Model-Agnostic Approximation of Constrained Forest Problems

Corinna Coupette
Aalto University, Finland
Max Planck Institute for Informatics, Saarbrücken, Germany

Alipasha Montaseri
Sharif University of Technology, Tehran, Iran

Christoph Lenzen
CISPA Helmholtz Center for Information Security, Saarbrückem, Germany

Abstract

Constrained Forest Problems (CFPs) as introduced by Goemans and Williamson in 1995 capture a wide range of network design problems with edge subsets as solutions, such as Minimum Spanning Tree, Steiner Forest, and Point-to-Point Connection. While individual CFPs have been studied extensively in individual computational models, a unified approach to solving general CFPs in multiple computational models has been lacking. Against this background, we present the *shell-decomposition algorithm*, a *model-agnostic meta-algorithm* that efficiently computes a $(2+\varepsilon)$-approximation to CFPs for a broad class of forest functions. The shell-decomposition algorithm isolates the problem-specific hardness of individual CFPs in a single computational subroutine, breaking the remainder of the computation into fundamental tasks that are studied extensively in a wide range of computational models. In contrast to prior work, our framework is compatible with the use of *approximate* distances.

To demonstrate the power and flexibility of this result, we instantiate our algorithm for three fundamental, NP-hard CFPs (Steiner Forest, Point-to-Point Connection, and Facility Placement and Connection) in three different computational models (Congest, PRAM, and Multi-Pass Streaming). For constant ε, we obtain the following $(2+\varepsilon)$-approximations in the Congest model:

(1) For Steiner Forest specified via input components (SF–IC), where each node knows the identifier of one of k disjoint subsets of V (the input components), we achieve a deterministic $(2+\varepsilon)$-approximation in $\widetilde{\mathcal{O}}(\sqrt{n}+D+k)$ rounds, where D is the hop diameter of the graph, significantly improving over the state of the art.

(2) For Steiner Forest specified via symmetric connection requests (SF–SCR), where connection requests are issued to pairs of nodes $u, v \in V$, we leverage randomized equality testing to reduce the running time to $\widetilde{\mathcal{O}}(\sqrt{n}+D)$, succeeding with high probability.

(3) For Point-to-Point Connection, we provide a $(2+\varepsilon)$-approximation in $\widetilde{\mathcal{O}}(\sqrt{n}+D)$ rounds.

(4) For Facility Placement and Connection, a relative of non-metric Uncapacitated Facility Location, we obtain a $(2+\varepsilon)$-approximation in $\widetilde{\mathcal{O}}(\sqrt{n}+D)$ rounds.

We further show how to replace the $\sqrt{n}+D$ term by the complexity of solving Partwise Aggregation, achieving (near-)universal optimality in any setting in which a solution to Partwise Aggregation in near-shortcut-quality time is known. Notably, all of our concrete results can be derived with relative ease once our model-agnostic meta-algorithm has been specified. This demonstrates the power of our modularization approach to algorithm design.

2012 ACM Subject Classification Theory of computation → Distributed algorithms; Mathematics of computing → Graph algorithms; Theory of computation → Models of computation

Keywords and phrases Distributed Graph Algorithms, Model-Agnostic Algorithms, Steiner Forest

Digital Object Identifier 10.4230/LIPIcs.DISC.2025.25

Related Version *Full Version*: `https://arxiv.org/abs/2407.14536` [10]

Funding *Corinna Coupette*: Supported by *Digital Futures* at KTH Royal Institute of Technology.

1 Introduction

The classic approach to determining the computational complexity of a task consists in deriving upper and lower bounds that are as tight as possible in a particular model of computation. On the upper-bound side, this can result in solutions that are specifically engineered toward the chosen computational model, such that the task at hand needs to be reexamined for every relevant model. Worse still, one might end up with fragile solutions that overcome only the challenges specifically represented by the model under study. At first glance, lower bounds might appear more robust in this regard, since they highlight an obstacle that *any* algorithm needs to overcome. However, many such lower bounds are shown for very specific network topologies that are rarely observed in practice. This might lead to a false sense of success in having classified the complexity of a given task – although the lower bound merely indicates that the parameters used to capture the computational complexity of the task are insufficient. In brief, traditional upper and lower bounds bear two limitations: *model specificity* and *existential optimality*.

A textbook illustration of these limitations is provided by the Steiner Forest problem (SF), which generalizes the well-known Steiner Tree problem (ST). In the classic SF formulation using input components (SF–IC), we are given a weighted graph, along with disjoint subsets of nodes $V_1, \dots, V_k \subseteq V$, and the goal is to determine a minimum-weight subgraph spanning each V_i, $i \in [k]$. This general and fundamental connectivity problem has been studied in depth in the classic centralized model of computation [1, 8, 16, 19, 28]. It is of significant practical importance, both due to direct application, see e.g., [36, Sec. 18.5.5], and being the key building block in variants of Steiner-type problems with broad real-world utility, e.g., Steiner Tree reoptimization [5] and Multicommodity Rent-or-Buy [20]. The state of the art in the CONGEST model is due to Lenzen and Patt-Shamir [30], who presented modified and adapted variants of a well-known centralized algorithm by Agrawal et al. [1]. One might suspect that algorithmic techniques underlying these tailored solutions could be transferred to other computational models, but this cannot be readily determined from their specialized solution (*model specificity*). Furthermore, for any fixed topology, there are inputs such that even their fastest algorithm runs for $\widetilde{\Omega}(\sqrt{n})$ rounds.[1] This is known to be necessary in the worst case (*existential optimality*): the Minimum Spanning Tree (MST) problem is a special case of the Steiner Forest Problem with $k = 1$ and $V_1 = V$, to which the $\widetilde{\Omega}(\sqrt{n})$ lower bound by Das Sarma et al. [11] applies. However, the topology of the lower-bound graph is highly contrived, and the technique of using *low-congestion shortcuts* has been demonstrated to overcome the $\sqrt{n}$ barrier for MST in many more realistic settings [13, 14, 24]. These findings motivate us to revisit the Steiner Forest problem, along with a much broader class of connectivity problems, in an attempt to overcome the abovementioned limitations.

Beyond Model Specificity and Existential Optimality

To go beyond of *model specificity* and *existential optimality*, two paradigms have emerged.

First, numerous works have pushed toward what we term *model agnosticism*, developing algorithms that can be readily instantiated in many computational models [6, 7, 21, 32, 35]. The individual steps of these algorithms are either core tasks like distance computation, identifying connected components, and sorting, which are well-studied across a wide range of models, or they are easily implemented at low cost in any notable model. Hence, we call

[1] We use $\widetilde{\mathcal{O}}$ and $\widetilde{\Omega}$ to suppress factors of $\log^{\mathcal{O}(1)} n$.

these algorithms *model-agnostic (meta-)algorithms*. Naturally, model-agnostic algorithms are more robust against model variations, and by allowing us to plug in optimized model-specific subroutines for the core computational problems, they sacrifice little performance over model-specific solutions. Thus, model-agnostic algorithms can be viewed as model-agnostic reductions to more basic computational tasks. Moreover, since these basic tasks can be solved by model-specific subroutines, they improve whenever progress is made on the current performance bottleneck in a given computational model.

Second, *universal optimality*, which was coined in the context of distributed computing, pushes for the design of topology-adaptive algorithms that are asymptotically worst-case optimal on *every* network topology, i.e., when varying input parameters *other* than the underlying communication graph [15, 27, 37].[2] One might argue that this idea is no different than taking into account more parameters, such as the network diameter, the node connectivity, or any other quantity meaningful for the computational task. However, parametrizing complexity by the input graph is extremely general, subsuming a large number of parameters that one might consider, and capturing any *topology-specific obstacle* for the task at hand in a given computational model.

Our Contributions in Brief

In this work, we study a general class of connectivity problems called *Constrained Forest Problems* (CFPs), introduced by Goemans and Williamson [16], through the lenses of *model agnosticism* and *universal optimality*. Intuitively, a CFP is specified by a Boolean function f that indicates, for each node subset $S \subseteq V$, whether it needs to be connected to its complement, i.e., if the output must contain an edge from S to $V \setminus S$. We require f to be *proper*, i.e., (1) $f(V) = 0$, (2) $f(S) = f(V \setminus S)$, and (3) $f(A) = f(B) = 0$ for disjoint $A, B \subseteq V$ implies that $f(A \cup B) = 0$. For example, the Steiner Forest problem can be specified by setting $f(S) = 1$ if and only if, for given disjoint node subsets $V_i \subseteq V$, there exists some $i \in [k]$ such that both $V_i \cap S$ and $V_i \cap (V \setminus S)$ are non-empty.

Model-Agnostic Algorithm for CFPs

We devise the *shell-decomposition algorithm*, a generic approximation algorithm for CFPs that is efficient if f can be evaluated efficiently.

▶ **Theorem 1** (Model-Agnostic Complexity of Constrained Forest Problems). *Given $0 < \varepsilon \leq 1$ and a graph $G = (V, E)$ with polynomially bounded edge weights $c\colon E \to \mathbb{N}$, a $(2+\varepsilon)$-approximation to a CFP with proper forest function $f\colon 2^V \to \{0,1\}$ can (up to book-keeping operations) be obtained at complexity $\widetilde{\mathcal{O}}\left((aSSSP + MST + RPS + FFE)\varepsilon^{-1}\right)$, with the terms in the sum denoting the complexities of solving (1) aSSSP: $(1+\varepsilon)$-approximate Set-Source Shortest-Path Forest, (2) MST: Minimum Spanning Tree, (3) RPS: Root-Path Selection, and (4) FFE: Forest-Function Evaluation for f.*

By *book-keeping operations*, we denote simple steps that extract the input for the next subroutine from the output of the previous ones and can be easily implemented at low complexity, where the complexity measure(s) depend on the model at hand (e.g., round

[2] Note that *instance optimality*, which requires that an algorithm is $\mathcal{O}(1)$-competitive with any other always-correct algorithm on each instance, including the best algorithm for the specific instance, is often unachievable [27]. In particular, one can test whether the input matches an abritrary fixed instance in D rounds. If so, a matching solution can be output without further computation; otherwise, a fallback algorithm is executed.

Table 1 CONGEST results for SF derived from our algorithm, for $\varepsilon \in \Theta(1)$. Deterministic and randomized results are marked with $\mathfrak{D}$ and $\mathfrak{R}$, respectively. Here, n is the number of nodes, D is the (unweighted) hop diameter, s is the shortest-path diameter (which may be different from the weighted diameter), t is the number of terminals, k is the number of SF input components, and Q is the shortcut quality of the input graph, cf. Table 3. In our results, the $\sqrt{n}+D$ terms can be replaced by $T^{PA}n^{o(1)}$, where $T^{PA} \in \mathcal{O}(\sqrt{n}+D)$, $T^{PA} \geq Q$, is the running time of Partwise Aggregation [25]. The $n^{o(1)}$ factors can be removed conditionally on the existence of certain cycle-cover algorithms [35]. For any values of the other parameters, s and t can be linear in n.[4]

Problem	LB		Previous Work			Our Work	
		Ref.	APX	Complexity	Ref.	APX	Complexity
SF(–IC)	$\widetilde{\Omega}(Q)$	[27]	$(2+\varepsilon)$ $\mathfrak{D}$	$\widetilde{\mathcal{O}}(sk+\sqrt{\min\{st,n\}})$	[30]	$(2+\varepsilon)$ $\mathfrak{D}$	$\widetilde{\mathcal{O}}(\sqrt{n}+D+k)$
			$\mathcal{O}(\log n)$ $\mathfrak{R}$	$\widetilde{\mathcal{O}}(\min\{s,\sqrt{n}\}+D+k)$			
PPC	$\widetilde{\Omega}(Q)$	[27]		—		$(2+\varepsilon)$ $\mathfrak{D}$	$\widetilde{\mathcal{O}}(\sqrt{n}+D)$
FPC	$\widetilde{\Omega}(Q)$	[27]		—		$(2+\varepsilon)$ $\mathfrak{D}$	$\widetilde{\mathcal{O}}(\sqrt{n}+D)$

complexity in CONGEST). MST is the special case of SF in which all nodes are to be connected by a lightest possible tree. The task of aSSSP requires us, given $\varepsilon > 0$ and $S \subseteq V$, to compute a forest that preserves distances to S up to a factor of $1+\varepsilon$, which ie equivalent to computing a $(1+\varepsilon)$-approximate shortest-path tree in the graph obtained by identifying all nodes in S. Finally, RPS (a special case of the more general *transshipment* problem) demands that, given a rooted forest $F \subseteq E$ and a set of marked nodes, we select all edges on paths from marked nodes to the roots of their respective trees.

