Computing Replacement Paths in the CONGEST Model

Vignesh Manoharan(✉) and Vijaya Ramachandran

The University of Texas at Austin, Austin, TX, USA
{vigneshm,vlr}@cs.utexas.edu

Abstract. We present several results on the round complexity of Replacement Paths and Second Simple Shortest Path which are basic graph problems that can address fault tolerance in distributed networks. These are well-studied in the sequential setting, and have algorithms [18, 20, 30, 34] that nearly match their fine-grained complexity [3, 33]. But very little is known about either problem in the distributed setting.

We present algorithms and lower bounds for these problems in the CONGEST model, with many of our results being close to optimal.

Keywords: Distributed Algorithms · Graph Algorithms · Shortest Paths

1 Introduction

Consider a communication network $G = (V, E)$, with two special nodes s and t in V, and with communication transmission from s to t along a shortest path P_{st}. In the distributed setting, it can be important to maintain efficient communication from s to t in the event that a link (i.e., edge) on this path P_{st} fails. This is the Replacement Paths (RPaths) problem, where for each edge e on P_{st}, we need to find a shortest path from s to t that avoids e. The RPaths problem has been extensively studied in the sequential setting [18, 20, 30]. The closely related problem of finding a second simple shortest path (2-SiSP), i.e., a shortest simple path from s to t avoids at least one edge on the original shortest path P_{st}, has also been studied in the sequential setting [34]. Surprisingly, there are virtually no results known in the distributed CONGEST model for RPaths or 2-SiSP. In this paper we address this lacuna by obtaining strong round complexity bounds for these problems in the CONGEST model, which are optimal or near-optimal in many cases.

Let $|V| = n$, $|E| = m$. Directed weighted RPaths and 2-SiSP are important problems in sequential fine-grained complexity, being part of the n^3 time complexity class [33] which contains many graph problems: weighted APSP, Negative Triangle Detection, Minimum Weight Cycle (MWC), Radius, Eccentricities and Betweenness Centrality [2]. RPaths and 2-SiSP are also in the fine-grained mn complexity class [3] which is of more relevance in the CONGEST model where

This work was supported in part by NSF grant CCF-2008241.

Y. Emek (Ed.): SIROCCO 2024, LNCS 14662, pp. 420–437, 2024.
https://doi.org/10.1007/978-3-031-60603-8_23

$O(\log n)$ bits of communication occur per round per edge in the graph, leading to $\tilde{O}(m)$[1] communication per round, and the goal is to minimize the number of rounds in the computation.

Despite RPaths and 2-SiSP being well-motivated problems for distributed networks, there is a lack of results in the CONGEST model (see Sect. 1.3 for prior work). In this paper we present several nontrivial results, in many cases near-optimal, for distributed RPaths and 2-SiSP. In addition to algorithms computing weights, we present algorithms that construct paths by computing routing tables in the full version [22].

Although RPaths and 2-SiSP have not been extensively studied in the CONGEST model, other related and more recently defined problems have been studied: CONGEST algorithms for fault-tolerant distance preservers [8,15], which are sparse subgraphs on the network in which replacement path distances are exactly preserved, and fault-tolerant spanners [13,26], which are sparse subgraphs in which replacement path distances are approximations of the distances in the original network have been reported (see Sect. 1.3 for details). The techniques in these results do not readily give efficient algorithms to explicitly compute replacement path weights or to construct a replacement path when a failed edge in known; further, these results mainly deal with undirected unweighted graphs.

Weighted directed RPaths and 2-SiSP are in the mn and n^3 sequential fine-grained complexity classes as discussed above. But these two problems are unique among the problems in these sequential classes in that they become simpler for undirected graphs [18] and for unweighted directed graphs [30]. The results we present in this paper for RPaths and 2-SiSP for the CONGEST model show a similar trend, with the added contribution of an unconditional near linear lower bound for the weighted directed case along with sublinear round algorithms for unweighted and undirected graphs (as long as the network diameter and length of P_{st} are sublinear). Thus we show a proven separation in complexity for RPaths and 2-SiSP between weighted directed graphs and either unweighted directed graphs or undirected graphs (even if weighted) in the CONGEST model. Our results for undirected graphs are in the full version [22].

1.1 Preliminaries

The CONGEST Model. In the CONGEST model [27], a communication network is represented by a graph $G = (V, E)$ where nodes model processors and edges model bounded-bandwidth communication links between processors. Each node has a unique identifier in $\{0, 1, \ldots n-1\}$ where $n = |V|$, and each node only knows the identifiers of itself and its neighbors in the network. Each node has infinite computational power. The nodes perform computation in synchronous rounds, where each node can send a message of up to $\Theta(\log n)$ bits to each neighbor and can receive the messages sent to it by its neighbors. The complexity of an algorithm is measured by the number of rounds until the algorithm terminates.

[1] We use the notation $\tilde{O}, \tilde{\Omega}, \tilde{\Theta}$ to hide poly-logarithmic factors.

We consider both weighted and unweighted graphs G in this paper, where in the weighted case each edge has an integer weight which is known to the vertices incident to the edge. The graph G can be directed or undirected. Following the convention for CONGEST algorithms [4,5,12,16,24], the communication links are always bi-directional and unweighted.

Notation. Let $G = (V, E)$ be a directed or undirected graph with $|V| = n$ and $|E| = m$. Let edge (u, v) have non-negative integer weight $w(u, v)$ according to a weight assignment function $w : E \rightarrow \{0, 1, \ldots W\}$, where $W = poly(n)$. Let $d(s, t)$ denote the weight of a shortest path P_{st} from s to t and h_{st} denote the number of edges (hop distance) on this shortest path. The undirected diameter D is the maximum shortest path distance between any two vertices in the underlying undirected unweighted graph of G.

Note 1. We use $SSSP$ and $APSP$ to denote the round complexity in the CONGEST model for weighted single source shortest paths (SSSP) and weighted all pairs shortest paths (APSP) respectively. The current best algorithm for weighted APSP runs in $\tilde{O}(n)$ rounds, randomized [7]. For weighted SSSP, recent results [9,31] provide an $\tilde{O}(\sqrt{n} + n^{2/5+o(1)}D^{2/5} + D)$ round randomized algorithm. The current best lower bounds are $\Omega\left(\frac{\sqrt{n}}{\log n} + D\right)$ for weighted SSSP [28] and $\Omega\left(\frac{n}{\log n}\right)$ for (weighted and unweighted) APSP [24].

We now define the two problems we consider in this paper.

