Approximate Minimum Directed Spanning Trees under Congestion

Christoph Lenzen$^{1[0000-0002-3290-0674]}$ and Hossein Vahidi$^{1,2(\boxtimes)[0000-0002-0040-1213]}$

^{1} MPI for Informatics, Germany, {clenzen,hovahidi}@mpi-inf.mpg.de
^{2} Saarbrücken Graduate School of Computer Science, Germany

Abstract. The *minimum directed spanning tree* (MDST) problem has until recently not been studied in distributed computing models. This fundamental task generalizes the well-studied minimum spanning tree problem, by asking for a minimum weight spanning tree rooted at some specified node of a directed network. In their DISC 2019 paper [9], Fischer and Oshman reduce the MDST problem to the single-source shortest path (SSSP) problem, with a polylogarithmic increase in running time. This holds both in the Congest and Congested Clique models.
Fischer and Oshman further suggest the possibility that an *approximate* SSSP algorithm could be leveraged in computing an *approximate* MDST. We extend their analysis to show that this is indeed the case: For $\varepsilon > 0$, using a $(1+\varepsilon)$-approximation to SSSP running in R rounds we can compute a $(1+\varepsilon)$-approximate MDST in $\tilde{O}(R)$ rounds.[3] In particular, this implies the following improvements in the state of the art for $(1+o(1))$-approximation of MDST.

- An $\tilde{O}(n^{1-2/\omega+o(1)}) \subset \tilde{O}(n^{0.158})$-round Congested Clique algorithm, where $\omega < 2.373$ is the fast matrix multiplication exponent [3].
- An $\tilde{O}(\lambda^2)$-round Congested Clique algorithm in graphs where each edge has an at most factor $\lambda \geq 1$ heavier reverse edge [1].
- An $\tilde{O}(\lambda^2(\sqrt{n}+D))$-round Congest algorithm in the same family of graphs [1]. For $\lambda \in \log^{O(1)} n$, the resulting running time of $\tilde{O}(\sqrt{n}+D)$ is unconditionally tight up to a polylogarithmic factor [21].

1 Introduction

Finding the minimum-weight spanning tree (MST) of a weighted undirected graph is a fundamental problem, which is well-studied in both sequential and distributed settings. In the more general minimum *directed* spanning tree (MDST) problem, we are given a weighted directed graph $G = (V, E, w)$ and a node $r \in V$. The goal is to determine a minimum-weight spanning tree rooted at r.

In this work, we study the task of finding a directed spanning tree (DST) rooted at r whose weight is at most by a factor of $1+\varepsilon$ larger than the optimum, or solving $(1+\varepsilon)$-approximate MDST for short, in the Congest and Congested Clique models.

[3] $\tilde{O}$-notation neglects polylogarithmic factors in the number n of nodes in the graph.

The Congest and Congested Clique models. In the Congest model, the network is represented by a weighted n-node graph $G = (V, E, w)$, where each node $v \in V$ has a unique $O(\log n)$-bit identifier. Initially, each node knows only its identifier and the weights of its incident edges, and it must determine which of its incident edges belong to the DST the algorithm chooses. Computation proceeds in synchronous compute-send-receive rounds, and we seek to minimize the number of communication rounds until all nodes have determined their local output and terminated. Message size is restricted to $O(\log n)$ bits, where each node may send a different message to each of its neighbors in each round. To avoid the complication that distances cannot be encoded by a single message, we follow the common convention to assume that edge weights are integers from a range that is polynomially bounded in n. The Congested Clique model is identical, except that each node may send an $O(\log n)$-bit message to each other node, not just its neighbors.

State of the art. While both the MST and the MDST problem have been extensively studied in the sequential setting, there is a surprising disparity in the distributed context. In all likelihood, the MST problem is *the* most well-studied task in the Congest model, and it received substantial attention in the Congested Clique as well. In contrast, despite its many applications, the community completely ignored the MDST problem until recently. Drawing inspiration from sequential [7, 10] and PRAM algorithms [17], Fischer and Oshman [9] presented a reduction of MDST to single-source shortest path in directed graphs (SSSP), the task of computing a shortest path tree with given root s. Their reduction incurs a round overhead of $\log^{O(1)} n$. Plugging in the currently best known algorithms for SSSP, one obtains a randomized $\tilde{O}(\sqrt{n}D^{1/4} + D)$-round Congest [4] and a deterministic $\tilde{O}(n^{1/3})$-round Congested Clique [3] algorithm, respectively.

Moreover, Fischer and Oshman show that in both models, MDST is at least as hard as finding a shortest path between two designated nodes $s, t \in V$. Thus, the round complexity of MDST is wedged in between those of SSSP and s-t path in both models. In Congest, even approximating s-t path in undirected graphs or an MST require $\tilde{\Omega}(\sqrt{n} + D)$ rounds [8, 20, 21]. In the Congested Clique, no non-trivial lower bounds are known, and finding even slightly super-constant such bounds would imply long-sought statements on circuit complexity [6].

However, the complexity of (directed) SSSP and s-t path are incompletely understood in either model, with polynomial gaps between upper and lower bounds. In addition, it is an open question whether *approximation* is easier. Currently, faster $(1 + \varepsilon)$-approximate SSSP algorithms for non-negative weights are known for the Congested Clique [3] and for special cases in Congest [1]. Thus, a reduction based on SSSP approximation is of interest. Indeed, Fischer and Oshman conjecture that a $(1 + \varepsilon)$-approximation to SSSP can be leveraged in their reduction to obtain a $(1 + \varepsilon)^{\lceil \log n \rceil}$-approximation to MDST [9].

Our contribution. We prove the stronger result that using a $(1+\varepsilon)$-approximate SSSP algorithm in the Fischer-Oshman framework, without modification, results in a $(1 + \varepsilon)$-approximation to MDST. This implies the following theorems.

Theorem 1. *For $\varepsilon > 0$, denote by $T_{\text{SSSP}}(n, D, \varepsilon)$ the round complexity of a $(1+\varepsilon)$-approximate SSSP Congest algorithm on directed n-node graphs with non-negative weights and (undirected) diameter D. Then a $(1+\varepsilon)$-approximate MDST can be computed in $\tilde{O}(T_{\text{SSSP}}(n, D, \varepsilon))$ rounds in the Congest model.*

Theorem 2. *For $\varepsilon > 0$, denote by $T_{\text{SSSP}}(n, D, \varepsilon)$ the round complexity of a $(1+\varepsilon)$-approximate SSSP Congested Clique algorithm on directed n-node graphs with non-negative weights and (undirected) diameter D. Then a $(1+\varepsilon)$-approximate MDST can be computed in $\tilde{O}(T_{\text{SSSP}}(n, D, \varepsilon))$ rounds in the Congested Clique.*

We note that the restriction to non-negative edge weights is natural when using approximation algorithms, as negative edge weights could cancel out with positive weights to result in a very light MDST despite heavy edges. This would decouple the notions of SSSP and MDST approximation to the point of meaninglessness, where, e.g., only an exact MDST has a negative total weight. Thus, any multiplicative MDST approximation would have to be an exact solution.

The above results are derived by generalizing the analysis of Fischer and Oshman in a model-independent way. Thus, whenever their framework can be applied, the same holds for our generalization.[4] However, for the sake of conciseness we confine the presentation to these two prominent models. In these models, pluggin in the most recent SSSP approximations yields the following.