MST, aSSSP, and RPS are well-understood in many models of computation. In contrast, f can be (ab)used to force the evaluation of an arbitrarily hard function g on the entire topology.[3] Thus, Theorem 1 can be viewed as confining the hardness of the task arising from the choice of f to $\mathcal{O}(\varepsilon^{-1}\log n)$ iterations of evaluating $f(C)$ for all $C \in \mathcal{C}$, where $\mathcal{C}$ is a set of disjoint connected components. Because for any such $\mathcal{C}$, one can enforce this evaluation by assigning weight 1 to edges inside each C and a large weight, say n^2, to all other edges, this is the best possible up to an $\mathcal{O}(\varepsilon^{-1}\log n)$ factor.

To illustrate the power and flexibility of our result, we apply our machinery in three models of computation – CONGEST, Parallel Random-Access Machine (PRAM), and Multi-Pass Streaming (MPS) – to three NP-hard CFPs: (1) *Steiner Forest (SF)*; (2) *Point-to-Point Connection (PPC)*, i.e., given $X, Y \subset V$ of equal cardinality, finding a lightest set of edges such that each induced component the number of nodes from X is the same as the number of nodes from Y; and (3) *Facility Placement and Connection (FPC)*, i.e., minimizing the cost of opening facilities at some nodes and connecting a set of clients $C \subseteq V$ to them. Table 1 summarizes our results in CONGEST; for all problems, our PRAM algorithms require $\widetilde{\mathcal{O}}(\varepsilon^{-3}m)$ work and $\widetilde{\mathcal{O}}(\varepsilon^{-3})$ depth, and our MPS algorithms need $\widetilde{\mathcal{O}}(\varepsilon^{-3})$ passes and $\widetilde{\mathcal{O}}(n)$ memory.

[3] For an input graph $G=(V,E)$ with uniform edge weights and any function g mapping such a graph to a desired range, choosing an arbitrary node $v \in V$ and setting $f(S)=1$ if and only if (i) $S=\{v\}$ or $S=V\setminus\{v\}$, and (ii) $g(G)=1$ results in a proper forest function f.

[4] Strictly speaking, the upper bounds from Lenzen and Patt-Shamir [30] are incomparable to ours, as it is possible to construct instances in which s, t, and k are small, yet T^{PA} is not. This corresponds to "easy" instances for the subroutines we use, so one could provide refined bounds. However, this is orthogonal to our goals in this work, as we seek to derive bounds that depend on the topology (V,E) only.

Table 2 CONGEST results for the four input variants of SF, with the notational conventions from Table 1. Regardless of the input representation, on any graph, a lower bound of $\widetilde{\Omega}(Q)$ applies to randomized algorithms; the listed lower bounds arising from the input representation are existential (ex.). As before, $\sqrt{n}+D$ terms can be replaced by $T^{PA}n^{o(1)}$.

Problem	Input	ex. LB	APX	Complexity
SF–IC	Each node v is given its component identifier $\lambda_v \in [k] \cup \{\perp\}$	$\widetilde{\Omega}(k)$ $\mathfrak{R}$	$(2+\varepsilon)$ $\mathfrak{D}$	$\widetilde{\mathcal{O}}(\sqrt{n}+D+k)$
SF–CIC	As in SF–IC, but node v knows λ_v *and* $\lvert\{u \in V \mid \lambda_u = \lambda_v\}\rvert$ (if $\lambda_v \neq \perp$)	$\widetilde{\Omega}(k)$ $\mathfrak{D}$	$(2+\varepsilon)$ $\mathfrak{R}$	$\widetilde{\mathcal{O}}(n^{2/3}+D)$
SF–CR	Each node v is given $\mathcal{R}_v \subseteq V \setminus \{v\}$	$\widetilde{\Omega}(t)$ $\mathfrak{R}$	$(2+\varepsilon)$ $\mathfrak{D}$	$\widetilde{\mathcal{O}}(\sqrt{n}+D+t)$
SF–SCR	$\mathcal{R} \subseteq \binom{V}{2}$; each node v knows $\mathcal{R}_v = \{u \in V \mid \{u,v\} \in \mathcal{R}\}$	$\widetilde{\Omega}(t)$ $\mathfrak{D}$	$(2+\varepsilon)$ $\mathfrak{R}$	$\widetilde{\mathcal{O}}(\sqrt{n}+D)$

Approaching Universal Optimality

In our upper bounds for CONGEST, $\sqrt{n}+D$ can largely be replaced by $T^{PA}n^{o(1)}$, where T^{PA} is the running time of an algorithm performing *Partwise Aggregation* (PA) [14, 25] – i.e., given a partition of V into connected subsets of nodes, computing the result of an aggregation function separately on each subset and outputting the result at each of its constituent nodes. Due to the respective hardness results [27], this implies that the running times of our solutions to PPC and FPC are universally optimal up to a factor of $n^{o(1)}$.

For SF–IC, this is true up to the additive term of k. Here, PA is insufficient: Evaluating f requires us to determine, for each set V_i, if it is contained in a single connected component induced by the set of edges that have been selected into the current (partial) solution, but the V_i may not induce connected components in G. Existential lower bounds demonstrate that this obstacle is unavoidable in general [30]. We suspect that studying Partwise Aggregation with disjoint components that may be internally disconnected – i.e., *Disjoint Aggregation* (DA) – is key to characterizing the universal complexity of SF–IC, but leave this to future work, noting that an upper bound of $\mathcal{O}(k+D)$ arises from standard pipelining techniques.

An orthogonal consideration, however, yields surprising results, as summarized in Table 2. For the SF problem, the input representation drives the problem complexity. It is known that if nodes are given the identifiers of other nodes they must connect to in the output to encode connectivity requirements (SF–CR), an existential lower bound of $\widetilde{\Omega}(t)$ applies [30], where t is the number of *terminals*, i.e., nodes that need to be connected to some other node. In our corresponding upper bound, the additive k then becomes an additive t, again resulting in existential but not universal optimality. Interestingly, this picture is turned upside down when inputs are *symmetric* in the following sense: If $u \in V$ knows that it must connect to $v \in V$ by the input, then also v knows that it must connect to u (SF–SCR). In this setting, we can exploit symmetry to efficiently evaluate f using randomized equality testing. This leads to an algorithm running in $\varepsilon^{-3}T^{PA}n^{o(1)}$ rounds, which is close to universal optimality, provided that an efficient partwise-aggregation routine is available. Similarly, we observe that the $\widetilde{\Omega}(k)$ bound can be circumvented if nodes receive not only the identifier of their input component V_i, but also the size $|V_i|$ of this input component (SF–CIC). We then obtain a randomized algorithm running in $\widetilde{\mathcal{O}}(\varepsilon^{-3}(\sqrt{n}+D)+\varepsilon^{-1}n^{2/3})$ time.

We note that a simple adaptation of the lower-bound construction from Lenzen and Patt-Shamir [30] shows the same hardness (i.e., $\widetilde{\Omega}(t)$ for SF–SCR and $\widetilde{\Omega}(k)$ for SF–CIC, respectively) for *deterministic* algorithms, based on a communication-complexity reduction

Table 3 Main notation used in this work. All notation involving nodes and edges makes implicit reference to the graph G.

Symbol	Definition	Meaning
$\lvert S\rvert$		Cardinality of a set S
2^S	$=\{X \mid X \subseteq S\}$	Set of all subsets (i.e., *power set*) of S
$\binom{S}{k}$	$=\{X \subseteq S \mid \lvert X\rvert = k\}$	Set of k-element subsets of S
$[k]$	$=\{i \mid i \in \mathbb{N}, i \leq k\}$	Set of nonnegative integers no larger than k
$[k]_0$	$=\{i \mid i \in \mathbb{N}_0, i \leq k\}$	Set of positive integers no larger than k
G	$=(V, E)$	Graph with node set V and edge set E
n	$=\lvert V\rvert$	Number of nodes
m	$=\lvert E\rvert$	Number of edges
$c(e)$	$\in \mathbb{N} = \{1, 2, \ldots\}$	Cost of edge e
$c(Z)$	$=\sum_{e\in Z} c(e)$	Cost of edge subset $Z \subseteq E$
$p(u, v)$	$=(e_1, \ldots, e_\ell)$ s.t. $\exists(v_1, \ldots, v_{\ell+1})$: $e_i = \{v_i, v_{i+1}\} \in E\ \forall i \in [\ell]$, $u = v_1$, $v = v_{\ell+1}$	ℓ-hop path between u and v (distinct nodes and distinct edges)
$h(u, v)$	$=\min\{i \mid \exists\ p(u, v)$ with $\lvert p(u, v)\rvert = i\}$	Hop (= unweighted) distance between u and v
D	$=\max\{h(u, v) \mid \{u, v\} \in \binom{V}{2}\}$	Hop diameter of G
$d(u, v)$	$=\min\{i \mid \exists\ p(u, v)$ with $c(p(u, v)\}$	Weighted distance between u and v
$P(u, v)$	$=p(u, v)$ with $c(p(u, v)) = d(u, v)$	Shortest path between u and v
s	$=\max\{\min\{\lvert P(u, v)\rvert \mid \{u, v\} \in \binom{V}{2}\}$	Shortest-path diameter of G
$\delta(S)$	$=\{e \in E \mid \lvert e \cap S\rvert = 1\}$	Set of edges with exactly one endpoint in $S \subset V$
$\mathcal{T}$	$=\{v \in V \mid f(\{v\}) = 1\}$	Terminals
t	$=\lvert\mathcal{T}\rvert$	Number of terminals
k		Number of input components in SF–IC

from 2-player equality testing. To the best of our knowledge, this is the first natural example of a provably large gap between the randomized and deterministic complexity in the CONGEST model for a *global* problem.

Structure

After introducing our main definitions in Section 2, we provide a technical overview of our results in Section 3. To conclude the main text, we discuss open questions in Section 4. A full specification of our model-agnostic algorithm and a detailed proof of its correctness are given in Section A. Model descriptions, model-specific results, and further related work are deferred to the additional appendices available in the extended version of this work [10].

2 Preliminaries

We begin by defining our basic notation as well as the class of problems we are interested in.

Basic Notation

Our basic notation is summarized in Table 3.

For a set S, we denote its cardinality by $\lvert S\rvert$, its power set by $2^S = \{X \mid X \subseteq S\}$, and the set of its k-element subsets by $\binom{S}{k}$. We extend functions $f\colon S \to \mathbb{R}$ to subsets $X \subseteq S$ in the natural way by setting $f(X) = \sum_{x\in X} f(x)$, and we write the sets of positive and nonnegative integers no greater than k as $[k] = \{i \in \mathbb{N} \mid i \leq k\}$ resp. $[k]_0 = \{i \in \mathbb{N}_0 \mid i \leq k\}$.

We consider weighted graphs $G = (V, E)$ with $n = |V|$ nodes, $m = |E|$ edges, and edge weights (*edge costs*) $c\colon E \to \mathbb{N}_0$ polynomially bounded in n.[5] Each node is equipped with a unique identifier of $\mathcal{O}(\log n)$ bits, which is also used to break ties. An ℓ-hop *path* from $u \in V$ to $v \in V$, denoted $p(u, v)$, is a sequence of distinct edges $(e_1, \dots e_\ell)$ arising from a sequence of distinct nodes $(v_1, \dots, v_{\ell+1})$ such that $e_i = \{v_i, v_{i+1}\} \in E$ for $i \in [\ell]$, $v_1 = u$, and $v_{\ell+1} = v$. The (unweighted) *hop distance* between u and v is the smallest number of hops needed to go from u to v, i.e., $h(u, v) = \min\{i \mid \exists\, p(u, v) \text{ with } |p(u, v)| = i\}$, and the *hop diameter* of G is $D = \max\{h(u, v) \mid \{u, v\} \in \binom{V}{2}\}$. The (weighted) *shortest-path distance* between u and v is $d(u, v) = \min\{i \mid \exists\, p(u, v) \text{ with } c(p(u, v)) = i\}$. A *shortest path* between u and v, denoted $P(u, v)$, is a path from u to v of length $d(u, v)$. The *shortest-path diameter* s of G is the maximum over all $\{u, v\} \in \binom{V}{2}$ of the minimum number of hops contained in a shortest path from u to v, i.e., $s = \max\{\min\{|P(u, v)| \mid \{u, v\} \in \binom{V}{2}\}\}$. Given a *cut* $(S, V \setminus S)$, the set of edges with exactly one endpoint in S is denoted as $\delta(S) = \{e \in E \mid |e \cap S| = 1\}$.

