Distributed Computing (2023) 36:475–499
https://doi.org/10.1007/s00446-023-00454-0

Almost universally optimal distributed Laplacian solvers via low-congestion shortcuts

Ioannis Anagnostides[1] · Christoph Lenzen[2] · Bernhard Haeupler[3] · Goran Zuzic[3] · Themis Gouleakis[4]

Received: 15 January 2023 / Accepted: 25 June 2023 / Published online: 31 July 2023

Abstract

In this paper, we refine the (almost) *existentially optimal* distributed Laplacian solver of Forster, Goranci, Liu, Peng, Sun, and Ye (FOCS '21) into an (almost) *universally optimal* distributed Laplacian solver. Specifically, when the topology is known (i.e., the Supported-CONGEST model), we show that any Laplacian system on an n-node graph with *shortcut quality* $\mathrm{SQ}(G)$ can be solved after $n^{o(1)}\mathrm{SQ}(G)\log(1/\epsilon)$ rounds, where $\epsilon > 0$ is the required accuracy. This almost matches our lower bound that guarantees that any correct algorithm on G requires $\widetilde{\Omega}(\mathrm{SQ}(G))$ rounds, even for a crude solution with $\epsilon \le 1/2$. Several important implications hold in the unknown-topology (i.e., standard CONGEST) case: for excluded-minor graphs we get an almost universally optimal algorithm that terminates in $D \cdot n^{o(1)}\log(1/\epsilon)$ rounds, where D is the hop-diameter of the network; as well as $n^{o(1)}\log(1/\epsilon)$-round algorithms for the case of $\mathrm{SQ}(G) \le n^{o(1)}$, which holds for most networks of interest. Moreover, following a recent line of work in distributed algorithms, we consider a hybrid communication model which enhances CONGEST with limited global power in the form of the node-capacitated clique model. In this model, we show the existence of a Laplacian solver with round complexity $n^{o(1)}\log(1/\epsilon)$. The unifying thread of these results, and our main technical contribution, is the development of near-optimal algorithms for a novel ρ-*congested* generalization of the standard *part-wise aggregation* problem, which could be of independent interest.

Keywords Distributed algorithms · Laplacian solvers · Low-congestion shortcuts · Universal optimality

1 Introduction

The *Laplacian paradigm* has emerged as one of the cornerstones of modern algorithmic graph theory. Integrating techniques from combinatorial optimization with powerful machinery from numerical linear algebra, it was originally pioneered by Spielman and Teng [1] who established the first nearly-linear time solvers for a (linear) Laplacian system. Thereafter, there has been a considerable amount of interest in providing simpler and more efficient solvers [2–4]. Indeed, this framework has led to some state of the art algorithms for a wide range of fundamental graph-theoretic problems; e.g., see [5–10], and references therein. In the distributed setting, a major breakthrough was recently made by Forster et al. [11]. In particular, the authors developed a distributed algorithm

The author ordering was randamized using https://www.aeaweb.org/journals/policies/random-author-order/generator. It is requested that citations of this work list the authors seperated by ⓡ instead of commas: Anagnostides ⓡ Lenzen ⓡ Haeupler ⓡ Zuzic ⓡ Gouleakis.

✉ Ioannis Anagnostides
ianagnos@cs.cmu.edu

Christoph Lenzen
lenzen@cispa.de

Bernhard Haeupler
haeuplb@ethz.ch

Goran Zuzic
goran.zuzic@inf.ethz.ch

Themis Gouleakis
tgoule@nus.edu.sg

[1] Department of Computer Science, Carnegie Mellon University, 5000 Forbes Avenue, Pittsburgh, PA 15213, USA

[2] CISPA Helmholtz Center for Information Security, Stuhlsatzenhaus 5, 66123 Saarbrücken, Germany

[3] Department of Computer Science, ETH Zürich, Rämistrasse 101, 8092 Zürich, Switzerland

[4] Department of Computer Science, National University of Singapore, 21 Lower Kent Ridge Road, Singapore 119077, Singapore

that solves any Laplacian system on an n-node graph after $n^{o(1)}(\sqrt{n} + D)\log(1/\varepsilon)$ rounds of the standard CONGEST model, where D represents the hop-diameter of the underlying network and $\varepsilon > 0$ is the error of the solver. Moreover, they showed that their algorithm is *existentially optimal*, up to the $n^{o(1)}$ factor, establishing a lower bound of $\widetilde{\Omega}(\sqrt{n}+D)$ rounds via a reduction from the $s-t$ connectivity problem [12].

This *existential* lower bound in the CONGEST model of distributed computing should hardly come as any surprise. Indeed, it is well-known by now that a remarkably wide range of *global* optimization problems, including minimum spanning tree (MST), minimum cut (Min-Cut), maximum flow, and single-source shortest paths (SSSP), require $\widetilde{\Omega}(\sqrt{n} + D)$ rounds[1] [12–14]. The same limitation generally applies to any non-trivial approximation and even under randomization. Nonetheless, these lower bounds are constructed on some pathological graph instances that arguably do not occur in practice. This begs the question: *Can we obtain more refined performance guarantees based on the underlying topology of the communication network?* The framework of *low-congestion shortcuts*, introduced by Ghaffari and Haeupler [15], demonstrated that bypassing the notorious $\Omega(\sqrt{n})$ lower bound is possible: MST and Min-Cut on *planar graphs* can be solved in $\widetilde{O}(D)$ rounds. This is crucial, given that in many graphs of practical significance the diameter is remarkably small; e.g., $D = \text{polylog}(n)$ (as is folklore, this holds for most social networks), implying *exponential improvements* over generic algorithms used for general graphs. In the context of the distributed Laplacian paradigm, we raise the following question:

> *Is there a faster distributed Laplacian solver under "non-worst-case" families of graphs in the* CONGEST *model?*

The only known technique in distributed computing for designing algorithms that go below the $\sqrt{n}$-bound is the low-congestion shortcut framework of Ghaffari and Haeupler [15], and the large ecosystem of tools built around it [16–22]. However, the "ρ-congested minor" primitive introduced and extensively used in the novel distributed Laplacian solver [11] is out of reach from the current set of tools available in the low-congestion shortcut framework. We address this issue by introducing an analogous primitive called *ρ-congested part-wise aggregation*, which greatly simplifies the interface used by Forster et al. [11]. We then extend the low-congestion shortcut framework with new techniques that enables it to near-optimally solve this primitive: we provide both an algorithm that utilizes the very recent hop-constrained expander decompositions for shortcut construction [22] to solve the primitive in general graphs with a linear dependence on ρ, as well as a very simple algorithm with a quadratic ρ-dependence for bounded-treewidth graphs. Finally, we settle our original question in the positive by establishing that our new primitive can be readily used to accelerate the distributed Laplacian solver for non-worst-case topologies.

Specifically, we show that our new techniques are sufficient to lift the existentially optimal algorithm [11] to a *universally optimal* algorithm—modulo $n^{o(1)}$ factor inherent in the prior approach—for distributedly solving a Laplacian system, meaning that, *for any topology*, our algorithm is essentially as fast as possible. In other words, for any graph, our algorithm almost matches the best possible (correct) algorithm for that graph. This result is unconditional in essentially all settings of interest (see Theorem 2 for details), but relies on conjectured improvements of current state-of-the-art constructions of low-congestion shortcuts to achieve unqualified universal optimality [18]—like all other results in the area.

Furthermore, another concrete way of bypassing the $\widetilde{\Omega}(\sqrt{n} + D)$ lower bound, besides investigating non-worst-case families of graphs, is by enhancing the local communication network with a limited amount of *global power*. Indeed, research concerning *hybrid* networks was recently initiated in the realm of distributed algorithms [23], although networks combining different communication modes have already found numerous applications in real-life computing systems; as such, hybrid networks have been intensely studied in other areas of distributed computing (see [24–26], and references therein). In this paper, we will enhance the standard CONGEST model with the recently introduced *node-capacitated clique* (henceforth NCC) [27]. The latter model enables all-to-all communication, but with severe capacity restrictions for every node. The integration of these models will be referred to as the HYBRID model for the rest of this work. This leads to the following central question:

> *Is there a faster distributed Laplacian solver in the* HYBRID *model?*

Our paper essentially settles this question by showing the same ρ-congested part-wise aggregation primitive can be efficiently solved in $\widetilde{O}(\rho)$ rounds of NCC, implying an almost optimal $n^{o(1)}$-round distributed algorithm for solving Laplacian systems in the HYBRID model. A conceptual contribution of our approach is that we treat both CONGEST, Supported-CONGEST, and HYBRID in a *unified way* through the lens of the low-congestion shortcut framework, by designing our algorithm using high-level

[1] As usual, we use the notation $\widetilde{O}(\cdot)$ and $\widetilde{\Omega}(\cdot)$ to suppress polylogarithmic factors on n.

primitives and leaving the model-specific translations to the framework itself. A similar unified view of PRAM (i.e., parallel) and CONGEST (i.e., distributed) graph algorithms through the same lens has led to very recent breakthroughs on long-standing open problems for both of these settings [28].

1.1 Overview of our contributions and techniques

The unifying thread and the main technical ingredient of our (almost) universally optimal distributed Laplacian solvers is a new fundamental communication primitive referred to as the *congested part-wise aggregation problem*. Specifically, we develop near-optimal algorithms for solving this problem in the (Supported-)CONGEST and the NCC model (Sect. 3), and then we utilize this primitive to develop almost universally optimal Laplacian solvers.

1.1.1 The congested part-wise aggregation problem

To introduce the congested part-wise aggregation problem, let us first give some basic background. The aforementioned Ghaffari-Haeupler framework of low-congestion shortcuts revolves around the so-called *part-wise aggregation problem* posed as follows: "The graph is partitioned into *disjoint* and individually-connected parts, and we need to compute some simple aggregate function for each part, e.g., the minimum of the values held by the nodes in a given part" [15] (see Definition 1 for a formal definition). Importantly, it has been shown that this primitive can be solved efficiently in *structured* topologies and that many problems (including the MST, shortest path, min-cut, etc.) reduce to a small number of calls to a part-wise aggregation oracle, leading to universally optimal algorithms. Unfortunately, it is not clear how to reduce solving a Laplacian system to (a small number of) part-wise aggregation calls; in this paper, we primarily address this issue.

Our first technical contribution is to extend the framework of low-congestion shortcuts by studying a more general primitive: one that incorporates *congestion* (of the input parts) into the underlying *part-wise aggregation* instance. More precisely, unlike the standard part-wise aggregation problem, we allow each node to participate in up to $\rho \in \mathbb{Z}_{\geq 1}$ aggregation parts (see Definition 6). We later show that efficient solutions to this primitive leads to efficient distributed Laplacian solvers.

We first remark that a natural strategy for solving congested part-wise aggregation instances does not work: congested instances *cannot*, in general, be directly reduced to a "small" collection of 1-congested instances, thereby necessitating a more refined approach. To this end, our approach is based on "lifting" the underlying communication network $\overline{G}$ into its ρ*-layered version* $\widehat{G}_{O(\rho)}$: every edge is replaced with a matching and every node with a ρ-clique. The importance of this transformation is that, as we show in Lemma 8, the ρ-congested part-wise aggregation problem can be reduced to a 1-congested instance on the ρ-layered graph (Sect. 3.1.1). This is first established under the assumption that individual parts correspond to simple paths, and then we extend our results to general parts by following the approach of Haeupler et al. [18]. In light of this reduction, we next focus on solving the 1-congested part-wise aggregation instance on the layered graph.

As a warm-up, we treat graphs with bounded *treewidth* $\operatorname{tw}(G)$ (Definition 5). It is known that on a graph G with treewidth $\operatorname{tw}(G)$, a 1-congested part-wise aggregation instance can be solved in $\widetilde{O}(\operatorname{tw}(G)D)$ rounds of CONGEST [17]. Keeping this in mind, we first show that the treewidth of the ρ-layered graph $\widehat{G}_\rho$ can only increase by a factor of ρ compared to the original graph (Lemma 13). Hence, we can solve 1-congested instances in $\widehat{G}_{O(\rho)}$ in $\widetilde{O}(\rho \operatorname{tw}(\overline{G})D)$ rounds (when the underlying network is $\widehat{G}_{O(\rho)}$), which in turn allows us to solve ρ-congested instances on $\overline{G}$ in $\widetilde{O}(\rho^2 \operatorname{tw}(G)D)$ time in G (another ρ factor is necessary to simulate $\widehat{G}_{O(\rho)}$ in $\overline{G}$). This positive result poses a natural question: can we achieve similar results on graphs with bounded *minor density* $\delta(G)$ (Definition 4)? However, the answer to this question is negative: *minor density* can blow up even for a 2-layered planar graph (see Observation 2), making such a result impossible.

Then, we look at arbitrary graphs G: it is known that 1-congested part-wise aggregation instances can be solved in a number of rounds that is controlled by $\mathrm{SQ}(G)$, where $\mathrm{SQ}(G)$ is the *shortcut quality* of G (a certain graph parameter we formalize in Definition 3). Specifically, it can be solved in $\widetilde{O}(\mathrm{SQ}(G))$ rounds when the topology is known in advance[2] [18] and $\operatorname{poly}(\mathrm{SQ}(G)) \cdot n^{o(1)}$ in general CONGEST [22]. The shortcut quality parameter is significant since many distributed problems (including the MST, shortest path, Min-Cut, and—as we show later—Laplacian solving) require $\widetilde{\Omega}(\mathrm{SQ}(G))$ rounds in CONGEST to be solved on G [18]. Therefore, algorithms that have an upper bound close to $\mathrm{SQ}(G)$ are *universally optimal*.

With the end goal of solving the 1-congested part-wise aggregations on layered graphs $\widehat{G}_\rho$ in time controlled by $\mathrm{SQ}(G)$, our main result establishes that *the shortcut quality of the* ρ*-layered graph does not increase* (modulo polylogarithmic factors) as compared to the original graph (Theorem 15). This has a plethora of important consequences: (1) when $\mathrm{SQ}(G) \leq n^{o(1)}$, we can unconditionally solve ρ-congested part-wise aggregation instances in $\rho \cdot n^{o(1)}$ CONGEST rounds using the state-of-the-art shortcut con-

[2] This model is also known as the *supported* CONGEST. That is, CONGEST under the assumption that the topology is known; see Sect. 2 for a formal description of the model. Our techniques also apply in the full generality of CONGEST, as we explain in the sequel.

struction [22], and (2) when the topology of G is known (i.e., Supported-CONGEST), there exists a distributed algorithm that solves any ρ-congested part-wise aggregation problem in $\rho \cdot \widetilde{O}(\mathrm{SQ}(G))$ rounds [18]. As a consequence of our general result, the shortcut quality of any 2-layered planar graph is $\widetilde{O}(D)$ since the shortcut quality of a planar graph is $\widetilde{O}(D)$ [15]. This is perhaps the most natural example of a graph whose minor density is very far from the shortcut quality; the only other example documented in the literature so far is that of *expander graphs*.

Our proof proceeds by employing alternative characterizations of the shortcut quality in terms of certain communication tasks. Specifically, shortcut quality can be shown to be equal (modulo polylogarithmic factors) to the following two-player max-min game: the first (max) player chooses k sources and k sinks in the graph such that we can find k node-disjoint paths matching the sources with the sinks; then the second (min) player finds the smallest so-called *quality* Q such that there exist k paths matching the sources with the sinks with the path lengths being at most Q and each edge of the underlying graph supporting at most Q of second player's paths. This characterization allows us to compare the shortcut quality of $\widehat{G}_\rho$ with $\overline{G}$ as follows: take the worst-case (first player's) set of sources and sinks in $\widehat{G}_\rho$. Project them to $\overline{G}$ and note they have node congestion ρ (due to the construction of $\widehat{G}_\rho$). Then, we show we can decompose (i.e., partition) these set of sources and sinks into $\widetilde{O}(\rho)$ pairs of sub-sources and sub-sinks that are node-disjointly connectable in G. However, each such set enjoys paths of quality $\mathrm{SQ}(G)$, hence embedding each such pair in a separate layer of $\widehat{G}_\rho$ shows that the shortcut quality of $\mathrm{SQ}(\widehat{G}_\rho)$ is at most $\widetilde{O}(\mathrm{SQ}(\overline{G}))$. Although this general approach improves over our result for treewidth-bounded graphs we previously described, our approach for the latter class of graphs is substantially simpler and more suited in practice.

1.1.2 Almost universally optimal Laplacian solvers

First, we note that any distributed Laplacian solver that always correctly outputs an answer on a fixed graph G must take at least $\widetilde{\Omega}(\mathrm{SQ}(G))$ rounds, giving us a lower bound to compare ourselves with. Our refined lower bound uses the hardness result recently shown by Haeupler et al. [18] for the spanning connected subgraph problem, applicable for *any* (i.e., non-worst-case) graph G. Specifically, we show that a Laplacian solver can be leveraged to solve the spanning connected subgraph problem, thereby substantially strengthening the lower bound due to Forster et al. [11].

Theorem 1 *Consider a graph $\overline{G}$ with shortcut quality $\mathrm{SQ}(\overline{G})$. Then, solving a Laplacian system on $\overline{G}$ with $\varepsilon \le \frac{1}{2}$ requires $\widetilde{\Omega}(\mathrm{SQ}(\overline{G}))$ rounds in both* CONGEST *and Supported-*CONGEST *models.*

On the upper-bound side, we utilize the congested part-wise aggregation primitive to improve and refine the Laplacian solver of Forster et al. [11], leading to a substantial improvement in the round complexity under *structured* network topologies.

Theorem 2 *Consider any n-node graph G with shortcut quality $\mathrm{SQ}(G)$ and hop-diameter D. There exists a distributed Laplacian solver with error $\varepsilon > 0$ with the following guarantees:*

- *In the Supported-*CONGEST *model, it requires $n^{o(1)}$ $\mathrm{SQ}(G)\log(1/\varepsilon)$ rounds.*
- *In the* CONGEST *model, it requires $n^{o(1)}$ $\mathrm{poly}(\mathrm{SQ}(G))$ $\log(1/\varepsilon)$ rounds.*
- *In the* CONGEST *model on graphs with minor density δ, it requires $n^{o(1)}\delta D\log(1/\varepsilon)$ rounds.*

We note that the above algorithm is almost (up to inherent $n^{o(1)}$ factors) universally optimality for most settings of interest. Since it is (almost) matching the $\mathrm{SQ}(G)$-lower-bound, it is unconditionally universally optimal when the topology is known in advance (i.e., Supported-CONGEST). Furthermore, in standard CONGEST, we give almost universally optimal $Dn^{o(1)}\log(1/\varepsilon)$-round algorithms for topologies that include planar graphs, $n^{o(1)}$-*genus* graphs, $n^{o(1)}$-treewidth graphs, excluded-minor graphs, since all of them are graphs with minor density $\delta(G) = n^{o(1)}$. Furthermore, for the realistic case of $D \le n^{o(1)}$, it holds for most networks of interest that $\mathrm{SQ}(G) \le n^{o(1)}$ (e.g., expanders, hop-constrained expanders, as well as all classes mentioned earlier), for which we get $n^{o(1)}\log(1/\varepsilon)$-round solvers. We stress that the conjectured improvements of the state-of-the-art of almost-optimal low-congestion shortcut constructions would immediately lift our results to be unconditionally universally optimal in CONGEST; this issue is orthogonal and not within the scope of this paper. It is also worth pointing out that both our algorithms for solving congested part-wise aggregations and the building blocks of the Laplacian solver of Forster et al. [11] are in general randomized, and so all of our guarantees apply with high probability.