Definition 1. ***Replacement Paths (RPaths)*** *: Given an n-node graph G, two vertices s, t and a shortest path P_{st} from s to t, for each edge $e \in P_{st}$, compute the weight $d(s, t, e)$ of a shortest simple path P_e from s to t that does not contain e.*

Second Simple Shortest Path (2-SiSP) *: Given an n-node graph G, two vertices s, t and a shortest path P_{st} from s to t, compute the weight $d_2(s, t)$ of a shortest simple path P_2 from s to t that differs from P_{st}.*

Our lower bounds for RPaths and 2-SiSP apply even when only one node in the graph is required to know each distance $d(s, t, e)$ or $d_2(s, t)$. In our algorithms, all vertices can learn the distances $d(s, t, e)$ or $d_2(s, t)$ using a simple broadcast in $O(h_{st} + D)$ rounds, which is within the round complexity bounds.

In our results, we assume that the shortest path P_{st} between the vertices s, t is part of the input and that each vertex in the network knows the identities of s and t, and the identities of vertices on P_{st}. The round bounds of our algorithms are unchanged if we are required to compute P_{st} using known CONGEST algorithms for SSSP and broadcast the necessary information in $O(h_{st} + D)$ rounds. Also, note that once we have the weights of the h_{st} replacement paths for P_{st} we can compute the 2-SiSP weight in additional $O(D)$ rounds with a convergecast.

1.2 Our Results

Table 1 lists our results for directed RPaths and 2-SiSP. Our upper bounds are either exact or $(1+\epsilon)$-approximation results, for arbitrarily small constant $\epsilon > 0$,

Table 1. Our results. *SSSP* and *APSP* refer to round complexity of weighted SSSP and APSP (See Note 1). Approximation results hold for approx. ratio α and $(1+\epsilon)$, where $\alpha > 1$ is an arbitrarily large constant, and $\epsilon > 0$ is an arbitrarily small constant.

Problem	Lower Bound	Upper Bound
Results are for directed graphs. Entries are (*approx. ratio, round bound*)		
Weighted RPaths, 2-SiSP	$1, \Omega\left(\frac{n}{\log n}\right)$ [Theorem 1.A]	$1, O(APSP) = \tilde{O}(n)$ [Theorem 1.B]
Approximate Weighted RPaths, 2-SiSP	$\alpha, \Omega(SSSP) = \Omega\left(\frac{\sqrt{n}}{\log n} + D\right)$ [Theorem 2.A]	$(1+\epsilon), \tilde{O}(\sqrt{nh_{st}} + D + \min(n^{2/3}, h_{st}^{2/5} n^{2/5+o(1)} D^{2/5}))$ [Theorem 1.C]
Unweighted RPaths, 2-SiSP	$\alpha, \Omega(SSSP)$ [Theorem 2.A]	$1, \tilde{O}(\min(n^{2/3} + \sqrt{nh_{st}} + D, SSSP \cdot h_{st}))$ [Theorem 2.B]

and our lower bounds are either exact or α-approximation results (for any constant $\alpha > 1$). Our main algorithms and bounds are for computing the *weights* of paths defined in Definition 1. However, we also have distributed algorithms that use routing tables to find such a path when an edge fails in the full version [22].

Directed Weighted Graphs. For an n-node directed weighted graph, we present an RPaths CONGEST algorithm that runs in near-linear $\tilde{O}(n)$ rounds (Sect. 2.2). The classic sequential $\tilde{O}(mn)$-time algorithm for 2-SiSP and RPaths [34] performs a sequence of h_{st} SSSP computations, and a near-linear bound is not achievable on the CONGEST model through implementing this algorithm. Instead, we formulate RPaths as an APSP computation (on an alternate graph) that can be efficiently computed within the APSP bound $\tilde{O}(n)$ in the CONGEST model. We show that our algorithm is nearly optimal by presenting an $\tilde{\Omega}(n)$ lower bound (even when the undirected diameter D is a constant) for both RPaths and 2-SiSP through a reduction from set disjointness (Sect. 2.1).

Our lower bound proof for RPaths is much more involved than the $\tilde{\Omega}(n)$ APSP lower bound in [24], and we also show that RPaths differs from APSP in efficient approximability: the $\tilde{\Omega}(n)$ APSP lower bound in [24] applies to α-approximation for any constant $\alpha > 1$, but for weighted directed RPaths we give in Sect. 3.3 an asymptotically improved algorithm for $(1+\epsilon)$-approximation (for any constant $\epsilon > 0$) that runs in sublinear rounds ($\tilde{O}(n^{1-c})$ for constant $c > 0$) whenever both h_{st} and D are sublinear.

Theorem 1. *Given a directed weighted graph G on n vertices with undirected diameter D and a shortest path P_{st} of hop length h_{st},*

A. Any randomized algorithm that computes RPaths or 2-SiSP for P_{st} requires $\Omega\left(\frac{n}{\log n}\right)$ rounds, even if D is constant.

B. RPaths and 2-SiSP for P_{st} can be computed in $O(APSP)$ rounds, and hence by a randomized algorithm in $\tilde{O}(n)$ rounds.

C. *There is a randomized algorithm that computes a* $(1+\epsilon)$*-approximation of RPaths and 2-SiSP in* $\tilde{O}(\sqrt{nh_{st}}+D+\min(n^{2/3}, h_{st}^{2/5}n^{2/5+o(1)}D^{2/5}))$ *rounds, for any constant* $\epsilon > 0$.
D. *Computing an* α*-approximation of RPaths or 2-SiSP for any constant* $\alpha > 1$ *requires* $\Omega\left(\frac{\sqrt{n}}{\log n} + D\right)$ *rounds.*

Directed Unweighted Graphs. In the case of directed unweighted graphs, the near linear lower bound for the weighted case no longer applies. We give an algorithm based on sampling and computing detours that takes $\tilde{O}(n^{2/3}+\sqrt{nh_{st}}+D)$ rounds. This gives us an algorithm that runs in sublinear rounds whenever both h_{st} and D are sublinear. We also have a simple algorithm taking $O(h_{st} \cdot SSSP)$ rounds that is more efficient when h_{st} is small (Sect. 3.2). We show a lower bound of $\Omega(SSSP) = \tilde{\Omega}(\sqrt{n}+D)$ for computing RPaths and 2-SiSP on directed unweighted graphs (Sect. 3.1), and our algorithm matches this $O(SSSP)$ bound when h_{st} is $O(1)$. This $\tilde{\Omega}(\sqrt{n}+D)$ lower bound shows that computing RPaths is harder in directed unweighted graphs than in undirected unweighted graphs, where we have an $O(D)$ round algorithm (see below).