An event occurring *with high probability* (w.h.p.) has probability at least $1 - 1/n^c$ for a freely chosen constant $c \geq 1$.

Constrained Forest Problems

We are interested in Constrained Forest Problems (CFPs) as introduced by Goemans and Williamson [16].[6] Given a graph $G = (V, E)$ with edge costs $c : E \to \mathbb{N}$ and a function $f : 2^V \to \{0, 1\}$, a CFP asks us to solve the integer program stated as Problem 1, whose dual relaxation is provided as Problem 2.

▶ **Problem 1** (CFP Primal IP).

$$\begin{aligned} &\min \sum_{e \in E} c(e) x_e \\ s.t.\quad &x(\delta(S)) \geq f(S) \quad \forall\, \emptyset \neq S \subset V \\ &x_e \in \{0, 1\} \quad \forall\, e \in E \end{aligned}$$

▶ **Problem 2** (CFP Dual LP).

$$\begin{aligned} &\max \sum_{S \subset V} f(S) y_S \\ s.t.\quad &\sum_{S : e \in \delta(S)} y_S \leq c(e) \quad \forall\, e \in E \\ &y_S \geq 0 \end{aligned}$$

That is, a CFP is a minimization problem whose optimal solution is a *forest*[7] of edges from the input graph G that meets the *constraints* imposed by the *forest function* f. Like Goemans and Williamson [16], we consider CFPs with *proper* functions f, which satisfy (1) *zero*, i.e., $f(V) = 0$, (2) *symmetry*, i.e., $f(S) = f(V \setminus S)$, and (3) *disjointness* (also called *maximality* [17]), i.e., if $A \cap B = \emptyset$ for two sets A and B, then $f(A) = f(B) = 0$ implies $f(A \cup B) = 0$.[8]

For a CFP with forest function f, a node v is called a *terminal* if $f(\{v\}) = 1$. Given a forest function f, we denote the set of terminals as $\mathcal{T} = \{\{v\} \mid v \in V, f(\{v\}) = 1\}$ and the number of terminals as $t = |\mathcal{T}|$.

[5] Assuming polynomially bounded edge weights allows us to encode polynomial sums of edge weights with $\mathcal{O}(\log n)$ bits, which means that we can encode edge weights in a single message (CONGEST) or memory word (PRAM and MPS). Zero-weight edges arise naturally when simulating contractions in the distributed setting. We can handle them by scaling all non-zero edge weights by n/ε (where w.l.o.g., $1/\varepsilon$ is polynomially bounded as well), and assigning weight 1 to all previously zero-weight edges.

[6] To simplify the technical exposition, like Goemans and Williamson [16], we disallow zero-weight edges. However, it is straightforward to extend our approach to zero-weight edges by scaling edge weights as discussed in Footnote 5.

[7] For any cycle and node set S, $\delta(S)$ cannot contain exactly one edge from the cycle. Hence, from any solution with a cycle, we can remove an edge.

[8] We also assume that f is *nontrivial*, i.e., that there exists at least one $S \subset V$ such that $f(S) = 1$.

Specific Constrained Forest Problems

To demonstrate the flexibility of our approach, we consider Survivable Network Design Problems (SNDPs) that originate from three different real-world challenges.

▶ **Definition 2** (Steiner Forest (SF)). *Given a partition of the terminals $\mathcal{T} = V_1 \dot\cup \ldots \dot\cup V_k$, find a minimum-cost edge subset connecting the nodes in each input component. The corresponding forest function evaluates to 1 on $S \subseteq V$ if and only if there is $i \in [k]$ such that $\emptyset \neq V_i \cap S \neq V_i$. We distinguish between several input representations:*

SF–CR *Terminal v is given the identifiers of other terminals as* connection requests *$R_v \subseteq \mathcal{T}$.*

SF–SCR *As SF-CR, but connection requests are symmetric, i.e., $u \in R_v \Leftrightarrow v \in R_u$.*

SF–IC *Terminal $v \in V_i$ is given a unique identifier (of size $\mathcal{O}(\log n)$) for its* input component.

SF–CIC *As SF-IC, but $v \in V_i$ is also given the* cardinality *$|V_i|$ of its input component as input.*

SF has practical relevance especially in infrastructure development [1], with MST ($\mathcal{T} = V_1 = V$) and ST ($\mathcal{T} = V_1 \subseteq V$) as special cases. It is NP-complete [29] and APX-hard [9].

▶ **Definition 3** (Point-to-Point Connection (PPC)). *Given a set of sources $X \subset V$ and a set of targets $Y \subset V$, find a minimum-cost edge subset such that in each connected component, the number of sources equals the number of targets. That is, for $S \subseteq V$, $f(S) = 1$ if and only if $|S \cap X| \neq |S \cap Y|$.*

PPC is motivated by challenges from circuit switching and VLSI design, and the problem is NP-complete [31].

Our last problem is Facility Placement and Connection (FPC), an NP-complete facility-location-type problem arising, e.g., in operations research. Intuitively, FPC can be stated as follows.

▶ **Definition 4** (FPC [intuitive]). *Given for each node $v \in V$ an opening cost $o_v \in \mathbb{N}$ and indication whether it is in the set of clients $C \subseteq V$, identify a subset $O \subseteq V$ of facilities to open and an edge set $F \subseteq E$ such that each client is connected to a facility by F, minimizing $\sum_{v \in O} o_v + \sum_{e \in F} c(e)$.*

To turn this task into a CFP matching our framework, we add one additional node $s \notin V$ and, for each $v \in V$, an edge $\{v, s\}$ of weight $c(\{v, s\}) = o_v$. The task then becomes to determine a (low-weight) Steiner Tree spanning $C \cup \{s\}$, i.e., the special case of SF with $k = 1$.

▶ **Definition 5** (FPC [rephrased]). *Given, for each node $v \in V$, an opening cost $o_v \in \mathbb{N}$ and an indication whether it is in the set of clients $C \subseteq V$, solve ST on $G = (V \dot\cup \{s\}, E \dot\cup \{\{v, s\} \mid v \in V\})$, with edge costs of $c(e)$ for $e \in E$ and $c(\{v, s\}) = o_v$ for $v \in V$, where the terminals are $\mathcal{T} = C \cup \{s\}$.*

Even in Congest, we can solve ordinary ST instances efficiently, regardless of the specific input representation. However, the *virtual node* s and its incident edges need to be *simulated* in the chosen model of computation. This is trivial in PRAM (simply modify the input representation in parallel) and Multi-Pass Streaming (since we use $\Omega(n \log n)$ bits of memory anyway) but nontrivial in Congest. For Congest, we show that we can simulate the virtual node efficiently using Partwise Aggregation.

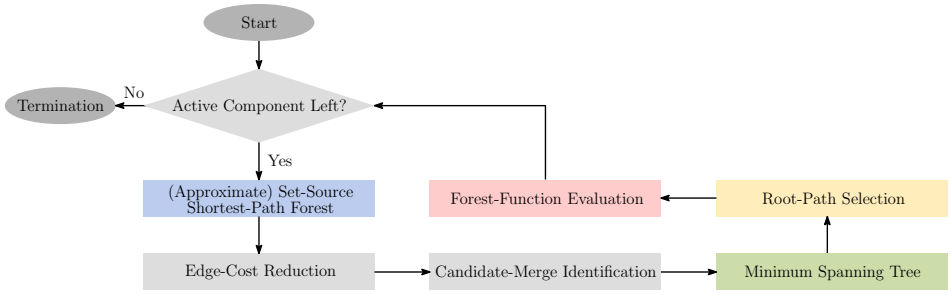

Figure 1 Overview of our model-agnostic shell-decomposition algorithm for approximating Constrained Forest Problems. Main tasks are colored; book-keeping operations are shaded in gray.

Partwise Aggregation and Shortcut Quality

Finally, we introduce a subroutine that we use as a black box to achieve near-universal optimality in cases where efficient solutions are known.

▶ **Definition 6** (Partwise Aggregation [13]). *For disjoint node sets $V_1, \dots, V_k \subseteq V$, suppose that V_i induces a connected subgraph. In the* Partwise Aggregation (PA) *problem, each $v \in V_i$ is given a unique identifier for V_i (of size $\mathcal{O}(\log n)$) and a second $\mathcal{O}(\log n)$-bit value $x(v) \in X$. For a specified associative and commutative operator $\bigoplus\colon X \times X \to X$, for each $i \in [k]$ and each $v \in V_i$, v needs to compute its output $\bigoplus_{w \in V_i} x(w)$.*

▶ **Definition 7** (Shortcut Quality). *The* shortcut quality *of G, denoted by Q, is the maximum over all feasible operators $\bigoplus$ and partitions of V of the minimum number of rounds in which a* Congest *algorithm with knowledge of the full topology can solve Partwise Aggregation. Put differently, the algorithm may preprocess the graph and the operator, but must then compute the output within Q rounds after the nodes have been given their inputs to the PA instance.*

Haeupler et al. [27] show that $\widetilde{\Omega}(Q)$ is a lower bound for MST (i.e., SF and ST with $k = 1$ and $V_1 = V$) and shortest s-t path (the special case of PPC with $|X| = |Y| = 1$) – regardless of the approximation ratio and also for randomized Las Vegas algorithms, i.e., those that guarantee a feasible output. They further prove that PA can be solved in $\widetilde{\mathcal{O}}(Q)$ rounds in the *Supported* Congest model, which is Congest with the unweighted graph topology given as part of the input.

3 Technical Overview

In this section, we outline our techniques and discuss the technical challenges to be overcome. We also use this opportunity to discuss the most relevant related work in context.

Model-Agnostic Algorithm for Proper Constrained Forest Problems

Our shell-decomposition algorithm, depicted in Figure 1, is based on the primal-dual formulation for general CFPs given by Goemans and Williamson [16] (GW algorithm), restated as Algorithm 1, which yields a $(2 - 2/t)$-approximation [16] for CFPs with proper forest functions. Our algorithm can also be seen as a model-agnostic generalization of the algorithm by Agrawal et al. [1], ported to the distributed setting by Lenzen and Patt-Shamir [30]. These algorithms have also been called *moat-growing* algorithms [3, 18, 30].

Intuitively, our algorithm operates as follows. We maintain connected *components* $\mathcal{C}$, initialized to the singletons $\mathcal{C} = \{\{v\} \mid v \in V\}$. A component $C \in \mathcal{C}$ is *active* if $f(C) = 1$. The algorithm concurrently grows balls around all *active components*, with respect to the

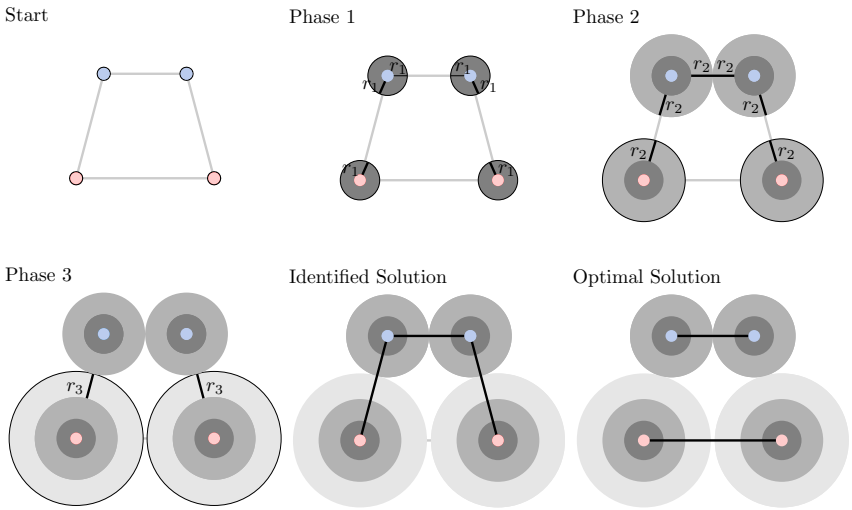

Figure 2 Illustration of our *shell-decomposition argument* on a small instance of Steiner Forest (SF–IC). Gray lines indicate original edges, black lines indicate (parts of) selected edges, black circle linings indicate active components, and node colors indicate input components. The illustrations provided by Goemans and Williamson [16] (Figs. 2–5) are stylistically similar, but our drawings clarify the phase-wise charging argument underlying our shell-decomposition algorithm.

metric induced by the then-current edge costs c'. In the growth process, two balls can touch only when at least one of their associated components is active, and only when the balls of *two active components* C, C' touch, adding edges to merge the components can make the resulting component inactive – otherwise, i.e., assuming $f(C \cup C') = f(C') = 0$, symmetry implies $f(V \setminus (C \cup C')) = 0$, disjointness implies $f(V \setminus C) = f((V \setminus (C \cup C')) \cup C') = 0$, and using symmetry again we get $f(C) = 0$, a contradiction.