Furthermore, in HYBRID we obtain an almost optimal complexity in *general graphs*:

Theorem 3 *Consider any n-node graph. There exists a distributed Laplacian solver in the* HYBRID *model with round complexity $n^{o(1)}\log(1/\varepsilon)$, where $\varepsilon > 0$ is the error of the solver.*

This implies a remarkably fast subroutine for solving a Laplacian system in HYBRID under arbitrary topologies. As a result, we corroborate the observation that a very limited amount of global power can lead to substantially faster algorithms for certain optimization problems, supplementing a recent line of work [23, 29–35]. Furthermore, our framework based on the congested part-wise aggregation problem allows for a unifying treatment of both (Supported-)CONGEST and HYBRID, and we consider this to be an important conceptual contribution of our work. Indeed, as we previously explained, both of our accelerated Laplacian solvers rely on faster algorithms for solving the congested part-wise aggregation problem. In particular, for (Supported-)CONGEST we have already described our approach in detail, while in the HYBRID model we employ certain communication primitives developed in [27] for dealing with congestion in part-wise aggregations. A byproduct of our results is that the framework of low-congestion shortcuts interacts particularly well with the HYBRID model, as was also observed in prior work [36].

1.2 Further related work

Our main reference point is the recent Laplacian solver of Forster et al. [11] with *existentially* almost-optimal complexity of $n^{o(1)}(\sqrt{n}+D)\log(1/\varepsilon)$ rounds, where $\varepsilon > 0$ represents the error of the solver. Specifically, they devised several new ideas and techniques to circumvent certain issues which mostly relate to the bandwidth restrictions of the CONGEST model; these building blocks, as well as the resulting Laplacian solver are revisited in our work to refine the performance of the solver. We are not aware of any previous research addressing this problem in the distributed context. On the other hand, the Laplacian paradigm has attracted a considerable amount of interest in the community of parallel algorithms [37, 38].

Research concerning hybrid communication networks in distributed algorithms was recently initiated by Augustine et al. [23]. Specifically, they investigated the power of a model which integrates the standard LOCAL model [39] with the recently introduced node-capacitated clique (NCC) [27], focusing mostly on distance computation tasks. Several of their results were subsequently improved and strengthened in subsequent works [30, 32] under the same model of computation. In our work we consider a substantially weaker model, imposing a severe limitation on the communication over the "local edges". This particular variant has been already studied in some recent works for a variety of fundamental problems [31, 33].

The NCC model, which captures the global network in all hybrid models studied thus far, was introduced by Augustine et al. [27] partly to address the unrealistic power of the *congested clique* (CLIQUE) [40]. In the latter model each node can communicate *concurrently and independently* with *all* other nodes by $O(\log n)$-bit messages. In contrast, the NCC model allows communication with $O(\log n)$ (arbitrary) nodes per round. As a result, in the HYBRID model and under a sparse local network, only $\widetilde{\Theta}(n)$ bits can be exchanged overall per round, whereas CLIQUE allows for the exchange of up to $\widetilde{\Theta}(n^2)$ (distinct) bits. As evidence for the power of CLIQUE we note that even slightly super-constant lower bounds would give new lower bounds in circuit complexity, as implied by a simulation argument due to Drucker et al. [41]. Finally, we remark a subsequent work that leverages tools from the Laplacian paradigm in the *broadcast* variant of the congested clique [42].

2 Preliminaries

General notation. We denote with $[k] := \{1, 2, \ldots, k\}$. Graphs throughout this paper are undirected. The nodes and the edges of a given graph G are denoted as $V(G)$ and $E(G)$, respectively. We also use $n := |V(G)|$ for brevity. The graphs are often weighted, in which case we assume (as is standard) that for all $e \in E(G)$, $\boldsymbol{w}(e) \in \{1, 2, \ldots, \text{poly}(n)\}$. We will denote the hop-diameter of a graph G with $D(G)$ (the hop-diameter ignores weights). Moreover, we use $A \uplus B$ to denote the multiset union, i.e., each element is repeated according to its multiplicity; this operation corresponds to disjoint unions when $A \cap B = \emptyset$.
Communication models. The communication network consists of a set of $\overline{n}$ entities with $[\overline{n}] := \{1, 2, \ldots, \overline{n}\}$ being the set of their IDs, and a local communication *topology* given by a graph $\overline{G}$.[3] We define $D := D(\overline{G})$ to be the (hop-)diameter of the underlying network. At the beginning, each node knows its own unique $O(\log \overline{n})$-bit identifier as well as the weights of the incident edges. Communication occurs in *synchronous rounds*, and in every round nodes have unlimited computational power to process the information they possess. We will consider models with both *local* and *global* communication modes.

The *local* communication mode will be modeled with the *CONGEST model* [43] and *Supported-CONGEST model* [44], for which in each round every node can exchange an $O(\log \overline{n})$-bit message with each of its neighbors in $\overline{G}$ via the *local* edges. In the (standard) CONGEST model, each node $v \in V(\overline{G})$ initially only knows the identifiers of each node in v's own neighborhood, but has no further knowledge about the topology of the graph. On the other hand, in the Supported-CONGEST model, all nodes know the entire topology of $\overline{G}$ upfront, but not the input.

The *global* communication mode will be modeled using *NCC* [27], for which in each round every node can exchange $O(\log \overline{n})$-bit messages with $O(\log \overline{n})$ arbitrary nodes via *global* edges. If the capacity of some channel is exceeded, i.e., too many messages are sent to the same node, it will only

[3] To avoid any possible confusion we point out that, for consistency with the nomenclature of Forster et al. [11], we henceforth reserve $\overline{G}$ to denote the underlying *communication network* while G is used in statements regarding arbitrary graphs.

receive an *arbitrary* (potentially adversarially selected) subset of the information based on the capacity of the network; the rest of the messages are dropped. In this context, we will let HYBRID be the integration of CONGEST and NCC (i.e., nodes have both a *local* and a *global* communication mode at their disposal).

At this point, it is worth pointing out that different models are suitable for different applications. The standard CONGEST model is appropriate in local communication networks under severely congested edges, in contrast to LOCAL, another popular model which captures local networks with edges of essentially unlimited capacity. On the other hand, NCC was introduced to model networks with global capabilities, but under severe restrictions on the amount of communication possible in each round; NCC has been put forward as a more realistic counterpart to CLIQUE, which we described earlier. Correspondingly, HYBRID is more appropriate to model networks with multiple communication modes. For an additional motivation for each of the above models, we refer to the papers that introduced them, and references therein.

The performance of a distributed algorithm will be measured in terms of its *round complexity*—the number of rounds required so that every node knows its part of the output. For randomized algorithms it will suffice to reach the desired state with high probability.[4] We will assume throughout this work that nodes have access to a common source of randomness; this comes without any essential loss of generality in our setting [45]. When talking about a distributed algorithm for a specific problem (e.g., Laplacian solving, part-wise aggregation, etc.) we assume the input is appropriately *distributedly stored* (i.e., each node will know its own part) and, upon termination, it will be required that the output is appropriately distributedly stored. The appropriate way to distributedly store the input and output will be explained in the problem definition.

Low-congestion shortcuts. A recurring scenario in distributed algorithms for global problems (e.g. MST) boils down to solving the following part-wise aggregation problem:

Definition 1 (Part-Wise Aggregation Problem) Consider an n-node graph G whose node set $V(G)$ is *partitioned* into k (disjoint) parts $P_1 \uplus \cdots \uplus P_k \subseteq V(G)$ such that each induced subgraph $G[P_i]$ is *connected*. In the *part-wise aggregation* problem, each node $v \in V$ is given its part-ID (if any) and an $O(\log n)$-bit value $\boldsymbol{x}(v)$ as input. The goal is that, for every part P_i, all nodes in P_i learn the part-wise aggregate $\bigoplus_{w \in P_i} \boldsymbol{x}(w)$, where $\bigoplus$ is an arbitrary pre-defined *aggregation function*.

[4] We say that an event holds with high probability if it occurs with probability at least $1 - 1/n^c$ for a (freely choosable) constant $c > 0$.

Throughout this paper, we will assume that the aggregation function $\bigoplus$ is commutative and associative (e.g. min, sum, logical-AND), although this is not strictly needed (e.g., see [21]). To solve such problems, Ghaffari and Haeupler [15] introduced a natural combinatorial graph structure that they refer to as *low-congestion shortcuts*.

Definition 2 (Low-Congestion Shortcuts) Consider a graph G whose node set $V(G)$ is *partitioned* into k (disjoint) parts $P_1 \uplus \cdots \uplus P_k \subseteq V(G)$ such that each induced subgraph $G[P_i]$ is *connected*. A collection of subgraphs $H_1, \ldots, H_k$ is a *shortcut* of G with *congestion* c and *dilation* d if the following properties hold: (i) the (hop) diameter of each subgraph $G[P_i] \cup H_i$ is at most d, and (ii) every edge is included in at most c many of the subgraphs H_i. The quantity $Q = c + d$ will be referred to as the *quality* of the shortcut.

Importantly, a shortcut of quality Q allows us to solve the part-wise aggregation problem in $\widetilde{O}(Q)$ rounds of CONGEST, as formalized below.

Proposition 4 *Suppose that $P_1, \ldots, P_k$ is any part-wise aggregation instance in a communication network $\overline{G}$. Given a shortcut of quality Q, we can solve with high probability the part-wise aggregation problem in $\widetilde{O}(Q)$* CONGEST *rounds.*

Proof Consider only one part P_i in isolation over the network $G[P_i] + H_i$. First, we claim that there exists a simple deterministic algorithm that computes the AND-aggregate (where each node $v \in P_i$ has a input bit $\boldsymbol{x}(v)$) in $O(d)$ rounds, where each edge is used to send at most $O(1)$ messages. Concretely, any node whose input is 0 will forward its input to all neighbors and deactivate itself. Any node which hears about the existence of an input-0 will forward this to all of its neighbors and deactivate itself. After $O(d)$ rounds, either all nodes have heard about the existence of a 0 or they can conclude all inputs are 1.

We continue considering only one part P_i in isolation. The next step is to elect a leader of P_i by finding the node with the smallest ID in P_i; then, (1) iterate from the most significant bit of the ID to the least significant bit of the ID; (2) compute the AND-aggregate of the current bit of all the nodes' IDs; (3) if the AND-aggregate is 0, all nodes whose current bit of the ID is 1 will drop out.

Putting these together we have a way of computing the aggregate of a part P_i in isolation in $\widetilde{O}(d)$ rounds with each edge carrying $\widetilde{O}(1)$ messages: First, we elect a leader of P_i. Then, the leader initiates the computation of a spanning BFS tree of $\overline{G}[P_i] + H_i$ by broadcasting from itself to all other nodes, and each node forwards the message to all neighbors; the neighbor from which it obtains the message first is the parent in the tree. Moreover, by performing a convergecast over the BFS tree, one can easily compute the aggregate in $O(d)$ rounds for a single part P_i.

Finally, we have to run the algorithms on all the parts $\{P_i\}_i$ simultaneously. However, this might incur congestion issues on some edges since algorithms associated with multiple parts want to send a message through the same edge in the same round. By definition of the congestion c, at most c messages need to be passed over any one edge e. Our job is to schedule the algorithms such that, indeed, all of them complete in $\widetilde{O}(c)$ rounds. To this end, we choose a uniformly random $\text{delay}(i)$ between 0 and $\widetilde{O}(c)$ for each part P_i. Then, we start the algorithm on P_i at round $\text{delay}(i)$—this technique is known as the randomized delay [45]. Randomly delaying all algorithms makes the *expected* number of messages crossing a given edge in a fixed round $\Theta(1)$. By a Chernoff bound, this number is bounded by $\widetilde{O}(1)$ with high probability. Therefore, by simulating each round of the algorithm using $\widetilde{O}(1)$ rounds of communication (where each round of communication carries at most a single message across an edge), we can schedule the algorithms on all parts simultaneously [45]. In turn, this allows us to complete all of the aggregates in $\widetilde{O}(d+c) = \widetilde{O}(Q)$ rounds. □

We recall that we are operating under the assumption that nodes share a common source of randomness, which is used in the above proof.

Shortcut quality and construction of shortcuts. Shortcut quality, introduced below, is a fundamental graph parameter that has been proven to characterize the complexity of many important problems in distributed computing.

Definition 3 Given a graph $G=(V,E)$, we define the *shortcut quality* $\text{SQ}(G)$ of G as the optimal (smallest) shortcut quality of the worst-case partition of V into disjoint and connected parts $P_1 \uplus P_2 \uplus \ldots \uplus P_k \subseteq V$.

For fundamental problems such as MST, SSSP, and Min-Cut any correct algorithm requires $\widetilde{\Omega}(\text{SQ}(\overline{G}))$ rounds on any network $\overline{G}$, even if we allow randomized solutions and (non-trivial) approximation factors. In fact, this limitation holds even when the network topology $\overline{G}$ is known to all nodes in advance [18]. We remark that $\widetilde{\Omega}(D(\overline{G})) \leq \text{SQ}(\overline{G}) \leq O(D(\overline{G}) + \sqrt{\overline{n}})$, and the upper bound is known to be tight in certain (pathological) worst-case graph instances [15].

Moreover, *assuming* fast distributed algorithms for constructing shortcuts of quality competitive with $\text{SQ}(\overline{G})$, all of the aforementioned problems can be solved in $\widetilde{O}(\text{SQ}(\overline{G}))$ rounds [15, 20, 21]. However, the key issue here is the algorithmic construction of the shortcuts upon which the above papers rely. While there has been a lot of recent progress in this regard, current algorithms are quite complicated and have sub-optimal guarantees. We recall below these state-of-the-art $\text{SQ}(\overline{G})$-competitive construction results.

Theorem 5 *There exists a distributed algorithm that, given any part-wise aggregation instance on any $\overline{n}$-node graph $\overline{G}$, computes with high probability a shortcut with the following guarantees:*

- *In* CONGEST*, the shortcut has quality* $\text{poly}\left(\text{SQ}(\overline{G})\right) \cdot \overline{n}^{o(1)}$ *and the algorithm terminates in* $\text{poly}\left(\text{SQ}(\overline{G})\right) \cdot \overline{n}^{o(1)}$ *rounds [22].*
- *In Supported-CONGEST, the shortcut has quality $\widetilde{O}(\text{SQ}(\overline{G}))$ and the algorithm terminates in $\widetilde{O}(\text{SQ}(\overline{G}))$ rounds [18].*

Universal optimality. A distributed algorithm is said to be *α-universally optimal* if, on every network graph $\overline{G}$, it is α-competitive with the fastest correct algorithm on $\overline{G}$ [18]. Even the existence of such algorithms is not at all clear as it would seem possible that vastly different algorithms are required to leverage the structure of different networks. Nevertheless, a remarkable consequence of Theorem 5 is that in Supported-CONGEST we can design $\widetilde{O}(1)$-universally optimal algorithms for many fundamental optimization problems. Moreover, efficient shortcut construction is the only obstacle towards achieving these results in the full generality of CONGEST, which is an orthogonal issue and out of scope for this paper. Still, the aforementioned results are sufficient to design $\overline{n}^{o(1)}$-universally optimal algorithms on graphs that have shortcut quality $\text{SQ}(\overline{G}) = \overline{n}^{o(1)}$.

Graphs excluding dense minors. It turns out that the crucial issue of efficient shortcut construction can be resolved with a near-optimal, simple, and even deterministic algorithm for the rich class of graphs with *bounded minor density*. Formally, let us first recall the following definition.

Definition 4 (Minor Density) The *minor density* $\delta(G)$ of a graph G is defined as

$$\delta(G) = \max\left\{\frac{|E'|}{|V'|} : H = (V', E') \text{ is a minor of } G\right\}.$$

Any family of graphs closed under taking minors (such as planar graphs) has a constant minor density. For such graphs, Ghaffari and Haeupler [19] established an efficient shortcut construction:

Theorem 6 ([19]) *Any graph G with hop-diameter D and minor density $\delta(G)$ admits shortcuts of quality $\widetilde{O}(\delta D)$, which can be constructed with high probability in $\widetilde{O}(\delta D)$ rounds of* CONGEST.

Some of our results apply for communication networks with *bounded treewidth*, so let us recall the following definition.

Definition 5 (Tree Decomposition and Treewidth) A *tree decomposition* of a graph G is a tree T with tree-nodes $X_1, \ldots, X_k$, where each X_i is a subset of $V(G)$ satisfying the following properties:

1. $V = \bigcup_{i=1}^{k} X_i$;
2. For any node $u \in V(G)$, the tree-nodes containing u form a connected subtree of T;
3. For every edge $\{u, v\} \in E(G)$, there exists a tree-node X_i which contains both u and v.

The *width* w of the tree decomposition is defined as $w := \max_{i\in[k]} |X_i| - 1$. Moreover, the *treewidth* $\operatorname{tw}(G)$ of G is defined as the minimum of the width among all possible tree decompositions of G.

Bounded-treewidth graphs inherit all of the nice properties guaranteed by Theorem 6, as implied by the following well-known fact.

Fact 7 *For any graph G, $\delta(G) \le \operatorname{tw}(G)$.*

The Laplacian matrix. Consider a weighted undirected graph $G = (V, E, \boldsymbol{w} > 0)$. The *Laplacian* of the graph G is defined as

$$\mathcal{L}(G)_{u,v} = \begin{cases} \sum_{\{u,z\}\in E} \boldsymbol{w}(u, z) & \text{If } u = v, \\ -\boldsymbol{w}(u, v) & \text{otherwise.} \end{cases}$$

The Laplacian matrix of a graph is (i) *symmetric* ($\mathcal{L}(G)^T = \mathcal{L}(G)$); (ii) *positive semi-definite* ($\boldsymbol{x}^T \mathcal{L}(G)\boldsymbol{x} \ge 0$ for any $\boldsymbol{x}$); and (iii) *weakly diagonally dominant* ($\mathcal{L}(G)_{u,u} \ge \sum_{v\neq u} |\mathcal{L}(G)_{u,v}|$).

Further notation. Consider two positive semi-definite matrices $\mathbf{A}, \mathbf{B} \in \mathbb{R}^{n\times n}$. For a vector $\boldsymbol{x} \in \mathbb{R}^n$ we define $\|\boldsymbol{x}\|_{\mathbf{A}} := \sqrt{\boldsymbol{x}^T \mathbf{A} \boldsymbol{x}}$ (Mahalanobis norm). We will write $\mathbf{A} \approx_\varepsilon \mathbf{B}$ if $\exp(-\varepsilon)\mathbf{A} \preceq \mathbf{B} \preceq \exp(\varepsilon)\mathbf{A}$, where $\mathbf{A} \preceq \mathbf{B}$ if and only if the matrix $\mathbf{B} - \mathbf{A}$ is positive semi-definite. For an edge $e = \{u, v\}$, we will let $\boldsymbol{b}(e) := \mathbb{1}_u - \mathbb{1}_v$, where $\mathbb{1}_u \in \mathbb{R}^n$ represents the characteristic vector of node u. For a graph G with *resistances* $\boldsymbol{r}(e)$, we define the *leverage scores* as $\operatorname{lev}_G(e) := \boldsymbol{r}(e)^{-1}\boldsymbol{b}^T(e)\mathcal{L}(G)^\dagger \boldsymbol{b}(e)$. Note that $0 \le \operatorname{lev}_G(e) \le 1$.