Theorem 2. *Given a directed unweighted graph* G *on* n *vertices with undirected diameter* D *and a shortest path* P_{st} *of hop length* h_{st}*,*

A. *Any randomized algorithm that computes RPaths or 2-SiSP requires* $\Omega\left(\frac{\sqrt{n}}{\log n} + D\right)$ *rounds, even if* h_{st} *and* D *are as small as* $\Theta(\log n)$*. These lower bounds also apply to an* α*-approximation, for any constant* $\alpha > 1$.
B. *There is a randomized algorithm that computes RPaths and 2-SiSP for* P_{st} *in* $\tilde{O}(\min(n^{2/3}+\sqrt{nh_{st}}+D, SSSP \cdot h_{st}))$ *rounds.*

We adapt our lower bound proof for directed unweighted RPaths to other basic directed graph problems such as s-t reachability and s-t shortest path in directed unweighted graphs. A folklore lower bound of $\tilde{\Omega}(\sqrt{n}+D)$ for these problems was attributed by [16] to *undirected* lower bound results in [32], and we give explicit proofs here. A lower bound of $\tilde{\Omega}(\sqrt{n}+D)$ for undirected weighted SSSP was known [14,32] (which also applies to directed weighted SSSP) but this does not apply to unweighted directed graphs. These problems are easier in undirected unweighted graphs since undirected BFS can be performed in $O(D)$ rounds. Thus our results indicate that basic problems in directed graphs are asymptotically harder than their undirected unweighted counterparts. We note that this difference is not very surprising since the underlying communication network is the undirected version of the graph regardless of whether the graph is directed or undirected.

Undirected Graphs. For undirected graphs, our upper and lower bounds for RPaths match the round complexity of SSSP (BFS for unweighted) in the CONGEST model, except for weighted RPaths which requires an additional $O(h_{st})$ rounds. The remaining gap between our upper and lower bounds is inherited from the gap between the current best bounds for SSSP. Due to space limitation, details of our undirected graph results are deferred to the full version [22].

Theorem 3. *Given an undirected weighted (or unweighted) graph G on n vertices with diameter D and a shortest path P_{st} of hop length h_{st},*

A. Any algorithm that computes RPaths or 2-SiSP requires:
   i. $\Omega(SSSP) = \Omega\left(\frac{\sqrt{n}}{\log n} + D\right)$ rounds if G is weighted, even if h_{st} is constant. This lower bound applies to α-approximation, for any $\alpha > 1$.
   ii. $\Omega(D)$ rounds if G is unweighted, which is a tight bound.

B. We can compute RPaths for P_{st} in $O(SSSP+h_{st}) = \tilde{O}(\sqrt{n}+n^{2/5+o(1)}D^{2/5}+D+h_{st})$ rounds. For 2-SiSP the bound is $O(SSSP)$. If G is unweighted, the bound is $O(D)$ rounds.

.

1.3 Prior Work

RPaths and the closely related 2-SiSP are well studied problems in the sequential setting. For weighted directed graphs the classical algorithm of Yen [34] runs in $\tilde{O}(mn)$ time and has a matching fine-grained lower bound of $\tilde{\Omega}(mn)$ assuming a sequential hardness result for MWC [3]. For unweighted directed graphs, a randomized $\tilde{O}(m\sqrt{n})$ algorithm is given in [30]. For undirected graphs, a near-linear time algorithm is given in [18], matching the running time for sequential SSSP. Our bounds for RPaths for these different graph classes in the CONGEST model follow a similar pattern: close to APSP for weighted directed graphs, close to SSSP for undirected graphs, and intermediate bounds for directed unweighted graphs. The more general problem of single source replacement paths (SSRP) was studied in the sequential setting in [10,11].

In the distributed setting, an $O(D \log n)$ algorithm for single source replacement paths in undirected unweighted graphs was given in [15]. We are not aware of any prior results in the CONGEST model for RPaths or 2-SiSP for directed graphs or for weighted graphs. Distributed constructions of fault-tolerant preservers, which construct a sparse subgraph that exactly preserves replacement path distances, have been studied in [8,25] but their constructions do not give an efficient procedure to compute replacement path distances. Fault-tolerant spanners, which construct a sparse subgraph that approximates replacement path distances, have also been studied in CONGEST [13,26].

There are unconditional $\tilde{\Omega}(n)$ CONGEST lower bounds for several graph problems in the sequential n^3 and mn fine-grained complexity class: APSP [24], diameter, radius [1,5], minimum weight cycle (MWC) [23]. CONGEST algorithms for these problems have also been studied: APSP [4,7,21], diameter and radius [1,5], MWC [23] and betweenness centrality [17].

1.4 Our Techniques

CONGEST Lower Bounds. Our lower bounds use reductions either from Set Disjointness or from other graph problems with known CONGEST lower bounds. Set Disjointness is a two party communication complexity problem, where two

players Alice and Bob are each given a k-bit string S_a and S_b respectively. Alice and Bob need to communicate and decide if the sets represented by S_a and S_b are disjoint, i.e., whether there is no bit position i, $1 \leq i \leq k$ with $S_a[i] = 1$ and $S_b[i] = 1$. A classical result in communication complexity is that Alice and Bob must exchange $\Omega(k)$ bits even if they are allowed shared randomness [6,19,29]. Lower bounds using such a reduction also hold against randomized algorithms.

Our reduction from Set Disjointness for the $\tilde{\Omega}(n)$ CONGEST lower bound for directed weighted RPaths is loosely inspired by a construction in a sparse sequential reduction from MWC to RPaths in [3]. Some of our lower bounds use known unconditional CONGEST lower bounds for problems like s-t Subgraph Connectivity and weighted s-t Shortest Path [32].

CONGEST Algorithms. Adapting the sequential algorithm for directed weighted replacement paths [34] directly to the CONGEST model requires up to n SSSP computations, which is not efficient. Instead, our CONGEST algorithm builds on a sequential sparse reduction from RPaths to Eccentricities in [3] which we tailor to work efficiently in the CONGEST model using weighted APSP [7] as a subroutine. Our algorithm for directed unweighted RPaths is loosely based on the sequential algorithm in [30], but we make significant changes to obtain efficiency in the distributed setting. We use a variety of techniques such as sampling, computing shortest paths in skeleton graphs and pipelined BFS.

Many of our algorithms do not use any randomness apart from that used by the randomized algorithms for SSSP and APSP. If we use deterministic CONGEST algorithms for these problems, such as for unweighted APSP [21] and weighted APSP [4], our algorithms will be deterministic as well. The exceptions to this are our directed unweighted RPaths algorithm and approximate weighted directed RPaths, both of which inherently use random sampling.