As illustrated in Figure 2, until the balls around two components touch, they are disjoint, witnessing that the dual problem has a solution with weight larger than the product of the current radius times the number of currently active components. Accordingly, when merging active components, we can afford to connect the terminals by adding a shortest path between them to the primal solution, paying a cost of at most twice the radius at merge. Because each merge we perform reduces the number of active components by at least one, the ball growth always witnesses sufficient additional weight in a dual solution to pay for future merges up to an approximation factor of 2. Upon termination, i.e., when all components are inactive, the set of edges we selected constitutes a feasible solution.

The intuition sketched above already suggests that the ball-growing process allows for substantial concurrency. To achieve high efficiency across the board in various models, our shell-decomposition algorithm performs several modifications to the centralized GW algorithm, which originally assumes a *final filtering step* to eliminate unneeded edges from the solution, as well as *exact distances* and *immediate forest-function evaluation*.

(A) *Incremental Solution-Set Construction.* We merge active components that touch *regardless* of whether this ultimately turns out to be necessary to satisfy connectivity requirements. This does not impact the approximation guarantee, which was implicit already in the contribution by Agrawal et al. [1] and is now made explicit in our reformulation of the GW framework [16]. As a result, we need to determine if f imposes further connectivity requirements only as often as we iterate through the loop in Figure 1.

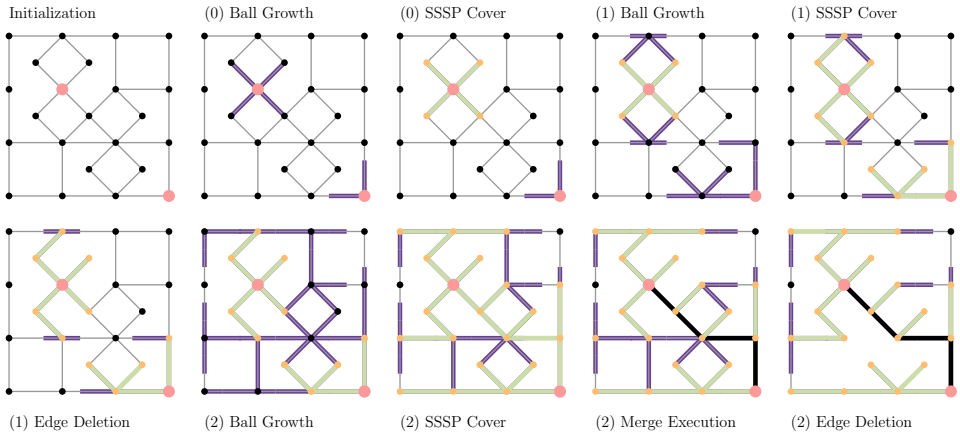

Figure 3 Operation of our shell-decomposition algorithm (Algorithm 2) on an s-t-shortest-path instance seeking to connect the red nodes, starting with $r_0 = 1/2$ (cf. Line 7), and working over phases 0, 1, and 2. Panels are labeled with their phase number and the illustrated step of Algorithm 2. Nodes absorbed by the SSSP forest are drawn in orange, edge-cost reduction is indicated in purple, edges selected into the SSSP forest are marked in green, and edges selected into the solution are marked in black. Distance approximations and deferred forest-function evaluation are not shown.

(B) *Approximate Distance Computations.* At the cost of a factor of $1 + \varepsilon$ in the approximation ratio, we replace exact distance computations with $(1 + \varepsilon)$-approximate distance computations. The challenge here is to ensure that we do not violate the dual constraints when charging dual variables (corresponding to cuts) based on the progress of the primal solution. This can be achieved by constructing the dual solution in the true metric space, rather than reusing the approximate distances leveraged by the primal solution. In contrast to the other modifications, this requires a comparatively involved argument, and it is the main technical novelty and contribution in this part of our work.

(C) *Deferred Forest-Function Evaluation.* Again at the cost of a factor of $1 + \varepsilon$ in the approximation ratio, we let components grow to radii that are integer powers of $1 + \varepsilon$ before reevaluating our forest function to update their activity status. This technique was introduced by Lenzen and Patt-Shamir [30] for the Steiner Forest problem specifically; we show its general correctness in the context of the GW algorithm. Using this technique, we can limit the number of loop iterations in Figure 1 to $\mathcal{O}(\varepsilon^{-1} \log n)$ (assuming polynomially bounded edge weights).

The pseudocode of the resulting model-agnostic shell-decomposition algorithm is given as Algorithm 2. Here, to increase clarity and highlight the primal-dual nature of our algorithm, we also compute and output the lower bound LB on the cost of an optimal solution, which is not required to determine the output forest F. An example execution of Algorithm 2 is depicted in Figure 3. In Section A, we prove that the algorithm maintains an approximation ratio of $2 + \varepsilon$.

Leveraging the modifications specified in Items A–C, to derive concrete algorithms in specific models of computation, what remains is to implement the individual steps in Figure 1. Up to simple book-keeping operations, this entails four main tasks (colored boxes in Figure 1):

(1) *(Approximate) Set-Source Shortest-Path Forest (aSSSP).* This is essentially computing a single-source shortest-path tree with a virtual source node, so we can plug in state-of-the-art algorithms for each model of interest [4, 35].

Algorithm 1 GW Algorithm for $(2 - 2/t)$-Approximation of Constrained Forest Problems [16].

Input: An undirected graph $G = (V, E)$, edge costs $c : E \to \mathbb{N}$, and a proper function f
Output: A forest F and a value LB

$F' \leftarrow \emptyset$
$LB \leftarrow 0$ // Implicitly set $y_S \leftarrow 0$ for all $S \subset V$
$\mathcal{C} \leftarrow \{\{v\} \mid v \in V\}$
foreach $v \in V$ **do**
  $r(v) \leftarrow 0$
while $\exists C \in \mathcal{C} : f(C) = 1$ **do**
  $E_{\mathcal{C}} \leftarrow \{\{i,j\} \in E \mid i \in C_i \in \mathcal{C}, j \in C_j \in \mathcal{C}, C_i \neq C_j, f(C_i) + f(C_j) > 0\}$
  $\eta, e \leftarrow \min_{e=\{i,j\} \in E_{\mathcal{C}}} \operatorname{argmin} \left\{ \frac{c'(e)}{f(C_i)+f(C_j)} \mid c'(e) = c(e) - r(i) - r(j) \right\}$
  $F' \leftarrow F' \cup \{e\}$
  foreach $v \in C_r \in \mathcal{C}$ **do**
    $r(v) \leftarrow r(v) + \eta \cdot f(C_r)$
  $LB \leftarrow LB + \eta \cdot \sum_{C \in \mathcal{C}} f(C)$ // Implicitly set $y_C \leftarrow y_C + \eta \cdot f(C)\ \forall C \in \mathcal{C}$
  $\mathcal{C} \leftarrow (\mathcal{C} \cup \{C_i \cup C_j\}) \setminus \{C_i, C_j\}$
$F \leftarrow \{e \in F' \mid f(N) = 1 \text{ for some connected component } N \text{ of } (V, F' \setminus \{e\})\}$
return F, LB

Algorithm 2 Our Shell-Decomposition Algorithm for $(2+\varepsilon)$-Approximation of Constrained Forest Problems. To obtain the desired approximation guarantee, we choose $\varepsilon', \varepsilon'' \leq \varepsilon/4$.

Input: A graph $G = (V, E)$, edge costs $c : E \to \mathbb{N}$, and a proper forest function f
Output: A forest F and a value LB

$F \leftarrow \emptyset$
$\mathcal{C} \leftarrow \{\{v\} \mid v \in V\}$
$\mathcal{T}^1 \leftarrow \{v \mid f(\{v\}) = 1\}$ // At the beginning, $|\mathcal{T}^1| = t$
foreach $e \in E$ **do**
  $c'(e) \leftarrow c(e)$ // c' keeps track of *reduced* costs
$LB \leftarrow 0$
$r \leftarrow \frac{\varepsilon''}{4}$ // For upper bound (edge selection), we could start with any $r \leq 1/2$
while $\exists C \in \mathcal{C} : f(C) = 1$ **do**
  $U \leftarrow$ Union of $(1+\varepsilon')$-approx. balls of radius r around nodes in $\mathcal{T}^1$ under edge costs c'[9]
  $F' \leftarrow (1+\varepsilon')$-approx. SSSP forest for edge cost c' with set source $\mathcal{T}^1$, restricted to U
  foreach $e \in E$ **do**
    $c'(e) \leftarrow \max\{0, c'(e) - c_U(e)\}$ // $c_U(e) = \sum_{v \in e} \max\{r - d_{F'}(\mathcal{T}^1, v), 0\}$
  $\mathcal{C}_U \leftarrow$ Connected components of (V, F') containing a (unique) $\tau \in \mathcal{T}^1$
  $M \leftarrow \{\{u,v\} \in E \mid u \in C_u \in \mathcal{C}_U, v \in C_v \in \mathcal{C}_U, C_u \neq C_v, c'(\{u,v\}) = 0\}$ // Merge cand.
  $A \leftarrow$ Arbitrary subset of M such that $F' \cup A$ is a forest spanning $(V, F' \cup M)$
  $F \leftarrow F \cup A \cup \bigcup_{v \in e \in A} \{p_v \mid p_v \text{ is the (unique) path in } F' \text{ from } v \text{ to some } \tau \in \mathcal{T}^1\}$
  // Merge
  $E \leftarrow (E \setminus \{e \in E \mid c'(e) = 0\}) \cup F' \cup A$ // Remove unneeded edges contained in U
  $\mathcal{C} \leftarrow$ Connected components of (V, F) // Update connected components
  $\mathcal{T}^1 \leftarrow \{\min\{v \in C \mid f(\{v\}) = 1\} \mid f(C) = 1, C \in \mathcal{C}\}$ // Update active terminals
  $LB \leftarrow LB + (1+\varepsilon')^{-1} r |\mathcal{T}^1|$
  $r \leftarrow (1+\varepsilon'')r$ // Update ball radius
return F, LB

[9] U as used in Line 9 contains all parts of edges in a $(1+\varepsilon)$-approximate SSSP forest that lie at distance at most r from their respective roots. Note that U can also contain *parts of* edges, whereas F' (Line 10), a $(1+\varepsilon)$-approximate SSSP forest truncated to U, only contains full edges.

(2) *Minimum Spanning Tree (MST).* This is another well-studied task for which near-optimal solutions are known in all prominent models [12, 27, 33, 34].

(3) *Root-Path Selection (RPS).* Here, we are given a forest rooted at sources (where each node knows its parent in its tree and its closest source), along with a number of marked nodes. Our goal is to select the edges on the path from each marked node to its closest source. This problem can be straightforwardly addressed by solving a much more general flow problem: *(Approximate) Transshipment*, in which flow demands are to be satisfied at minimum cost. Again, we can plug in state-of-the-art algorithms for each model of interest [4, 35]. Note that in our case, the solution is restricted to containing the edges of a predetermined forest, such that the solution is unique, edge weights play no role, and the challenge becomes to determine the feasible flow quickly.

(4) *Forest-Function Evaluation (FFE).* The remaining subtask is to assess if $f(C) = 1$, for each component $C \subset V$ in a set of disjoint, internally connected components $\mathcal{C}$– i.e., to determine which of the components still need to be connected to others. This is the only step that depends on f, requiring an implementation of f matching the given model of computation.

Note that f is an arbitrary proper function, so evaluating it can be arbitrarily hard. Thus, our algorithm confines the hardness of the task that comes from the choice of f to $O(\varepsilon^{-1} \log n)$ evaluations of $f(C)$ for disjoint connected components $C \in \mathcal{C}$ (cf. Item A). In contrast, the other three subtasks can be solved by state-of-the-art algorithms from the literature in a black-box fashion.

To illustrate the power of our result, we apply our machinery in three models of computation – Congest, Parallel Random-Access Machine (PRAM), and Multi-Pass Streaming (MPS) – to three Constrained Forest Problems. Our results follow from Theorem 1 with (1) the referenced results on aSSSP, MST, and RPS, (2) model- and problem-specific subroutines for FFE, and (3) model-specific subroutines for book-keeping operations.

In Table 1, we highlight the improvements over the state of the art achieved for our example problems by instantiating the shell-decomposition algorithm in the Congest model.