3 Congested part-wise aggregations

This section is concerned with a *congested* generalization of the standard part-wise aggregation problem (Definition 1), formally introduced below.

Definition 6 (Congested Part-Wise Aggregation Problem) Consider an n-node graph G with a collection of k subsets of nodes $P_1, \ldots, P_k \subseteq V(G)$ called *parts* such that each induced subgraph $G[P_i]$ is *connected* and each node $v \in V(G)$ is contained in at most $\rho \in \mathbb{Z}_{\ge 1}$ many parts, i.e., $\forall v \in V(G)\ |\{i : P_i \ni v\}| \le \rho$. In the *$\rho$-congested part-wise aggregation* problem, each node v is given the following as input: for each part $P_i \ni v$ node v knows the part-ID i and an $O(\log n)$-bit part-specific value $\boldsymbol{x}_i(v)$. The goal is that, for each part P_i, all nodes in P_i learn the part-wise aggregate $\bigoplus_{w\in P_i} \boldsymbol{x}_i(w)$, where $\bigoplus$ is a pre-defined *aggregation function*.

This congested generalization of the standard part-wise aggregation problem that we study in this section turns out to be a central ingredient in our refined Laplacian solver; this is further explained in Sect. 4. The remainder of this section is organized as follows. In Sect. 3.1 we establish near-optimal algorithms for solving congested part-wise aggregations in CONGEST, which is also the main focus of this section. We conclude by pointing out the construction for NCC in Sect. 3.2.

3.1 Solving congested instances in the CONGEST model

The first natural strategy for solving the ρ-congested part-wise aggregation problem of Definition 6 is through a reduction to $\operatorname{poly}(\rho)$ 1-congested instances. However, this approach immediately fails even if we allow $\rho = 2$. Indeed, there exist congested part-wise aggregation instances for which every two (distinct) parts share a common node, even when $\rho = 2$, leading to the following observation.

Observation 1 For an infinite family of values $\overline{n}$, there exists an $\overline{n}$-node planar graph $\overline{G}$ and a 2-congested part-wise aggregation instance $\mathcal{I}$ with $k = \Theta(\sqrt{\overline{n}})$ parts such that reducing $\mathcal{I}$ to the union of k' 1-congested part-wise aggregation instances on $\overline{G}$ requires $k' = \Omega(\sqrt{\overline{n}})$.

Such a pattern is illustrated in Fig. 1. Indeed, in that 2-congested part-wise aggregation instance every two distinct parts share a common node. As a result, directly employing a 1-congested part-wise aggregation oracle is of little use since it would introduce an overhead depending on the number of parts. In light of this, we develop a more refined approach that leverages what we refer to as the *layered graph*.

3.1.1 The layered graph

Here we introduce the *layered graph* $\widehat{G}_\rho$, associated with the underlying graph $\overline{G}$. Then, we reduce any ρ-congested part-wise aggregation on $\overline{G}$ to a 1-congested instance on $\widehat{G}_{O(\rho)}$.

The layered graph. Consider an underlying network $\overline{G}$ and some $\rho \in \mathbb{Z}_{\ge 1}$, corresponding to the congestion parameter in Definition 6. The *layered graph* $\widehat{G}_\rho$ is constructed in the following way. First, we let $\widehat{G}_\rho$ be a disjoint union of ρ copies of $\overline{G}$ (called *layers*), namely $\overline{G}_1, \overline{G}_2, \ldots, \overline{G}_\rho$. Each node $v \in V(\overline{G})$ is associated with its copies $v_1, v_2, \ldots, v_\rho \in V(\widehat{G}_\rho)$. We also add an edge between each two copies that originate from the same node (i.e., we add a clique to $\widehat{G}_\rho$ on the set of copies associated with the same node $v \in V(\overline{G})$); this construction is illustrated in Fig. 2. The layered graph induces

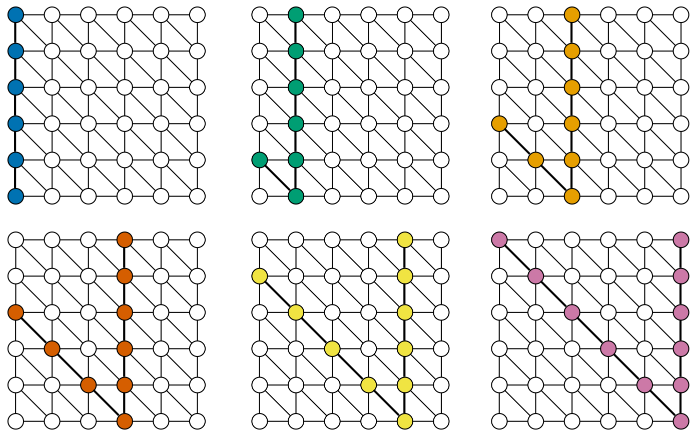

Fig. 1 A 2-congested part-wise aggregation problem on a 6×6 grid (the instance immediately extends to an $\sqrt{n} \times \sqrt{n}$ topology). Different colors highlight different parts of the instance

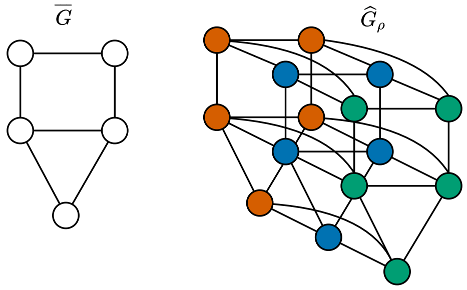

Fig. 2 An example of a transformation from $\overline{G}$ to the layered graph $\widehat{G}_\rho$ with $\rho = 3$. We have highlighted with different colors different layers of the graph

a natural *projection* operation $\pi : V(\widehat{G}_\rho) \to V(\overline{G})$ which maps a copy v_i to its original node $v = \pi(v_i)$. Furthermore, we often talk about simulating $\widehat{G}_\rho$ in $\overline{G}$, by which we mean that each node v simulates—learns all the inputs and can generate all outputs—for its copies $v_1, \ldots, v_\rho$. Throughout this paper, we will assume that $\rho = \text{poly}(\overline{n})$ so that any $O(\log n)$-bit message on $\widehat{G}_\rho$ can be sent within $O(1)$ rounds in $\overline{G}$; this also keeps the $\widetilde{O}$-notation well-defined.

The main goal of this section is to establish that the ρ-congested part-wise aggregation problem on $\overline{G}$ can be reduced to a 1-congested instance on $\widehat{G}_{O(\rho)}$, as formalized below.

Lemma 8 (Unrestricted Congested Part-Wise Aggregation) *Let $\overline{G}$ be an $\overline{n}$-node graph and let $\mathbb{Z}_{\geq 1} \ni \rho \leq \text{poly}(\overline{n})$. Suppose that any (1-congested) part-wise aggregation on $\widehat{G}_{O(\rho)}$ can be solved with a τ-round CONGEST algorithm on $\widehat{G}_{O(\rho)}$. Then, there exists an $\widetilde{O}(\rho \cdot \tau)$-round CONGEST algorithm on $\overline{G}$ that solves any ρ-congested part-wise aggregation instance on $\overline{G}$.*

Towards establishing this reduction, we first point out that any CONGEST algorithm on $\widehat{G}_\rho$ can be simulated with only a ρ multiplicative overhead in the round complexity.

Lemma 9 (Simulating $\widehat{G}_\rho$ in $\overline{G}$) *For any $\overline{G}$ and any $\mathbb{Z}_{\geq 1} \ni \rho \leq \text{poly}(\overline{n})$, we can simulate any τ-round CONGEST algorithm on $\widehat{G}_\rho$ with a $(\rho \cdot \tau)$-round CONGEST algorithm on $\overline{G}$.*

Proof Let us consider one round of communication in $\widehat{G}_\rho$. Each node v will simulate (learn all messages coming into) its copies $v_1, \ldots, v_\rho \in V(\widehat{G}_\rho)$. Therefore, in each round node $v \in V(\overline{G})$ needs to learn all messages sent to v's copies $v_1, \ldots, v_\rho \in V(\widehat{G}_\rho)$ from their neighbors in $\widehat{G}_\rho$. Note that, by definition, v already knows the messages sent between any two copies v_i and v_j. Hence, in a single round v can learn all messages sent to any fixed v_i. As a result, ρ rounds of communication in $\overline{G}$ suffice to simulate a single round in $\widehat{G}_\rho$. □

Furthermore, we will use a folklore result showing how to color a (multi)graph of maximum degree Δ in $O(\Delta)$ colors in $O(\log n)$ rounds of CONGEST.

Fact 10 (Folklore, [46]) *Given a (multi)graph G with n nodes and maximum degree $\Delta \leq \text{poly}(n)$, there exists a randomized CONGEST algorithm that colors the edges of G with $O(\Delta)$ colors and completes in $O(\log n)$ rounds, with high probability. The coloring is proper, i.e., two edges that share an endpoint are assigned a different color.*

By multigraph here we simply mean that there can be multiple parallel edges between the same pair of nodes, and every such edge can carry an independent message per round. For completeness, we provide the simple proof below.

Proof of Fact 10 A simple edge-coloring algorithm presented by Johansson [46] works by choosing a color uniformly at random from the set $\{1, \ldots, O(\Delta)\}$ for each edge. Each edge will, with constant probability, choose a color not used by its neighbors. Then, this color stays fixed and the edge drops out. Hence, after $O(\log n)$ iterations the edges will be properly colored. Implementation-wise, we can assume there is an additional node in the middle of each edge which represents that edge (this only makes the problem harder). Each edge randomly chooses and sends its color to its endpoints which, in turn, inform on whether there is a conflict. Then, the edges send back to its endpoints whether it dropped out. This iteration is then repeated until we reach a proper coloring. □

Using this lemma, we first prove a version of our main reduction (Lemma 8), but with the slight twist that we restrict each part of the ρ-congested part-wise aggregation problem to be a simple path.

Lemma 11 (Path-Restricted Congested Part-Wise Aggregation) *Let $\overline{G}$ be an $\overline{n}$-node graph and let $\mathbb{Z}_{\geq 1} \ni \rho \leq \mathrm{poly}(\overline{n})$. Suppose that there exists a τ-round CONGEST algorithm solving the (1-congested) part-wise aggregation on $\widehat{G}_{O(\rho)}$. Then, there exists an $\widetilde{O}(\rho \cdot \tau)$-round CONGEST algorithm on $\overline{G}$ that solves any ρ-congested part-wise aggregation instance on $\overline{G}$ when each part is* restricted to be a simple path[5] *(nodes are not repeated in simple paths).*

Proof Let $\mathcal{P} = \{P_1, P_2, \ldots, P_k\}$ be subsets of nodes in $\overline{G}$ comprising the parts of some ρ-congested part-wise aggregation on $\overline{G}$. We will construct paths $\mathcal{P}' = \{P_1', P_2', \ldots, P_k'\}$ in $\widehat{G}_{O(\rho)}$ in a way that solving a part-wise aggregation on $\mathcal{P}'$ corresponds to solving a ρ-congested part-wise aggregation on $\mathcal{P}$.

Let E_i be the set of edges of $\overline{G}$ comprising the simple path traversing all the nodes in P_i, and consider the graph $G' := (V(\overline{G}), \biguplus_{i=1}^{k} E_i)$. First, we observe that the degree of any node in $v \in V(G') = V(\overline{G})$ is at most 2ρ since at most ρ many parts contain v and each part contributes at most 2 to the degree (since P_i is a simple path). Furthermore, we can simulate any ψ-round CONGEST algorithm on G' with a $(\psi \cdot \rho)$-round CONGEST algorithm on $\overline{G}$ as each edge $e \in E(\overline{G})$ appears at most ρ times in $E(G')$ due to the part-wise aggregation instance being at most ρ-congested. Therefore, using Fact 10 we can distributedly color the edges of G' into at most $O(\rho)$ colors in $O(\log n)$ CONGEST rounds on G', which translates to $\widetilde{O}(\rho)$ CONGEST rounds on $\overline{G}$. Suppose that the algorithm assigns a color $\boldsymbol{c}(e) \in \{1, \ldots, O(\rho)\}$ to each edge $e \in \biguplus_i E_i$.

We now construct $P_i' \subseteq \widehat{G}_{O(\rho)}$ as follows: consider each edge $\{u, v\} \in E_i$ and add both $u_{\boldsymbol{c}(\{u,v\})}, v_{\boldsymbol{c}(\{u,v\})} \in V(\widehat{G}_{O(\rho)})$ to P_i' (i.e., the $\boldsymbol{c}(\{u, v\})$-th copy of both u and v). By construction, P_i' induces a connected subgraph and the projection P_i' to $\overline{G}$ is exactly P_i. Next, we invoke the (1-congested) part-wise aggregation τ-round algorithm for $\{\mathcal{P}_1', \ldots, \mathcal{P}_k'\}$ on $\widehat{G}_\rho$, which can be converted to an $\widetilde{O}(\tau \cdot \rho)$-round algorithm on $\overline{G}$ (Lemma 9). Thus, we obtain an $\widetilde{O}(\tau \cdot \rho)$-round CONGEST algorithm on $\overline{G}$ which solves any path-restricted ρ-congested part-wise aggregation problem. □

[5] I.e., there exists a simple path traversing all the nodes of the part, and each node knows the corresponding incident edges of that path.

Finally, our reduction claimed in Lemma 8 follows via [18, Lemma 7.2], as we formalize below.

Proof of Lemma 8 Armed with Lemma 11, the claim follows by leveraging [18, Lemma 7.2 in the Full Version]. For completeness, their result states the following: Suppose we are given a collection of part-wise aggregation parts $\{P_i\}_i$ in any graph H. Then, one can solve the corresponding part-wise aggregate problem by reducing it to $\widetilde{O}(1)$-many (1-congested) part-wise aggregations between disjoint parts restricted to be simple paths $\mathcal{P}_i' = \{P_{i,j}'\}_{j=1}^{\widetilde{O}(1)}$. Here, $P_{i,j}'$ is a collection of node-disjoint simple paths restricted to P_i. Note that, for each j, $\bigcup_i P_{i,j}'$ is a collection of node-disjoint simple paths (since $P_{i,j}' \subseteq P_i$). Hence, it is sufficient to solve part-wise aggregation on a collection of node-disjoint simple paths on H.

For our proof, we use the above result for $H = \overline{G}$. Moreover, $\bigcup_i P_{i,j}'$ is ρ-congested since at most ρ parts P_i use any node v, and within each such P_i, every oracle call uses the node v at most once (since they are disjoint). By Lemma 11, the part-wise aggregation problem on ρ-congested node-disjoint simple paths is exactly handled in $\widetilde{O}(\rho \cdot \tau)$ CONGEST rounds. Therefore, the original problem can also be solved in $\widetilde{O}(\rho \cdot \tau)$ CONGEST rounds, as required. □

3.1.2 Treewidth-bounded graphs

Here we leverage the reduction we established in Lemma 8 to obtain a simple algorithm for solving the congested part-wise aggregation problem in treewidth-bounded graphs. The crucial observation is that the treewidth of the layered graph can only grow by a factor of ρ compared to the treewidth of the underlying graph, as we show below.

Claim 12 $D(\widehat{G}_\rho) \leq D(\overline{G}) + 1$.

Proof First, consider any two nodes $u_i, v_j \in V(\widehat{G}_\rho)$ such that $\pi(u_i) \neq \pi(v_j)$, with $i, j \in [\rho]$. By construction of the layered graph $\overline{G}_i$, there exists a path of length at most $D(\overline{G})$ in the i-th layer of $\widehat{G}_\rho$ between u_i to v_i. Thus, it follows that the (hop) distance between u_i and v_j is at most $D(\overline{G})$ given that v_j and v_i, with $i \neq j$, are adjacent—the copies form a clique in the layered graph. This also implies that the distance between any two nodes u_i and u_j, with $\pi(u_i) = \pi(u_j)$, is 1, concluding the proof. □

Lemma 13 *If the treewidth of $\overline{G}$ is $\mathrm{tw}(\overline{G})$, then $\mathrm{tw}(\widehat{G}_\rho) \leq \rho\, \mathrm{tw}(\overline{G}) + \rho - 1$.*

Proof Consider a tree decomposition (in the sense of Definition 5) of $\overline{G}$ into tree-nodes $\{X_j\}_{j=1}^{k}$ such that the width of the decomposition satisfies $w = \mathrm{tw}(\overline{G})$. We will show that there exists a tree decomposition on the graph $\widehat{G}_\rho$ with width at most $\rho(w+1)-1$, which in turn will imply that $\text{tw}(\widehat{G}_\rho) \le \rho(w+1)-1 = \rho(\text{tw}(\overline{G})+1)-1$. Indeed, consider the following sets:

$$\widehat{X}_j := \{u_i : u \in X_j, i \in [\rho]\},$$

for all $j \in [k]$. In words, each node $V(\overline{G}) \ni u \in X_j$ is replaced by all of its copies u_i in $\widehat{X}_j$. Observe that, by construction, $|\widehat{X}_j| = \rho|X_j|$. Thus, it suffices to show that the collection of sets $\{\widehat{X}_j\}_{j=1}^k$ forms a legitimate tree decomposition. First, since $V(\overline{G}) \subseteq \bigcup_j X_j$, it follows that $V(\widehat{G}_\rho) \subseteq \bigcup \widehat{X}_j$. Moreover, consider any two sets $\widehat{X}_j, \widehat{X}_\ell$, both containing a node $u_i \in V(\widehat{G}_\rho)$ for some $i \in [\rho]$. Then, we know that all the tree-nodes in the (unique) path between X_j and X_ℓ based on the original tree decomposition include u since X_j and X_ℓ both include u and $\{X_j\}$ is a tree decomposition of $\overline{G}$. In turn, this implies that all the tree-nodes in the path between $\widehat{X}_j$ and $\widehat{X}_\ell$ also contain u_i. Thus, the tree-nodes containing u_i form a connected subtree. Finally, we know that for every edge $\{u, v\} \in E(\overline{G})$ there exists a subset X_j such that $u, v \in X_j$. Hence, we can infer that for every edge in $E(\widehat{G}_\rho)$ there is a tree-node $\widehat{X}_j$ which includes both incident endpoints. As a result, we have constructed a tree decomposition in $\widehat{G}_\rho$ with width $\max_{j\in[k]}|\widehat{X}_j| - 1 \le \rho(w+1)-1$. □

Combining this guarantee with Fact 7, Lemma 8, Theorem 6, and Claim 12, we obtain the following immediate consequence.

