1/n^4)$. The first condition is satisfied with high probability because one can show $|E_{i+1}| \leq 3|E_i|/4$ with probability $1 - O(1/n^2)$ by the straightforward application of Chernoff bound. Applying the union bound again on the failing events of the first and second conditions for all i, we obtain the proposition. □

B Adaptive decoding of deterministic $\mathcal{S}_{f,T}$-outdetect labeling scheme based on the Reed–Solomon code

In this Appendix, we show that our $\mathcal{S}_{f,T}$-outdetect labeling scheme attains the adaptiveness without any modification or transformation. The key idea is a nice property of Reed-Solomon Code: We define $L^{\mathsf{RS}(k)}_{H,k'}$ for $k' \leq k$ as the labeling function which assign $v \in V_H$ with the prefix of $L^{\mathsf{RS}(k)}_H(v)$ up to the k'-th element. Then the following proposition holds:

Proposition 24 *For any $k' \leq k$, $L^{\mathsf{RS}(k)}_{H,k'} = L^{\mathsf{RS}(k')}_H$ holds.*

Proof The proof trivially follows the definition of the parity check matrix C_{2k}. Let c_{ij} be the (i, j)-element of C_{2k} $(0 \leq i \leq |\mathcal{E}|, 0 \leq j \leq 2k - 1)$ and ω be a primitive element of $\mathbb{F}$. Then the each element of C_{2k} is defined as $c_{ij} = \omega^{ij}$. That is, the submatrix formed by the first $2k'$ columns of C_{2k} is is equal to $C_{2k'}$. □

Intuitively, the proposition above implies that the $O(k' \log n)$-bit prefixes of the labels assigned by our k-threshold outdetect labeling scheme also work as the labels of the k'-threshold outdetect labeling scheme. It is also possible to make the sparsification hierarchy adaptive because our construction does not use the upper bound f at all, i.e., the construction is universal for every f: We explained in the previous section that E_{i+1} is constructed in the way that it becomes the hitting set of $\mathcal{Z}_{i,f,cf^2 \log n}$ (where c is a hidden constant). In reality, it also becomes the hitting set of $\mathcal{Z}_{i,f',c(f')^2}$ for any $f' > 0$. Hence it is guaranteed that for any $S \in \mathcal{S}_{|\partial_T(S)|,T}$ there exists an index i such that $0 < |\partial_{G_i}(S)| \leq c|\partial_T(S)|^2 \log n$ holds (in the deterministic case).

Acknowledgements The authors express their gratitude to the anonymous reviewers for their careful reading and many valuable comments. This work was supported by JSPS KAKENHI Grant Nos. 21H05854, 22H03569, 23H04385, 20H04139, and 19H04085.

Author Contributions All authors are equally contributed to this work.

Funding Open Access funding provided by Osaka University.

Data availability No datasets were generated or analysed during the current study.

Declarations

Conflict of interest The authors declare no conflict of interest.

References

1. Abraham, I., Chechik, S., Gavoille, C.: Fully dynamic approximate distance oracles for planar graphs via forbidden-set distance labels. In: Proceedings of the 44th Annual ACM Symposium on Theory of Computing (STOC), pp. 1199–1218 (2012)
2. Abraham, I., Chechik, S., Gavoille, C., Peleg, D.: Forbidden-set distance labels for graphs of bounded doubling dimension. ACM Trans. Algorithms **12**(2), 1–17 (2016)