(I) *Steiner Forest (SF).* From $\widetilde{\mathcal{O}}(\varepsilon^{-1}(sk+\sqrt{\min\{st, n\}}))$ time[10] for a $(2+\varepsilon)$-approximation and $\widetilde{\mathcal{O}}(\min\{s, \sqrt{n}\} + D + k)$ time for an $\mathcal{O}(\log n)$-approximation to SF–IC obtained by Lenzen and Patt-Shamir [30] to $(2+\varepsilon)$-approximations in time (1) $\widetilde{\mathcal{O}}(\varepsilon^{-3}(\sqrt{n} + D) + \varepsilon^{-1}k)$ for SF–IC, (2) $\widetilde{\mathcal{O}}(\varepsilon^{-3}(\sqrt{n} + D))$ for SF–SCR, and (3) $\widetilde{\mathcal{O}}(\varepsilon^{-3}(\sqrt{n} + D) + \varepsilon^{-1} \min\{n^{2/3}, k\})$ for SF–CIC. For SF–CR, we incur the same additive running-time overhead of $\mathcal{O}(t)$ as Lenzen and Patt-Shamir [30], matching the existential lower bound they showed.

(II) *Point-to-Point Connection (PPC).* We are not aware of prior work providing Congest algorithms for the PPC problem. Here, we obtain a $(2+\varepsilon)$-approximation in $\widetilde{\mathcal{O}}(\varepsilon^{-3}(\sqrt{n} + D))$ time.

(III) *Facility Placement and Connection (FPC).* We do not know of any existing Congest algorithms for the FPC problem, and we realize a $(2+\varepsilon)$-approximation in $\widetilde{\mathcal{O}}(\varepsilon^{-3}(\sqrt{n} + D))$ time.

We also derive algorithms for SF, PPC, and FPC in the PRAM and Multi-Pass Streaming models, taking $\widetilde{\mathcal{O}}(\varepsilon^{-3}m)$ work and $\widetilde{\mathcal{O}}(\varepsilon^{-3})$ depth in the PRAM model, as well as $\widetilde{\mathcal{O}}(\varepsilon^{-3})$ passes and $\widetilde{\mathcal{O}}(n)$ space in the Multi-Pass Streaming model. To the best of our knowledge,

[10] Recall that n denotes the number of nodes, k the number of input components, t the number of terminals, D the unweighted (hop) diameter, and s the shortest-path diameter. Both t and s can be up to $\Omega(n)$, regardless of k or D.

these algorithms are either the first ones to perform these tasks in their respective models or they cover more general classes of instances than the state of the art (see extended version). Notably, we obtain our diverse results with relative ease once the analysis of our model-agnostic shell-decomposition algorithm is in place – which contrasts with the challenges of directly designing specific algorithms for specific problems in specific models.

Taking Shortcuts

In the CONGEST model, the $\sqrt{n}$ term in the complexity is due to the fact that, in general, it is not possible to solve *Partwise Aggregation* (PA) in $\widetilde{o}(\sqrt{n})$ rounds. As formalized in Definition 6, PA denotes the task of performing an aggregation and broadcast operation on each subset in a partition of V that induces connected components [14]. We can leverage PA, *inter alia*, to determine a leader and distribute its ID, or to find and make known a heaviest outgoing edge, for each component (a.k.a. *part*) in parallel. PA-type operations are key to efficient MST construction, and any T^{PA}-round solution to Partwise Aggregation lets us solve MST in $\widetilde{\mathcal{O}}(T^{PA} + D) = \widetilde{\mathcal{O}}(T^{PA})$ rounds [13, 14].

As MST computation is a CFP, it might not surprise that Partwise Aggregation can serve as a key subroutine for other CFPs in the CONGEST model as well. We show that the $\sqrt{n}$ term in the complexity of CFPs can be replaced by T^{PA}. Since this is already known for $(1+\varepsilon)$-approximate Set-Source Shortest-Path Forest [35], Minimum Spanning Tree [13], and $(1+\varepsilon)$-approximate Transshipment [35] (which can be used to solve Root-Path Selection),[11] again we need to show this for Forest-Function Evaluation only. This is straightforward for PPC and FPC, and simple algorithms evaluate the forest function for SF-IC and SF-CR in $\mathcal{O}(T^{PA} + k)$ and $\mathcal{O}(T^{PA} + t)$ rounds, respectively.

The substantial literature on *low-congestion shortcuts* provides a large array of results on solving Partwise Aggregation in time comparable to the *shortcut quality* of the input graph, i.e., the best possible running time T^{PA} for Partwise Aggregation [2,14,15,22,23,26,27,35,37]. In particular, $T^{PA} \in \widetilde{\mathcal{O}}(D)$ if the input graph does not contain a fixed $\widetilde{\mathcal{O}}(1)$-dense minor, without precomputations or further knowledge of G [14]. Examples of graphs without $\widetilde{\mathcal{O}}(1)$-dense minors are planar graphs and, more generally, graphs of bounded genus. Moreover, in *Supported* CONGEST (where (V, E) is known – or rather, can be preprocessed), $T^{PA} \in \widetilde{\mathcal{O}}(Q)$, i.e., within a polylogarithmic factor of the optimum. Due to the modular structure of our results, any future results on Partwise Aggregation and low-congestion shortcuts will automatically improve the state of the art for CFPs in the CONGEST model.

The Quest for Universal Optimality

In the CONGEST model, it is known that even on a *fixed* network topology, $\widetilde{\Omega}(T^{PA})$ rounds are required to obtain *any* non-trivial approximation for a large class of problems. This class includes MST, a special case of ST, which reduces to FPC by making all but the opening cost of a single node very high, and shortest s-t-path [27], a special case of PPC. Thus, this *universal* lower bound applies to our example tasks as well. In particular, our results on PPC and FPC have almost universally tight running times.

In contrast, the additive terms of k and t in the running times of our algorithms for SF-IC and SF-CR, respectively, are only *existentially* optimal [30]: The lower-bound graph – a double star – has shortcut quality $\mathcal{O}(1)$, but in a fully connected graph, it is trivial to evaluate f for

[11] The results for approximate Set-Source Shortest-Path Forest and approximate Transshipment are conditional on the existence of fast $(\widetilde{\mathcal{O}}(1), \widetilde{\mathcal{O}}(1))$-cycle-cover algorithms; otherwise, we incur an $n^{o(1)}$ overhead (see Table 1).

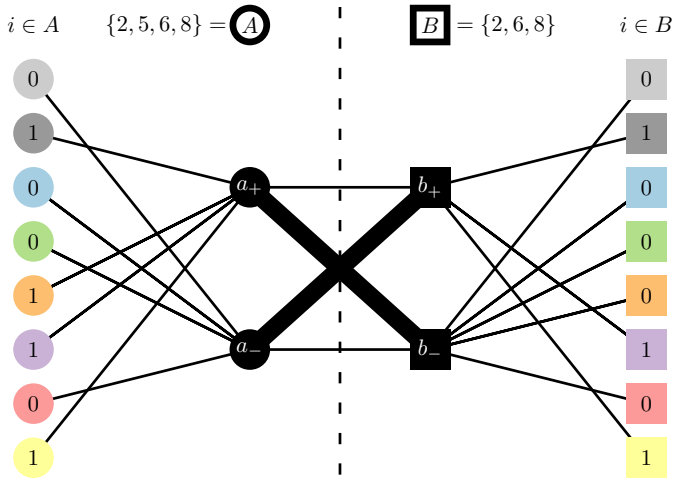

Figure 4 Example of a lower-bound graph used to reduce Equality Testing to Steiner Forest, for an instance with $n = 8$. Node colors indicate input components (SF–CIC) resp. connection requests (SF–SRC), and node shapes indicate which nodes are simulated by which player.

all current components in two rounds. This motivates us to split Forest-Function Evaluation for Steiner Forest into two parts: (1) determining, for each input component $V_i \subseteq V$ or connection request $r \in \mathcal{R}$, respectively, whether the specified connectivity requirements are *met*, and (2) determining, for each current component $C \in \mathcal{C}$, whether it contains a terminal whose connectivity requirements are *not met*. While the second part can be solved via standard Partwise Aggregation, the first part requires aggregating information within k (or t) disjoint components that may be *internally disconnected*. We call this task *Disjoint Aggregation on p parts*, DA(p), achieve the trivial bound $T^{DA(p)} \in \mathcal{O}(D + p)$ via pipelining over a BFS tree, and hypothesize that the universal complexity of DA(p) merits further investigation in future work.

As an orthogonal approach, we explore the effect of the input specification on our SF results. The assumptions that (1) pairs of terminals know that they need to be connected (SF–SCR), or (2) the size of each input component is known to its constituent terminals (SF–CIC) both are plausible, and they change the basis of the applicable existential lower bounds from 2-party *set disjointness* [30] to 2-party *equality*, as depicted in Figure 4. We provide the details in the extended version.

Assuming SF–SCR, we show how to achieve a running time of $\widetilde{\mathcal{O}}(\min\{T^{PA}n^{o(1)}, \sqrt{n}+D\})$ using efficient randomized 2-party equality testing. We sketch a solution assuming *shared* randomness here; standard techniques achieve the same without shared randomness. For each terminal-request pair $\{u, v\}$, we flip a fair independent coin and denote the result by $c_{\{u,v\}} \in \{0, 1\}$. Now each component C aggregates $\sum_{u\in C}\sum_{v\in\mathcal{R}_v} c_{\{u,v\}} \bmod 2$. Observe that if $u, v \in C$ for request pair $\{u, v\}$, then for SF–SCR, it holds that $v \in \mathcal{R}_u$ and $u \in \mathcal{R}_v$. Hence, if C contains, for each request pair, either both terminals or none of the terminals, then the sum is guaranteed to be 0 modulo 2. Otherwise, fix a request pair $\{u, v\}$ with $u \in C$ and $v \in V \setminus C$. After evaluating the sum up to coin $c_{u,v}$, it is either 0 or 1, and hence, by independence of $c_{u,v}$, with probability $1/2$, the sum is 1 modulo 2. Therefore, performing the process $\mathcal{O}(\log n)$ times in parallel, we can distinguish between $f(C) = 0$ and $f(C) = 1$ with high probability. This computation can be performed by a single aggregation, where the aggregation operator $\bigoplus$ is given by bit-wise addition modulo 2.

Our strategy for SF–CIC is similar, but results in a much weaker bound of $\widetilde{\mathcal{O}}(n^{2/3}+D)$; our main point here is to demonstrate that the $\widetilde{\Omega}(k)$ can be beaten. We distinguish three cases. (1) Components C of size at most $n^{2/3}$ are spanned by a rooted tree of size $n^{2/3}$. Here, we can aggregate the terminal counts, for all input components, within C at C's root in $\mathcal{O}(n^{2/3})$ rounds to determine activity status. (2) For each input component of size at least $n^{1/3}$, we globally aggregate if there are two distinct component IDs with a terminal from that input component. This requires one aggregation for each such input component, all of which can be completed within $\mathcal{O}(n^{2/3}+D)$ rounds via a BFS tree, as there can be at most $n^{2/3}$ input components of this size. (3) For input components of size $s < n^{1/3}$, each component C of size larger than $n^{2/3}$ uses the same strategy as for SF–SCR. However, only input components of the same size can be handled in a single aggregation, as the summation is now modulo s. Hence, $\widetilde{\mathcal{O}}(n^{1/3})$ aggregations by at most $n^{1/3}$ components of size larger than $n^{2/3}$ are required, which can again be performed in $\widetilde{\mathcal{O}}(n^{2/3}+D)$ rounds.

4 Discussion

In this work, we presented a general model-agnostic framework for the $(2+\varepsilon)$-approximation of Constrained Forest Problems (CFPs) and demonstrated its utility on three NP-hard CFPs in three models of computation. We conclude with a number of open questions – beyond applying our framework to other graph problems and computational models – in increasing order of generality:

(Q1) **Can the running time of SF–CIC in Congest be improved to nearly T^{PA}?**

(Q2) Many of our results are near-universally optimal in Congest, even for Las Vegas algorithms – i.e., randomized with guaranteed success – but our algorithms for SF–SCR and SF–CIC are Monte Carlo. Due to existential lower bounds based on the communication complexity of 2-party equality, this is required to (always) achieve small running times w.h.p.
Is $\widetilde{\Omega}(Q)$ also a lower bound for Las Vegas Congest algorithms?

(Q3) **How hard is Disjoint Aggregation, i.e., how can we characterize $T^{DA(p)}$?**

(Q4) The FPC problem minimizes the sum of opening and forest costs, disregarding the (often distance-based) *connection costs* considered in other facility-location-type problems.
Does our approach to FPC generalize to problems with connection costs?