Corollary 14 *Let $\overline{G}$ be an $\overline{n}$-node communication network of diameter at most D and treewidth $\text{tw}(\overline{G})$. Then, we can solve with high probability any ρ-congested part-wise aggregation problem in $\overline{G}$ within $\widetilde{O}(\rho^2 \cdot \text{tw}(\overline{G}) \cdot D)$ rounds of* CONGEST.

Minor density in the layered graph. In light of Lemma 13, a natural question is whether an analogous bound holds with respect to the minor density of the underlying graph; i.e., whether $\delta(\widehat{G}_\rho) = \text{poly}(\rho)\delta(G)$. Unfortunately, this is not possible, as illustrated in Fig. 3.

Observation 2 There exists an n-node graph G with minor density $\delta(G) = \widetilde{O}(1)$, but its 2-layered version $\widehat{G}_2$ has minor density $\delta(\widehat{G}_2) = \Omega(\sqrt{n})$.

3.1.3 General graphs

We conclude with our main result of Sect. 3.1: a near-optimal distributed algorithm for solving the ρ-congested part-wise aggregation problem in general graphs. In light of our reduction in Lemma 8, the technical crux is to control the degradation in the shortcut quality incurred by the transformation into the layered graph. Surprisingly, we show that the shortcut quality of $\widehat{G}_\rho$ does not increase by more than a polylogarithmic factor even when the number of layers is polynomial:

Theorem 15 *For any $\overline{n}$-node graph $\overline{G}$ and any $\mathbb{Z}_{\ge 1} \ni \rho \le \text{poly}(\overline{n})$, we have that $\text{SQ}(\widehat{G}_\rho) = \widetilde{O}(\text{SQ}(\overline{G}))$.*

This theorem improves over our previous result for treewidth-bounded graphs (Lemma 13) since the latter guarantee inevitably induces a linear factor of ρ in the shortcut quality of $\widehat{G}_\rho$; in contrast, Theorem 15 guarantees merely a polylogarithmic in ρ degradation in the shortcut quality. While this will not affect the asymptotic performance of the Laplacian solver, this improvement might prove to be important for future applications. Assuming that we have shown Theorem 15, we can then utilize the efficient shortcut constructions given in Theorem 5 to solve ρ-congested part-wise aggregations on any graph.

Corollary 16 *There exists a randomized distributed algorithm that, for any $\overline{n}$-node graph $\overline{G}$ and $\rho \in \mathbb{Z}_{\ge 1} \le \text{poly}(\overline{n})$, solves with high probability any ρ-congested part-wise aggregation instance on $\overline{G}$ with the following guarantees:*

- *In the* CONGEST*, the algorithm terminates in at most $\rho \cdot \text{poly}\left(\text{SQ}(\overline{G})\right) \cdot \overline{n}^{o(1)}$ rounds.*
- *In the* CONGEST *model on graphs with minor density δ, it requires $\widetilde{O}(\rho \cdot \delta \cdot D)$ rounds.*
- *In the Supported-*CONGEST*, the algorithm terminates in $\widetilde{O}(\rho \cdot \text{SQ}(\overline{G}))$ rounds.*

The rest of this subsection is dedicated to the proof of Theorem 15. First, to argue about the shortcut quality of the layered graph, we need to develop several generalized notions of node connectivity.

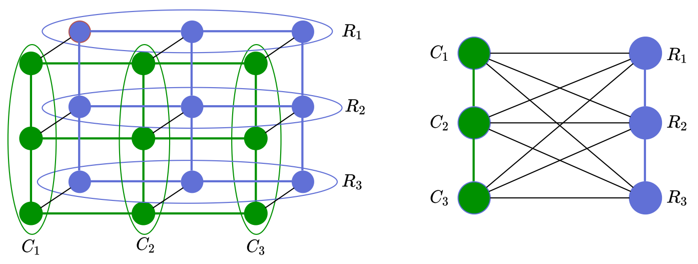

Fig. 3 The layered graph $\widehat{G}_\rho$ of a 3×3 grid with every node having congestion $\rho = 2$ (left), and a minor of $\widehat{G}_\rho$ induced by the connected components $\{C_1, C_2, C_3, R_1, R_2, R_3\}$ (right)

Pair node connectivity. Given a (multi)set of source-sink pairs $\mathcal{P} = \{(s_i, t_i)\}_{i=1}^{k}$ in G, we say that $\mathcal{P}$ has pair node connectivity ρ if there exist paths $P_1, \ldots, P_k$, with s_i and t_i being the endpoints of each P_i, such that every node $v \in V(G)$ is contained in at most ρ many paths, i.e., for all v we have $|\{i : V(P_i) \ni v\}| \leq \rho$. If $\mathcal{P}$ has pair node connectivity 1 we say that the pairs in $\mathcal{P}$ are *node-disjointly connectable*.

Any-to-any node connectivity. Suppose that we are given multisets of k *sources* $S = \{s_1, \ldots, s_k\}$ and k *sinks* $T = \{t_1 \ldots, t_k\}$. We say that (S, T) have any-to-any node connectivity ρ if there is a permutation $\pi : \{1, \ldots, k\} \to \{1, \ldots, k\}$ such that the pairs $\{(s_i, t_{\pi(i)})\}_{i=1}^{k}$ have pair node connectivity ρ. If (S, T) have any-to-any node connectivity 1 we say that the multisets (S, T) are any-to-any *node-disjointly connectable*.

The following decomposition lemma states that two sets with any-to-any node connectivity ρ can be decomposed into $\widetilde{O}(\rho)$ many pairs of subsets that are any-to-any node-disjointly connectable.

Lemma 17 *Given a graph G, suppose we are given any two multisets of nodes $S \subseteq V(G)$ and $T \subseteq V(G)$ of size $k := |S| = |T|$ that have any-to-any node connectivity ρ. Then, we can partition $S = S_1 \uplus S_2 \uplus \ldots \uplus S_{O(\rho \log k)}$ and $T = T_1 \uplus T_2 \uplus \ldots T_{O(\rho \log k)}$ such that $|S_i| = |T_i|$ and (S_i, T_i) are any-to-any node-disjointly connectable.*

Proof Suppose that each edge in G has infinite capacity while each node in G has unit capacity. Then, let us connect a super-source s to each node $x \in S$ with a unit-capacity edge, and a super-sink t to each node $x \in T$ with a unit capacity edge. By assumption, we know that there exists a flow f over $E(G)$ which sends k units of flow from s to t with edge congestion 1 and node congestion at most ρ. Therefore, the flow f/ρ sending k/ρ units of flow from s to t is a feasible solution of the maximum flow linear program with node constraints (i.e., it satisfies both edge and node capacity constraints). Since that linear program is integral (i.e., has an integrality gap of 1), there exists an integral flow f' which sends at least k/ρ units of flow and satisfies both node and edge capacity restrictions. In other words, there exist at least k/ρ node disjoint paths (with the exception of the endpoints) between s and t. Let $S_1 \subseteq S$ ($T_1 \subseteq T$) be the set of nodes on these paths immediately following the super-source (just before the super-sink, respectively). Clearly, by construction, (S_1, T_1) are any-to-any node-disjointly connectable. Finally, we define $S' \leftarrow S \backslash S_1$, $T' \leftarrow T \backslash T_1$ and proceed iteratively as above (producing S_2, T_2 instead of S_1, T_1). In each step, the size of S' and T' decreases by at least a multiplicative factor of $1 - 1/\rho$. Hence, $O(\rho \log k)$ steps suffice so that $S' = T' = \emptyset$. □

Next, we introduce two communication tasks that will be useful for characterizing the shortcut quality.

Multiple-unicast problem. Suppose that we are given k source-sink pairs $\mathcal{P} = \{(s_i, t_i)\}_{i=1}^{k}$. The goal is to find the smallest possible *completion time* τ such that there are k paths $P_1, \ldots, P_k$ for which (1) the endpoints of each P_i are exactly s_i and t_i; (2) the dilation is τ, i.e., each path P_i has at most τ hops; and (3) the congestion is τ, i.e., each edge $e \in E(G)$ is contained in at most τ many paths.

Any-to-any-cast problem. Suppose we are given k *sources* $S = \{s_1, \ldots, s_k\}$ and k *sinks* $T = \{t_1 \ldots, t_k\}$. The goal is to find the smallest τ so that there is a permutation $\pi : \{1, \ldots, k\} \to \{1, \ldots, k\}$ for which the multiple-unicast problem on $\{(s_i, t_{\pi(i)})\}_{i=1}^{k}$ has at most τ *completion time*.

Finally, we now recall (a reinterpretation of) a result characterizing shortcut quality from [18, 47]. Shortcut quality was originally defined as the smallest completion-time of the worst-case generalized (with respect to parts) multiple-unicast (i.e., multi-commodity) problem over a *pair* node-disjointly connectable instance (Definition 3). Using recent network coding gap results, we can equivalently express shortcut quality as the smallest completion-time of the worst-case any-to-any-cast (i.e., single-commodity) problem over sources and sinks that are *any-to-any* node-disjointly connectable. The formal statement follows.

Theorem 18 ([18, 47]) *Consider any graph G and let τ be the worst-case completion time of any-to-any-cast problems taken over all any-to-any node-disjointly connectable sets $(S \subseteq V(G), T \subseteq V(G))$. Then, $\tau = \widetilde{\Theta}(\mathrm{SQ}(G))$.*

Proof It was proven in [18, Lemma 2.8 in the Full Version] that $\mathrm{SQ}(G)$ is, up to $\widetilde{\Theta}(1)$ factors, equal to the completion time C of some multiple-unicast instance with respect to some source-sink pairs $\mathcal{P} := \{(s_i, t_i)\}_{i=1}^{k}$ that are pair node-disjointly connectable. We note that, since sources and sinks are disjoint, it follows that $k = \mathrm{poly}(n)$ and $O(\log k) = O(\log n)$. Furthermore, Haeupler et al. [47] proved that there exists a sub-instance $\mathcal{P}' = \{(s'_i, t'_i)_{i=1}^{k'}\} \subseteq \mathcal{P}$ such that $\mathrm{SQ}(G)$ is (up to $\widetilde{\Theta}(1)$ factors) equal to the completion time τ of the any-to-any-cast problem with respect to $(\{s'_i\}_{i=1}^{k'}, \{t'_i\}_{i=1}^{k'})$. One side of the claim is clear: for any sub-instance $\mathcal{P}' \subseteq \mathcal{P}$ we have that $\tau \leq C$. The other direction is harder and we sketch its proof here using the terminology by Haeupler et al. [47]. By definition and strong duality, $\mathrm{Cut}_{\mathcal{P}}(2C) = \mathrm{ConcurrentFlow}_{\mathcal{P}}(2C) \leq 1$. Furthermore, $\mathrm{Cut}_{\mathcal{P}}(C/10) = \mathrm{Cut}_{\mathcal{P}}(2C)/20 \leq 1/10$. Hence, by [18, Lemma 2.6] there is a sub-instance $\mathcal{P}' \subseteq \mathcal{P}$ with a moving cut of distance $\tau := \widetilde{\Omega}(C)$ and capacity less than $|\mathcal{P}'|$. Therefore, this proves that the completion time of any-to-any-cast problem on $\mathcal{P}'$ is at least τ. With this in mind, we have that $\widetilde{\Omega}(\mathrm{SQ}(G)) = \widetilde{\Omega}(C) = \tau \leq C = \widetilde{\Theta}(\mathrm{SQ}(G))$.

Finally, since $\mathcal{P} = \{(s_i, t_i)\}_{i=1}^{k}$ is pair node-disjointly connectable, it follows from the definition that the sub-instance $(\{s'_i\}_{i=1}^{k'}, \{t'_i\}_{i=1}^{k'})$ is any-to-any node-disjointly connectable. Therefore, $(\{s'_i\}_{i=1}^{k'}, \{t'_i\}_{i=1}^{k'})$ satisfies the constraints of this result and has completion-time $\tau = \widetilde{\Theta}(\mathrm{SQ}(G))$, as required. It is also clear that, by shortcut quality, any any-to-any node-disjointly connectable instance has completion time at most $\mathrm{SQ}(G)$ using the node-disjoint paths that witness the any-to-any node-disjointness as parts of the shortcut, making $(\{s'_i\}_{i=1}^{k'}, \{t'_i\}_{i=1}^{k'})$ the worst-case such instance (modulo poly-logarithmic factors). □

Finally, combining all of the previous ingredients, we are ready to show Theorem 15.

Proof of Theorem 15 Let $S \subseteq V(\widehat{G}_\rho)$ and $T \subseteq V(\widehat{G}_\rho)$ be any-to-any node-disjointly connectable sets such that the completion time of any-to-any-cast between S and T is $\widetilde{\Theta}(\mathrm{SQ}(\widehat{G}_\rho))$ (Theorem 18). Let $k := |S| = |T|$, and suppose that $S' := \biguplus_{s \in S}\{\pi(s)\} \subseteq V(\overline{G})$ and $T' := \biguplus_{t \in T}\{\pi(t)\} \subseteq V(\overline{G})$ are the multisets induced by projecting S and T to $\overline{G}$, respectively. By construction of $\widehat{G}_\rho$, S' and T' have any-to-any node connectivity ρ; to see this, consider the witness paths disjointly connecting them in $\widehat{G}_\rho$ and project them to $\overline{G}$. Therefore, we can partition $S' = S'_1 \uplus \ldots \uplus S'_{O(\rho \log k)}$ and $T' = T'_1 \uplus \ldots \uplus T'_{O(\log k)}$ such that $|S'_i| = |T'_i|$ and (S'_i, T'_i) are any-to-any node-disjointly connectable in $\overline{G}$ (Lemma 17).

By definition of shortcut quality, for each $i \in \{1, \ldots, O(\rho \log k)\}$ there exists a set of paths $(P^i_j)_{j=1}^{|S'_i|}$ in $\overline{G}$ between S'_i and T'_i of quality (i.e., both congestion and dilation) at most $\mathrm{SQ}(\overline{G})$. Then, we inject the first $O(\log k)$ collections of paths $(P^1_j)_j, (P^2_j)_j, \ldots, (P^{O(\log k)}_j)_j$ to the first layer $\overline{G}_1$ of $\widehat{G}_\rho$; the second $O(\log k)$ collections to the second layer $\overline{G}_2$, and so on, until we finally inject the last $O(\log k)$ collections to the last layer $\overline{G}_\rho$. Note that only the paths on the same layer interact, so both the congestion and dilation after injecting all paths into $\widehat{G}_\rho$ is $O(\mathrm{SQ}(\overline{G}) \log k)$. Hence, the same applies for the shortcut quality. Finally, to solve the any-to-any-cast problem on S and T one might need to add an between-layer edge at the beginning and at the end since each injected path is restricted to some adversarially chosen layer. However, this only increases the congestion and dilation by $O(1)$. Hence, the completion time of any-to-any-cast between S and T is $\widetilde{O}(\mathrm{SQ}(\overline{G}))$, implying that $\mathrm{SQ}(\widehat{G}_\rho) = \widetilde{O}(\mathrm{SQ}(\overline{G}))$. □

3.2 The NCC model

We next turn our attention to the NCC model. In particular, we observe that the ρ-congested part-wise aggregation problem admits a solution in $\mathrm{poly}(\rho, \log \overline{n})$ rounds of NCC:

Lemma 19 *Let $\overline{G}$ be an $\overline{n}$-node graph. Then, we can solve with high probability any ρ-congested part-wise aggregation problem on $\overline{G}$ after $O(\rho + \log \overline{n})$ rounds of* NCC.

This lemma is established after appropriately translating the communication primitives established for NCC by Augustine et al. [27]. In particular, let us first describe one of their key communication primitives.

The aggregation problem. In the *aggregation problem*, as defined by Augustine et al. [27], we are given a distributive function and a set of *aggregation parts* $\{P_1, \ldots, P_k\}$, with $P_i \subseteq V(\overline{G})$ for all i. Every aggregation part is associated with some target node $t_i \in P_i$.[6] Assuming that every node holds *exactly one* input value for each aggregation part of which it is a member, the goal is to let all the target nodes learn the aggregate values with respect to the associated aggregation parts. This setting allows a node to be part of multiple groups, and in particular, we let ℓ be the *local load*: the number of groups a given node may be included in—or an upper bound thereof. In addition, if $L = \sum_{i=1}^{k} |P_i|$ represents the *global load* of the aggregation problem, [27, Theorem 2.3] established the following result.

Lemma 20 ([27]) *There exists an aggregation algorithm which solves with high probability the aggregation problem in $O(L/\overline{n} + \ell/\log \overline{n} + \log \overline{n})$ rounds of* NCC.

In the context of the ρ-congested part-wise aggregation problem (Definition 6), it is clear that $\ell \leq \rho$ and $L \leq \rho\overline{n}$. Thus, we are now ready to establish Lemma 19.

Proof of Lemma 19 We first employ the communication protocol of Lemma 20 so that after $O(\rho + \log \overline{n})$ rounds of NCC each target node learns with high probability the aggregate values with respect to the associated aggregation parts. Next, we can reverse in time the previous communication pattern, but instead using the aggregate values as determined by the target nodes. As a result, every node will know with high probability the aggregate value for each of its aggregation parts after $O(\rho + \log \overline{n})$ rounds of NCC. □

4 Almost universally optimal Laplacians

In this section, we relate the congested part-wise aggregation problem we studied in the previous section with the Laplacian solver of Forster et al. [11]. To present a unifying analysis for both CONGEST and HYBRID, as well as for future applications and extensions, we analyze the distributed Laplacian solver under the following hypothesis.

Assumption 1 Consider a model of computation which incorporates CONGEST. We assume that we can solve with high probability any ρ-congested part-wise aggregation problem in $Q(\rho) = O(\rho^c Q)$ rounds, for some universal constant $c \geq 1$.

[6] In [27] the target node does not have to belong to the corresponding aggregation part, but this additional flexibility will not be required for our purposes.

The important connection between the congested part-wise aggregation problem (Definition 6) and the distributed Laplacian solver of Forster et al. [11] revolves around the concept of a *low-congestion minor*, a central component in the work of Forster et al. [11].

Definition 7 ([11]) A graph G is a *minor* of $\overline{G}$ if the following properties hold:

1. For every node $u^G \in V(G)$ there exists:
   (i) A subset of nodes of $\overline{G}$, which is termed as a *super-node*, $S^{G\to\overline{G}}(u^G)$, with a leader node $\ell(u^G) \in S^{G\to\overline{G}}(u^G)$;
   (ii) A connected subgraph of $\overline{G}$ on $S^{G\to\overline{G}}(u^G)$, for which we maintain a spanning tree $T^{G\to\overline{G}}(u^G)$.
2. There exists a mapping of the edges of G onto edges of $\overline{G}$, or self-loops, such that for any $\{u^G, v^G\} \in E(G)$, the mapped edge $\{u, v\}$ satisfies $u \in S^{G\to\overline{G}}(u^G)$ and $v \in S^{G\to\overline{G}}(v^G)$.

Moreover, we say that this minor G has *congestion* ρ, or G is a ρ-minor, if:

1. Every node $u \in \overline{G}$ is contained in at most ρ super-nodes $S^{G\to\overline{G}}(u^G)$, for some $u^G \in V(G)$;
2. Every edge of $\overline{G}$ appears as the image of an edge of G or in one of the trees connecting super-nodes (i.e., $T^{G\to\overline{G}}(u^G)$ for some u^G) at most ρ times.