Corollary 1 ([3, 11]). *For $\varepsilon > 0$, in the Congested Clique a $(1+\varepsilon)$-approximate MDST with non-negative weights can be found in $n^{1-2/\omega+o(1)} \subset O(n^{0.158})$ rounds, where ω is the matrix multiplication exponent in the Congested Clique.*

Corollary 2 ([1]). *Suppose $G = (V, E, w)$ satisfies that for each $(u, v) \in E$, it holds that $(v, u) \in E$ and $0 \leq w(u, v) \leq \lambda w(v, u)$. Then, for $\varepsilon > 0$, in the Congested Clique a $(1+\varepsilon)$-approximate MDST can be found in $\tilde{O}(\lambda^2/\varepsilon^2)$ rounds.*

Corollary 3 ([1]). *Suppose $G = (V, E, w)$ satisfies that for each $(u, v) \in E$, it holds that $(v, u) \in E$ and $0 \leq w(u, v) \leq \lambda w(v, u)$. Then, for $\varepsilon > 0$, in the Congest model a $(1+\varepsilon)$-approximate MDST can be found in $\tilde{O}((\sqrt{n}+D)\lambda^2/\varepsilon^{3/2})$ rounds w.h.p.*[5] *Deterministic correctness (i.e., only the running time bound could be violated) can be achieved in $\tilde{O}((\sqrt{n}+D)\lambda^2/\varepsilon^3)$ rounds w.h.p.*

Note that for $\lambda \in \log^{O(1)} n$ this matches the lower bound of $\tilde{\Omega}(\sqrt{n}+D)$ for shortest s-t path approximation from [21] up to a polylogarithmic factor.

On the technical level, we obtain our results by generalizing the analysis of the Fischer-Oshman framework. Intuitively, one might expect that it is sufficient to plug in the approximate SSSP algorithm instead of an exact routine. This turns out to be correct, but proving it requires to overcome a technical hurdle.

[4] For example, [9] also discusses a modification to the Congest model in which directed edges enable communication in one direction only. Similarly, we expect that the approach is efficient in the k-machine and semi-streaming models.

[5] W.h.p. stands for with high probability, which means with probability at least $1 - 1/n^c$ for a freely chosen, but fixed constant $c > 0$.

The exact MDST algorithm by Edmonds [7] heavily exploits that the graph manipulations it performs do not change the "remaining" MDST. More precisely, it first computes an edge set that may contain cycles, contracting these cycles whenever they are formed. It then updates edge weights in such a way that the set of MDST edges that are still to be selected are not affected by such changes. Once the selected subgraph is spanning, i.e., all nodes can be reached from r, Edmonds iteratively uncontracts all selected cycles and determines for each of them which edge has to be removed to obtain an MDST.

Fischer and Oshman simulate Edmonds algorithm in a manner that parallelizes well, yet can be efficiently implemented in distributed models. As their algorithm computes the same MDST as the one by Edmonds, the main challenge they face is efficient implementation. In contrast, our main obstacle is to relate the computed DST to an optimal MDST. Since we do not solve SSSP exactly, we choose different edges as Edmonds' algorithm, which in turn means that future SSSP instances might differ wildly from those in the exact algorithm. Hence, while we can build on [9] for an efficient implementation, our challenge is to argue that the computed DST is actually a good approximation to an MDST.

Facing this challenge from scratch would be a difficult task. As the immediate connection to an MDST breaks down, it is unclear how the complex evolution of the intermediate subgraphs could be tied to an MDST by an ad hoc argument. Even if one could be found, it would likely result in a proof repeating many of the steps for the exact setting.

We follow a different route, by re-interpreting the run of the approximation algorithm as a run of the *exact* algorithm on an *approximation* of the input graph. The effect of the modifications of the graph on the weight of an MDST can be easily bounded. The modifications of the graph are limited to scaling all edge weights by factor $1+\varepsilon$ and performing some simple changes to the graph forcing the exact algorithm to contract the exact same regions as the approximation algorithm. This also means that the exact algorithm incurs the same cost as the approximation algorithm on the original graph. Since the MDST of the modified graph is at most by factor $1+\varepsilon$ more costly than the original graph (as the original edge set is still present, albeit with factor $1+\varepsilon$ larger cost), the same follows for the approximate solution on the original graph. Overall, this results in a proof of the conjecture by Fischer and Oshman, without the need for any non-trivial modification to their algorithmic framework.

Further Related Work

Sequential MDST computation. Gabow et al. [10] provided an efficient implementation of an approach proposed independently by Edmonds [7], Bock [2] and, Chu and Liu [5], with step complexity $O(m+n\log n)$. The algorithm goes through a series of steps, also called Edmonds steps, and at each step, every vertex (except the root r) selects the lightest incoming edge, remembers its weight, and subtracts it from the weight of all the incoming edges. The idea is that to reach a vertex, we need to pay at least as much as its lightest incoming edge.

Then, all zero-weight cycles are contracted and we continue these steps recursively until there is only one component left. The sum of the weights subtracted by vertices through these steps is equal to the cost of an MDST of the graph. Finding an MDST requires a careful but simple unpacking of the contractions.

Lovasz' algorithm. The framework by Fischer and Oshman can be seen as a less aggressive way of parallelizing Edmonds steps than in Lovasz' PRAM algorithm [17]. For $\lceil \log n \rceil$ iterations, Lovasz performs an all-pairs shortest paths computation to find shortest paths that, similar to Edmonds approach, could be selected sequentially into an MDST. As in each of the iterations, each of the components induced by the currently selected edges but the one containing r is guaranteed to connect to another component, in the end only a single component remains. Then a similar unpacking procedure yields an MDST. The key observation by Fischer and Oshman is that it is perfectly sufficient to find for each component the shortest path leaving it (when flipping the directions of all edges). This more conservative approach avoids that the computations of different components "overlap," which causes the need for an all-pairs shortest path algorithm for Lovasz, yet also ensures termination within $\lceil \log n \rceil$ iterations.

MST in Congest. While the MDST problem has been the neglected child in the family of global Congest problems, its little brother, finding an MST in an undirected graph, has been showered with attention; we make no attempt at covering these results here, see [19] for a recent survey. In part, this is likely due to habit, as MST is a canonical global problem that lends itself well to studying new models and complexity measures. On the other hand, the problem is closely related to testing whether a given subgraph is connected. Any MST algorithm can solve this task, and the lower bound bound of $\tilde{\Omega}(\sqrt{n} + D)$ due to Das Sarma et al. [21] applies to both tasks. Due to the $\tilde{O}(\sqrt{n} + D)$-round algorithm by Kutten and Peleg [15], this implies that there appears to be virtually no difference between the two problems. As many Congest algorithms for global problems need to test subgraphs for connectivity, this close connection means that insights on MST construction frequently transfer to tasks which appear unrelated at first sight. It also explains why so many problems turn out to have round complexity $\tilde{\Theta}(\sqrt{n} + D)$.

We remark that for MST, it is known that allowing approximate solutions or randomization does not make the job easier [8, 21]. As discussed above, for the MDST problem this will depend on whether either makes SSSP (or possibly s-t path) easier in the considered models, with remains open to date.

MST in the Congested Clique. The Congested Clique was introduced by Lotker et al. in 2003 [16].[6] Naturally, the MST problem served as their guinea pig, and they provided an $O(\log \log n)$-round algorithm. After more than a decade of silence, a flurry of results [12, 13, 14] culminated in a deterministic constant-round solution [18].

[6] In the MST problem, heavy edges can be added without changing the solution. Hence, decoupling problem and communication graph was formalized only later.