2 Directed Weighted Replacement Paths

In this section, we prove near-linear unconditional CONGEST lower bounds for directed weighted RPaths and 2-SiSP (Sect. 2.1). We complement this result with an algorithm that runs in a near-linear number of rounds (Sect. 2.2).

2.1 Directed Weighted RPaths Lower Bound

We prove Theorem 1.A by showing an unconditional $\Omega\left(\frac{n}{\log n}\right)$ lower bound for computing 2-SiSP (which extends to RPaths) in directed weighted graphs. Our proof is based on a new reduction from set disjointness pictured in Fig. 1, partly inspired by a sparse sequential reduction from MWC to RPaths in [3].

Consider an instance of the Set Disjointness problem where the players Alice and Bob are given k^2-bit strings S_a and S_b respectively representing sets of at most k^2 elements (the i'th bit being 1 indicates element i belongs to the set). The problem is to determine whether $S_a \cap S_b = \phi$, i.e., that for all indices $1 \leq i \leq k^2$ either $S_a[i] = 0$ or $S_b[i] = 0$. A classical result in communication complexity

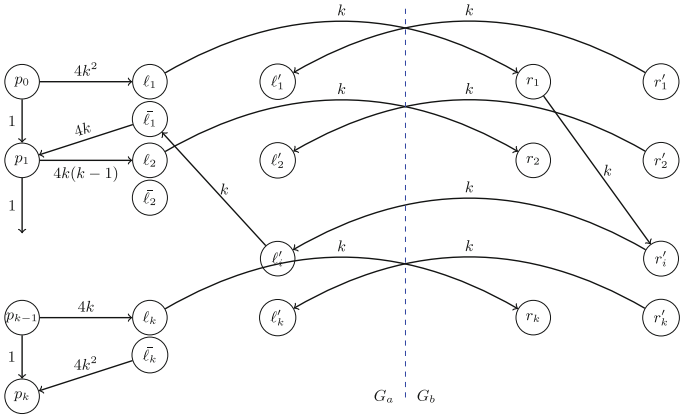

Fig. 1. Directed weighted RPaths, 2-SiSP lower bound construction

is that Alice and Bob must exchange $\Omega(k)$ bits even if they are allowed shared randomness [6,19,29]. Our reduction constructs the graph $G = (V, E)$ described below and we show in Lemma 1 that G has a low-weight 2-SiSP if and only if the sets S_a and S_b are not disjoint.

We will construct $G = (V, E)$ with six sets of vertices (see Fig. 1): $L = \{\ell_i \mid 1 \leq i \leq k\}, L' = \{\ell'_i \mid 1 \leq i \leq k\}, R = \{r_i \mid 1 \leq i \leq k\}, R' = \{r'_i \mid 1 \leq i \leq k\}$, $\overline{L} = \{\overline{\ell_i} \mid 1 \leq i \leq k\}$, $P = \{p_i \mid 0 \leq i \leq k\}$. Note that the number of vertices is $n = 6k + 1$. We set $s = p_0$ and $t = p_k$ and for each $1 \leq i \leq k$, we add the edges (p_{i-1}, p_i) with weight 1; this is the input shortest path $\mathcal{P} = P_{st}$. We add directed edges (ℓ_i, r_i) and (r'_i, ℓ'_i) for each $1 \leq i \leq k$. Each of these edges has weight k. For each $1 \leq i \leq k$, we add the edges (p_{i-1}, ℓ_i) with weight $4k(k - i + 1)$ and $(\overline{\ell}_i, p_i)$ with weight $4ki$. This is our base graph.

We now add edges to G based on the set disjointness inputs S_a,S_b. We encode each integer q, $1 \leq q \leq k^2$, as an ordered pair (i, j) such that $q = (i-1) \cdot k + j$. If $S_a[q] = 1$, we add the edge $(\ell'_j, \overline{\ell_i})$ with weight k, if $S_b[q] = 1$, we add the edge (r_i, r'_j) with weight k. For the 2-SiSP problem, the desired output is $d_2(p_0, p_k)$, the weight of a second simple shortest path from p_0 to p_k.

Lemma 1. *If $S_a \cap S_b \neq \emptyset$, then $d_2(p_0, p_k) \leq (4k^2 + 9k - 1)$. Otherwise, if $S_a \cap S_b = \emptyset$, then $d_2(p_0, p_k) \geq (4k^2 + 12k)$.*

Proof. If the sets S_a, S_b are not disjoint, then there exists $1 \leq i, j \leq k$ such that $S_a[(i-1) \cdot k + j] = S_b[(i-1) \cdot k + j] = 1$. Then, the path $\langle p_{i-1}, \ell_i, r_i, r'_j, \ell'_i, \overline{\ell}_i, p_i \rangle$ provides a detour for the edge (p_{i-1}, p_i). This can be used along with the shortest paths from p_0 to p_{i-1} and p_i to p_k to obtain a simple path of weight $4k(k+1) + 4k + k - 1$ that does not use edge (p_{i-1}, p_i). So, the second simple shortest path from p_0 to p_k has weight at most $4k^2 + 9k - 1$.

Assume the strings are disjoint. Let $\mathcal{P}_2$ be a second simple shortest path, and let (p_{i-1}, p_i) be the first edge that is not in $\mathcal{P}_2$ but is in the p_0-p_k shortest path $\mathcal{P}$. Since the only other outgoing edge from p_{i-1} is (p_{i-1}, ℓ_i) (with weight

$4k(k-i+1)$), this edge must be on $\mathcal{P}_2$. Let p_j $(j \geq i)$ be the next vertex from $\mathcal{P}$ that is also on path $\mathcal{P}_2$, such a vertex must exist as p_k is on $\mathcal{P}$ and $\mathcal{P}_2$. By the construction of G, edge $(\overline{\ell}_j, p_j)$ (with weight $4kj$) must be in $\mathcal{P}_2$ which means the path $\mathcal{P}_2$ has weight at least $4k(k-i+1)+4kj$ not including edges in the path from ℓ_i to $\overline{\ell}_j$. We also observe that any path from ℓ_i to $\overline{\ell}_j$ requires at least 4 edges, with total weight $4k$. If we have $j > i$, we immediately conclude that $\mathcal{P}_2$ has weight at least $4k(k-i+1+j)+4k \geq 4k(k+1)+8k$. If $j = i$, then $\mathcal{P}_2$ contains a path from ℓ_i to $\overline{\ell}_i$. This path can have length 4 if and only if the edges (r_i, r'_j) and $(\ell'_j, \overline{\ell}_i)$ simultaneously exist for some j, which means $S_a[(i-1)\cdot k+j] = S_b[(i-1)\cdot k+j] = 1$. This contradicts the assumption that strings S_a and S_b are disjoint. So, this ℓ_i to $\overline{\ell}_i$ path has length at least 8, which means $\mathcal{P}_2$ has weight at least $4k(k+1)+8k = 4k^2+12k$. □