Finally, we say that G is *ρ-minor distributed* over $\overline{G}$ if every $u \in V(\overline{G})$ stores:

1. All $u^G \in V(G)$ for which $u \in S^{G\to\overline{G}}(u^G)$;
2. For every edge e incident to u, (i) all the nodes u^G for which $e \in T^{G\to\overline{G}}(u^G)$, and (ii) all edges e^G that map to it.

We remark that the basis of Definition 7 was the earlier concept of a *distributed cluster graph* [48]. Now the upshot is that the congested part-wise aggregation problem we introduced is the central ingredient that allows performing certain "local" operations on a graph ρ-minor distributed into the underlying communication network. Indeed, the following lemma is a direct consequence of Definition 7.

Lemma 21 *Let $G = (V, E)$ be an n-node graph ρ-minor distributed into an $\overline{n}$-node communication network $\overline{G} = (\overline{V}, \overline{E})$ for which Assumption 1 holds for some $Q = Q(\rho)$. Then, we can perform with high probability the following operations simultaneously for all $u^G \in V(G)$, within $O(Q(\rho))$ rounds:*

1. *Every leader $\ell(u^G)$ sends an $O(\log \overline{n})$-bit message to all the nodes in $S^{G\to\overline{G}}(u^G)$;*
2. *All the nodes in $S^{G\to\overline{G}}(u^G)$ compute an aggregation function on $O(\log \overline{n})$-bit inputs.*

Armed with this connection, our next crucial observation is that the performance of the Laplacian solver of Forster et al. [11] can be parameterized in terms of the complexity of the congested part-wise aggregation problem. Indeed, we revisit and refine the main building blocks of their solver in Section A, leading to our main result below.

Theorem 22 *Consider a weighted $\overline{n}$-node graph $\overline{G}$ for which Assumption 1 holds for some $Q(\rho) = O(\rho^c \mathcal{Q})$, where c is a universal constant and $\mathcal{Q} = \mathcal{Q}(\overline{G})$ is some parameter. Then, we can solve any Laplacian system after $\overline{n}^{o(1)}\mathcal{Q}\log(1/\varepsilon)$ rounds.*

Combining this theorem with Corollary 16 and Lemma 19 yields the following immediate consequences.
Lower bound in Supported-CONGEST. Finally, we complement our positive results with an almost-matching lower bound on any graph $\overline{G}$, applicable even under the Supported-CONGEST model, thereby establishing universal optimality up to an $\overline{n}^{o(1)}$ factor. Our reduction leverages the refined hardness result established by Haeupler et al. [18] for the *spanning connected subgraph* problem [12]. In this problem a subgraph $\overline{H}$ of $\overline{G}$ is specified with nodes knowing all of the incident edges belonging to $\overline{H}$. The goal is to let every node learn whether $\overline{H}$ is connected and spans the entire network.

Theorem 23 ([18]) *Let $\mathcal{A}$ be any algorithm which is always correct with probability*[7] *at least $\frac{2}{3}$ for the spanning connected subgraph problem, and $T(\overline{G}) = \max_{\mathcal{I}} T_{\mathcal{A}}(\mathcal{I}; \overline{G})$ be the worst-case round-complexity of $\mathcal{A}$ under $\overline{G}$. Then, $T(\overline{G}) = \widetilde{\Omega}(\mathrm{SQ}(\overline{G}))$.*

In this context, we show that a Laplacian solver can be leveraged to solve the spanning connected subgraph problem, leading to the following lower bound.

Proof First of all, as pointed out in [11, Theorem 2], it suffices to establish the lower bound for a high-precision solver, i.e. for a sufficiently small $\varepsilon = 1/\mathrm{poly}(\overline{n})$. Indeed, a low-accuracy solver ($\varepsilon \leq \frac{1}{2}$) can always be "boosted" with only an $O(\log \overline{n})$ overhead in the overall complexity.

In this context, let $\overline{H}$ be the input to the spanning connected subgraph problem. We construct a resistor network H' so that $\boldsymbol{r}(e) = 1$ if $e \in E(\overline{H})$, and $\boldsymbol{r}(e) = \overline{n}^4$ for every edge $e \notin E(\overline{H})$. Moreover, let us select arbitrarily a node $v \in V(\overline{G})$. The key idea of the proof is to consider as input

[7] Note that Haeupler et al. [18] only proved this for always-correct algorithms with probability 1, but the extension we claim here follows readily from their argument.

to the Laplacian solver a vector $\boldsymbol{b} \in \mathbb{R}^{\overline{n}}$ such that $\boldsymbol{b}(u) = -1$ for all $u \in V(\overline{G})\backslash\{v\}$, while $\boldsymbol{b}(v) = \overline{n} - 1$.

To analyze the output of that Laplacian system, we first analyze the simpler Laplacian system with input a vector $\boldsymbol{\chi}_{v,u} \in \mathbb{R}^{\overline{n}}$ for which the coordinate corresponding to node v is 1; the coordinate corresponding to node u is -1; and any other coordinate is set to 0. We recall the following well-known facts.

Fact 24 *Let $\boldsymbol{\phi} = \mathcal{L}(H')^{\dagger}\boldsymbol{\chi}_{v,u}$. Then, for any node $w \in V(\overline{G})$ it holds that $\boldsymbol{\phi}(v) \geq \boldsymbol{\phi}(w) \geq \boldsymbol{\phi}(u)$.*

In the statement above, we use $\mathcal{L}(H')^{\dagger}$ to denote the *Moore-Penrose pseudo-inverse* of matrix $\mathcal{L}(H')$.

Fact 25 *Let $\boldsymbol{\phi} = \mathcal{L}(H')^{\dagger}\boldsymbol{\chi}_{v,u}$. Then, the $v-u$ effective resistance is such that $\mathrm{res}_{H'}(v, u) = \boldsymbol{\phi}(v) - \boldsymbol{\phi}(u)$.*

As argued by Forster et al. [11], the output of the Laplacian with input $\boldsymbol{\chi}_{v,u}$ and a sufficiently small error $\varepsilon = 1/\operatorname{poly}(\overline{n})$ can be used to determine whether v and u are connected. Indeed, the following arguments have been extracted from their lower bound.

Claim 26 *If u and v are connected in $\overline{H}$ it follows that $\mathrm{res}_{H'}(v, u) \leq \overline{n} - 1$.*

Proof It is well-known that the effective resistances satisfy the triangle inequality. Moreover, given that v and u are connected in $\overline{H}$, it follows that there exists a path of length at most $\overline{n} - 1$ in H' so that every edge has resistance 1 (by construction of the resistor network H'). As a result, the triangle inequality implies that $\mathrm{res}_{H'}(v, u) \leq \overline{n} - 1$. □

Claim 27 *If v and u are not connected in $\overline{H}$ it follows that $\mathrm{res}_{H'}(v, u) \geq \overline{n}^2$.*

Proof Suppose that $e_1, \ldots, e_k$ are the edges leaving the connected component of v in $\overline{H}$, for some $k \leq \overline{n}^2$. Then, the Nash-Williams inequality implies that

$$\mathrm{res}_{H'}(v, u) \geq \frac{1}{\sum_{i=1}^{k} \frac{1}{\boldsymbol{r}(e_i)}} \geq \overline{n}^2,$$

by construction of the resistor network. □

The next step of the proof is to incorporate in the analysis the error of the solver. To this end, let $\boldsymbol{\phi}'$ be an ε-approximate solution to the linear system $\mathcal{L}(H')\boldsymbol{\phi} = \boldsymbol{\chi}_{v,u}$ in the sense that

$$\|\boldsymbol{\phi}' - \mathcal{L}(H')^{\dagger}\boldsymbol{\chi}_{v,u}\|_{\mathcal{L}(H')} \leq \varepsilon\|\boldsymbol{\chi}_{v,u}\|_{\mathcal{L}(H')^{\dagger}} = \varepsilon\sqrt{\mathrm{res}_{H'}(v, u)}.$$

Moreover, since the Laplacian matrix has integer resistances up to range $\operatorname{poly}(\overline{n})$, it follows that for any $\boldsymbol{x}$, $\|\boldsymbol{x}\|_{\infty} \leq \operatorname{poly}(\overline{n})\|\boldsymbol{x}\|_{\mathcal{L}}$. Thus, by setting $\varepsilon = 1/\operatorname{poly}(\overline{n})$ to be sufficiently small, we have that

$$\mathrm{res}_{H'}(v, u) - \frac{1}{\overline{n}} \leq \boldsymbol{\phi}'(v) - \boldsymbol{\phi}'(u) \leq \mathrm{res}_{H'}(v, u) + \frac{1}{\overline{n}}.$$

Now we will use these bounds to argue about the initial Laplacian system with input vector $\boldsymbol{b}$. By linearity, a solution of the Laplacian system with input $\boldsymbol{b}$ can be expressed as the sum of solutions of Laplacians with input $\boldsymbol{\chi}_{v,u}$ over all $u \in V(\overline{G})\backslash\{v\}$. Next, we let $\boldsymbol{\phi} = \mathcal{L}(H')^{\dagger}\boldsymbol{b}$, and $\boldsymbol{\phi}'$ be the output of the Laplacian solver for a sufficiently small $\varepsilon = 1/\operatorname{poly}(\overline{n})$. Our analysis distinguishes between the following cases.

Case I. Suppose that $\overline{H}$ is connected. In turn, this implies that v is connected with any node $u \in V(\overline{G})$. As a result, it follows from Fact 24, Fact 25 and Claim 26 that for any node u,

$$\boldsymbol{\phi}'(v) - \boldsymbol{\phi}'(u) \leq (\overline{n} - 1)^2 + 1. \tag{1}$$

Case II. In the contrary case, there must be node u such that v and u are disconnected on $\overline{H}$. By Claim 27 and Fact 24 this yields that

$$\boldsymbol{\phi}'(v) - \boldsymbol{\phi}'(u) \geq \overline{n}^2 - 1. \tag{2}$$

Thus, (1) and (2) imply that the output $\boldsymbol{\phi}'$ of the Laplacian solver contains enough information to determine whether $\overline{H}$ is connected or not since $\overline{n}^2 - 1 > (\overline{n} - 1)^2 + 1$ for any $\overline{n} \geq 2$.

To leverage this in the CONGEST model we proceed as follows. First, node v sends to every other node in the graph its own part of the output from the Laplacian solver. This step can be clearly completed after $D(\overline{G})$ rounds. Then, each node u inspects whether the value $\boldsymbol{\phi}'(v) - \boldsymbol{\phi}'(u)$ is larger than $\overline{n}^2 - 1$. In that case, node u can transmit this information to the entire network; this step is easily seen to be implementable in $D(\overline{G})$ rounds. As a result, assuming that $\mathrm{SQ}(\overline{G}) \geq 3D(\overline{G})$, the proof follows immediately from Theorem 23. But the contrary case is also immediate since on *any* topology solving a Laplacian system trivially requires $\Omega(D(\overline{G}))$ rounds. This completes the proof. □

5 Conclusions

In this paper, we have established almost universally optimal Laplacian solvers for both the (Supported-)CONGEST and the HYBRID model. One of our main technical contributions was to introduce and study a congested generalization of the standard part-wise aggregation problem, which we believe may find further applications beyond the Laplacian paradigm in the future. For example, one candidate problem would be to refine the distributed algorithm for max-flow due to Ghaffari et al. [48]. We also hope that our accelerated Laplacian solvers will be used as a basic primitive for obtaining improved distributed algorithms for other fundamental optimization problems as well.

Funding Open Access funding provided by Carnegie Mellon University. Bernhard Haeupler was supported in part by NSF grants CCF-1814603, CCF-1910588, NSF CAREER award CCF-1750808, a Sloan Research Fellowship, funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation program (ERC grant agreement 949272), and the Swiss National Foundation (project grant 200021-184735). Goran Zuzic was supported in part by the Swiss National Foundation (project grant 200021-184735). Themis Gouleakis was supported in part by an NRF Fellowship for AI (R-252-000-A33-133). Part of the work was done while visiting the Simons Institute for Theory of Computing.

Appendix A: The Laplacian solver

In this section, we describe the basic building blocks of the distributed Laplacian solver of Forster et al. [11]. Our goal will be to cast their guarantees within our more general framework, leading to the proof of Theorem 22. First, let us introduce some further concepts and notation related to Laplacian systems.

Definition 8 (Schur Complement) For a symmetric matrix $\mathbf{A} \in \mathbb{R}^{n \times n}$ and a partition of $[n]$ into $\mathcal{T}$ and $\mathcal{S}$, permute the rows and columns of $\mathbf{A}$ such that

$$\mathbf{A} = \begin{bmatrix} \mathbf{A}_{[\mathcal{S},\mathcal{S}]} & \mathbf{A}_{[\mathcal{S},\mathcal{T}]} \\ \mathbf{A}_{[\mathcal{T},\mathcal{S}]} & \mathbf{A}_{[\mathcal{T},\mathcal{T}]} \end{bmatrix}.$$

Then, the *Schur complement* of $\mathbf{A}$ onto $\mathcal{T}$ is defined as $\mathbf{SC}(\mathbf{A}, \mathcal{T}) := \mathbf{A}_{[\mathcal{T},\mathcal{T}]} - \mathbf{A}_{[\mathcal{T},\mathcal{S}]}\mathbf{A}^{\dagger}_{[\mathcal{S},\mathcal{S}]}\mathbf{A}_{[\mathcal{S},\mathcal{T}]}$, where we recall that $\mathbf{M}^{\dagger}$ denotes the Moore-Penrose pseudo-inverse of matrix $\mathbf{M}$. For a graph G and a subset $\mathcal{T} \subseteq V(G)$, we will write $\mathbf{SC}(G, \mathcal{T}) := \mathbf{SC}(\mathcal{L}(G), \mathcal{T})$.

A.1 The Laplacian building blocks

To keep the exposition reasonably self-contained, here we review the basic ingredients of the distributed Laplacian solver developed by Forster et al. [11]. Our main goal is to extend their guarantees under Assumption 1. Then, we will combine these pieces in Section A.2 to complete the construction.

A.1.1 Ultra-sparsification

As is standard in the Laplacian paradigm, we will require a preconditioner in the form of an *ultra-sparsifier*. In particular, the following lemma is established in Section B.2, and it is a refinement of [11, Lemma 4.9]:

Lemma 28 (Ultra-Sparsification) *Consider an n-node m-edge graph G which is ρ-minor distributed into an $\overline{n}$-node communication network $\overline{G}$ for which Assumption 1 holds for some $Q = Q(\rho)$. Then,* UltraSparsify(G, k) *takes as input a parameter k and returns after $n^{o(1)} Q(\rho)$ rounds a graph H such that*

1. *H is a subgraph of G;*
2. *H has $n - 1 + m 2^{O(\sqrt{\log n \log\log n})}/k$ edges;*
3. *$\mathcal{L}(G) \preceq \mathcal{L}(H) \preceq k\mathcal{L}(G)$.*

Moreover, the algorithm returns $\widehat{G}, \mathbf{Z}_1, \mathbf{Z}_2, C$ such that

1. *$\widehat{G}$ 1-minor distributes into H such that $\widehat{G} = \mathbf{SC}(H, C)$, with $|C| = m 2^{O(\sqrt{\log n \log\log n})}/k$;*
2. *The operators $\mathbf{Z}_1$ and $\mathbf{Z}_2$ can be evaluated in $O(Q(\rho) \log n)$ rounds, and are such that*

$$\mathcal{L}(H)^{\dagger} = \mathbf{Z}_1^T \begin{bmatrix} \mathbf{Z}_2 & 0 \\ 0 & \mathcal{L}(\widehat{G})^{\dagger} \end{bmatrix} \mathbf{Z}_1.$$

Let us briefly review the pieces required for this lemma. First, we need the distributed implementation of the low-stretch spanning tree algorithm of Alon et al. [49], which is due to Ghaffari et al. [48]. Then, this spanning tree is augmented with off-tree edges based on the sampling procedure of Koutis et al. [50], leading to a graph with a spectral approximation guarantee with respect to the original graph. Finally, the parallel elimination procedure of Blelloch et al. [38] is used to perform a series of contractions, leading to a subset with size analogous to the number of off-tree edges. We revisit these steps in detail in Section B.2.

A.1.2 Sparsified cholesky

The next building block is the sparsified Cholesky algorithm of Kyng et al. [51], which manages to effectively eliminate in every iteration a non-negligible fraction of the nodes. In the distributed context, we state the following lemma which is a refinement of [11, Lemma 4.10].

Lemma 29 (Sparsified Cholesky) *Let G be an n-node graph ρ-minor distributed into a communication network $\overline{G}$ for which Assumption 1 holds for some $Q = Q(\rho)$. Then, for a given parameter d and error ε, the algorithm* Eliminate (G, d, ε) *runs in $O(Q(\rho)(\log^c n/\varepsilon^c)^d)$ rounds, where c rep-*

resents some universal constant, and returns a subset $\mathcal{T} \subset V(G)$ and access to operators $\mathbf{Z}_1$ and $\mathbf{Z}_2$ such that

1. $|\mathcal{T}| \le (49/50)^d |V(G)|$;
2. *The operators $\mathbf{Z}_1, \mathbf{Z}_1^T, \mathbf{Z}_2$ can be applied to vectors in $O(Q(\rho)(\log^c n/\varepsilon^c)^d)$ rounds;*
3.

$$(1-\varepsilon)^d \mathcal{L}(G)^\dagger \preceq \mathbf{Z}_1^T \begin{bmatrix} \mathbf{Z}_2 & 0 \\ 0 & \mathbf{SC}(G,\mathcal{T})^\dagger \end{bmatrix} \mathbf{Z}_1 \preceq (1+\varepsilon)^d \mathcal{L}(G)^\dagger.$$

This lemma is established based on a distributed implementation of the *sparsified Cholesky* algorithm of Kyng et al. [51]. In particular, the Cholesky decomposition essentially reduces solving a Laplacian to inverting (i) any sub-matrix of the Laplacian induced on a set S, and (ii) the Schur complement on $V \setminus S$. Thus, Kyng et al. [51] initially develop a procedure for identifying an "almost independent" subset of nodes F (more precisely, a *strongly* diagonally dominant subset) for which inverting the Laplacian restricted on F can be done efficiently through preconditioning (e.g. via the *Jacobi method*), while F also contains at least a constant fraction of the nodes. Next, a combinatorial view of the Schur complement based on a certain family of random walks (see [52]) is employed to construct a spectral sparsifier of the Schur complement on $\mathcal{T} = V \setminus F$. This process is then repeated for d iterations, leading to Lemma 29. Several technical challenges that arise are discussed in Section B.3. Next, the main idea is to recurse on the set of terminals $\mathcal{T}$. However, in our context this requires maintaining the invariant that the underlying subgraph is cast as a minor (with a reasonable congestion) of $\overline{G}$. This is ensured in the following subsection.