3. Ahn, K.J., Guha, S., McGregor, A.: Analyzing graph structure via linear measurements. In: Proceedings of the Twenty-Third Annual ACM-SIAM Symposium on Discrete Algorithms (SODA), pp. 459–467. SIAM (2012)
4. Ahn, K.J., Guha, S., McGregor, A.: Graph sketches: sparsification, spanners, and subgraphs. In: Proceedings of the 31st ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems (PODS), pp. 5–14 (2012)
5. Bilò, D., Choudhary, K., Gualà, L., Leucci, S., Parter, M., Proietti, G.: Efficient oracles and routing schemes for replacement paths. In: Proceedings of 35th Symposium on Theoretical Aspects of Computer Science (STACS), volume 96 of Leibniz International Proceedings in Informatics (LIPIcs), pp. 13:1–13:15 (2018)
6. Bar-Natan, A., Charalampopoulos, P., Gawrychowski, P., Mozes, S., Weimann, O.: Fault-tolerant distance labeling for planar graphs. In: Proceedings of International Colloquium on Structural Information and Communication Complexity (SIROCCO), volume 12810 of Lecture Notes in Computer Science, pp. 315–333 (2021)
7. Bilò, D., Gualà, L., Leucci, S., Proietti, G.: Compact and fast sensitivity oracles for single-source distances. In: Sankowski, P., Zaroliagis, C.D. (eds.) Proceedings of 24th Annual European Symposium on Algorithms (ESA), volume 57 of Leibniz International Proceedings in Informatics (LIPIcs), pp. 13:1–13:14 (2016)
8. Bernstein, A., Karger, D.: A nearly optimal oracle for avoiding failed vertices and edges. In: Proceedings of the 41st Annual ACM Symposium on Theory of Computing (STOC), pp. 101–110 (2009)
9. Baswana, S., Khanna, N.: Approximate shortest paths avoiding a failed vertex: near optimal data structures for undirected unweighted graphs. Algorithmica **66**(1), 18–50 (2013)
10. Chechik, S., Cohen, S., Fiat, A., Kaplan, H.: $(1+\epsilon)$-Approximate f-sensitive distance oracles. In: Proceedings of the 2017 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA), pp. 1479–1496 (2017)
11. Chuzhoy, J., Gao, Y., Li, J., Nanongkai, D., Peng, R., Saranurak, T.: A deterministic algorithm for balanced cut with applications to dynamic connectivity, flows, and beyond. In: 61st IEEE Annual Symposium on Foundations of Computer Science (FOCS), pp. 1158–1167. IEEE (2020)
12. Chechik, S.: Fault-tolerant compact routing schemes for general graphs. In: 38th International Colloquium on Automata, Languages, and Programming, (ICALP), pp. 101–112 (2011)
13. Chechik, S., Langberg, M., Peleg, D., Roditty, L.: f-sensitivity distance oracles and routing schemes. Algorithmica **63**(4), 861–882 (2012)
14. Chazelle, B., Matoušek, J.: On linear-time deterministic algorithms for optimization problems in fixed dimension. J. Algorithms **21**(3), 579–597 (1996)
15. Courcelle, B., Twigg, A.: Compact forbidden-set routing. In: Proceedings of the 24th Annual Conference on Theoretical Aspects of Computer Science, STACS'07, pp. 37–48 (2007)
16. Courcelle, B., Twigg, A.: Constrained-path labellings on graphs of bounded clique-width. Theory Comput. Syst. **47**(2), 531–567 (2010)
17. Dodis, Y., Ostrovsky, R., Reyzin, L., Smith, A.: Fuzzy extractors: how to generate strong keys from biometrics and other noisy data. SIAM J. Comput. **38**(1), 97–139 (2008)
18. Duan, R, Pettie, S: Dual-failure distance and connectivity oracles. In: Proceedings of the 20th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA), pp. 506–515 (2009)
19. Duan, R., Pettie, S.: Connectivity oracles for failure prone graphs. In: Proceedings of the 42nd ACM Symposium on Theory of Computing (STOC), pp. 465–474 (2010)
20. Duan, R., Pettie, S.: Connectivity oracles for graphs subject to vertex failures. SIAM J. Comput. **49**(6), 1363–1396 (2020)
21. Dory, M., Parter, M.: Fault-tolerant labeling and compact routing schemes. In: Proceedings of the 2021 ACM Symposium on Principles of Distributed Computing (PODC), pp. 445–455 (2021)
22. Demetrescu, C., Thorup, M.: Oracles for distances avoiding a link-failure. In: Proceedings of the 13th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA), pp. 838–843 (2002)
23. Feigenbaum, J., Karger, D.R., Mirrokni, V.S., Sami, R.: Subjective-cost policy routing. Theor. Comput. Sci. **378**(2), 175–189 (2007)
24. Ghaffari, M., Kuhn, F.: Distributed MST and broadcast with fewer messages, and faster gossiping. In: Proceedings of 32nd International Symposium on Distributed Computing (DISC), volume 121 of LIPIcs, pp. 30:1–30:12 (2018)
25. Gibb, D., Kapron, B.M., King, V., Thorn, N.: Dynamic graph connectivity with improved worst case update time and sublinear space. CoRR, arXiv:1509.06464 (2015)
26. Ghaffari, M., Parter, M.: Mst in log-star rounds of congested clique. In: Proceedings of the 2016 ACM Symposium on Principles of Distributed Computing (PODC), pp. 19–28 (2016)
27. Gmyr, R., Pandurangan, G.: Time-message trade-offs in distributed algorithms. In: Proceedings of 32nd International Symposium on Distributed Computing (DISC), volume 121 of LIPIcs, pp. 32:1–32:18 (2018)
28. Gu, Y., Ren, H.: Constructing a distance sensitivity oracle in $o(n^{2.5794}m)$ time. In: Bansal, N., Merelli, E., Worrell, J. (eds.) 48th International Colloquium on Automata, Languages, and Programming, (ICALP), volume 198 of LIPIcs, pp. 76:1–76:20 (2021)
29. Grandoni, F., Williams, V.V.: Faster replacement paths and distance sensitivity oracles. ACM Trans. Algorithms **16**(1), 1–25 (2019)
30. Holm, J., de Lichtenberg, K., Thorup, M.: Poly-logarithmic deterministic fully-dynamic algorithms for connectivity, minimum spanning tree, 2-edge, and biconnectivity. J. ACM **48**(4), 723–760 (2001)
31. Henzinger, M.R., King, V.: Randomized fully dynamic graph algorithms with polylogarithmic time per operation. J. ACM **46**(4), 502–516 (1999)
32. Hegeman, J.W., Pandurangan, G., Pemmaraju, S.V., Sardeshmukh, V.B., Scquizzato, M.: Toward optimal bounds in the congested clique: graph connectivity and mst. In: Proceedings of the 2015 ACM Symposium on Principles of Distributed Computing (PODC), pp. 91–100 (2015)
33. Haussler, D., Welzl, E.: Epsilon-nets and simplex range queries. Discrete Comput. Geom. **2**, 127–151 (1987)
34. Jurdzinski, T., Nowicki, K.: MST in $O(1)$ rounds of the congested clique. In: Proceedings of the 29th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA), pp. 2620–2632 (2018)
35. Kulkarni, J., Govindarajan, S.: New epsilon-net constructions. In: 22nd Annual Canadian Conference on Computational Geometry (CCCG), pp. 159–162 (2010)
36. Kapron, B.M., King, V., Mountjoy, B.: Dynamic graph connectivity in polylogarithmic worst case time. In: Proceedings of the 24th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA), pp. 1131–1142 (2013)
37. King, V., Kutten, S., Thorup, M.: Construction and impromptu repair of an mst in a distributed network with $o(m)$ communication. In: Proceedings of the ACM Symposium on Principles of Distributed Computing (PODC), pp. 71–80 (2015)
38. Kapralov, M., Lee, Y.T., Musco, C., Musco, C., Sidford, A.: Single pass spectral sparsification in dynamic streams. In: Proceedings of 55th IEEE Annual Symposium on Foundations of Computer Science (FOCS), pp. 561–570 (2014)
39. Kannan, S., Naor, M., Rudich, S.: Implicit representation of graphs. SIAM J. Discrete Math. **5**(4), 596–603 (1992)
40. Kapralov, M., Woodruff, D.: Spanners and sparsifiers in dynamic streams. In: Proceedings of the 2014 ACM Symposium on Principles of Distributed Computing (PODC), pp. 272–281 (2014)