2 Preliminaries

Let $G = (V, E, w)$ be an n-node directed graph with edge weight function w. Let $S \subseteq V(G)$ be a vertex set. Then, the corresponding induced subgraph in G is denoted by $G[S] = (S, E')$, where $E' = \{(u, v) \mid u, v \in S \text{ and } (u, v) \in E(G)\}$. Moreover, the weight of subgraph $H \subseteq G$, denoted by $w_G(H)$, is the sum of the weight of its edges. For any graph H, $V(H)$ and $E(H)$ denote its vertex and edge set, respectively, and w_H its weight function. A subgraph is weakly connected, if its underlying undirected graph is connected. Weakly connected components are defined accordingly. For $u, v \in V$, let $\mathrm{dist}_G(u, v)$ be the weight of the shortest path from u to v in G. Denote by $B_G(v, r)$ the ball of radius r around node v, i.e., $\{u \mid \mathrm{dist}_G(u, v) \leq r\}$. An s-t path P can be represented as a tuple of its vertices in the order of appearance, e.g., $P = (s, v_1, \dots, v_k, t)$.

Contraction of an edge $e = (u, v)$ is the following operation. We combine u and v into a supernode x that keeps all the incident edges of u and v. Then, self-loops are removed and in the case of parallel edges between two nodes we only keep the lightest edge. Contracting a ball $B_G(v, r)$ is similar, in that all nodes inside the ball are merged into a supernode x. However, an edge $(u, u') \in (G \setminus B_G(v, r)) \times B_G(v, r)$ gives rise to edge (u, x) of weight $w_G(u, u') + \mathrm{dist}_G(u', v) - r$. Analogously, outgoing edge $(u, u') \in B_G(v, r) \times (G \setminus B_G(v, r))$ results in edge (x, u') of weight $w_G(u, u') + \mathrm{dist}_G(v, u) - r$. Again, self-loops are removed and only the lightest edge is kept. Finally, we will need to contract "approximate" balls, where we consider $B_T(v, r)$ in a graph G, with T being a tree. In such a case, we replace the dist_G terms by dist_T, even for edges that are present in G, but not in T. Note that this can result in negative edge weights. In such a case, the resulting contracted edge will be assigned weight 0.

When referring to an *uncontraction,* this means to reverse the above process. If we uncontract a graph with a selected spanning tree T that resulted from contraction of G, each edge of T induces a marked edge in G, which caused it to be assigned its weight. For convenience, we refer to this as *uncontracting* T.

3 Exact MDST Computation

We will modify the Fischer-Oshman approach [9] to provably work with approximate SSSP computations. For an intuitive understanding as well as a formal proof, it is instructive to revisit their technique.

In the following and throughout this paper, w.l.o.g. let us make the following assumptions. First, the input graph $G = (V, E, w)$ has indeed a spanning tree rooted at r, i.e., an MDST rooted at r exists. Otherwise the algorithm will simply fail to compute such an MDST, and this can be verified fast enough in the considered models. Second, we assume that there is no edge with endpoint r, as no DST with root r contains such an edge. Finally, we assume that all edge weights are different; any kind of consistent tie-breaking mechanism results in equivalent behavior.

A variant of Edmonds' algorithm. In MST construction, it is commonly exploited that the lightest edge of a cut, and in particular the lightest incident edge to each node, are part of the MST. Crucially, these statements assume a consistent tie-breaking mechanism in place, which ensures that no cycle can be closed when selecting all such edges concurrently: it does not matter which edge of a cycle in which all edges have the same weight is selected, but one edge must be excluded. In the directed setting, it is no longer arbitrary which edge of a directed cycle of "candidate edges" is excluded, as for reachability it now matters at which node the path from the root to the cycle in the MDST ends. Lemma 1 provides a highly useful structural property of MDSTs corresponding to these observations.

Lemma 1 (implicit in [7]). *Define $G' = (V, E, w')$ by setting $w'(u, v) := w(u, v) - \min_{(u',v)\in E}\{w(u', v)\}$ for all $(u, v) \in E$. Then an MDST T of G is also an MDST of G', where $w(T) = w'(T) + \sum_{v\in V\setminus\{r\}} \min_{(u',v)\in E}\{w(u', v)\}$. Moreover, let $E_0 := \{(u, v) \in E \mid w'(u, v) = 0\}$ denote the pseudo-forest*[7] *given by the 0-weight edges of G'. Then for each cycle C in E_0 there is a unique edge $(u, v) \in C \setminus T$.*

Proof. Each non-root node has exactly one incoming edge in any DST, so the weight of *each* DST rooted at r changes by $\sum_{v\in V\setminus\{r\}} \min_{(u',v)\in E}\{w(u', v)\}$ when replacing w by w'. For the second claim, observe that any cycle C in E_0 must have at least one of its edges not be part of T. So assume for contradiction that there is a cycle C such that $|T \cap (V \setminus C \times C)| > 1$. Let $(u, v) \in T \cap (V \setminus C \times C)$ and denote by $(u', v) \in E_0$ the unique edge in E_0 with endpoint v. Then $T' := T \setminus (u, v) \cup (u', v)$ is a DST: there is still some node in C reachable from r in $T \setminus (u, v)$, which then inductively applies to all its successors in C, including u'. However, $w'(u, v) > 0$ and $w'(u', v) = 0$ by choice of E_0, implying that $w'(T') < w'(T)$, a contradiction to the already established claim that T is an MDST of G'. □

Lemma 1 suggests the following procedure to compute an MDST. For each non-root node select the cheapest incoming edge and subtract its weight from all incoming edges of the node. Contract the resulting 0-weight cycles and repeat the process on all the newly created supernodes until no non-root node without a selected incoming edge remains. At this point, the 0-weight edges form an MDST of the current graph (this will be shown in the proof of Lemma 2). To get an MDST of the original graph, we go in the reverse order, uncontracting the 0-weight cycles of the previous step, and adjusting the MDST to the graph resulting from uncontraction. To achieve the latter, for each new node without an incoming edge in the old MDST, we pick a 0-weight incoming edge from the uncontracted cycle. Lemma 2 shows that this procedure, whose pseudocode is given in Algorithm 1, indeed yields an MDST of G.

[7] By the above assumptions, each non-root node has exactly one 0-weight incoming edge, while the root has none. However, E_0 might contain cycles.

Algorithm 1 Finding an MDST of a Graph

```
1: procedure EDMONDS($G = (V, E, w)$)
2:     $i := 0, G^{(0)} := (V^{(0)}, E^{(0)}, w^{(0)}) := G$
3:     while $(V^{(i)}, \{e_v^{(i)} \mid v \in V^{(i)}\})$ is not a DST of $G^{(i)}$ do
4:         $i \leftarrow i + 1$
5:         Each node $v \in V^{(i-1)} \setminus \{r\}$ sets $e_v^{(i)} := \operatorname{argmin}_{(u,v)\in E^{(i-1)}} \{w^{(i-1)}(u, v)\}$
6:         For each $(u, v) \in E^{(i-1)}$, set $w^{(i)}(u, v) := w^{(i-1)}(u, v) - w^{(i-1)}(e_v^{(i)})$
7:         Contract all 0-weight cycles, resulting in $G^{(i)} = (V^{(i)}, E^{(i)}, w^{(i)})$
8:     Set $T^{(i)} := \{e_v^{(i)} \mid v \in V^{(i)}\}$                    ▷ uncontraction phase
9:     while $i > 0$ do
10:        Initialize $T^{(i-1)}$ by the set of edges obtained by uncontracting $G^{(i)}$ to $G^{(i-1)}$
11:        Let $X$ be the set of nodes that have been in a zero-cycle before contraction
12:        for each $v \in X$ without an incoming edge in $T^{(i)}$ do
13:            $T^{(i-1)} \leftarrow T^{(i-1)} \cup \{e_v^{(i-1)}\}$          ▷ add its incoming zero-weight edge
14:        $i \leftarrow i - 1$
15:    return $T^{(0)}$
```