A.1.3 Minor schur complement

This subsection introduces a subroutine that will be invoked after the ELIMINATE algorithm to return a low-congestion minor based on the set of terminals $\mathcal{T}$ returned by ELIMINATE; while doing so, the algorithm will incur a small overhead in the spectral guarantee, and a limited growth in the number of nodes with respect to $\mathcal{T}$. This increase will be eventually negligible due to the selection of parameter d in ELIMINATE. In this context, the following lemma is a refinement of [11, Theorem 3].

Lemma 30 *Let G be an n-node graph ρ-minor distributed into an $\overline{n}$-node communication network for which Assumption 1 holds for some $Q = Q(\rho)$. Then, for an error parameter $0 < \varepsilon < 0.1$ and a subset $\mathcal{T}$ of nodes, the algorithm* APPROXSC *returns with high probability a graph H as a ρ-minor distribution into $\overline{G}$ such that*

1. $\mathcal{T} \subseteq V(H)$;
2. *H has $O(|\mathcal{T}| \log^2 n/\varepsilon^2)$ edges;*
3. $\mathbf{SC}(H, \mathcal{T}) \approx_\varepsilon \mathbf{SC}(G, \mathcal{T})$.

This algorithm requires $O(\log^{10} n/\varepsilon^3)$ calls to a distributed Laplacian solver to accuracy $1/\operatorname{poly}(n)$ on graphs that 2ρ-minor distribute into $\overline{G}$, and an overhead of $O(Q(\rho)\log^{10} \overline{n}/\varepsilon^3)$ rounds.

This result builds upon the work of Li and Schild [53], who (roughly speaking) established that randomly contracting an edge with probability equal to its leverage score (and otherwise deleting) would suffice. In the distributed context, Forster et al. [11] devise a parallelized implementation of this scheme based on the *localization of electrical flows* [54]. More precisely, they manage to identify a non-negligible subset of edges—which they refer to as *steady edges*—with small mutual (electrical) "correlation", allowing for independent (and hence highly parallelized) contractions/deletions within this set. This approach employs the recursive and sketching-based method of random projections due to Spielman and Srivastava [55], similarly to the approach of Li and Schild [53], to estimate quantities such as leverage scores and electrical correlation. These steps are carefully reviewed in Section B.4.

A.1.4 Schur complement chain

Finally, let us introduce the concept of a Schur complement chain, and explain how it can be employed to produce a Laplacian solver.

Definition 9 For an n-node graph G, $\{(G_i, \mathbf{Z}_{i,1}, \mathbf{Z}_{i,2}, \mathcal{T}_i)\}_{i=1}^t$ is a (γ, ε)-*Schur complement chain* if the following conditions hold:

1. $G_1 = G$;
2. $\mathcal{T}_i \subset V(G_{i+1}) \subset V(G_i)$ and $\mathbf{SC}(G_i, \mathcal{T}_i) \approx_\varepsilon \mathbf{SC}(G_{i+1}, \mathcal{T}_i)$;
3. $|V(G_{i+1})| \le |V(G_i)|/\gamma$ for $i < t$, and $|V(G_t)| \le \gamma$.
4.

$$(1-\varepsilon)\mathcal{L}(G_i)^\dagger \preceq \mathbf{Z}_{i,1}^T \begin{bmatrix} \mathbf{Z}_{i,2} & 0 \\ 0 & \mathbf{SC}(G_i, \mathcal{T}_i)^\dagger \end{bmatrix} \mathbf{Z}_{i,1} \preceq (1+\varepsilon)\mathcal{L}(G_i)^\dagger.$$

In the sequel, a Schur complement chain will be developed through Lemmas 28 to 30. Next, the following lemma implies a solution to the Laplacian system based on a suitable Schur complement chain.

Lemma 31 ([11]) *Consider an $\overline{n}$-node communication network for which Assumption 1 holds for some $Q = Q(\rho)$, and let $\{(G_i, \mathbf{Z}_{i,1}, \mathbf{Z}_{i,2}, \mathcal{T}_i)\}_{i=1}^t$ be a (γ, ε)-Schur complement chain for an n-node graph G for some $\gamma \ge 2$ and $\varepsilon \le 1/(C \log n)$, for a sufficiently large constant C, such that for all i:*

1. *G_i ρ-minor distributes into $\overline{G}$;*
2. *The linear operators $\mathbf{Z}_{i,1}$ and $\mathbf{Z}_{i,2}$ can be evaluated in at most $\overline{n}^{o(1)}Q(\rho)$ rounds.*

Then, for any given vector $\boldsymbol{b}$, there is an algorithm which computes a vector $\boldsymbol{x}$ in $\overline{n}^{o(1)}Q(\rho)$ rounds such that

$$\|\boldsymbol{x} - \mathcal{L}(G)^{\dagger}\boldsymbol{b}\|_{\mathcal{L}(G)} \leq \varepsilon \log n \|\boldsymbol{b}\|_{\mathcal{L}(G)^{\dagger}}.$$

A.2 Putting everything together

In this subsection, we combine the building blocks we previously developed to establish Theorem 22. The distributed Laplacian solver of Forster et al. [11] is given in Algorithm 1. We also include below the formal version of Theorem 22.

Algorithm 1 Distributed Laplacian Solver [11]: SOLVER(G, ε)

```
Input: An undirected weighted graph $G$
$G'$ := SPECTRALSPARSIFY($G$)
$(G_1, \mathbf{Z}_{1,1}, \mathbf{Z}_{1,2}, \mathcal{T}_1, G_2)$ := ULTRASPARSIFY($G', k$)        ▷ Lemma 28
$\{(G_i, \mathbf{Z}_{i,1}, \mathbf{Z}_{i,2}, \mathcal{T}_i)\}_{i=2}^{t}$ := BUILDCHAIN($G_2, d, \varepsilon, k$)
Solve $\mathcal{L}(G)\boldsymbol{x} = \boldsymbol{b}$ via Chebyshev preconditioning        ▷ Lemma 31
Procedure BUILDCHAIN($G, d, \varepsilon, k$)
if $|V(G)| \leq k$ return $\emptyset$
$(\mathbf{Z}_1, \mathbf{Z}_2, C)$ := ELIMINATE($G, d, \varepsilon$)        ▷ Lemma 29
$H$ := APPROXSC($G, C, \varepsilon$)        ▷ Lemma 30
return $(G, \mathbf{Z}_1, \mathbf{Z}_2, C) \cup$ BUILDCHAIN($H, d, \varepsilon, k$)
```

Theorem 32 (Full-Version of Theorem 22) *Consider a weighted $\overline{n}$-node graph $\overline{G}$ for which Assumption 1 holds for some $Q(\rho) = O(\rho^c \mathcal{Q}(\overline{G}))$, where c is a universal constant and $\mathcal{Q} = \mathcal{Q}(\overline{G})$ is some parameter. Then, for any vector $\boldsymbol{b} \in \mathbb{R}^{\overline{n}}$ stored on its nodes and a sufficiently small error parameter $\varepsilon > 0$,* SOLVER$(\overline{G}, \varepsilon)$ *returns after $\overline{n}^{o(1)}\mathcal{Q}\log(1/\varepsilon)$ rounds a vector $\boldsymbol{x}$ distributed on its nodes such that*

$$\|\boldsymbol{x} - \mathcal{L}(G)^{\dagger}\boldsymbol{b}\|_{\mathcal{L}(G)} \leq \varepsilon \|\boldsymbol{b}\|_{\mathcal{L}(G)}.$$

The proof of this theorem is included in Section B.5. We note that a guarantee with respect to the $\mathcal{L}(G)^{\dagger}$-norm—as in Lemma 31—can be translated to a guarantee in the $\mathcal{L}(G)$-norm. This incurs only a logarithmic multiplicative overhead since it is assumed that the weights are polynomially bounded and the dependence on $1/\varepsilon$ is logarithmic [56, pp. 19–20]. Thus, the overhead is subsumed by the factor $\overline{n}^{o(1)}$.

Appendix B: Omitted proofs

In this section, we include all of the proofs deferred from Section A. We commence by introducing some additional helpful routines.

B.1 Useful routines

Before diving into the proofs of the Laplacian building blocks, it will be useful to present several operations that can be performed efficiently under Assumption 1. We stress that the proofs related to the Laplacian solver closely follow the approach in [11]. Our goal here is to translate them into our more general setting.

Corollary 33 (Matrix–Vector Products) *Consider a matrix $\mathbf{A}$ with non-zeroes supported on the edges of an n-node graph G which is ρ-minor distributed over a communication network $\overline{G}$ for which Assumption 1 holds for some $Q = Q(\rho)$, with values stored in the endpoints of the corresponding edges, and a vector $\boldsymbol{x} \in \mathbb{R}^n$ stored on the nodes $\ell(u^G)$ for $u^G \in V(G)$. Then, we can compute the vector $\mathbf{A}\boldsymbol{x} \in \mathbb{R}^n$ stored on the leader nodes $\ell(u^G)$ for all $u^G \in V(G)$ after $O(Q(\rho))$ rounds with high probability.*

The proof of this corollary follows the one by [11, Corollary 4.4], but nonetheless we state it here for completeness.

Proof of Corollary 33 The first step is to use Assumption 1 to disseminate the coordinates of vector $\boldsymbol{x}$ to the corresponding super-nodes after $Q(\rho)$ rounds; that is, for every $u^G \in V(G)$ the leader $\ell(u^G)$ passes to $S^{G\to\overline{G}}(u^G)$ the corresponding coordinate. Then, every node performs locally all the multiplications for its corresponding indices, and after ρ rounds the node can deliver this information to the corresponding super-node. Observe that this is possible because $\mathbf{A}$ is supported on edges of G, and Definition 7 imposes an edge-congestion bound. Finally, we invoke again Assumption 1 to sum all of the values of each super-node to the leader node, which gives the desired output requirement. □

Another important corollary of Assumption 1 is that we can simulate the spectral sparsification algorithm of Koutis (henceforth SPECTRALSPARSIFY) on G [57]:

Corollary 34 (Spectral Sparsification) *Consider an n-node graph G that ρ-minor distributes into an $\overline{n}$-node communication network $\overline{G}$ for which Assumption 1 holds for some $Q = Q(\rho)$. Then, for any $0 < \varepsilon < 0.1$ we can implement the* SPECTRALSPARSIFY *algorithm of Koutis for G after $O(Q(\rho)\log^7 n/\varepsilon^2)$ rounds, which returns with high probability a graph $\widetilde{G}$ distributed as a ρ-minor into $\overline{G}$ such that*

- *$\mathcal{L}(G) \approx_{\varepsilon} \mathcal{L}(\widetilde{G})$ (Spectral approximation);*
- *$\widetilde{G}$ is a reweighted subgraph of G with $O(n\log^6 n/\varepsilon^2)$ edges in expectation.*

The proof of this corollary is fairly simple (see [11, Corollary 4.4]), but we give a sketch for completeness.

Proof of Corollary 34 The SPECTRALSPARSIFY algorithm of Koutis iteratively uses the spanner scheme of Baswana and

Sen [58]. The latter algorithm gradually grows clusters. In particular, in each round clusters are sampled at random—a "leader" node determines whether the cluster is included in the sample, and then forwards the information to the rest of the cluster. Then, nodes compare the weights of their incident edges to decide whether they will join some cluster, and which incident edges will be added to the spanner. As a result, all the operations of the Baswana-Sen algorithm can be performed via the routine of Assumption 1, and the claim follows. □

Composition of Minors. We also state the extensions of [11, Lemma 4.6] and [11, Corollary 4.7], which are related to the composition of ρ-minors.

Lemma 35 (Composing Minors) *Consider a graph G_2 which is ρ_2-minor distributed into a communication network $\overline{G}$ for which Assumption 1 holds for some $Q = Q(\rho)$, and a graph G_1 which is ρ_1-minor distributed into G_2. Then, we can compute with high probability and after $\widetilde{O}(Q(\rho_1\rho_2))$ rounds a $(\rho_1 \times \rho_2)$-minor distribution of G_1 into G.*

Corollary 36 (Parallel Contraction) *Consider a graph G which is ρ-minor distributed into a communication network $\overline{G}$ for which Assumption 1 holds for some $Q = Q(\rho)$. If F represents a subset of the edges of graph G, we can obtain with high probability a ρ-minor distribution of G/F into $\overline{G}$ in $\widetilde{O}(Q(\rho))$ rounds.*

Recall that the notation G/F implies the graph obtained from G after contracting all the edges in the set $F \subseteq E(G)$.

B.2 Ultra-sparsification: proof of Lemma 28

The first ingredient required for Lemma 28 is a distributed version of the celebrated Alon-Karp-Peleg-West (AKPW) low-stretch spanning tree construction [49], which is due to Ghaffari et al. [48]. We commence by stating their definition of a *distributed N-node cluster graph*, which incidentally was the basis for Definition 7.

Definition 10 (Distributed Cluster Graph, [48]) A distributed N-node cluster graph is a 5-tuple $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{L}, \mathcal{T}, \psi)$ satisfying the following properties:

1. $\mathcal{V} = \{S_1, \ldots, S_N\}$ forms a partition of the node set into N clusters;
2. $\mathcal{E}$ represents a multi-set of (weighted) edges;
3. $\mathcal{L}$ is the set of leaders such that every cluster S_i has *exactly* one leader $\ell_i \in \mathcal{L}$. The ID of the leader node will also serve as the ID of the cluster, while it is assumed that nodes know the ID of their leader, as well as the size of their cluster;
4. $\mathcal{T} = \{T_1, \ldots, T_N\}$ is a set of cluster trees such that each cluster tree $T_i = (S_i, E_i)$ is a (rooted) spanning tree of the induced subgraph $G[S_i]$ of G, with root the leader of the cluster $\ell_i \in S_i$ (observe that this implies that the subgraph induced by each cluster S_i is connected);
5. $\psi : \mathcal{E} \to E$ is a bijective function that maps every edge $\{S_i, S_j\} \in \mathcal{E}$ to some edge $\{u_i, u_j\} \in E$ connecting the corresponding clusters; i.e., it holds that $u_i \in S_i$ and $u_j \in S_j$. It is assumed that the two nodes u_i and u_j know that the edge $\{u_i, u_j\}$ is used to connect their respective clusters, as well as its weight.

Having introduced the concept of a distributed cluster graph, we state the following lemma, which is a direct corollary of the communication primitives we previously described.

Lemma 37 *Let $G = (V, E)$ be an n-node graph ρ-minor distributed into an $\overline{n}$-node communication network $\overline{G} = (\overline{V}, \overline{E})$ for which Assumption 1 holds for some $Q = Q(\rho)$. If $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{L}, \mathcal{T}, \psi)$ is a distributed cluster graph for G, the following operations can be performed in $\widetilde{O}(Q(\rho))$ rounds:*

1. *The leader ℓ_i of each cluster S_i broadcasts an $O(\log \overline{n})$-bit message to every node in S_i;*
2. *Computing aggregation functions on $O(\log \overline{n})$-bit inputs simultaneously for all clusters, assuming the tree T_i is known.*

Proof The definition of a distributed N-node cluster graph (Definition 10) implies that $\mathcal{G}$ is 1-minor distributed over G, and in turn ρ-minor distributed into $\overline{G}$. Note that the induced distributed mapping can be obtained using $\widetilde{O}(Q(\rho))$ rounds of communication by virtue of Lemma 35. Thus, Assumption 1 leads to the desired claim. □

As a result, it follows that the SPLITGRAPH algorithm of Ghaffari et al. [48] can be simulated on a graph G ρ-minor distributed into $\overline{G}$ after $n^{o(1)} Q(\rho)$ communication rounds—under Assumption 1. In particular, this observation directly gives a distributed construction of a low-stretch spanning tree:

Lemma 38 ([11, 48]) *Consider an n-node m-edge graph G which is ρ-minor distributed into an $\overline{n}$-node communication network $\overline{G}$ for which Assumption 1 holds for some $Q = Q(\rho)$. Then, we can construct a spanning tree T of G after $n^{o(1)} Q(\rho)$ rounds such that the nodes know upper bounds on the corresponding stretches that sum to at most $m 2^{O(\sqrt{\log n \log \log n})}$.*

Importantly, it turns out that the guarantee of Lemma 38 suffices to sample edges by stretch, as implied by the following lemma.

Lemma 39 ([50]) *Consider an n-node graph G and a tree T such that the nodes know upper bounds on the corresponding* *stretches that sum up to α. Then, for any parameter k there is a sampling procedure, implementable locally, that gives a graph H which satisfies with high probability the following:*

1. *$\mathcal{L}(G) \preceq \mathcal{L}(H) \preceq k\mathcal{L}(G)$;*
2. *H contains the edges of T and $O(\alpha \log n/k)$ additional edges.*

The final step for establishing Lemma 28 uses the parallel elimination procedure of Blelloch et al. [38], which requires a logarithmic number of rounds under the PRAM model of computation. Thus, we can show the following lemma:

Lemma 40 *Consider an n-node graph H which is ρ-minor distributed into an $\overline{n}$-node communication network $\overline{G}$ for which Assumption 1 holds for some $Q = Q(\rho)$. Moreover, let T be a spanning tree of H and W be the set of off-tree edges of H with respect to T. Then, there is an algorithm which runs in $O(Q(\rho)\log n)$ rounds and returns a graph $\widehat{G}$, 1-embeddable into H, satisfying the following:*

1. *$\widehat{G}$ contains $O(|W|)$ nodes and edges;*
2. *There are operators $\mathbf{Z}_1$ and $\mathbf{Z}_2$, which can be evaluated in $O(Q(\rho)\log n)$ rounds, such that*

$$\mathcal{L}(H)^\dagger = \mathbf{Z}_1^T \begin{bmatrix} \mathbf{Z}_2 & 0 \\ 0 & \mathcal{L}(\widehat{G})^\dagger \end{bmatrix} \mathbf{Z}_1.$$

With these pieces in place, Lemma 28 follows directly from Lemmas 38 to 40.

B.3 Sparsified Cholesky: proof of Lemma 29

The proof of Lemma 29 mainly relies on a distributed implementation of the *Schur complement chain* (SCC) construction of Kyng et al. [51]. In particular, the first step is to formalize a notion of almost-independence:

Definition 11 A matrix $\mathbf{M}$ is α-diagonally dominant (henceforth α-DD) if

$$\mathbf{M}_{i,i} \geq (1+\alpha)\sum_{j\neq i}|\mathbf{M}_{i,j}|, \quad \forall i.$$

Moreover, an index set F is α-DD if $\mathbf{M}_{[F,F]}$ is α-DD.

An important observation is that computing the inverse $\mathbf{M}^{-1}[F, F]$ for an α-DD set can be efficiently performed using a preconditioned gradient descent method. In this context, Kyng et al. [51] give a simple sampling algorithm for finding "large" α-DD sets given a Laplacian matrix. More precisely, their algorithm initially selects a random subset of nodes, and then it filters out these which do not met the condition of Definition 11. This leads to the following:

Lemma 41 *Let G be an n-node graph ρ-minor distributed into a communication network $\overline{G}$ for which Assumption 1 holds for some $Q = Q(\rho)$. Then, if $\mathcal{L}$ is the Laplacian matrix of G and $\alpha \geq 0$ some parameter, there is an algorithm which computes an α-DD subset F of $\mathcal{L}$ of size at least $n/(8(1+\alpha))$ in $O(Q(\rho)\log n)$ rounds with high probability.*

Indeed, the algorithm of Kyng et al. [51] determines an α-DD subset of size $n/(8(1+\alpha))$, while the round-complexity guarantee follows similarly to the proof in [11, Lemma 6.7]. Here we should note that the global aggregation steps required in the distributed implementation of [11, Lemma 6.7] can be trivially performed in $O(Q(1))$ rounds.