41. Matoušek, J.: Derandomization in computational geometry. J. Algorithms **20**(3), 545–580 (1996)
42. Mustafa, N.H., Dutta, K., Ghosh, A.: A simple proof of optimal epsilon nets. Combinatorica **38**(5), 1269–1277 (2018)
43. Mashreghi, A., King, V.: Broadcast and minimum spanning tree with $o(m)$ messages in the asynchronous CONGEST model. Distrib. Comput. **34**(4), 283–299 (2021)
44. Patrascu, M., Demaine, E.D.: Logarithmic lower bounds in the cell-probe model. SIAM J. Comput. **35**(4), 932–963 (2006)
45. Parter, M., Petruschka, A.: Near-optimal distributed computation of small vertex cuts. In: 36th International Symposium on Distributed Computing (DISC), vol. 246, pp. 31:1–31:21 (2022)
46. Parter, M., Petruschka, A.: Õptimal dual vertex failure connectivity labels. In: 36th International Symposium on Distributed Computing (DISC 2022), pp. 32:1–32:19 (2022)
47. Parter, M., Petruschka, A., Pettie, S.: Connectivity labeling and routing with multiple vertex failures. In: 56th Annual ACM Symposium on Theory of Computing (STOC), pp. 823–834 (2024)
48. Patrascu, M., Thorup, M.: Planning for fast connectivity updates. In: Proceedings of 48th Annual IEEE Symposium on Foundations of Computer Science, pp. 263–271 (2007)
49. Pritchard, D., Thurimella, R.: Fast computation of small cuts via cycle space sampling. ACM Trans. Algorithms **7**(4), 1–30 (2011)
50. Rajan, V.: Space efficient edge-fault tolerant routing. In: D'Souza, D., Kavitha, T., Radhakrishnan, J. (eds.) Proceedings of IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, (FSTTCS), volume 18 of Leibniz International Proceedings in Informatics (LIPIcs), pp. 350–361 (2012)
51. Thorup, M.: Near-optimal fully-dynamic graph connectivity. In: The 32nd Annual ACM Symposium on Theory of Computing (STOC), pp. 343–350 (2000)
52. Wulff-Nilsen, C.: Faster deterministic fully-dynamic graph connectivity. In: The 2013 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA), pp. 1757–1769