(Q5) As we reduce most of our tasks to few PA instances, in Congest, $T^{PA} \approx Q$ rounds are both necessary and sufficient to achieve near-universal optimality. Since PA can be solved in $\mathcal{O}(n)$ work and $\mathcal{O}(\log n) = \widetilde{\mathcal{O}}(1)$ depth in PRAM, PA-based algorithms also yield good solutions in PRAM. However, in the streaming model, while PA can be solved in $\widetilde{\mathcal{O}}(n)$ memory and two passes, it is unclear if this yields optimal results (unless we insist that the output needs to be held in memory), as we are mostly limited to existential $\widetilde{\Omega}(\sqrt{n})$ lower bounds.
Are there $\tilde{o}(n)$-memory streaming algorithms with $\widetilde{\mathcal{O}}(1)$ passes? Better yet: Is there an analog to universal optimality in the MPS model?
Note that T^{PA} seems inadequate as a parameter here, as each part will require *some* memory for a few-pass implementation, but there may be $\Omega(n)$ parts.

(Q6) **Does our approach generalize to CFPs on hypergraphs?**

(Q7) Beyond proper functions, the primal-dual method has proven useful for *uncrossing* functions [17]. One of the main features of optimization problems with uncrossing functions is that they are guaranteed to feature an optimal dual solution that is *laminar*.
Can our approach be extended to uncrossing or other non-proper functions?

References

1 Ajit Agrawal, Philip N. Klein, and R. Ravi. When trees collide: An approximation algorithm for the generalized steiner problem on networks. *SIAM J. Comput.*, 24(3):440–456, 1995. doi:10.1137/S0097539792236237.

2 Ioannis Anagnostides, Christoph Lenzen, Bernhard Haeupler, Goran Zuzic, and Themis Gouleakis. Almost universally optimal distributed laplacian solvers via low-congestion shortcuts. *Distributed Comput.*, 36(4):475–499, 2023. doi:10.1007/s00446-023-00454-0.

3 Aaron Archer, MohammadHossein Bateni, MohammadTaghi Hajiaghayi, and Howard J. Karloff. Improved approximation algorithms for prize-collecting steiner tree and TSP. *SIAM J. Comput.*, 40(2):309–332, 2011. doi:10.1137/090771429.

4 Ruben Becker, Sebastian Forster, Andreas Karrenbauer, and Christoph Lenzen. Near-optimal approximate shortest paths and transshipment in distributed and streaming models. *SIAM J. Comput.*, 50(3):815–856, May 2021. doi:10.1137/19M1286955.

5 Davide Bilò. New algorithms for steiner tree reoptimization. *Algorithmica*, 86(8):2652–2675, 2024. doi:10.1007/S00453-024-01243-2.

6 Joakim Blikstad, Jan van den Brand, Yuval Efron, Sagnik Mukhopadhyay, and Danupon Nanongkai. Nearly optimal communication and query complexity of bipartite matching. In *63rd IEEE Annual Symposium on Foundations of Computer Science, FOCS 2022, Denver, CO, USA, October 31 - November 3, 2022*, pages 1174–1185. IEEE, 2022. doi:10.1109/FOCS54457.2022.00113.

7 Yi-Jun Chang, Manuela Fischer, Mohsen Ghaffari, Jara Uitto, and Yufan Zheng. The complexity of $(\Delta+1)$ coloring in congested clique, massively parallel computation, and centralized local computation. In Peter Robinson and Faith Ellen, editors, *Proceedings of the 2019 ACM Symposium on Principles of Distributed Computing, PODC 2019, Toronto, ON, Canada, July 29 - August 2, 2019*, pages 471–480. ACM, 2019. doi:10.1145/3293611.3331607.

8 Chandra Chekuri and F. Bruce Shepherd. Approximate integer decompositions for undirected network design problems. *SIAM J. Discret. Math.*, 23(1):163–177, 2008. doi:10.1137/040617339.

9 Miroslav Chlebík and Janka Chlebíková. The steiner tree problem on graphs: Inapproximability results. *Theor. Comput. Sci.*, 406(3):207–214, 2008. doi:10.1016/j.tcs.2008.06.046.

10 Corinna Coupette, Alipasha Montaseri, and Christoph Lenzen. Model-agnostic approximation of constrained forest problems, 2024. doi:10.48550/arXiv.2407.14536.

11 Atish Das Sarma, Stephan Holzer, Liah Kor, Amos Korman, Danupon Nanongkai, Gopal Pandurangan, David Peleg, and Roger Wattenhofer. Distributed verification and hardness of distributed approximation. *SIAM J. Comput.*, 41(5):1235–1265, 2012. doi:10.1137/11085178X.

12 Joan Feigenbaum, Sampath Kannan, Andrew McGregor, Siddharth Suri, and Jian Zhang. On graph problems in a semi-streaming model. *Theor. Comput. Sci.*, 348(2-3):207–216, 2005. doi:10.1016/j.tcs.2005.09.013.

13 Mohsen Ghaffari and Bernhard Haeupler. Distributed algorithms for planar networks II: low-congestion shortcuts, mst, and min-cut. In Robert Krauthgamer, editor, *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2016, Arlington, VA, USA, January 10-12, 2016*, pages 202–219. SIAM, 2016. doi:10.1137/1.9781611974331.ch16.

14 Mohsen Ghaffari and Bernhard Haeupler. Low-congestion shortcuts for graphs excluding dense minors. In Avery Miller, Keren Censor-Hillel, and Janne H. Korhonen, editors, *PODC '21: ACM Symposium on Principles of Distributed Computing, Virtual Event, Italy, July 26-30, 2021*, pages 213–221. ACM, 2021. doi:10.1145/3465084.3467935.

15 Mohsen Ghaffari and Goran Zuzic. Universally-optimal distributed exact min-cut. In Alessia Milani and Philipp Woelfel, editors, *PODC '22: ACM Symposium on Principles of Distributed Computing, Salerno, Italy, July 25 - 29, 2022*, pages 281–291. ACM, 2022. doi:10.1145/3519270.3538429.

16 Michel X. Goemans and David P. Williamson. A general approximation technique for constrained forest problems. *SIAM J. Comput.*, 24(2):296–317, 1995. doi:10.1137/S0097539793242618.

17 Michel X Goemans and David P Williamson. The primal-dual method for approximation algorithms and its application to network design problems. In *Approximation Algorithms for NP-Hard Problems*, pages 144–191. PWS Publishing Co., 1996.

18 Anupam Gupta and Amit Kumar. A constant-factor approximation for stochastic steiner forest. In Michael Mitzenmacher, editor, *Proceedings of the 41st Annual ACM Symposium on Theory of Computing, STOC 2009, Bethesda, MD, USA, May 31 - June 2, 2009*, pages 659–668. ACM, 2009. doi:10.1145/1536414.1536504.

19 Anupam Gupta and Amit Kumar. Greedy algorithms for steiner forest. In Rocco A. Servedio and Ronitt Rubinfeld, editors, *Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing, STOC 2015, Portland, OR, USA, June 14-17, 2015*, pages 871–878. ACM, 2015. doi:10.1145/2746539.2746590.

20 Anupam Gupta, Amit Kumar, Martin Pál, and Tim Roughgarden. Approximation via cost-sharing: A simple approximation algorithm for the multicommodity rent-or-buy problem. In *44th Symposium on Foundations of Computer Science, FOCS 2003, Cambridge, MA, USA, October 11-14, 2003, Proceedings*, pages 606–615. IEEE Computer Society, 2003. doi:10.1109/SFCS.2003.1238233.

21 Bernhard Haeupler, D. Ellis Hershkowitz, and Thatchaphol Saranurak. Maximum length-constrained flows and disjoint paths: Distributed, deterministic, and fast. In Barna Saha and Rocco A. Servedio, editors, *Proceedings of the 55th Annual ACM Symposium on Theory of Computing, STOC 2023, Orlando, FL, USA, June 20-23, 2023*, pages 1371–1383. ACM, 2023. doi:10.1145/3564246.3585202.

22 Bernhard Haeupler, Taisuke Izumi, and Goran Zuzic. Low-congestion shortcuts without embedding. In George Giakkoupis, editor, *Proceedings of the 2016 ACM Symposium on Principles of Distributed Computing, PODC 2016, Chicago, IL, USA, July 25-28, 2016*, pages 451–460. ACM, 2016. doi:10.1145/2933057.2933112.

23 Bernhard Haeupler, Taisuke Izumi, and Goran Zuzic. Near-optimal low-congestion shortcuts on bounded parameter graphs. In Cyril Gavoille and David Ilcinkas, editors, *Distributed Computing - 30th International Symposium, DISC 2016, Paris, France, September 27-29, 2016. Proceedings*, volume 9888 of *Lecture Notes in Computer Science*, pages 158–172. Springer, 2016. doi:10.1007/978-3-662-53426-7_12.

24 Bernhard Haeupler, Taisuke Izumi, and Goran Zuzic. Low-congestion shortcuts without embedding. *Distributed Comput.*, 34(1):79–90, 2021. doi:10.1007/s00446-020-00383-2.

25 Bernhard Haeupler and Jason Li. Faster distributed shortest path approximations via shortcuts. In Ulrich Schmid and Josef Widder, editors, *32nd International Symposium on Distributed Computing, DISC 2018, New Orleans, LA, USA, October 15-19, 2018*, volume 121 of *LIPIcs*, pages 33:1–33:14. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2018. doi:10.4230/LIPIcs.DISC.2018.33.

26 Bernhard Haeupler, Harald Räcke, and Mohsen Ghaffari. Hop-constrained expander decompositions, oblivious routing, and distributed universal optimality. In Stefano Leonardi and Anupam Gupta, editors, *STOC '22: 54th Annual ACM SIGACT Symposium on Theory of Computing, Rome, Italy, June 20 - 24, 2022*, pages 1325–1338. ACM, 2022. doi:10.1145/3519935.3520026.

27 Bernhard Haeupler, David Wajc, and Goran Zuzic. Universally-optimal distributed algorithms for known topologies. In Samir Khuller and Virginia Vassilevska Williams, editors, *STOC '21: 53rd Annual ACM SIGACT Symposium on Theory of Computing, Virtual Event, Italy, June 21-25, 2021*, pages 1166–1179. ACM, 2021. doi:10.1145/3406325.3451081.

28 Kamal Jain and Vijay V. Vazirani. Approximation algorithms for metric facility location and k-median problems using the primal-dual schema and lagrangian relaxation. *J. ACM*, 48(2):274–296, 2001. doi:10.1145/375827.375845.

29 Richard M. Karp. Reducibility among combinatorial problems. In Raymond E. Miller and James W. Thatcher, editors, *Proceedings of a symposium on the Complexity of Computer Computations, held March 20-22, 1972, at the IBM Thomas J. Watson Research Center, Yorktown Heights, New York, USA*, The IBM Research Symposia Series, pages 85–103. Plenum Press, New York, 1972. doi:10.1007/978-1-4684-2001-2_9.

30 Christoph Lenzen and Boaz Patt-Shamir. Improved distributed Steiner forest construction. In Magnús M. Halldórsson and Shlomi Dolev, editors, *ACM Symposium on Principles of Distributed Computing, PODC '14, Paris, France, July 15-18, 2014*, pages 262–271. ACM, 2014. doi:10.1145/2611462.2611464.

31 Chung-Lun Li, S. Thomas McCormick, and David Simchi-Levi. The point-to-point delivery and connection problems: complexity and algorithms. *Discret. Appl. Math.*, 36(3):267–292, 1992. doi:10.1016/0166-218X(92)90258-C.

32 Sagnik Mukhopadhyay and Danupon Nanongkai. Weighted min-cut: Sequential, cut-query, and streaming algorithms. In Konstantin Makarychev, Yury Makarychev, Madhur Tulsiani, Gautam Kamath, and Julia Chuzhoy, editors, *Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing, STOC 2020, Chicago, IL, USA, June 22-26, 2020*, pages 496–509. ACM, 2020. doi:10.1145/3357713.3384334.

33 Seth Pettie and Vijaya Ramachandran. An optimal minimum spanning tree algorithm. *J. ACM*, 49(1):16–34, 2002. doi:10.1145/505241.505243.

34 Seth Pettie and Vijaya Ramachandran. A randomized time-work optimal parallel algorithm for finding a minimum spanning forest. *SIAM J. Comput.*, 31(6):1879–1895, 2002. doi:10.1137/S0097539700371065.