The next step is to construct an operator that approximates $\mathcal{L}^{-1}_{[F,F]}$, where F is an α-DD set, and can be efficiently applied to vectors. This is ensured by the following lemma:

Lemma 42 ([11]) *Let G be a graph ρ-minor distributed into an $\overline{n}$-node communication network $\overline{G}$ for which Assumption 1 holds for some $Q = Q(\rho)$. Moreover, let $\mathcal{L}$ be the Laplacian matrix associated with G, and F be a subset of $V(G)$ such that $\mathcal{L}_{[F,F]}$ is α-DD for some $\alpha \geq 4$. Then, for any vector $\boldsymbol{b}$ stored on the leaders of the super-nodes, there is an algorithm which returns in $O(Q(\rho)\log(1/\varepsilon))$ rounds the vector $\mathbf{Z}\boldsymbol{b}$ stored on the same nodes, where $\mathbf{Z}$ is a linear operator such that*

$$\mathcal{L}_{[F,F]} \preceq \mathbf{Z}^{-1} \preceq \mathcal{L}_{[F,F]} + \varepsilon \cdot \mathbf{SC}(\mathcal{L}, F),$$

for any sufficiently small $\varepsilon > 0$.

Again, this lemma follows from the guarantee of Kyng et al. [51] regarding the *Jacobi procedure*, as well as by directly adapting the distributed implementation of Forster et al. [11] using Corollary 33.
Approximating the Schur Complement. Moreover, α-DD sets will be useful in the approximation of the Schur complement induced by the complementary subset of nodes. First, let us recall a combinatorial view of the Schur complement as a Laplacian matrix with weights estimated by certain random walks:

Lemma 43 ([52]) *Let G be an n-node weighted graph and a subset of nodes $\mathcal{T}$. Moreover, consider parameters $0 < \varepsilon < 1$ and $\mu = O(\log n/\varepsilon^2)$. If H is an initially empty graph, repeat for every edge $\{u, v\} \in E(G)$ and for μ iterations the following procedure:*

1. *Simulate a random walk starting from u until it first hits $\mathcal{T}$ at some node t_1;*
2. *Simulate a random walk starting from v until it first hits $\mathcal{T}$ at some node t_2;*
3. *Combine these two walks to get a walk $t_1 = u_0, \ldots, u_\ell = t_2$, where ℓ is the length of the combined walk.*

4. Add the edge $\{t_1, t_2\}$ to H with weight

$$\frac{1}{\mu \sum_{i=0}^{\ell-1} 1/\boldsymbol{w}(u_i, u_{i+1})}.$$

Then, the resulting graph H satisfies $\mathcal{L}(H) \approx_\varepsilon \mathbf{SC}(G, \mathcal{T})$ with high probability.

It should be noted that the random walks in the lemma are implied in the usual sense, wherein a step from a node is taken with probability proportional to the edge-weights of the incident edges. In the sequel, we will compute an α-DD set F via Lemma 41, and then the goal will be to approximate the Schur complement on the set $\mathcal{T} = V \setminus F$. Importantly, given that F is α-DD, we can guarantee that the random walks required in Lemma 43 will be short in expectation. Nonetheless, a challenge that arises in the distributed context—and in particular under the CONGEST model—is that the expected congestion of an edge may by prohibitively large. This issue will be resolved by incorporating new nodes to the terminals whenever they exceed some threshold of congestion. At the same time, however, we also have to limit the node-congestion since G is minor distributed into $\overline{G}$, and we can only deal with limited congestion. This will be addressed by invoking the spectral sparsification algorithm, ensuring that the average degree, and subsequently the congestion, remains limited.

Before we proceed with the approximation of the Schur complement, we note that we can implement the random walks of Lemma 43 in $\widetilde{O}(Q(\rho))$ rounds under Assumption 1, as implied by the approach by Forster et al. [11].

Lemma 44 ([11]) *Let G be an n-node graph ρ-minor distributed into an $\overline{n}$-node communication network $\overline{G}$ for which Assumption 1 holds for some $Q = Q(\rho)$. Moreover, let F be an α-DD set, $\mathcal{T} = V \backslash F$ the set of terminals, $\varepsilon \in (0, 1)$ some error parameter, and $\gamma \geq 1$ the congestion parameter. Then, the algorithm* RandomWalkSchur *runs in $O(\alpha^{-1}\gamma Q(\rho) \log^2 n/\varepsilon^2)$ rounds, and returns a graph H along with its $(\alpha^{-1}\gamma \log n\rho)$-minor distribution into $\overline{G}$ such that*

$$\mathcal{L}(H) \approx_\varepsilon \mathbf{SC}(G, \widehat{\mathcal{T}}),$$

with high probability, where $\widehat{\mathcal{T}} \supseteq \mathcal{T}$ has size at most $n - |F| + O(\alpha^{-1}m\varepsilon^{-2} \log^2 n/\gamma)$.

Proof Let us briefly describe the RandomWalkSchur algorithm. First, we compute the expected congestion of the family of random walks W predicted by Lemma 43 with respect to the set of terminals $\mathcal{T}$. This is done by propagating the congestion to neighbors for $O(\alpha^{-1} \log n)$ steps. Then, we create a new set $\widehat{\mathcal{T}}$ which includes $\mathcal{T}$ as a subset, as well as all the nodes which exceeded the congestion threshold of γ based on the estimation procedure of the previous step. Note that the congestion of a node with respect to W is simply the number of times this particular node participates in some random walk of W. By construction, it follows that the size of $\widehat{\mathcal{T}}$ is $n - |F|$ along with all the nodes that exceeded the congestion threshold of γ. However, since F is an α-DD set it follows that the length of a random walk is $O(\alpha^{-1} \log n)$ with high probability, while for every edge we simulate $\mu = O(\log n/\varepsilon^2)$ random walks (this is related to the concentration of the corresponding random variables, as implied by Lemma 43), in turn implying that the total congestion generated by these random walks is $O(\alpha^{-1}m\varepsilon^{-2} \log^2 n)$. As a result, only $O(\alpha^{-1}m\varepsilon^{-2} \log^2 n/\gamma)$ nodes can have congestion more than γ, verifying the assertion regarding the size of $\widehat{\mathcal{T}}$. Next, the algorithm implements the random walks of Lemma 43, but with respect to the augmented set of terminals $\widehat{\mathcal{T}}$. A Chernoff bound argument assures us that all nodes in $V\backslash\widehat{\mathcal{T}}$ will have congestion $O(\gamma)$ with high probability.

In terms of the distributed implementation, estimating the congestion can be implemented in $O(\alpha^{-1}Q(\rho) \log^2 n/\varepsilon^2)$ rounds; this follows since every walk has length $O(\alpha^{-1} \log n)$ with high probability, and we execute $\mu = O(\log n/\varepsilon^2)$ iterations for every edge. Also note that a single step in the procedure estimating the congestion can be implemented in $O(Q(\rho))$ rounds. Next, the generation of the random walks with respect to the augmented set $\widehat{T}$ can be performed in $O(\alpha^{-1}\gamma Q(\rho) \log^2 n/\varepsilon^2)$ rounds with high probability; this uses the aforementioned guarantee for the congestion. The final step is to minor-distribute the graph H with weights as dictated by Lemma 43. This is done by assigning to the terminals the leaders of all intermediate (non-terminal) nodes. The congestion guarantee ensures that the resulting mapping is an $O(\alpha^{-1}\gamma \log n\rho)$-minor distribution into $\overline{G}$. □

Proof of Lemma 29 The Eliminate algorithm proceeds in d rounds, initializing $\mathbf{M}^{(0)}$ to be an ε-spectral sparsifier of $\mathcal{L}(G)$ (recall Corollary 34). In every round $i \geq 1$, (i) we compute an α-DD set F_i with $\alpha := 4$; (ii) we employ Lemma 42 to have access to an operator that approximates $\mathbf{M}^{(i-1)}_{[F,F]}$; and (iii) we compute an ε-spectral sparsifier $\mathbf{M}^{(i)}$ of the Schur complement $\mathbf{SC}(\mathbf{M}^{(i-1)}, \widehat{\mathcal{T}_i})$ approximated via Lemma 44; here, $\widehat{\mathcal{T}_i} = \widehat{\mathcal{T}}_{i-1} - F_i + U_i$, where U_i represents the set of extra nodes added to ensure low congestion. In particular, Lemma 44 is invoked with congestion parameter $\gamma := 1000C\alpha^{-1} \log^8 n/\varepsilon^4$, where C is a sufficiently large constant. The sparsification algorithm of Koutis (Corollary 34) tells us that the number of edges will be $m = (n \log^6 n/\varepsilon^2)$, in turn implying that the number of nodes drops by at least a multiplicative factor of 49/50.

In terms of the distributed implementation, notice that due to the selection of the parameters the approximation of the Schur complement (Lemma 44) can be performed in $O(Q(\rho) \log^{10} n/\varepsilon^6)$ rounds. Next, the spectral sparsification step can be implemented in $O(Q(\rho') \log^7 n/\varepsilon^2)$, where $\rho' =$

$\alpha^{-1}\gamma \log n\rho = O(\log^9 n/\varepsilon^4)\rho$. Thus, by virtue of Assumption 1 we can infer that $Q(\rho') = O(\log^{c'} n/\varepsilon^{c'})Q(\rho)$, where c' is some universal constant. Thus, after d iterations the cost of these operations is bounded by $O(Q(\rho)(\log^c n/\varepsilon^c)^d)$, where c is some universal constant. Finally, the error guarantee follows directly from Lemmas 42 to 44, after a direct argument bounding the accumulation of the error. □

B.4 Minor schur complement: proof of Lemma 30

We commence this subsection by introducing the notion of *steady edges*, which are in a sense edges which are mutually "uncorrelated":

Definition 12 ([11]) A stochastic subset of edges $Z \subseteq E$ is called (α, δ)-*steady* with respect to an m-edge graph H if

1. $\mathbb{E}_Z\left[\sum_{e\in Z} \boldsymbol{r}(e)^{-1}\boldsymbol{b}(e)\boldsymbol{b}(e)^T\right] \preceq \alpha\mathcal{L}(H)$;
2. For all $e \in Z$ we have $\sum_{e\neq f\in Z} \frac{|\boldsymbol{b}(e)^T\mathcal{L}(H)^\dagger\boldsymbol{b}(f)|}{\sqrt{\boldsymbol{r}(e)}\sqrt{\boldsymbol{r}(f)}} \leq \delta$;
3. For all $e \in Z$ it holds that

$$\boldsymbol{r}(e)^{-1}\boldsymbol{b}(e)^T\mathcal{L}(H)^\dagger\begin{bmatrix}\mathbf{SC}(H,\mathcal{T}) & 0\\ 0 & 0\end{bmatrix}\mathcal{L}(H)^\dagger\boldsymbol{b}(e) \leq \frac{32|\mathcal{T}|}{m}.$$

In words, the first constraint ensures that no edge will be selected in the steady set with too high of a probability; the second corresponds to the localization constraint, circumscribing the (mutual) correlation of edges within the set; and the final constraint imposes a bound on the variance, and will be used in the martingale analysis (to apply Freedman's inequality). It should be stressed that the existence of such objects is highly non-trivial, and follows from the localization of electrical flows recently shown by Schild et al. [54]. In the distributed setting, the following result will be established:

Lemma 45 ([11]) *Let G be an n-node m-edge graph ρ-minor distributed into $\overline{G}$ for which Assumption 1 holds for some $Q = Q(\rho)$. For a constant $\delta \in (0,1)$ and a subset of terminals $\mathcal{T} \subseteq V(G)$, there exists an algorithm which has access to a distributed Laplacian solver, and returns with high probability a set of at least $\delta m/(2000C\log^2 m)$ edges in expectation which is $(\delta/(1000C\log^2 m), \delta)$-steady, where C is a sufficiently large constant. This algorithm requires $O(\log^2 n)$ calls to a distributed Laplacian solver to $1/\operatorname{poly}(n)$ accuracy on graphs that 2ρ-minor distribute into $\overline{G}$, and $O(Q(\rho)\log^2 n)$ communication rounds.*

The first step towards establishing this lemma is to approximate the correlation of edges within some arbitrary set:

Lemma 46 ([11]) *Let G be an n-node graph with resistances $\boldsymbol{r}$, ρ-minor distributed into a communication network $\overline{G}$ for which Assumption 1 holds for some $Q = Q(\rho)$. Then, there is an algorithm, with access to a distributed Laplacian solver, which for any subset $W \subseteq E(G)$ and any edge $e \in W$ returns with high probability the quantity*

$$\sum_{e\neq f\in W}\frac{|\boldsymbol{b}(e)^T\mathcal{L}(G)^\dagger\boldsymbol{b}(f)|}{\sqrt{\boldsymbol{r}(e)}\sqrt{\boldsymbol{r}(f)}}$$

to within a factor of 2. This algorithm requires $O(\log^2 n)$ calls to a distributed Laplacian solver on graphs that ρ-minor distribute into $\overline{G}$ to accuracy $1/\operatorname{poly}(n)$, and an additional $O(Q(\rho)\log^2 n)$ communication rounds.

The proof of this lemma follows directly from [11, Lemma 5.13], and leverages the ℓ_1-sketch of Indyk [59]. Similarly, a sketch can be employed to estimate the effect of each edge on the Schur complement:

Lemma 47 ([11]) *Let G be an n-node with resistances $\boldsymbol{r}_e$, ρ-minor distributed into a communication network $\overline{G}$ for which Assumption 1 holds for some $Q = Q(\rho)$. Then, for a subset $\mathcal{T} \subseteq V(G)$, there exists an algorithm which returns with high probability an estimate of*

$$\boldsymbol{r}(e)^{-1}\boldsymbol{b}(e)^T\mathcal{L}(G)^\dagger\begin{bmatrix}\mathbf{SC}(G,\mathcal{T}) & 0\\ 0 & 0\end{bmatrix}\mathcal{L}(G)^\dagger\boldsymbol{b}(e)$$

to within a factor of 2. This algorithm requires $O(\log n)$ calls to a distributed Laplacian solver to accuracy $1/\operatorname{poly}(n)$ on graphs that 2ρ-minor distribute into $\overline{G}$, and $O(Q(\rho)\log n)$ communication rounds.

As a result, Lemma 45 is established based on the algorithm FINDSTEADY of Forster et al. [11], with the round complexity guarantee following directly from Lemma 46 and Lemma 47.

The next ingredient is a pre-processing step which ensures that all the edges have leverage scores bounded away from 0 and 1.

Lemma 48 ([53]) *Let G be an n-node graph ρ-minor distributed into a communication network $\overline{G}$ for which Assumption 1 holds for some $Q = Q(\rho)$. If 1.1-approximate leverage scores $\widetilde{\operatorname{lev}}_G(e)$ for the edges in G are known, then there exists a process which returns after $\widetilde{O}(Q(\rho))$ rounds a graph H such that*

1. *H is electrically equivalent to G;*[8]
2. *H is 2ρ-minor distributed into $\overline{G}$;*
3. *All the leverage scores of edges in H are between $[3/16, 13/16]$.*

[8] For the definition of electrical equivalence we refer to the work of Li and Schild [53], as it is not important for the purpose of this work.

Moreover, there exists a procedure which takes as input G and returns in $O(Q(\rho))$ rounds a graph resulting from collapsing paths and parallel edges, and removing non-terminal leaves, along with a ρ-minor distribution into $\overline{G}$.

The distributed implementation of this lemma is fairly simple, and relies on Lemma 35. We will also use the following lemma, which is based on the random projection scheme of Spielman and Srivastava [55]:

Lemma 49 ([11]) *Let G be an n-node graph ρ-minor distributed into a communication network $\overline{G}$ for which Assumption 1 holds for some $Q = Q(\rho)$. Then, there is an algorithm with access to a distributed Laplacian solver which for all edges $e \in E(G)$ approximates the leverage score $\mathrm{lev}_G(e)$ to within a factor of $1+\delta$ with high probability. This algorithm requires $O(\log n/\delta^2)$ calls to a distributed Laplacian solver on graphs which ρ-minor distribute into $\overline{G}$ to accuracy $1/\mathrm{poly}(n)$, as well as $O(Q(\rho)\log n/\delta^2)$ communication rounds.*

The proof of this lemma follows directly from [11, Lemma 5.4], and uses Achliopta's variant of the Johnson-Lindenstrauss lemma [60]. With these pieces at hand, we are ready to describe the algorithm for computing a minor Schur complement. At each iteration we first determine a set of steady edges via Lemma 45. Then, we estimate the leverage scores via the random projection scheme of Lemma 49, and each edge in the steady set is contracted (independently) with probability given by its (approximate) leverage scores; otherwise, the edge is deleted (for this we will use Corollary 36). We also employ Lemma 48 in every iteration to ensure that leverage scores are bounded away from 0 and 1. This process is repeated as long as the number of edges exceeds a threshold, leading to the algorithm ApproxSC [11]. The next theorem was shown by Forster et al. [11] using matrix martingale analysis:

Lemma 50 ([11]) *The algorithm* ApproxSC *takes as input a graph G with a set of terminals $\mathcal{T}$ and an error parameter ε, and returns with high probability a graph H satisfying $|E(H)| = O(|\mathcal{T}|\log^2 n/\varepsilon^2)$ and $\mathbf{SC}(H,\mathcal{T}) \approx_\varepsilon \mathbf{SC}(G,\mathcal{T})$.*

Proof of Lemma 30 First, the algorithm only performs deletions and contractions, implying that it indeed returns a minor. Moreover, the correctness follows directly from Lemma 50. To bound the requirements of the algorithm note that ApproxSC executes $O(\log m/\alpha)$ iterations, where $\alpha := \delta/(1000C\log^2 m) = O(\varepsilon/\log^4 m)$, with high probability. In each iteration the dominant cost in terms of calls to a distributed Laplacian solver follows from the subroutine approximating leverage scores, which requires $O(\log n/\delta^2) = O(\log^5 n/\varepsilon^2)$. Thus, we may conclude that ApproxSC requires $O(\log^{10} n/\varepsilon^3)$ calls to a distributed Laplacian solver. The bound in terms of the round complexity follows similarly. □

B.5 Putting everything together: proof of Theorem 32

Proof The correctness of the algorithm follows directly from Lemmas 28 to 31, so let us focus on the round complexity. By the guarantee of Lemma 28 we know that the UltraSparsify routine returns a graph G_2 such that $|V(G_2)| = |V(G_1)|2^{O(\sqrt{\log n \log\log n})}/k$; this follows since we have sparsified the graph in the first step. Thus, for $k = 2^{(\log \overline{n})^{2/3}}$ we can infer that $|V(G_2)| \leq |V(G_1)|/k^{1-o(1)}$. Next, with regards to the Schur complement chain, Lemmas 29 and 30 imply that $|V(G_{i+1})| \leq |V(G_i)| O(0.98^d \log^2 n/\varepsilon^2)$. Hence, setting $d = 2^{(\log\log\overline{n})^2}$ and $\varepsilon = 1/(\log\overline{n})^2$ gives us that $|V(G_{i+1})| \leq |V(G_i)|2^{-\Theta((\log\log\overline{n})^2)}$.