Lemma 2 (implicit in [7]). *Algorithm 1 computes an MDST of G. Denoting by $i_{\max}$ the maximum value of i throughout the procedure, its weight equals $\sum_{i=1}^{i_{\max}} \sum_{v\in V^{(i-1)}\setminus\{r\}} w^{(i-1)}(e_v^{(i)})$.*

Proof. Observe that contracting a cycle of weight 0 in a graph without negative edge weights cannot change the cost of an MDST: Regardless of how an MDST of the graph after contraction looks like, we can connect all nodes in the cycle at cost 0 in the original graph, and we can delete a non-cycle edge (of cost at least 0) for each cycle this closes. Because Lines 5 and 6 ensure that the new weights satisfy this property, applying Lemma 1 inductively to the first while loop shows that $\sum_{i=1}^{i_{\max}} \sum_{v\in V^{(i-1)}\setminus\{r\}} w^{(i-1)}(e_v^{(i)})$ is precisely the cost of an MDST of G.

Hence, it remains to show that the computed edge set $T^{(0)}$ is indeed an MDST. To this end, we show by induction that $T^{(i)}$ is an MDST of $G^{(i)}$ for all $i \in \{0, \dots, i_{\max}\}$. To anchor the induction at $i = i_{\max}$, note that by Lines 5 and 6, it holds that $w^{(i_{\max})}(e_v^{(i)}) = 0$ for all $v \in V^{(i_{\max})}$ and $w^{(i_{\max})}(u, v) \geq 0$ for all $(u, v) \in E^{(i_{\max})}$. Hence, by the halting criterion of the first loop, $T^{(i_{\max})}$ is indeed an MDST of $G^{(i_{\max})}$. Assuming that the claim holds for $i > 0$, Line 13 and the fact that $T^{(i)}$ is a tree ensure that each non-root node has indegree 1. Moreover, as $T^{(i)}$ is spanning, we can reach each node $v \in V^{(i)}$ from r by taking the edges in $E^{(i-1)}$ corresponding to the respective path in $T^{(i)}$ and, wherever a path node gets uncontracted into a cycle C, adding the path in C connecting the endpoint of the incoming edge to the node with the outgoing edge (or v if it is on the cycle). Thus, $T^{(i-1)}$ is a spanning pseudo-forest, implying that it must be a DST. The minimality of $w(T^{(i-1)})$ follows from the fact that $T^{(i)}$ is an MDST of $G^{(i)}$, the weight changes of Line 13, and Lemma 1. □

The Fischer-Oshman framework. The above observations are promising in that in each iteration, all (weakly) connected components can operate concurrently.

Despite the technical obstacle that cycle contraction is problematic in Congest, because the communication graph does not change, Fischer and Oshman show how to perform all operations with sufficient efficiency. However, a second, more important hurdle is that the above procedure might require a lot of iterations. In the worst case, each added edge closes another cycle rather than reducing the number of connected components. This means that $\Omega(n)$ contractions could be performed sequentially, resulting in a slow algorithm.

More concretely, note that after selecting the lightest incoming edges, the resulting subgraph has exactly one cycle in each weakly connected non-root component. After contraction, we get a forest whose roots are the nodes resulting from cycle contraction. These roots are exactly the "new" nodes selecting new edges, while all other nodes stick to their previously selected edges whose reduced weight is 0. Such a node selecting a new edge and immediately contracting the resulting cycle is referred to as an *Edmonds step.* If both endpoints of the edge selected by an Edmonds step are in the same weakly connected component, another cycle within this component is formed, whose contraction might eliminate no more than a few nodes.

Let H be a non-root component formed by the currently selected edges and let C_H be the unique cycle at the heart of H. The crucial insight Fischer and Oshman exploit is the following. The iterations of the above Algorithm 1 on H until an edge into the component is selected are equivalent to running Dijkstra's algorithm on the component, where the contracted cycle, C_H, is the source, and we reverse edges. Thus, the selected edges contain a shortest path P of weight β_H from outside the connected component to the (original) cycle. Equivalently, we can find β_H and contract $B_H(C_H, \beta_H)$, i.e., a ball of radius β_H centered at C_H, where edges that are only partially inside the ball lose a respective share of their weight. Note that this does not affect other components, until the resulting 0-weight edge connecting to another component is contracted. Hence, this operation can be performed on all weakly connected non-root components concurrently, and Fisher and Oshman formalize how to achieve this using a call to a single, globally operating SSSP instance. Doing so constitutes a *mega-step* in the Fischer-Oshman terminology. As each component gets merged with at least one other component in a mega-step, the process terminates after at most $\lceil \log n \rceil$ mega-steps.

Fischer and Oshman prove that the necessary bookkeeping and the later uncontractions can be efficiently implemented in both the Congest and Congested Clique models, rendering the SSSP algorithm the subroutine that dominates the round complexity. In Congest this implementation is rather involved. Fortunately, as we will show that we can replace the exact SSSP computation with an approximate one *in a blackbox fashion,* we can confine our presentation to the abstract viewpoint of explicitly performing contractions and uncontractions. We summarize the relevant structural results by Fischer and Oshman as follows.

Lemma 3 (Fischer and Oshman [9]). *Denote by $G^{(i)} = (V^{(i)}, E^{(i)}, w^{(i)})$ the graph after i mega-steps. Denote by $H \not\ni \{r\}$ a weakly connected component* *of the selected edges and by C_H its unique node with indegree 0. Then contracting $B_{G^{(i)}}(C_H, \beta_H)$ simulates multiple Edmonds steps on H.*

A mega-step is performed by executing these operations on each weakly connected non-root component and then contracting 0-weight cycles. We stress that, while the intuition as to why the algorithm is correct remains the same as for Algorithm 1 – both can be decomposed into a sequence of Edmonds steps – the iterations of the first loop of Algorithm 1 cannot be consistently mapped to mega-steps.

Nonetheless, constructing an MDST after contracting the graph into a root component is done in a similar fashion. One uncontracts in reverse order of the mega-steps, operating on all components in parallel. In this process, one maintains the invariant that the current tree $T^{(i)}$ is an MDST of the current graph $G^{(i)}$, starting with $G^{(i_{\max})}$. After uncontracting to $G^{(i-1)}$, we need to connect to the cycle of each component H, which is done by selecting the computed shortest path π_H from outside of H to its cycle. Finally, the remaining nodes without incoming edge add their previosly selected edge of weight 0. This ensures reachability of all nodes in the component, as (i) all path nodes are reachable from the "parent" component, (ii) all nodes on the cycle are reachable from the endpoint of the path along the cycle edges, and (iii) the remaining nodes are attached to the path and cycle nodes by a forest of 0-weight edges. Because $T^{(i)}$ was a DST, so is $T^{(i-1)}$, and as the weight of the added edges in $G^{(i-1)}$ sums up to $\sum_{H\in\mathcal{H}} w^{(i-1)}(\pi_H)$, $T^{(i-1)}$ is an MDST of $G^{(i-1)}$.