To complete the reduction from set disjointness, assume that there is a CONGEST algorithm $\mathcal{A}$ that takes $R(n)$ rounds to compute the weight of a 2-SiSP path in a directed weighted graph on n vertices. Consider the vertex partition V_a, V_b of V with $V_a = L \cup L' \cup \overline{L} \cup P$ and $V_b = R \cup R'$, and let $G_a(V_a, E_a), G_b(V_b, E_b)$ be the subgraphs of G induced by the vertex sets V_a, V_b respectively. Note that G_a is completely determined by S_a and G_b is completely determined by S_b. Alice and Bob will communicate to simulate $\mathcal{A}$ on G. Alice will simulate the computation done in nodes in V_a, and Bob will simulate the computation done in nodes in V_b. If the algorithm communicates from a node in V_a to a node in V_b, Alice sends all the information communicated along this edge to Bob. Since there are $2k$ cut edges, and $\mathcal{A}$ can send $O(\log n)$ bits through each edge per round, Alice and Bob communicate up to $O(2k \cdot \log n)$ bits per round, for a total of $O(2k \cdot \log n \cdot R(n))$ bits. After the simulation, Alice knows $d_2(p_0, p_k)$ and can determine if the sets are disjoint by checking if $d_2(p_0, p_k) > 4k^2+9k-1$ (Lemma 1). Since any communication protocol for set disjointness must use at least $\Omega(k^2)$ bits and $n = \Theta(k)$, $R(n)$ is $\Omega\left(\frac{n}{\log n}\right)$.

Our lower bound also applies to the RPaths problem, since given an algorithm $\mathcal{A}$ that computes replacement path for each edge, Alice can compute the minimum of those to get the second simple shortest path weight and then use Lemma 1 as before. This lower bound applies even for graphs with constant undirected diameter: we can add a 'sink' vertex with incoming edges from all vertices in G, so that Lemma 1 still holds and the undirected diameter is 2.

2.2 Directed Weighted RPaths Algorithm

In this section we present an $\tilde{O}(n)$ round CONGEST algorithm for computing RPaths and 2-SiSP in directed weighted graphs, which is nearly optimal given the near linear lower bound in Sect. 2.1. Our main tool is a reduction from RPaths to weighted APSP that can be simulated efficiently in the CONGEST network. This reduction is inspired by a sequential fine-grained reduction from RPaths to Eccentricities in [3], though some care is needed to ensure that the reduction can be efficiently mapped to the underlying CONGEST network. We present our algorithm to compute replacement path weights, and the second simple shortest

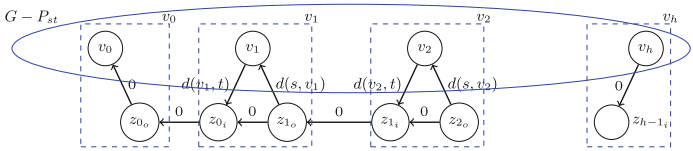

Fig. 2. Directed weighted RPaths reduction to APSP. $v_0 = s$ and $v_h = t$.

path weight $d_2(s, t)$ can be computed by taking the minimum weight replacement path among those computed, with an additional $O(D)$ rounds.

Our algorithm constructs a graph G' pictured in Fig. 2, and runs a weighted APSP algorithm on G'. We show later how communication in the newly constructed G' can be simulated efficiently in the underlying CONGEST network of G, so that the APSP algorithm can be applied to G' in $\tilde{O}(n)$ rounds. The algorithm uses the $\tilde{O}(n)$ round weighted APSP algorithm [7] as a subroutine and has $O(n)$ additive overhead, giving our $\tilde{O}(n)$ round bound.

We construct graph $G'(V', E')$ with $V' = V \cup Z_o \cup Z_i$, where $Z_o = \{z_{j_o} \mid 0 \leq j < h\}$, $Z_i = \{z_{j_i} \mid 0 \leq j < h\}$. We denote the nodes on the shortest path P_{st} by $s = v_0, v_1, \ldots v_h = t$. E' contains all edges in E with their original weights, except the edges from the given shortest path P_{st} which are removed. Additionally, E' contains directed edges $(z_{j_o}, v_j), (z_{j_i}, z_{j_o}), (v_{j+1}, z_{j_i})$ for $0 \leq j < h$. Edge (z_{j_o}, v_j) has weight $d(s, v_j)$, edge (v_{j+1}, z_{j_i}) has weight $d(v_{j+1}, t)$ and edge (z_{j_i}, z_{j_o}) has weight 0 — recall that $d(s, v_j)$ denotes the shortest path distance from s to v_j in G. We use d' to denote shortest path distances in G'. The following lemma shows that we can compute replacement paths in G using distances in G'.

Lemma 2. *The shortest path distance $d'(z_{j_o}, z_{j_i})$ in G' (Figure 2) is equal to the replacement path weight $d(s, t, (v_j, v_{j+1}))$ in the original graph G.*

Proof. Let $\mathcal{P}$ be a replacement path for the edge (v_j, v_{j+1}) with weight $d(s, t, (v_j, v_{j+1}))$. We will construct a path from z_{j_o} to z_{j_i} that has the same weight as $\mathcal{P}$. Let v_a be the first vertex where $\mathcal{P}$ deviates from P_{st}, v_b be the first vertex after v_a where $\mathcal{P}$ rejoins P_{st}. Note that $a \leq j$ and $b \geq j + 1$ as it is a replacement path for edge (v_j, v_{j+1}), and $\mathcal{P}$ contains a subpath $\mathcal{P}_{ab}$ from v_a to v_b that does not contain any edge from P_{st}. Construct the path $\langle z_{j_o}, \ldots z_{a_o}, v_a \rangle \circ \mathcal{P}_{ab} \circ \langle v_b, z_{b-1_i}, \ldots z_{j_i} \rangle$, which has weight $w(z_{a_o}, v_a) + w(\mathcal{P}_{ab}) + w(v_b, z_{b-1_i}) = d(s, a) + w(\mathcal{P}_{ab}) + d(b, t) = w(\mathcal{P})$. Thus, we have $d'(z_{j_o}, z_{j_i}) \leq d(s, t, (v_j, v_{j+1}))$.