As a result, BuildChain returns a $(2^{\Theta((\log\log\overline{n})^2)}, \varepsilon)$-Schur complement chain, which in turn implies that this chain has length $O(\log\overline{n}/(\log\log\overline{n})^2)$. Thus, Lemma 31 implies that we can use this chain to produce a solution in $\rho\overline{n}^{o(1)}Q(\rho)$ rounds, where ρ represents the maximum congestion of a graph along the chain; it will be establish that $\rho = \overline{n}^{o(1)}$.

Let $f(n,\rho)$ represent the number of rounds required by Solver on a graph with n nodes which ρ-minor distributes into $\overline{G}$, and $g(n,\rho)$ the number of rounds required by BuildChain with input an n-node graph which ρ-minor distributes into $\overline{G}$. Then, if we ignore lower order terms, it follows that

$$f(n,\rho) = \overline{n}^{o(1)}Q(\rho) + g(n/k^{1-o(1)}, \rho),$$

where we used that $|V(G_2)| \leq |V(G_1)|/k^{1-o(1)}$. Moreover, we have that

$$\begin{aligned} g(n,\rho) &= O\left((\log^c n/\varepsilon^c)^{(\log\log\overline{n})^2} Q(\rho)\right) \\ &\quad + f(n,2\rho)O(\log^{10} n/\varepsilon^3) + g(n/2^{\Theta((\log\log\overline{n})^2)}, \rho) \\ &= \overline{n}^{o(1)}Q(\rho) + \mathrm{polylog}(\overline{n})f(n,2\rho) \\ &\quad + g(n/2^{\Theta((\log\log\overline{n})^2)}, \rho), \end{aligned}$$

where we used that $|V(G_{i+1})| \leq |V(G_i)|2^{-\Theta((\log\log\overline{n})^2)}$, and we ignored lower order terms. As a result, the overall increase in congestion is $2^{O(\log\overline{n}/(\log\log\overline{n})^2)} = \overline{n}^{o(1)}$. That is, all the graphs constructed $(\overline{n}^{o(1)})$-minor distribute into $\overline{G}$. Finally, the theorem follows since by Assumption 1 the dependence of $Q(\rho)$ on ρ is polynomial. □

References

1. Spielman, D.A., Teng, S.: Nearly linear time algorithms for preconditioning and solving symmetric, diagonally dominant linear systems. SIAM J. Matrix Anal. Appl. **35**(3), 835–885 (2014). https://doi.org/10.1137/090771430

2. Koutis, I., Miller, G.L., Peng, R.: Approaching optimality for solving SDD linear systems. SIAM J. Comput. **43**(1), 337–354 (2014). https://doi.org/10.1137/110845914
3. Kelner, J.A., Orecchia, L., Sidford, A., Zhu, Z.A.: A simple, combinatorial algorithm for solving SDD systems in nearly-linear time. In: Symposium on Theory of Computing Conference, STOC'13, 2013, pp. 911–920 (2013). https://doi.org/10.1145/2488608.2488724
4. Kyng, R., Sachdeva, S.: Approximate gaussian elimination for Laplacians - fast, sparse, and simple. In: IEEE 57th Annual Symposium on Foundations of Computer Science, FOCS 2016, pp. 573–582 (2016). https://doi.org/10.1109/FOCS.2016.68
5. Axiotis, K., Madry, A., Vladu, A.: Faster sparse minimum cost flow by electrical flow localization. In: 62nd IEEE Annual Symposium on Foundations of Computer Science, FOCS 2021, pp. 528–539 (2021). https://doi.org/10.1109/FOCS52979.2021.00059
6. Madry, A.: Computing maximum flow with augmenting electrical flows. In: IEEE 57th Annual Symposium on Foundations of Computer Science, FOCS 2016, pp. 593–602 (2016). https://doi.org/10.1109/FOCS.2016.70
7. Brand, J., Lee, Y.T., Nanongkai, D., Peng, R., Saranurak, T., Sidford, A., Song, Z., Wang, D.: Bipartite matching in nearly-linear time on moderately dense graphs. In: 61st IEEE Annual Symposium on Foundations of Computer Science, FOCS 2020, pp. 919–930 (2020). https://doi.org/10.1109/FOCS46700.2020.00090
8. Kelner, J.A., Lee, Y.T., Orecchia, L., Sidford, A.: An almost-linear-time algorithm for approximate max flow in undirected graphs, and its multicommodity generalizations. In: Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2014, pp. 217–226 (2014). https://doi.org/10.1137/1.9781611973402.16
9. Peng, R.: Approximate undirected maximum flows in $O(m\text{polylog}(n))$ time. In: Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2016, pp. 1862–1867 (2016). https://doi.org/10.1137/1.9781611974331.ch130
10. Axiotis, K., Madry, A., Vladu, A.: Circulation control for faster minimum cost flow in unit-capacity graphs. In: 61st IEEE Annual Symposium on Foundations of Computer Science, FOCS 2020, pp. 93–104 (2020). https://doi.org/10.1109/FOCS46700.2020.00018
11. Forster, S., Goranci, G., Liu, Y.P., Peng, R., Sun, X., Ye, M.: Minor sparsifiers and the distributed laplacian paradigm. In: 62nd IEEE Annual Symposium on Foundations of Computer Science, FOCS 2021, pp. 989–999 (2021). https://doi.org/10.1109/FOCS52979.2021.00099
12. Das Sarma, A., Holzer, S., Kor, L., Korman, A., Nanongkai, D., Pandurangan, G., Peleg, D., Wattenhofer, R.: Distributed verification and hardness of distributed approximation. In: Proceedings of the Forty-Third Annual ACM Symposium on Theory of Computing. STOC '11, pp. 363–372 (2011). https://doi.org/10.1145/1993636.1993686
13. Peleg, D., Rubinovich, V.: A near-tight lower bound on the time complexity of distributed mst construction. In: 40th Annual Symposium on Foundations of Computer Science, pp. 253–261 (1999). https://doi.org/10.1109/SFFCS.1999.814597
14. Elkin, M.: Unconditional lower bounds on the time-approximation tradeoffs for the distributed minimum spanning tree problem. In: Proceedings of the Thirty-Sixth Annual ACM Symposium on Theory of Computing. STOC '04, pp. 331–340 (2004). https://doi.org/10.1145/1007352.1007407
15. Ghaffari, M., Haeupler, B.: Distributed algorithms for planar networks II: low-congestion shortcuts, mst, and min-cut. In: Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2016, pp. 202–219 (2016). https://doi.org/10.1137/1.9781611974331.ch16
16. Haeupler, B., Izumi, T., Zuzic, G.: Low-congestion shortcuts without embedding. In: Giakkoupis, G. (ed.) Proceedings of the 2016 ACM Symposium on Principles of Distributed Computing, PODC 2016, pp. 451–460 (2016). https://doi.org/10.1145/2933057.2933112
17. Haeupler, B., Izumi, T., Zuzic, G.: Near-optimal low-congestion shortcuts on bounded parameter graphs. In: Distributed Computing—30th International Symposium, DISC 2016. Lecture Notes in Computer Science, vol. 9888, pp. 158–172 (2016). https://doi.org/10.1007/978-3-662-53426-7_12
18. Haeupler, B., Wajc, D., Zuzic, G.: Universally-optimal distributed algorithms for known topologies. In: STOC '21: 53rd Annual ACM SIGACT Symposium on Theory of Computing, pp. 1166–1179 (2021). https://doi.org/10.1145/3406325.3451081
19. Ghaffari, M., Haeupler, B.: Low-congestion shortcuts for graphs excluding dense minors. In: PODC '21: ACM Symposium on Principles of Distributed Computing, 2021, pp. 213–221 (2021). https://doi.org/10.1145/3465084.3467935
20. Zuzic, G., Goranci, G., Ye, M., Haeupler, B., Sun, X.: Universally-optimal distributed shortest paths and transshipment via graph-based ℓ_1-oblivious routing. In: Proceedings of the 2022 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA), pp. 2549–2579 (2022). https://doi.org/10.1137/1.9781611977073.100. SIAM
21. Ghaffari, M., Zuzic, G.: Universally-optimal distributed exact min-cut. In: PODC '22: ACM Symposium on Principles of Distributed Computing, 2022, pp. 281–291 (2022). https://doi.org/10.1145/3519270.3538429
22. Haeupler, B., Räcke, H., Ghaffari, M.: Hop-constrained expander decompositions, oblivious routing, and distributed universal optimality. In: STOC '22: 54th Annual ACM SIGACT Symposium on Theory of Computing, 2022, pp. 1325–1338 (2022). https://doi.org/10.1145/3519935.3520026
23. Augustine, J., Hinnenthal, K., Kuhn, F., Scheideler, C., Schneider, P.: Shortest paths in a hybrid network model. In: Proceedings of the 2020 ACM-SIAM Symposium on Discrete Algorithms, SODA 2020, pp. 1280–1299 (2020). https://doi.org/10.1137/1.9781611975994.78
24. Chen, T., Gao, X., Chen, G.: The features, hardware, and architectures of data center networks: a survey. J. Parallel Distrib. Comput. **96**, 45–74 (2016). https://doi.org/10.1016/j.jpdc.2016.05.009
25. Wang, G., Andersen, D.G., Kaminsky, M., Papagiannaki, K., Ng, T.S.E., Kozuch, M., Ryan, M.: C-through: Part-time optics in data centers. In: Proceedings of the ACM SIGCOMM 2010 Conference. SIGCOMM '10, pp. 327–338 (2010). https://doi.org/10.1145/1851182.1851222
26. Kar, U.N., Sanyal, D.K.: An overview of device-to-device communication in cellular networks. ICT Express **4**(4), 203–208 (2018). https://doi.org/10.1016/j.icte.2017.08.002
27. Augustine, J., Ghaffari, M., Gmyr, R., Hinnenthal, K., Scheideler, C., Kuhn, F., Li, J.: Distributed computation in node-capacitated networks. In: The 31st ACM on Symposium on Parallelism in Algorithms and Architectures, SPAA 2019, pp. 69–79 (2019). https://doi.org/10.1145/3323165.3323195
28. Rozhon, V., Grunau, C., Haeupler, B., Zuzic, G., Li, J.: Undirected $(1+\epsilon)$-shortest paths via minor-aggregates: near-optimal deterministic parallel and distributed algorithms. In: STOC '22: 54th Annual ACM SIGACT Symposium on Theory of Computing, 2022, pp. 478–487 (2022). https://doi.org/10.1145/3519935.3520074
29. Censor-Hillel, K., Leitersdorf, D., Polosukhin, V.: On sparsity awareness in distributed computations. In: SPAA '21: 33rd ACM Symposium on Parallelism in Algorithms and Architectures, pp. 151–161 (2021). https://doi.org/10.1145/3409964.3461798
30. Kuhn, F., Schneider, P.: Computing shortest paths and diameter in the hybrid network model. In: Proceedings of the 39th Symposium

on Principles of Distributed Computing. PODC '20, pp. 109–118 (2020). https://doi.org/10.1145/3382734.3405719
31. Feldmann, M., Hinnenthal, K., Scheideler, C.: Fast hybrid network algorithms for shortest paths in sparse graphs. In: 24th International Conference on Principles of Distributed Systems, OPODIS 2020. LIPIcs, vol. 184, pp. 31–13116 (2020). https://doi.org/10.4230/LIPIcs.OPODIS.2020.31
32. Censor-Hillel, K., Leitersdorf, D., Polosukhin, V.: Distance computations in the hybrid network model via oracle simulations. In: 38th International Symposium on Theoretical Aspects of Computer Science, STACS 2021. LIPIcs, vol. 187, pp. 21–12119 (2021). https://doi.org/10.4230/LIPIcs.STACS.2021.21
33. Götte, T., Hinnenthal, K., Scheideler, C., Werthmann, J.: Time-optimal construction of overlay networks. In: PODC '21: ACM Symposium on Principles of Distributed Computing, pp. 457–468 (2021). https://doi.org/10.1145/3465084.3467932
34. Kuhn, F., Schneider, P.: Routing schemes and distance oracles in the hybrid model. In: 36th International Symposium on Distributed Computing, DISC 2022. LIPIcs, vol. 246, pp. 28–12822 (2022). https://doi.org/10.4230/LIPIcs.DISC.2022.28
35. Coy, S., Czumaj, A., Feldmann, M., Hinnenthal, K., Kuhn, F., Scheideler, C., Schneider, P., Struijs, M.: Near-shortest path routing in hybrid communication networks. In: 25th International Conference on Principles of Distributed Systems, OPODIS 2021. LIPIcs, vol. 217, pp. 11–11123 (2021). https://doi.org/10.4230/LIPIcs.OPODIS.2021.11
36. Anagnostides, I., Gouleakis, T.: Deterministic distributed algorithms and lower bounds in the hybrid model. In: 35th International Symposium on Distributed Computing, DISC 2021. LIPIcs, vol. 209, pp. 5–1519 (2021). https://doi.org/10.4230/LIPIcs.DISC.2021.5
37. Peng, R., Spielman, D.A.: An efficient parallel solver for SDD linear systems. In: Symposium on Theory of Computing, STOC 2014, pp. 333–342 (2014). https://doi.org/10.1145/2591796.2591832
38. Blelloch, G.E., Gupta, A., Koutis, I., Miller, G.L., Peng, R., Tangwongsan, K.: Nearly-linear work parallel SDD solvers, low-diameter decomposition, and low-stretch subgraphs. Theory Comput. Syst. **55**(3), 521–554 (2014). https://doi.org/10.1007/s00224-013-9444-5
39. Linial, N.: Locality in distributed graph algorithms. SIAM J. Comput. **21**(1), 193–201 (1992)
40. Lotker, Z., Pavlov, E., Patt-Shamir, B., Peleg, D.: MST construction in $O(\log \log n)$ communication rounds. In: SPAA 2003: Proceedings of the Fifteenth Annual ACM Symposium on Parallelism in Algorithms and Architectures, 2003, pp. 94–100 (2003). https://doi.org/10.1145/777412.777428
41. Drucker, A., Kuhn, F., Oshman, R.: On the power of the congested clique model. In: ACM Symposium on Principles of Distributed Computing, PODC '14, pp. 367–376 (2014). https://doi.org/10.1145/2611462.2611493
42. Forster, S., Vos, T.: The Laplacian paradigm in the broadcast congested clique. In: PODC '22: ACM Symposium on Principles of Distributed Computing, Salerno, 2022, pp. 335–344 (2022). https://doi.org/10.1145/3519270.3538436
43. Peleg, D.: Distributed Computing: A Locality-sensitive Approach. Society for Industrial and Applied Mathematics, Philadelphia (2000)
44. Schmid, S., Suomela, J.: Exploiting locality in distributed SDN control. HotSDN 2013—Proceedings of the 2013 ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking, pp. 121–126 (2013). https://doi.org/10.1145/2491185.2491198
45. Ghaffari, M.: Near-Optimal Scheduling of Distributed Algorithms. In: Proceedings of the 2015 ACM Symposium on Principles of Distributed Computing, pp. 3–12 (2015). https://doi.org/10.1145/2767386.2767417
46. Johansson, Ö.: Simple distributed $\delta + 1$-coloring of graphs. Inf. Process. Lett. **70**(5), 229–232 (1999)
47. Haeupler, B., Wajc, D., Zuzic, G.: Network coding gaps for completion times of multiple unicasts. In: 2020 IEEE 61st Annual Symposium on Foundations of Computer Science (FOCS), pp. 494–505 (2020). https://doi.org/10.1109/FOCS46700.2020.00053 . IEEE
48. Ghaffari, M., Karrenbauer, A., Kuhn, F., Lenzen, C., Patt-Shamir, B.: Near-optimal distributed maximum flow: extended abstract. In: Proceedings of the 2015 ACM Symposium on Principles of Distributed Computing, PODC 2015, pp. 81–90 (2015). https://doi.org/10.1145/2767386.2767440
49. Alon, N., Karp, R.M., Peleg, D., West, D.: A graph-theoretic game and its application to the k-server problem. SIAM J. Comput. **24**(1), 78–100 (1995). https://doi.org/10.1137/S0097539792224474
50. Koutis, I., Miller, G.L., Peng, R.: Approaching optimality for solving sdd linear systems. In: 2010 IEEE 51st Annual Symposium on Foundations of Computer Science, pp. 235–244 (2010). https://doi.org/10.1109/FOCS.2010.29
51. Kyng, R., Lee, Y.T., Peng, R., Sachdeva, S., Spielman, D.A.: Sparsified cholesky and multigrid solvers for connection laplacians. In: Proceedings of the Forty-Eighth Annual ACM Symposium on Theory of Computing. STOC '16, pp. 842–850. Association for Computing Machinery, New York, NY, USA (2016). https://doi.org/10.1145/2897518.2897640
52. Durfee, D., Gao, Y., Goranci, G., Peng, R.: Fully dynamic spectral vertex sparsifiers and applications. In: Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing, STOC 2019, pp. 914–925 (2019). https://doi.org/10.1145/3313276.3316379
53. Li, H., Schild, A.: Spectral subspace sparsification. In: 59th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2018, pp. 385–396 (2018). https://doi.org/10.1109/FOCS.2018.00044
54. Schild, A., Rao, S., Srivastava, N.: Localization of electrical flows. In: Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2018. SODA '18, pp. 1577–1584. Society for Industrial and Applied Mathematics, USA (2018). https://doi.org/10.1137/1.9781611975031.103
55. Spielman, D.A., Srivastava, N.: Graph sparsification by effective resistances. In: Dwork, C. (ed.) Proceedings of the 40th Annual ACM Symposium on Theory of Computing, 2008, pp. 563–568 (2008). https://doi.org/10.1145/1374376.1374456
56. Vishnoi, N.: Lx=b. Laplacian solvers and their algorithmic applications. Found. Trends Theor. Comput. Sci. (2012). https://doi.org/10.1561/0400000054
57. Koutis, I.: Simple parallel and distributed algorithms for spectral graph sparsification. In: 26th ACM Symposium on Parallelism in Algorithms and Architectures, SPAA '14, pp. 61–66 (2014). https://doi.org/10.1145/2612669.2612676
58. Baswana, S., Sen, S.: A simple and linear time randomized algorithm for computing sparse spanners in weighted graphs. Random Struct. Algorithms **30**(4), 532–563 (2007). https://doi.org/10.1002/rsa.20130
59. Indyk, P.: Stable distributions, pseudorandom generators, embeddings, and data stream computation. J. ACM **53**(3), 307–323 (2006). https://doi.org/10.1145/1147954.1147955
60. Achlioptas, D.: Database-friendly random projections: Johnson–lindenstrauss with binary coins. J. Comput. Syst. Sci. **66**(4), 671–687 (2003). https://doi.org/10.1016/S0022-0000(03)00025-4