Lemma 4 (follows from Lemma 8 in [9]). *Let $G^{(i-1)}$ be a graph with a set of 0-weight components $\mathcal{H}$ in accordance with Lemma 3 after $i-1 \geq 0$ mega-steps. Let $G^{(i)}$ be the graph obtained by contracting for each $H \in \mathcal{H}$ the ball $B_H(C_H, \beta_H)$, where β_H is the length of a shortest path π_H that has only its first node outside H and ends in C_H. Let $T^{(i)}$ be an MDST of $G^{(i)}$.*

Then setting $T^{(i-1)}$ to the edge set resulting from uncontraction of $T^{(i)}$ and performing the following operation for each $H \in \mathcal{H}$ yields an MDST:

- *uncontract C_H (into a 0-weight cycle),*
- *add the edges of π_H inside H (the first edge of π_H has been selected in the uncontraction of $T^{(i)}$), and*
- *add each 0-weight edge of H and C_H whose endpoint is not a node of π_H.*

We stress that the proof from [9] requires that the paths π_H are shortest paths only to establish that the cost of the weight of the constructed DST is minimal. Thus, we have the following corollary.

Corollary 4. *Assume the same setting as in Lemma 4, except that the paths π_H are not necessarily shortest paths. Then the construction of the lemma yields a DST of weight equal to $w^{(i)}(T^{(i)}) + \sum_{H\in\mathcal{H}} w^{(i-1)}(\pi_H)$.*

In summary, the challenge for obtaining an approximation algorithm lies in relating the weight of Corollary 4 to the weight of an MDST.

4 $(1+\varepsilon)$-approximate MDST from SSSP Approximation

Throughout this section, we assume that all edge weights are non-negative. The algorithm maintains this invariant when modifying edge weights. Hence, we can rely on SSSP approximation algorithms that assume non-negative edge weights as well. Formally, such an algorithm provides the following output.

Definition 1 ($(1+\varepsilon)$-approximate SSSP). *For a given graph G and root r, a $(1+\varepsilon)$-approximate directed SSSP algorithm returns a DST T of G such that $\mathrm{dist}_T(r,v) \leq (1+\varepsilon)\cdot \mathrm{dist}_G(r,v)$ for all $v \in V(G)$. The output is given by each $v \in V(G)\setminus\{r\}$ learning about its parent and $\mathrm{dist}_T(r,v)$.*

For the remainder of the section, $\mathcal{A}$ denotes an algorithm following the Fischer-Oshman framework [9] for computing an MDST, while $\mathcal{A}'$ denotes our approximate version, which is obtained by replacing the exact directed SSSP solution by a $(1+\varepsilon)$-approximation and using the distances in the tree(s) instead of the exact ones. Recall that we enforce that the minimum edge weight resulting from a contraction is 0, cf. Section 2, so the approximate SSSP algorithm will always operate on graphs with non-negative weights.

In this section, we will establish the following theorem.

Theorem 3. *If $w(u,v) \geq 0$ for all $(u,v) \in E$, $\mathcal{A}'$ computes a $(1+\varepsilon)$-approximate MDST.*

Theorems 1 and 2 readily follow from this theorem and the running time bounds from [9]. In more detail, besides calling the SSSP subroutine, their framework uses $\tilde{O}(\sqrt{n}+D)$ and $\log^{O(1)} n$ rounds per mega-step in Congest and the Congested Clique, respectively, yielding running times of $\tilde{O}(T_{\mathrm{SSSP}}(n,D,\varepsilon)+\sqrt{n}+D)$ in Congest and $\tilde{O}(T_{\mathrm{SSSP}}(n,D,\varepsilon)$ in the Congested Clique. In Congest, the lower bound of $\tilde{\Omega}(\sqrt{n}+D)$ on any polynomial approximation to SSSP [21] implies that $\tilde{O}(T_{\mathrm{SSSP}}(n,D,\varepsilon)+\sqrt{n}+D) = \tilde{O}(T_{\mathrm{SSSP}}(n,D,\varepsilon))$.

Why analyzing the use of approximate SSSP is challenging. We would like to replace the exact SSSP computation in the Fischer-Oshman framework with an approximate one, assuming that the graph has non-negative weights. The trouble with that lies in the analysis of the algorithm. Where Fischer and Oshman argue that they simulate Edmonds steps, we run into the obstacle of relating the computed solution to an optimal one. The sequence of contractions and, accordingly, uncontractions performed can be vastly different even with only minor changes in the outcome of the SSSP computation.

A simple attempt at fixing this issue could be to modify the graph to enforce that the approximately shortest paths found by the subroutine become actual shortest paths in a slightly distorted topology. Unfortunately, simply scaling down the edge weights of such paths (or scaling up the non-path edges' weights) might not achieve this. A tree that approximates distances to the source up to factor $1+\varepsilon$ can still contain edges (p,c) for which parent p and child c satisfy that $\mathrm{dist}_G(p,c) \ll w(p,c)$ – the local error can be amortized over a much larger distance to the root. This means that we do not have enough information to adjust edge lengths such that (i) the computed paths become shortest paths and (ii) the weight of an MDST changes little.

We overcome the above obstacle by modifying both the topology and, at least on a formal level, also generalizing the solved problem. Intuitively, we still follow the strategy given above, but we accept that we introduce additional edges and nodes into the graph. Fortunately, these modifications are necessary only to create an execution of the Fischer-Oshman algorithm that we can compare to in order to establish the approximation guarantee; no change whatsoever is needed in the actual algorithm beyond the discussed replacement of the SSSP subroutine and performing contractions of the resulting "approximate" balls.

4.1 Shortcuts and Dummy Nodes

The modifications we make to the graph used by the exact algorithm require some bookkeeping. Denote for each node $u \in V$ and each mega-step i by $S_u^{(i)}$ the total amount that node u (or the supernodes resulting from it) would subtract from each of its incoming edges from outside the contracted balls during contractions up to and including mega-step i.[8] In other words, adding $S_u^{(i)}$ to the current weight of an incoming edge restores its weight in the original graph. Similarly, we apply this to the shortcut edges. We only ever need to add edges (u, v) that have residual weight $\beta > 0$ after the first $i-1$ mega-steps. To generate a corresponding edge for $U^{(i)}$, we pick arbitrary nodes $u', v' \in V$ that ultimately get contracted into u and v, respectively, and set the weight of (u', v') in $U^{(i)}$ to $\beta + S_{v'}^{(i-1)}$.

Simply put, our construction first stretches all edges by factor $1 + \varepsilon$ for $\mathcal{A}$ to obtain $U^{(0)}$, so that $\operatorname{opt}(U^{(0)}) = (1 + \varepsilon) \operatorname{opt}(G)$.[9] Then, in mega-step i, we add "shortcuts" to obtain $U^{(i)}$ from $U^{(i-1)}$. These shortcuts satisfy that (i) they do not affect any already performed mega-steps when using $U^{(i)}$ as input rather than $U^{(i-1)}$ and (ii) $\mathcal{A}$ performs the same contractions (i.e., with the same ball centers and radii) on $U^{(i)}$ as $\mathcal{A}'$ on $G^{(i-1)}$ in mega-step i. Thus, after the last iteration $i_{\max}$, we have a one-on-one mapping of the sequence of contractions of both algorithms.

Recall that the ball radii are also the cost the exact algorithm charges to its contractions (cf. Lemma 3), summing up to the weight of the computed MDST (cf. Lemma 4). $\mathcal{A}'$ charges the same cost to its contractions, corresponding to the weights of the *approximately* shortest paths its SSSP subroutine found. As these weights add up to the cost of the computed DST in the same way as for $\mathcal{A}$, the weight of the DST of G computed by $\mathcal{A}'$ equals the weight of an MDST of $U^{(i_{\max})}$. Finally, we show that going from $U^{(i-1)}$ to $U^{(i)}$ can only decrease the weight of an MDST, implying that $\operatorname{opt}(U^{(i_{\max})}) \leq \operatorname{opt}(U^{(0)}) = (1 + \varepsilon) \operatorname{opt}(G)$.