35 Václav Rozhon, Christoph Grunau, Bernhard Haeupler, Goran Zuzic, and Jason Li. Undirected $(1+\varepsilon)$-shortest paths via minor-aggregates: Near-optimal deterministic parallel and distributed algorithms. In Stefano Leonardi and Anupam Gupta, editors, *STOC '22: 54th Annual ACM SIGACT Symposium on Theory of Computing, Rome, Italy, June 20 - 24, 2022*, pages 478–487. ACM, 2022. doi:10.1145/3519935.3520074.

36 Stefan Voß. Steiner tree problems in telecommunications. In Mauricio G. C. Resende and Panos M. Pardalos, editors, *Handbook of Optimization in Telecommunications*, pages 459–492. Springer, 2006. doi:10.1007/978-0-387-30165-5_18.

37 Goran Zuzic, Gramoz Goranci, Mingquan Ye, Bernhard Haeupler, and Xiaorui Sun. Universally-optimal distributed shortest paths and transshipment via graph-based ℓ_1-oblivious routing. In Joseph (Seffi) Naor and Niv Buchbinder, editors, *Proceedings of the 2022 ACM-SIAM Symposium on Discrete Algorithms, SODA 2022, Virtual Conference / Alexandria, VA, USA, January 9 - 12, 2022*, pages 2549–2579. SIAM, 2022. doi:10.1137/1.9781611977073.100.

A Model-Agnostic Algorithm

A.1 Modified Centralized $(2+\varepsilon)$-Approximation

▶ **Theorem 8.** *For a graph $G=(V,E)$, edge costs $c\colon E\to\mathbb{N}$, and proper function f, the modified GW algorithm with incremental solution-set construction, $(1+\varepsilon')$-approximate distance computations, and $(1+\varepsilon'')$-deferred forest-function evaluation (Algorithm 2) yields a $(2-2/t)(1+\varepsilon')(1+\varepsilon'')^2$-approximation to the optimal solution, i.e., a $(2+\varepsilon)$-approximation for $\varepsilon',\varepsilon''\leq\varepsilon/4\leq 1/4$.*

Denote by $i=0,1,\dots$ the iterations of the while loop in Algorithm 2, calling each iteration a *merge phase*, and for each phase by (i) $r_i:=(1+\varepsilon'')^i\cdot\frac{\varepsilon''}{4}$ the radius r at the *beginning* of phase i (with $r_{-1}:=(1+\varepsilon'')^{-1}\cdot\frac{\varepsilon''}{4}$), (ii) U_i, c'_i F_i, F'_i, A_i, E_i, and $\mathcal{T}_i^1$ the values of the respective variables at the *end* of phase i, and (iii) $a_i:=|\mathcal{T}_i^1|$ the number of active components at the *end* of the phase i (with $a_{-1}:=t$). We prepare our proof in five lemmas.

▶ **Lemma 9.** *For all $i \in \mathbb{N}_0$ s.t. Algorithm 2 does not terminate at the end of phase i, we have (i) $r_i = (1+\varepsilon'')r_{i-1}$, (ii) $c'_{i+1}(e) \leq c'_i(e) \leq c(e)$ for each $e \in E_i$, (iii) U_i is spanned by $F'_i \cup A_i$, (iv) $U_i \subseteq U_{i+1}$, (v) $F'_i \cup A_i \subseteq F'_{i+1}$, (vi) $F_i \subseteq F_{i+1}$, (vii) $F_i \cup F'_i = F'_i \cup A'_i$ and (viii) (V, E_i) is connected and the weighted diameter w.r.t. reduced costs c' is decreasing.*

Proof. The first and the second statement hold by construction due to Lines 21 and 12, respectively. For the third statement, observe that F'_i connects each node in U_i to some node in $\mathcal{T}_i^1$, and the set M_i includes all edges that are both contained in U_i (i.e., whose reduced cost has become 0) and connect different connectivity components with respect to F'_i of U_i. Thus, the choice of A_i ensures that U_i is spanned by $F'_i \cup A_i$. This readily implies the forth statement, since $F'_i \cup A_i \subseteq E_i$, as well as the fifth, since all other edges e of reduced cost $c'(e) = 0$ are removed to obtain E_i, forcing any approximate SSSP solution to reselect the edges from $F'_i \cup A_i$ into F'_{i+1}. The sixth statement holds because the algorithm only adds edges to F, and the seventh statement holds by induction, using that $F_i = F_{i-1} \cup A_i \cup X$ for some set $X \subseteq F'_i$. Finally, for the eighth statement, observe that since U_i is spanned by $F'_i \cup A_i$, whose reduced cost is 0, and any edges not fully contained in U_i remain in E_i, the shortest path between any pair of nodes w.r.t. c' only becomes shorter: For any edge contained in U_i, there is now a path of reduced weight 0 between its endpoints, while any edge e with reduced cost $c'_i(e) > 0$ is still present and retains at most its original cost $c(e)$. ◀

▶ **Lemma 10.** *For $\varepsilon', \varepsilon'' \in n^{-\mathcal{O}(1)}$, Algorithm 2 terminates in $\mathcal{O}(\frac{\log n}{\varepsilon''})$ iterations of its loop.*

Proof. Since each of the operations within the loop terminate (assuming correct subroutines), it suffices to show the claimed bound on the number of loop iterations. To exit the while-loop, we must have $f(C) = 0$ for each $C \in \mathcal{C}$, where $\mathcal{C}$ is the set of connected components induced by our current edge set F. Recall that edge weights and hence the weighted diameter of the graph are polynomially bounded in n. As $\varepsilon'', \varepsilon' \in n^{-\mathcal{O}(1)}$, there is an $j \in \mathcal{O}(\frac{\log n}{\varepsilon''})$ for which $r_j = (1+\varepsilon'')^j \cdot \frac{\varepsilon''}{4}$ exceeds the weighted diameter of G times $1+\varepsilon'$. By Lemma 9 (viii), this entails that if we reach this phase, then U_j contains the entire connected graph (V, E_j). By Lemma 9 (iii), U_j (and thus V) is spanned by $F'_j \cup A_j$. The merge operation in Line 16 hence ensures that all terminals in $\mathcal{T}_{j-1}^1$ are connected by F_j. Denote by C the connectivity component of (V, F_j) containing the nodes in $\mathcal{T}_{j-1}^1$. For each terminal $\tau \notin \mathcal{T}_{j-1}^1 \setminus C$, τ lies in a connected component C' of (V, F_{j-1}) satisfying that $f(C') = 0$. This entails that we can partition $V \setminus C$ into two types of sets: (i) components C' of (V, F_{j-1}) satisfying that $f(C') = 0$, and (ii) non-terminals $v \in V \setminus \mathcal{T}$, which satisfy $f(\{v\}) = 0$ by definition. By disjointness of the proper forest function f, it follows that $f(V \setminus \mathcal{C}) = 0$, and by symmetry it follows that $f(C) = 0$. Finally, by Lemma 9 (vi), we have $F_{j-1} \subseteq F_j$. Therefore, each connectivity component C' of (V, F_j) other than C decomposes into connectivity components of (V, F_{j-1}), satisfying that $f(C') = 0$, and non-terminals, which again by disjointness implies that $f(C') = 0$. Thus, Algorithm 2 terminates by the end of phase $j \in \mathcal{O}(\frac{\log n}{\varepsilon''})$. ◀

▶ **Lemma 11.** *The edge set F output by Algorithm 2 is primal feasible.*

Proof. When Algorithm 2 terminates, we must have exited its while-loop, implying that we have $f(C) = 0$ for each $C \in \mathcal{C}$, where $\mathcal{C}$ is the set of connected components induced by our output set F. Consider any set $S \subseteq V$. If S non-trivially intersects a component $C \in \mathcal{C}$, i.e., $\emptyset \neq S \cap C \neq C$, then there is an edge of F in the cut defined by S, i.e., $|\delta(S) \cap F| \geq 1 \geq f(S)$. Otherwise, $S = \bigcup_{C \in \mathcal{C}_S} C$ for some $\mathcal{C}_S \subseteq \mathcal{C}$. As f satisfies disjointness and $f(C) = 0$ for each $C \in \mathcal{C}_S$, it follows that $|\delta(S) \cap F| \geq 0 = f(S)$ in this case, too. Thus, F is primal feasible. ◀

▶ **Lemma 12.** *The value LB output by Algorithm 2 is the cost of a feasible dual solution.*

Proof. Recall that $r = r_i = (1+\varepsilon'')^i \frac{\varepsilon''}{4}$ in phase i, and denote by j the index of the final phase Abbreviating $\Delta_i \coloneqq (1+\varepsilon')^{-1} r_i$, we can write the value LB as

$$LB = (1+\varepsilon')^{-1} \sum_{i=0}^{j} (1+\varepsilon'')^i \frac{\varepsilon''}{4} a_i = (1+\varepsilon')^{-1} \sum_{i=0}^{j} r_i a_i = \sum_{i=0}^{j} \Delta_i a_i \ .$$

Denote the cost reduction of an edge achieved in phase i as $\bar{c}_i(e) \coloneqq c_{U_i}(e) = c'_{i-1}(e) - c'_i(e)$, with $c'_{-1}(e) \coloneqq c(e)$, and the amount that y is increased in phase i for some set $S \subset V$ as $y_i(S)$. Now observe that to construct a feasible dual solution of value LB, it suffices to, in each phase $i \in [j]_0$, for each component C surviving phase i, increase the dual variables associated with the subsets $S \subset C$ in sum by Δ_i, while ensuring that for each $e \in E$, we maintain $\sum_{S\colon \delta(S) \ni e} y_i(S) \leq \bar{c}_i(e)$.

Since f is proper, we know that for each component C surviving phase i, there exists at least one component C' active in phase i such that $C' \subseteq C$. Therefore, to allocate Δ_i to dual variables associated with C, we can trace the construction of C from the set of components $\mathcal{C}' \coloneqq \{C' \subseteq C \mid f(C') = 1,\ C' \text{ is a connected component of } (V, F_{i-1})\}$ (where $F_{-1} \coloneqq \emptyset$) as follows. Starting with $\mathcal{C}'$ as it resulted from phase $i-1$, we increase the radii of all components $C' \in \mathcal{C}'$ at rate ρ and the y-variables of $C' \in \mathcal{C}'$ at rate $\frac{\rho}{1+\varepsilon'}$. Thus, we gradually construct U_i and reduce the costs of all edges in the affected cuts (i.e., increase $c_{U_i} = \bar{c}_i(e)$) until the first edge achieves $c'_i(e) = 0$ (or the phase ends). This happens *at the latest* when the first dual constraint in phase i becomes tight – it can happen earlier as the edge cost reductions associated with our radius increases are based on $(1+\varepsilon')$-approximate distances. When an edge achieves $c'_i(e) = 0$, we update F and $\mathcal{C}'$ to ensure that both endpoints of e lie in the same component, and we iterate the process described above.

Since C survived phase i, this process does not add an edge to F that lies in the cut $(C, V \setminus C)$ until we have increased the radius by r_i. We claim that at no point in this process, $\mathcal{C}'$ becomes empty. Assuming otherwise, we would have that $C = \bigcup_{C' \in \mathcal{C}'} C'$ with $f(C') = 0$ at the respective point of the process, implying the contradiction that $f(C) = 0$ by disjointness of f. Accordingly, the total increase of y-variables that we attribute to C during phase i is at least Δ_i, as desired. Moreover, our direct coupling of the y-variable increases with edge-cost reductions further ensures that the y-variables relevant for any individual edge e increase by at most $\bar{c}_i(e)$, i.e., its cost reduction in phase i, guaranteeing feasibility.

Summing over the a_i components surviving phase i, we increase the dual variables by at least $\Delta_i a_i$ per merge phase, concluding the proof. ◀

▶ **Lemma 13.** *Denoting by j the phase after which Algorithm 2 terminates, it holds that $c(F) \leq \left(2 - \frac{2}{t}\right)(1+\varepsilon'')^2 \sum_{i=0}^{j-1} r_i a_i$.*

Proof. Recall that merge phase $i \geq 0$ is the phase in which $r = r_i = (1+\varepsilon'')^i \cdot \frac{\varepsilon''}{4}$, and a_i is the number of active components *surviving* phase i. Moreover, note that all edges e added to F satisfy that $c'(e) = 0$ on termination, and that the cost reduction for each edge in phase i is at most $2r_i$, as each endpoint of the edge is contained in at most one tree of F'. Hence, the cost of each edge in F can be amortized over the phases i in which its cost is reduced by the algorithm, i.e., which it starts with $c'_{i-1}(e) > 0$, and in which it intersects U_i. Accordingly, F decomposes into a set of nested *shells* $U_i \setminus U_{i-1}$ (with $U_{-1} \coloneqq \emptyset$), which the algorithm iteratively and implicitly constructs around active components. This *shell-decomposition argument* is illustrated in Figure 2.