Now, consider any shortest path P from z_{j_o} to z_{j_i}. Observe that any such P must use a unique edge of the form (z_{a_o}, v_a) and a unique edge (z_b, z_{b-1_i}) in order to reach z_{j_i} from z_{j_o}, where $a \leq j$ and $b \geq j + 1$. Denote the subpath of P from v_a to v_b as P_{ab}. Now, consider the path $\mathcal{P}$ in G obtained by concatenating s-v_a shortest path, P_{ab} and v_b-t shortest path — this is a replacement path for edge (v_j, v_{j+1}) since $a \leq j$, $b \geq j + 1$ and P_{ab} does not contain any edge on P_{st}. The weight of $\mathcal{P}$ is equal to $d(s, v_a) + w(P_{ab}) + d(v_b, t)$ which is equal to $w(P)$

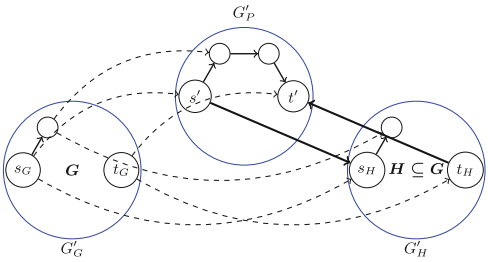

Fig. 3. Directed unweighted RPaths, 2-SiSP lower bound graph G'

since the only nonzero weight edges on P that are outside subpath P_{ab} are the ones with weight $d(s, v_a)$ and $d(v_b, t)$. Hence $d(s, t, (v_j, v_{j+1})) \leq d'(z_{j_o}, z_{j_i})$. □

To simulate an APSP algorithm on G' using the communication network G, we assign vertices v_i, z_{i-1_i}, z_{i_o} of G' to be simulated by CONGEST node v_i of G—this is represented by the dashed boxes in Fig. 2. This ensures that any edge of G' corresponds to either a communication link between nodes in the CONGEST network of G, or the edge is within the same node of G. We can compute the weights required to simulate G' after two SSSP computations with s and t as sources, and use an $\tilde{O}(n)$ algorithm to compute APSP in G' [7]. We show how to augment this algorithm to construct replacement paths using routing tables in the full version [22].

When the hop length of the s-t path h_{st} is small, the simple algorithm of performing h_{st} shortest path computations with each edge on the s-t path removed gives us an $O(h_{st} \cdot SSSP)$ round algorithm. We can obtain an improved round complexity if we only require a $(1+\epsilon)$-approximation of the replacement path weight. We defer the presentation of this approximation algorithm for directed weighted RPaths to Sect. 3.3 since it uses techniques that build on the directed unweighted RPaths algorithm.

3 Directed Unweighted Replacement Paths

In Sect. 3.1 we show a lower bound of $\tilde{\Omega}(\sqrt{n} + D)$ for computing RPaths and 2-SiSP in directed unweighted graphs, which also proves a folklore lower bound for directed single source reachability. We present an algorithm for directed unweighted RPaths in Sect. 3.2, and extend it to $(1+\epsilon)$-approximate directed weighted RPaths (Sect. 3.3).

3.1 Directed Unweighted RPaths Lower Bound

Our lower bound method uses a reduction from the undirected s-t *subgraph connectivity* problem defined in [32] as follows: Given an undirected CONGEST network G with n vertices, a subgraph H of G, and two vertices s, t, determine whether s and t are in the same connected component of H. The input subgraph

H is given by letting each vertex know which of its incident edges are in H. It is shown in [32] that this problem has a lower bound of $\Omega\left(\frac{\sqrt{n}}{\log n} + D\right)$ in graphs with D as small as $\Theta(\log n)$. We assume WLOG that network G is connected.

Proof (Proof of Theorem 3.A and Theorem 1.D). Given an instance of s-t subgraph connectivity with undirected network G, vertices s and t and subgraph H, our first attempt is to construct a directed unweighted graph G' with two copies of $V(G)$: G'_H contains only the edges in H, with bidirectional edges, and G'_P contains only a directed shortest path from s' to t' made of edges in G where s', t' are copies of s, t. These copies are connected with directed edges (s', s_H) and (t_H, t') (Fig. 3 without copy G'_G).

This construction has the property that there is a second directed path from s' to t' in G' (apart from the one in G'_P) if and only if there is an s_H-t_H path in G_H. So 2-SiSP weight in G is finite iff s,t are connected in H. But, this construction could have high undirected diameter as we have no control over the diameter of H, and fails to give a meaningful lower bound.

To obtain small undirected diameter, we add a third copy of G, denoted G'_G, which has all edges of G as bidirectional edges, pictured in Fig. 3. This copy is connected to the others with directed edges (v_G, v_H) and (v_G, v') where v_G is the copy in G'_G of $v \in G$. The undirected diameter of G' is now $(D + 2)$ (D is the diameter of G) as we can connect any pair of vertices using a bidirectional path in G'_G along with at most 2 connecting edges. This addition does not add any new directed paths from s' to t'.

Any communication in G' can be simulated in a constant number of rounds in the underlying network of G, as each node v in the network can simulate vertices v_G, v_H, v' of G', and all edges in G' are either within the same node or have an underlying undirected edge of G. Constructing G' requires only an $O(D)$-round computation of undirected shortest path from s to t in G.

This completes the reduction and establishes a lower bound of $\Omega\left(\frac{\sqrt{n}}{\log n} + D\right)$ for 2-SiSP (and RPaths) in unweighted directed graphs by additionally noting that $\Omega(D)$ rounds are necessary, as with other global problems in the distributed model [32], for information to travel to the farthest vertices to determine 2-SiSP. Our lower bound also applies to any α-approximation algorithm ($\alpha > 1$) since we distinguish between 2-SiSP of length $\leq n + 2$ and infinite length. □

Other Directed Unweighted Graph Problems. Our lower bound for directed unweighted RPaths can be adapted to give the same lower bound for other graph problems on directed unweighted graphs, including the basic problems of s-t directed reachability and s-t directed shortest path. These folklore lower bounds [16] have been attributed to [32] which only deals with undirected graphs, and we make these results explicit.