[8] "Would" here indicates that nodes might be inside a contracted region without edges to the outside.

[9] While this may result in non-integral edge weights, they can still be easily represented with $O(\log n)$ bits.

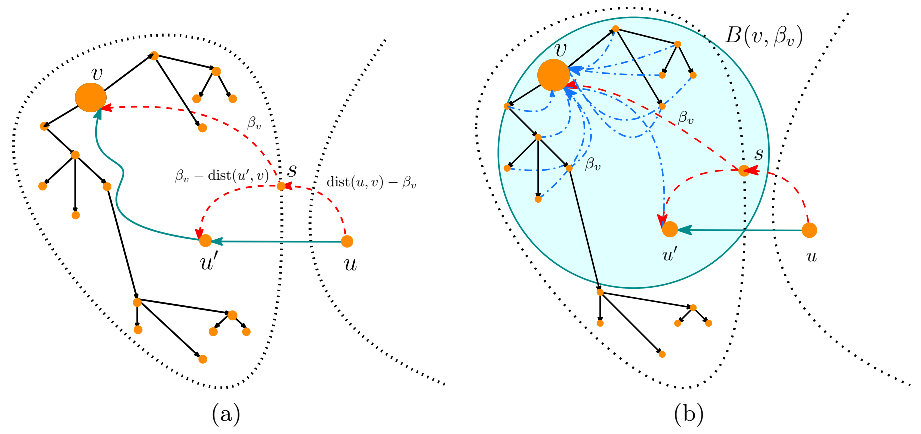

Fig. 1: Dashed red edges represent gadget replacement at a boundary edge and dash dotted blue edges represent shortcut edges. Note that the contraction of $B(v, \beta_v)$ eliminates all the added edges and nodes again; the construction merely ensures that the exact algorithm contracts exactly $B(v, \beta_v)$. Note that to obtain the graph before contractions, the introduced edges will be connected to some nodes inside the contracted supernodes (see Definition 2).

Meeting the requirements (i) and (ii) concurrently is tricky. We achieve this by introducing a gadget that subdivides boundary edges according to our needs, without affecting the weight of an MDST.[10] Together with the right shortcut edges, the resulting gadget shapes $B_{U^{(i)}}(v, \beta_v)$ in the right way without interfering with the algorithm's prior execution or the weight of an MDST.

Definition 2 (Shortcuts with Dummy Nodes). *Consider the graph $G^{(i-1)}$ after $i - 1$ mega-steps of the exact algorithm on $U^{(i-1)}$. For each $v \neq r$ without selected incoming edge denote by β_v the ball radius computed using the approximate SSSP algorithm (i.e., the distance for leaving the weakly connected component in the tree computed by the SSSP approximation algorithm). For each edge $(u, u') \in (V(G^{(i-1)}) \setminus B_{G^{(i-1)}}(v, \beta_v)) \times B_{G^{(i-1)}}(v, \beta_v)$, we replace the edge in $G^{(i-1)}$ by the following gadget (see Figure 1):*

- *A new dummy node s.*
- *An edge (u, s) of weight $\mathrm{dist}_{G^{(i-1)}}(u, v) - \beta_v$.*
- *An edge (s, u') of weight $\beta_v - \mathrm{dist}_{G^{(i-1)}}(u', v)$.*
- *An edge (s, v) of weight β_v.*

We then obtain $U^{(i)}$ from $U^{(i-1)}$ by adding the dummy nodes and changing the edge set of $U^{(i)}$ as follows.

[10] This holds true under the assumptions that the spanning tree needs not contain the added vertices, which is sufficient for our purposes.

- *Remove all edges from nodes that are contracted into u to nodes that are contracted into u'.*
- *For each new node s and each of its edges, denote by x the endpoint that is not s and by w its weight. Choose an arbitrary node $y \in V$ that got contracted into x (or x itself if $x \in V$). Add an edge with endpoints s and y of matching orientation and weight $w + S_y^{(i-1)}$ to $U^{(i)}$.*
- *For each $u \in B_{G^{(i-1)}}(v, \beta_v)$, denote by $x, y \in V$ nodes that got contracted into u and v, respectively. Add (x, y) with weight $\beta_v + S_y^{(i-1)}$ to $U^{(i)}$.*

Dummy nodes need not be spanned by a DST, i.e., $\mathrm{opt}(U^{(i)})$ denotes the minimum cost of a tree rooted at r spanning $V(G)$. To match our needs, $\mathcal{A}$ is a slighty different version of the Fischer-Oshman algorithm, where dummy nodes do not seek to select edges "on their own." However, they can take part in the shortest paths the algorithm selects. These changes are exactly those that make the algorithm "behave the same way" on $U^{(i)}$ and $U^{(i-1)}$ until mega-step i.

Corollary 5. *$\mathcal{A}$ computes a lightest tree that is rooted at r and spans $V(G)$.*

Proof (sketch). A simple check of the arguments in Section 3 shows that a tree spanning all non-dummy nodes is computed. To see optimality, note that it still holds that *if* a dummy node takes part in the tree, we need to pay at least the weight of its lightest incoming edge to include it (cf. Lemma 1). Thus, for each 0-degree non-root regular node we must pay at least as much as the weight of the shortest path connecting to it from outside its weakly connected component, as it needs to get connected to the root component in some way (cf. Lemma 3). □

To relate $\mathrm{opt}(U^{(i)})$ to $\mathrm{opt}(G)$, we first show that the replacement of Definition 2 does not introduce negative cycles or otherwise unduly distort the distance structure of the graph.

Lemma 5. *For any $i \geq 0$, in the graph resulting from applying gadgets of Definition 2 to $G^{(i)}$ the following holds:*

1. *no negative-weight cycle exists,*
2. *for each edge $(u, u') \in (V(G^{(i)}) \setminus B_{G^{(i)}}(v, \beta_v)) \times B_{G^{(i)}}(v, \beta_v)$, the distance from u to v does not change,*
3. *for each $u' \in B_{G^{(i)}}(v, \beta_v)$ the distance from u' to v does not change.*

Proof. The proof is by induction on i. Note that initially all edges have non-negative weight, so no negative-weight cycle is present. Let $E_i = \{e_1, \ldots, e_m\}$ be the set of boundary edges that need to be replaced by a gadget according to Definition 2. We perform an induction over the individual replacements, where we maintain the above invariants. For $j \in [m]$, let $\hat{G}_j$ be the graph resulting from replacement of the first j edges of E_i in $G^{(i)}$ (in particular, $\hat{G}_0 = G^{(i)}$).

Observe that, after replacement of $e_j = (u, u')$, any cycle involving s contains (u, s) and either (s, u') or (s, v). By definition 2, we have that $w_{\hat{G}_j}(u, s) + w_{\hat{G}_j}(s, v) = \mathrm{dist}_{G^{(i)}}(u, v) \geq 0$, so no negative cycle can be formed containing

(u,s) and (s,v). By the induction hypothesis, in $\hat{G}_{j-1}$ distances are well-defined (i.e., non-negative and satisfying the triangle inequality). Therefore, any cycle involving (u,s) and (s,u') is of weight at least

$$\begin{aligned}
&w_{\hat{G}_j}(u,s) + w_{\hat{G}_j}(s,u') + \mathrm{dist}_{\hat{G}_{j-1}}(u',u) \\
=&\,\mathrm{dist}_{G^{(i)}}(u,v) - \mathrm{dist}_{G^{(i)}}(u',v) + \mathrm{dist}_{\hat{G}_{j-1}}(u',u) && \text{Definition 2} \\
\geq&\,\mathrm{dist}_{G^{(i)}}(u,v) - \mathrm{dist}_{G^{(i)}}(u',v) + \mathrm{dist}_{\hat{G}_{j-1}}(u',v) - \mathrm{dist}_{\hat{G}_{j-1}}(u,v) && \Delta\text{-inequality} \\
=&\,0 && \text{I.H.}
\end{aligned}$$

This shows the first part of the invariant for index j. In particular, w.l.o.g. we may consider only simple path for the remainder of the proof.