Since in phase i, the remaining edges $F \setminus F_{i-1}$ (where $F_{-1} \coloneqq \emptyset$) to be added to F form a forest with a_i active components as nodes, and the average degree of such a forest is at most $2 - 2/a_i$, we can bound the cost of the forest F output by Algorithm 2 from above as

$$c(F) \leq \sum_{i=0}^{j} r_i a_{i-1} \left(2 - \frac{2}{a_{i-1}}\right) \leq \left(2 - \frac{2}{t}\right)(1+\varepsilon'') \sum_{i=0}^{j} r_{i-1} a_{i-1} = \left(2 - \frac{2}{t}\right)(1+\varepsilon'') \sum_{i=-1}^{j-1} r_i a_i \,,$$

where $a_{-1} \coloneqq t$. Since F is a feasible primal solution by Lemma 11, each terminal must be incident with at least one edge from F, and the minimum edge weight is at least 1, we have that $c(F) \geq t/2$. On the other hand, $a_{-1} r_{-1} = \frac{\varepsilon'' t}{4(1+\varepsilon'')}$, yielding that
$c(F) \leq (1+\varepsilon'')c(F) - \frac{\varepsilon'' t}{2} = (1+\varepsilon'')(c(F) - 2a_{-1}r_{-1}) \leq \left(2 - \frac{2}{t}\right)(1+\varepsilon'')^2 \sum_{i=0}^{j-1} r_i a_i$. ◀

Using the above lemmas, we can now prove Theorem 8.

Proof of Theorem 8. Assume that Algorithm 2 terminates at the end of phase j, i.e., $a_j = 0$; by Lemma 10, such a phase exists. Observe that the value LB output by Algorithm 2 then satisfies $LB = (1+\varepsilon')^{-1} \sum_{i=0}^{j} r_i a_i = (1+\varepsilon')^{-1} \sum_{i=0}^{j-1} r_i a_i$. By Lemma 11, F is a feasible primal solution. As LB is the cost of a feasible dual solution by Lemma 12, using Lemma 13, we obtain the desired approximation guarantee as

$$\frac{c(F)}{LB} \leq \frac{(2 - 2/t)(1+\varepsilon'')^2 \sum_{i=0}^{j-1} r_i a_i}{(1+\varepsilon')^{-1} \sum_{i=0}^{j-1} r_i a_i} = \left(2 - \frac{2}{t}\right)(1+\varepsilon')(1+\varepsilon'')^2 \,. \quad ◀$$

A.2 Specification of our Model-Agnostic Meta-Algorithm

The task of $(2+\varepsilon)$-approximating CFPs in any specific model reduces to simulating Algorithm 2. By Lemma 10, we can divide the computation into $\mathcal{O}(\frac{\log n}{\varepsilon''})$ phases, where, starting at phase 0, phase i grows components by radius $r_i = (1+\varepsilon'')^i \cdot \frac{\varepsilon''}{4}$. Each phase tackles six abstract problems, corresponding to the six building blocks of our algorithm (see Figure 1).

A.2.1 Problems Used as Building Blocks

▶ **Problem 3** (α-approximate Set-Source Shortest-Path Forest). *Given a connected graph $G = (V, E)$ with edge costs $c\colon E \to \mathbb{N}_0$ and a set of sources $S \subseteq V$, compute a forest F' spanning G such that for all nodes $v \in V$, $d_{F'}(S, v) \leq \alpha d(S, v)$, where $d(S, v) = \min_{u \in S}\{d(u, v)\}$, and $d_{F'}(u, v)$ is the weighted distance between u and v in F.*

Note that in phase i, we can confine ourselves to computing an α-approximate set-source shortest-path forest *up to distance* r_i (see Algorithm 2, Lines 9–10).

▶ **Problem 4** (Candidate-Merge Identification). *Given a graph $G = (V, E)$, edge costs $c\colon E \to \mathbb{N}_0$, and a rooted forest F' with a subset of its trees marked such that each node v in a marked tree knows that its tree is marked as well as the identity of its root τ_v, identify all edges $e = \{u, v\}$ that are in distinct marked trees and satisfy $c(e) = 0$.*

▶ **Problem 5** (Minimum Spanning Tree). *Given a graph $G = (V, E)$ with edge costs $c\colon E \to \mathbb{N}_0$, compute the Minimum Spanning Tree of G.*

▶ **Problem 6** (Root-Path Selection). *Given a graph $G = (V, E)$ with edge costs $c\colon E \to \mathbb{N}_0$, a rooted forest, and a set of marked nodes $S \subseteq V$, select the forest edges connecting each marked node to its root.*[12]

▶ **Problem 7** (Edge-Cost Reduction). *Given a graph $G = (V, E)$ with edge costs $c\colon E \to \mathbb{N}_0$, a radius r, and an output of Problem 3, compute, for each edge $e \in E$, $c'(e) = \max\left\{0, c(e) - \sum_{v \in e} \max\{r - d_{F'}(S, v), 0\}\right\}$.*

▶ **Problem 8** (Forest-Function Evaluation). *Given a graph $G = (V, E)$, a partition $\mathcal{C}$ of the node set such that each $C \in \mathcal{C}$ is connected, and a proper forest function $f\colon 2^V \to \{0, 1\}$, evaluate $f(C)$ for each component $C \in \mathcal{C}$.*

A.2.2 Meta-Algorithm Using the Building Blocks

Initialize F, $\mathcal{C}$, $\mathcal{T}^1$, c', and r as in Algorithm 2. Throughout our algorithm, we maintain a set of connected components $\mathcal{C}$ with activity statuses $f(C)$ for each $C \in \mathcal{C}$. At the beginning of phase 0, $\mathcal{C}$ contains exactly the singleton sets corresponding to all nodes, i.e., $\mathcal{C} = \{\{v\} \mid v \in V\}$, and the active components are the terminals. Each phase i of our algorithm (i.e., one loop iteration in Figure 1, simulating one while-loop iteration of Algorithm 2) then consists of the following steps, executed for $r = (1 + \varepsilon'')^i \cdot \frac{\varepsilon''}{4}$.

(1) *Approximate Set-Source Shortest-Path Forest (aSSSP).* Assign as temporary edge weight to $e \in E$ the reduced cost $c'(e)$ if $c'(e) > 0$ or $e \in F \cup F'$ (where $F' := \emptyset$ in phase 0). Compute a $(1 + \varepsilon')$-approximate (r-restricted) SSSP forest F', using the active terminals $\mathcal{T}^1 = \{\min\{v \mid v \in \mathcal{T} \cap C\} \mid C \in \mathcal{C}, f(C) = 1\}$ as sources, i.e., for each active component, $\mathcal{T}^1$ contains the terminal with the minimum identifier. After Step 1, for each node $v \in V \setminus \mathcal{T}^1$, we know its parent in the truncated SSSP forest, its closest source $u \in \mathcal{T}^1$ in the respective shortest-path tree (if any), and its distance $d_{F'}(u, v)$ to that source.

(2) *Edge-Cost Reduction (ECR).* Using the approximate r-restricted SSSP forest and the distances computed in Step 1, update the edge costs in accordance with Problem 7.[13]

(3) *Candidate-Merge Identification (CMI).* Using that nodes' parents and reduced edge costs are known, identify candidate merges M for adjacent trees of the aSSSP forest (Step 1).

(4) *Minimum Spanning Tree (MST).* Compute a Minimum Spanning Tree T of G with the following edge weights: (i) 0 for edges in F', i.e., the tree edges in the output of Step 1, (ii) 1 for edges in M, i.e., those determined in Step 3, and (iii) $+\infty$ (or a large value) for all other edges. Mark all selected edges of T that are also in the set M known from Step 3, i.e., the edges constituting A, and add them to F (thus *excluding* all edges with temporary weight greater than 1). For each connected component C' of the forest constituted by the selected edges of temporary weight 0 or 1 that contains a terminal $\tau \in C'$, set $\min\{v \in C' \mid f(\{v\}) = 1\}$ as the new identifier of the component C to be created from C', making it known to all $v \in C'$.

[12] Root-Path Selection can be reduced to *approximate transshipment* as follows: (i) count the number of marked nodes in each tree; (ii) set the demand of each marked node to -1 and the demand of the root of each tree to the number of marked nodes in the tree; (iii) set edge costs to 0 for tree edges and to $+\infty$ for all other edges; (iv) solve the approximate transshipment problem for these demands and edge weights; and (v) select all edges with non-zero flow in the output. Note that the only non-trivial step of the reduction is the computation of the demand, which boils down to a single Partwise Aggregation.

[13] We can keep the edge costs in $\mathbb{N}_0$ by making sure that phases end with integral values of r. Note that $\varepsilon' \geq n^{-\mathcal{O}(1)}$, or distance computations must be exact. Scale all weights by $\lceil 1/\varepsilon' \rceil$. Now rounding r up to the next integer has marginal impact on the approximation guarantee, as overgrowing by factor $(1 + \varepsilon')$ plus an additive 1 is not worse than overgrowing by factor $(1 + 2\varepsilon')$.

(5) *Root-Path Selection (RPS).* Connect the marked edges identified in Step 4 to the roots (i.e., the node with the same identifier as the component) of the components they connect by adding the necessary edges to F.

(6) *Forest-Function Evaluation (FFE).* Using the new component memberships known from Step 4, update the set $\mathcal{C}$ and evaluate $f(C)$ for each updated $C \in \mathcal{C}$. If $f(C) = 0$ for all such components, terminate and output F– else, continue with the next loop iteration.

A.3 Correctness and Complexity of our Model-Agnostic Meta-Algorithm

▶ **Theorem 14** (Model-Agnostic Shell-Decomposition Algorithm). *For $\varepsilon, \varepsilon', \varepsilon''$ as in Theorem 8, a graph $G = (V, E)$, edge costs $c\colon E \to \mathbb{N}$, and proper function f, the modular shell-decomposition algorithm (A.2.2) yields a feasible $(2+\varepsilon)$-approximation to the optimal solution.*

Proof. We prove the claim by induction on the phase i, going step by step through the algorithm given in Section A.2.2 and arguing why the computed objects, in particular F, match those of Algorithm 2. We augment the induction hypothesis by the claim that at the beginning of a phase, in Algorithm 2, E contains exactly the edges of non-zero reduced cost and $F_{i-1} \cup F'_{i-1}$. The induction anchor (phase $i = -1$) is given by the identical initialization of objects. For the step to phase $i \in \mathbb{N}_0$, observe first that by the induction hypothesis (in particular the additional claim), Step 1 computes the same F'_i and the same distances as Algorithm 2, and hence Step 2 yields the same c'_i. It follows that Step 3 computes the same set M of candidate merges, implying that Step 4 correctly determines A_i and adds it to F. Note that the latter step also updates component memberships and component identifiers, but does not yet evaluate whether $f(C) = 1$ for the new components. This is finally done in Step 6, such that in Step 1 of the next phase, the correct set $\mathcal{T}_i^1$ will be used. It remains to prove the additional claim that E_i contains exactly the edges of non-zero reduced cost and $F_{i-1} \cup F'_{i-1}$, which now is immediate from Line 17 and Lemma 9 (vii).

We conclude that both algorithms terminate at the end of the same phase j, returning the same forest $F = F_j$, which by Theorem 8 is a $(2 + \varepsilon)$-approximation. ◀

The proof of our main theorem now follows immediately.

Proof of Theorem 1. By Theorem 14, our modular shell-decomposition algorithm (Algorithm 2) delivers the desired approximation guarantee. Without loss of generality, we may assume that $\varepsilon \in n^{-\mathcal{O}(1)}$, as this is enough to enforce that ε times the cost of an optimal solution is smaller than 1, i.e., a 2-approximation is guaranteed. Thus, it is sufficient to instantiate the algorithm with $\varepsilon', \varepsilon'' \in n^{-\mathcal{O}(1)}$, such that by Lemma 10, the algorithm will terminate after $\mathcal{O}(\frac{\log n}{\varepsilon''}) = \mathcal{O}(\frac{\log n}{\varepsilon}) = \widetilde{\mathcal{O}}(\varepsilon^{-1})$ while-loop iterations. In each of these iterations (up to bookkeeping operations), aSSSP, MST, RPS, and FFE computations are performed exactly once, yielding the desired model-agnostic complexity. ◀