Lemma 3. *Any algorithm computing s-t directed reachability or s-t directed shortest path in a directed unweighted graph requires $\Omega\left(\frac{\sqrt{n}}{\log n} + D\right)$ rounds, even if the graph has undirected diameter as low as $\Theta(\log n)$.*

Proof. We use a simpler version of the construction in Fig. 3 by removing G'_P from the graph G' to form a graph G''. A directed path from s_H to t_H exists in G'' if and only if s and t are connected in the subgraph H. Using the same arguments as the RPaths lower bound, we note that G'' has undirected diameter $O(D)$ when the network G has undirected diameter D and G'' can be efficiently simulated on the original network G. So we get the desired reduction for both problems from s-t subgraph connectivity [32]. □

3.2 Directed Unweighted RPaths Algorithm

In the sequential setting, there are two approaches to compute replacement paths in directed graphs: (1) remove each edge in the input path P_{st} and compute shortest paths in the resulting graphs separately, using h_{st} shortest path computations [34], (2) compute shortest detour distances in order to compute replacement paths: A *detour* from a to b, where a, b are vertices on P_{st}, is a simple path from a to b with no edge in common with P_{st}. Any replacement path for edge $e \in P_{st}$ can be characterized as the concatenation of an initial s-a subpath of P_{st}, a detour from a to b, and a final b-t subpath of P_{st}, where a, b are vertices in P_{st} such that e is contained in the a-b subpath of P_{st} [30,34].

Our distributed algorithm uses both these approaches for different ranges of h_{st}, D (as in line 4 of Algorithm 1). In the first method, used in Case 1 of Algorithm 1, we compute replacement paths in $O(h_{st} \cdot SSSP)$ rounds using the obvious algorithm of removing one of the h_{st} edges on the input shortest path P_{st} and computing SSSP from s. We use a directed weighted SSSP algorithm with the weight of the removed edge set to ∞. We do not use unweighted directed BFS since an s-t shortest path could have up to $n-1$ hops after edge removal.

In the second method, used in Case 2 of Algorithm 1, we present a distributed detour-based algorithm that runs in $\tilde{O}(n^{2/3} + \sqrt{nh_{st}} + D)$ rounds.

To compute short detours (hop length $\leq h$, parameter h determined in line 4), our distributed algorithm exploits pipelining to compute h-hop limited BFS from each vertex on P_{st} in $O(h_{st} + h)$ rounds [17,21]. We compute these distances in the graph $G - P_{st}$, which is the graph G with edges on P_{st} removed. We denote shortest path distances in graph $G - P_{st}$ by $d^-(u, v)$.

For long detours (hop length $> h$), we sample $\Theta(p)$ vertices (p determined in line 4) in line 6 and compute a 'skeleton graph' on the set of sampled vertices: for $u, v \in S$, we add a directed edge (u, v) to the skeleton graph with weight $d^-(u, v)$ if there is an h-hop directed shortest path from u to v in $G - P_{st}$. The edges of the skeleton graph are computed using an h-hop BFS in line 9. The h-hop distances between all pairs of sampled vertices, and between sampled vertices and vertices on P_{st} are broadcast to all vertices in line 10. Algorithm 2, described below, is run at each vertex $a \in P_{st}$. It uses the h-hop distances (broadcast in line 10 of Algorithm 1) to *locally* compute at vertex a all detours starting from a. It then computes at a the best candidate replacement path among paths first deviating from P_{st} at a for each edge $e \in P_{st}$ that occurs after a on P_{st}, denoted $d^a(s, t, e)$. Finally, Algorithm 1 performs a pipelined minimum operation along P_{st} in line 15 to compute shortest replacement path distances for all h_{st} edges among candidate replacement paths computed by Algorithm 2 at each $a \in P_{st}$.

Algorithm 1 Directed Unweighted RPaths Algorithm

Input: Graph $G = (V, E)$, vertices $s, t \in V$, s-t shortest path P_{st}.
Output: Replacement path distance $d(s, t, e)$ known at s for each $e \in P_{st}$.

1: **Case 1.** $D \leq n^{1/4}, h_{st} \leq n^{1/6}$ or $n^{1/4} < D \leq n^{2/3}, h_{st} \leq n^{1/3}$
2: Perform h_{st} directed weighted SSSP computations in sequence, with each edge $e \in P_{st}$ having its weight set to ∞ to compute $d(s, t, e)$.
3: **Case 2.** $D \leq n^{1/4}, h_{st} > n^{1/6}$ or $n^{1/4} < D \leq n^{2/3}, h_{st} > n^{1/3}$ or $D > n^{2/3}$
4: Fix parameter $p = n^{1/3}$ if $h_{st} < n^{1/3}$ and $p = \sqrt{n/h_{st}}$ if $h_{st} \geq n^{1/3}$, and fix $h = n/p$.
5: Let graph $G - P_{st}$ be G but with all edges in P_{st} removed.
6: Sample each vertex $v \in G$ with probability $\Theta\left(\frac{\log n}{h}\right)$, let the set of sampled vertices be S.
7: **for each** vertex $v \in P_{st} \cup S$ **do**
8: ▷ *Comment: Line 9 computes h-hop $S \times V(P_{st})$ distances and h-hop $S \times S$ distances (edges of the skeleton graph on S)* ◁
9: Perform BFS starting from v on $G - P_{st}$, and the reversed graph, up to h hops to compute unweighted shortest paths: for $u \in P_{st} \cup S$, both v and u know the h-hop limited distance $d^-(v, u)$. Since we have $(|S| + h_{st})$ sources, this takes $O(|S| + h_{st} + h)$ rounds.
10: Broadcast $\{d^-(v, u) \mid u \in S \text{ or } v \in S\}$. At most $(|S|^2 + h_{st}|S|)$ values are broadcast, taking $O(|S|^2 + h_{st}|S| + D)$ rounds.
11: **for each** vertex $a \in P_{st}$ **do**
12: ▷ *Comment: Internally compute replacement paths $d^a(s, t, e)$ that deviate from P_{st} at a, using distances broadcast in line 10* ◁
13: ComputeLocalRPaths(a) ▷ *Algorithm 2*
14: **for** edge $e \in P_{st}$ **do**
15: Compute $d(s, t, e) \leftarrow \min_a d^a(s, t, e)$ by propagating values from $a \in P_{st}$ up the path P_{st}. The minimum for a single e over all $a \in P_{st}$ takes $O(h_{st})$ rounds, and the computation for all $e \in P_{st}$ can be pipelined in $O(h_{st})$ rounds.

Computation in Algorithm 2 (local computation at each $a \in P_{st}$). Algorithm 2 at vertex $a \in P_{st}$ takes as input the h-hop distances to and from a, computed in line 9 of Algorithm 1, and the h-hop distances broadcast in line 10 of Algorithm 1. In Algorithm 2, all pairs shortest path distances $d^-(u, v)$ for $u, v \in S$ in the skeleton graph are locally computed in line 3 using the h-hop skeleton graph edge distances. These distances, along with h-hop distances $d^-(u, b)$ for $u \in S, b \in P_{st}$, are used to compute long detours in line 5. Short detours are computed at a using the h-hop distance $d^-(a, b)$ to each vertex $b \in P_{st}$. With the best detour distances $\delta(a, b)$ computed in line 5, a locally computes replacement paths using detours starting from a in line 7, which gives the best candidate replacement path distance $d^a(s, t, e)$ among paths that first deviate from P_{st} at a, for each edge e after a on P_{st}.