For the second part, it is sufficient to show that the distance from u to v does not change, as then the same follows for all other considered edges by the induction hypothesis. To see that this holds true, observe first that the path (u,s,v) has weight $\mathrm{dist}_{G^{(i)}}(u,v)$ by construction, implying that $\mathrm{dist}_{\hat{G}_j}(u,v) \leq \mathrm{dist}_{G^{(i)}}(u,v)$. To prove that also $\mathrm{dist}_{\hat{G}_j}(u,v) \geq \mathrm{dist}_{G^{(i)}}(u,v)$, consider the simple paths from u to v. If they do not contain s, the induction hypothesis implies that they are not too light. The remaining paths are (u,s,v) (which we considered) and simple paths containing (u,s,u'). Any of the latter has weight at least

$$\begin{aligned}
&w_{\hat{G}_j}(u,s) + w_{\hat{G}_j}(s,u') + \mathrm{dist}_{\hat{G}_{j-1}}(u',v) \\
=&\,\mathrm{dist}_{G^{(i)}}(u,v) - \mathrm{dist}_{G^{(i)}}(u',v) + \mathrm{dist}_{\hat{G}_{j-1}}(u',v) = \mathrm{dist}_{\hat{G}_{j-1}}(u,v) \qquad I.H.
\end{aligned}$$

Thus, the second part of the invariant holds for index j.

It remains to show the third part of the invariant. Since the gadget replacement does not affect paths within $B_{G^{(i)}}(v,\beta_v)$, for each $u' \in B_{G^{(i)}}(v,\beta_v)$ we have that $\mathrm{dist}_{\hat{G}^{(j)}}(u',v) \leq \mathrm{dist}_{G^{(\hat{j}-1)}}(u',v)$. Assuming for contradiction that $\mathrm{dist}_{\hat{G}^{(j)}}(u',v) < \mathrm{dist}_{G^{(\hat{j}-1)}}(u',v)$, this must be due to a (simple) path containing s. By the already established second part of the invariant for index j, such a path cannot contain both u and u', as the subpath from u to u' would have weight at least $\mathrm{dist}_{\hat{G}_{j-1}}(u,u')$, i.e., the invariant would be violated for index $j-1$. However, by Definition 2 the edge (s,v) has weight β_v, which equals the weight of (s,u') plus $\mathrm{dist}_{G^{(i)}}(u',v)$. Thus, if (u,s,v) would be too light, so would be some path involving both u and u'. The third part of the invariant follows. □

4.2 Proving Theorem 3

Denote by $G^{(i)'}$ the graph algorithm $\mathcal{A}'$ computed after i mega-steps. Denote by $T_v^{(i)}$ an approximate shortest path tree of $G^{(i-1)'}$ rooted at v returned by approximate SSSP algorithm. For each $v \in V \setminus \{r\}$ with indegree 0, let β_v be the cost of minimum approximate shortest path entering the weakly connected component of v, according to $T_v^{(i)}$. We first prove a one-to-one correspondence between the mega-steps of $\mathcal{A}$ and $\mathcal{A}'$.

Lemma 6. *The gadget construction from Definition 2 ensures that after i mega-steps, $V(G^{(i)}) = V(G^{(i)'})$, $E(G^{(i)}) = E(G^{(i)'})$, and $\mathrm{dist}_{G^{(i)}} \geq (1+\varepsilon)\mathrm{dist}_{G^{(i)'}}$. Moreover, each contraction uses the same value of β_v in $\mathcal{A}$ and $\mathcal{A}'$.*

Proof. We prove the claim by induction, where the base case of $i = 0$ is trivial. For the step from $i-1$ to i, we first note that (i) changing $U^{(i-1)}$ to $U^{(i)}$ does not affect which balls are contracted during the first $i-1$ mega-steps and (ii) after $i-1$ megasteps, results in the graph created from $G^{(i-1)}$ by applying the gadget construction and adding for each $r \neq v \in V(G^{(i-1)})$ of indegree 0 and each $u \in B_{T_v^{(i)}}(v, \beta_v)$ an edge (u, v) of weight β_v. This holds true, because the edges that are added due to the gadget construction were not inside any previously contracted balls, and the values $S_x^{(i)}$ are chosen precisely such that they account for any weight loss of the new edges during the first $i-1$ mega-steps.

Hence, denote the graph resulting from applying the gadget construction and adding the above edges to $G^{(i-1)}$ by $\hat{G}$, and fix some $r \neq v \in V(G^{(i-1)})$ of indegree 0. Observe first that $B_{T_v^{(i)}}(v, \beta_v) \subseteq B_{\hat{G}}(v, \beta_v)$, as each $u \in B_{T_v^{(i)}}(v, \beta_v)$ has an edge (u, v) of weight β_v. In particular, this includes the endpoint of the path of length β_v giving rise to the contraction performed by $\mathcal{A}'$ on $G^{(i-1)'}$, implying that $\mathcal{A}$ will contract a ball of radius at most β_v around v. Moreover, for any dummy node s that has been introduced when replacing an edge (u, u') with $u' \in B_{T_v^{(i)}}(v, \beta_v)$, there is an edge (s, v) of weight β_v. Hence $s \in B_{\hat{G}}(v, \beta_v)$.

By the induction hypothesis, it holds that $\mathrm{dist}_{G^{(i-1)}} \geq (1+\varepsilon)\mathrm{dist}_{G^{(i-1)'}}$, yielding $B_{G^{(i-1)}}(v, \beta_v) \leq B_{G^{(i-1)'}}(v, \beta_v/(1+\varepsilon))$. Due to the approximation gurantee of the SSSP algorithm, there can be no path shorter than $\beta_v/(1+\varepsilon)$ reaching v from outside its weakly connected component. By Lemma 5, each node outside $B_{G^{(i-1)}}(v, \beta_v)$ satisfies that its distance to v is not changed by the gadget construction. Thus, $\mathcal{A}$ must contract exactly $B_{G^{(i-1)}}(v, \beta_v) = B_{\hat{G}}(v, \beta_v)$. Moreover, any edge resulting from (u, u') and the contraction of $B_{G^{(i-1)}}(v, \beta_v) \ni u'$ will satisfy that its weight is at least $\mathrm{dist}_{G^{(i-1)}}(u, v) - \beta_v$, implying that distances in the graph after contraction are at least as large as if we performed the contractions in $G^{(i-1)}$. Hence, $\mathrm{dist}_{G^{(i)}} \geq (1+\varepsilon)\mathrm{dist}_{G^{(i)'}}$ follows from the facts that $\mathrm{dist}_{G^{(i-1)}} \geq (1+\varepsilon)\mathrm{dist}_{G^{(i-1)'}} \geq \mathrm{dist}_{T_v^{(i)}}$ and that $\mathcal{A}'$ setting a negative edge weight to 0 can only happen for edges at the boundary of the contracted balls, which in $\mathcal{A}$ are assigned a positive weight. □

This establishes the desired relation between the weight of the trees constructed by $\mathcal{A}$ and $\mathcal{A}'$, respectively. Denote by $i_{\max}$ the number of mega-steps $\mathcal{A}'$ performs on G.

Corollary 6. *$\mathcal{A}'$ on G constructs a DST of the same weight as $\mathcal{A}$ on $U^{(i_{\max})}$.*

Proof. By Lemma 6, $\mathcal{A}$ on $U^{(i_{\max})}$ and $\mathcal{A}'$ on G perform the same sequence of contractions with the same ball radii. By inductive application of Corollary 4, they hence compute DSTs of the same weight. □

Hence, it remains to show that the gadget replacements do not increase the weight of a tree spanning all (non-dummy) nodes.

Lemma 7. *For all i,* $\operatorname{opt}(U^{(i)}) \leq (1+\varepsilon)\operatorname{opt}(G)$.

Proof. As $U^{(0)} = (V, E, (1+\varepsilon)w_G)$, we have that $\operatorname{opt}(U^{(0)}) = (1+\varepsilon)\operatorname{opt}(G)$. Thus, it is sufficient to show that $\operatorname{opt}(U^{(i)}) \leq \operatorname{opt}(U^{(i-1)})$ for all $i > 0$. We show first that in the graph $G^{(i-1)}$, the modifications by introducing the gadgets do not increase the weight of the MDST. To see this, consider an MDST T of $G^{(i-1)}$. If a gadget removes an edge (u, u') of T, we replace it by the edges (u, s) and (s, u') of the corresponding gadget. By Lemma 5, distances in $G^{(i-1)}$ are well-defined. Hence, we can apply the triangle inequality to see that the combined weight of these edges satisfies

$$\begin{aligned} w_{G^{(i-1)}}(u,s) + w_{G^{(i-1)}}(s,u') &= \operatorname{dist}_{G^{(i-1)}}(u,v) - \operatorname{dist}_{G^{(i-1)}}(u',v) \\ &\leq \operatorname{dist}_{G^{(i-1)}}(u,u') \leq w_{G^{(i-1)}}(u,u'). \end{aligned}$$

Thus, we obtain a tree T' of weight at most $w_{G^{(i-1)}}(T)$ spanning all but possibly some of the dummy nodes introduced by the gadgets. Recalling that we are not required to span dummy nodes, denoting by $\hat{G}$ the graph after gadget replacement we conclude that $\operatorname{opt}(\hat{G}) \leq \operatorname{opt}(G^{(i-1)})$.

To complete the proof, we invoke Lemma 6, showing that $\mathcal{A}$ performs the same sequence of contractions with the same ball radii in both $U^{(i-1)}$ and $U^{(i)}$. By inductively applying Corollary 4, we conclude that

$$\operatorname{opt}(U^{(i-1)}) - \operatorname{opt}(U^{(i)}) = \operatorname{opt}(\hat{G}) - \operatorname{opt}(G^{(i-1)}) \geq 0,$$

i.e., $\operatorname{opt}(U^{(i)}) \leq \operatorname{opt}(U^{(i-1)})$. □

By Lemma 4, the DST computed by $\mathcal{A}$ is actually an MDST of $U^{(i_{\max})}$. Hence, by Lemma 7, its weight is $\operatorname{opt}(U^{(i_{\max})}) \leq (1+\varepsilon)\operatorname{opt}(G)$. Thus, this completes the proof of Theorem 3.

Bibliography

[1] Becker, R., Karrenbauer, A., Krinninger, S., Lenzen, C.: Near-Optimal Approximate Shortest Paths and Transshipment in Distributed and Streaming Models. In: Richa, A.W. (ed.) Symposium on Distributed Computing (DISC). pp. 7:1–7:16 (2017)

[2] Bock, F.: An algorithm to construct a minimum directed spanning tree in a directed network. Developments in operations research pp. 29–44 (1971)

[3] Censor-Hillel, K., Kaski, P., Korhonen, J.H., Lenzen, C., Paz, A., Suomela, J.: Algebraic methods in the congested clique. Distributed Computing **32**(6), 461–478 (2019)

[4] Chechik, S., Mukhtar, D.: Single-Source Shortest Paths in the CONGEST Model with Improved Bound. In: Emek, Y., Cachin, C. (eds.) Symposium on Principles of Distributed Computing (PODC). pp. 464–473. ACM (2020)

[5] Chu, Y., Liu, T.: On the shortest arborescence of a directed graph. Scientia Sinica **14**, 1396–1400 (1965)

[6] Drucker, A., Kuhn, F., Oshman, R.: On the Power of the Congested Clique Model. In: Halldórsson, M.M., Dolev, S. (eds.) Symposium on Principles of Distributed Computing (PODC). pp. 367–376. ACM (2014)

[7] Edmonds, J.: Optimum branchings. Journal of Research of the national Bureau of Standards B **71**(4), 233–240 (1967)

[8] Elkin, M.: An unconditional lower bound on the time-approximation trade-off for the distributed minimum spanning tree problem. SIAM Journal on Computing **36**(2), 433–456 (2006)

[9] Fischer, O., Oshman, R.: A distributed algorithm for directed minimum-weight spanning tree. In: DISC (2019)

[10] Gabow, H.N., Galil, Z., Spencer, T., Tarjan, R.E.: Efficient algorithms for finding minimum spanning trees in undirected and directed graphs. Combinatorica **6**(2), 109–122 (1986)

[11] Gall, F.L.: Powers of Tensors and Fast Matrix Multiplication. In: Nabeshima, K., Nagasaka, K., Winkler, F., Szántó, Á. (eds.) Symposium on Symbolic and Algebraic Computation (ISSAC). pp. 296–303 (2014)

[12] Ghaffari, M., Parter, M.: MST in log-star rounds of congested clique. In: Proceedings of the 2016 ACM Symposium on Principles of Distributed Computing. p. 19–28. PODC '16 (2016)

[13] Hegeman, J.W., Pandurangan, G., Pemmaraju, S.V., Sardeshmukh, V.B., Scquizzato, M.: Toward optimal bounds in the congested clique: Graph connectivity and MST. In: Proceedings of the 2015 ACM Symposium on Principles of Distributed Computing. p. 91–100. PODC '15 (2015)

[14] Jurdziński, T., Nowicki, K.: MST in $O(1)$ rounds of congested clique. In: Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms. p. 2620–2632. SODA '18 (2018)

[15] Kutten, S., Peleg, D.: Fast Distributed Construction of Small k-Dominating Sets and Applications. J. Algorithms **28**(1), 40–66 (1998)

[16] Lotker, Z., Pavlov, E., Patt-Shamir, B., Peleg, D.: MST construction in $O(\log \log n)$ communication rounds. p. 94–100. SPAA '03 (2003)

[17] Lovasz, L.: Computing ears and branchings in parallel. In: 2013 IEEE 54th Annual Symposium on Foundations of Computer Science. pp. 464–467 (1985)

[18] Nowicki, K.: A Deterministic Algorithm for the MST Problem in Constant Rounds of Congested Clique. CoRR **abs/1912.04239** (2019)

[19] Pandurangan, G., Robinson, P., Scquizzato, M.: The Distributed Minimum Spanning Tree Problem. Bull. EATCS **125** (2018)

[20] Peleg, D., Rubinovich, V.: A near-tight lower bound on the time complexity of distributed minimum-weight spanning tree construction. SIAM Journal on Computing **30**(5), 1427–1442 (2000)

[21] Sarma, A.D., Holzer, S., Kor, L., Korman, A., Nanongkai, D., Pandurangan, G., Peleg, D., Wattenhofer, R.: Distributed verification and hardness of distributed approximation. SIAM Journal on Computing pp. 1235–1265 (2012)