Lemma 4. *The local computation in Algorithm 2 at $a \in P_{st}$ correctly computes $d^a(s, t, e)$, the minimum weight replacement path for $e \in P_{st}$ among paths that first deviate from P_{st} at a.*

Algorithm 2 Local computation at $a \in P_{st}$ of candidate replacement paths deviating from vertex a

Input: Graph $G = (V, E)$, shortest path P_{st}, subset $S \subseteq V$. The following distances in graph $G - P_{st}$ are known to a: h-hop distances $d^-(b, u)$, $d^-(u, b)$ for any $b \in P_{st}, u \in S$, $d^-(u, v)$, for $u, v \in S$, and $d^-(a, b)$ for $b \in P_{st}$.

Output: Vertex a computes for each $e \in P_{st}$ after a on P_{st}, the shortest replacement path distance $d^a(s, t, e)$ among paths that first deviate from P_{st} at a.

1: **procedure** ComputeLocalRPaths(a)
2: ▷ *Comment: All computation is done internally using distances known to a.* ◁
3: Locally compute all pairs distances in skeleton graph on S: compute $d^-(y, z)$ for each $y, z \in S$ using the skeleton graph h-hop edge distances.
4: **for each** vertex $b \in P_{st}$ after a along P_{st} **do**
5: Compute the best (short or long) detour $\delta(a, b)$ from a to b as $\delta(a, b) = \min\left(d^-(a, b), \min_{u,v \in S}\left(d^-(a, u) + d^-(u, v) + d^-(v, b)\right)\right)$
6: **for** edge $e = (x, y) \in P_{st}$ such that a that appears before x on P_{st} or $a = x$ **do**
7: $d^a(s, t, e) = \min_{b \in P_{st}}(d(s, a) + \delta(a, b) + d(b, t))$ (the minimum is over vertices b that appear after y on P_{st} or $b = y$)

Proof. We assume that a knows the correct h-hop distances specified as input to Algorithm 2. Note that the vertices and distances along P_{st} are known to a as part of RPaths input.

Due to our sampling probability, any shortest path between $u, v \in S$ can be decomposed into h-hop subpaths between sampled vertices w.h.p. in n. So in line 3, vertex a correctly computes all pairs shortest path distances between sampled vertices in S using h-hop skeleton graph distances.

Now we show that line 5 computes a shortest detour P^d_{ab} from a to each $b \in P_{st}$ that occurs after a on P_{st}, whose distance is denoted $\delta(a, b)$. If P^d_{ab} is a short detour, with hop length $\leq h$, its distance is equal to the h-hop distance $d^-(a, b)$ which is part of the input to a.

If P^d_{ab} is a long detour, with hop length $> h$, we use the fact that due to our sampling probability, any path of h hops contains a sampled vertex in S w.h.p. in n. We can find a sampled vertex u on the detour within h hops from a and a sampled vertex v on the detour within h hops from b. We will assume WLOG that v occurs after u or $u = v$. Then, the detour distance is $\delta(a, b) = d^-(a, u) + d^-(u, v) + d^-(v, b)$, and line 5 correctly computes this distance.

In any replacement path for edge $e \in P_{st}$ first deviating from P_{st} at a, there is a vertex $b \in P_{st}$ where it rejoins P_{st}. We can characterize such a replacement path as the concatenation of the s-a subpath of P_{st}, a detour P^d_{ab} from a to b, and the b-t subpath of P_{st}. This path has weight $d(s, a) + \delta(a, b) + d(b, t)$. We then compute the minimum over all valid detour endpoints b in line 7. This correctly computes $d^a(s, t, e)$ for edges $e = (x, y)$ that are on the a-b subpath of P_{st} □

Lemma 5. *Algorithm 1 computes replacement path weights in a directed unweighted graph in* $\tilde{O}(\min(n^{2/3} + \sqrt{nh_{st}} + D, h_{st} \cdot SSSP))$ *rounds.*

Proof. We focus on the analysis of Case 2 since Case 1 is straightforward.

Correctness: The inputs used by Algorithm 2 at vertex $a \in P_{st}$ are correctly computed in Algorithm 1: the h-hop distances from a to other vertices $b \in P_{st}$ and h-hop distances from sampled vertices are computed in line 9. After Algorithm 2 correctly computes $d^a(s,t,e)$, line 15 computes $d(s,t,e) = \min_{a \in P_{st}} d^a(s,t,e)$ for each edge e as the minimum distance among all valid replacement paths which may deviate at any $a \in P_{st}$.

Round Complexity: Recall that local computation does not contribute to the cost of an algorithm in the CONGEST model. So we can ignore Algorithm 2 for the round complexity analysis. In line 9 of Algorithm 1, we use the k-source h-hop BFS algorithm for directed graphs which runs in $O(k+h)$ rounds using pipelining [17,21]. As we have $k = p + h_{st}$ sources and h hops, this takes $O(p + h_{st} + h)$ rounds. We use the standard broadcast operation in line 10 which broadcasts $(p + h_{st}) \cdot p$ values in $O(p^2 + p \cdot h_{st} + D)$ rounds [27]. The global minimum in line 15 involves h_{st} convergecast operations which can be pipelined to take $O(h_{st} + D)$ rounds. The total round complexity is $O(p^2 + p \cdot h_{st} + h + D)$.

Setting parameters $h = n^{2/3}, p = n^{1/3}$ gives us a round complexity of $\tilde{O}(n^{2/3} + n^{1/3} h_{st} + D)$. When $h_{st} \geq n^{1/3}$, the parameters $h = \sqrt{nh_{st}}, p = \sqrt{n/h_{st}}$ are more favorable, giving a round complexity of $\tilde{O}(\sqrt{nh_{st}} + n/h_{st} + D) = \tilde{O}(\sqrt{nh_{st}} + D)$ (since $h_{st} \geq n^{1/3}$). The input parameter h_{st} can be shared to all nodes with a broadcast, so all vertices can choose the setting of h, p appropriately. Thus, Case 2 takes $\tilde{O}(n^{2/3} + \sqrt{nh_{st}} + D)$ rounds. □

We augment Algorithm 1, which computes only weights, to also construct replacement paths using routing tables in the full version [22].

3.3 Approximate Directed Weighted RPaths Algorithm

We present a $(1+\epsilon)$-approximation algorithm for directed weighted RPaths that runs in $\tilde{O}\left(\sqrt{nh_{st}} + D + \min\left(n^{2/3}, h_{st}^{2/5} n^{2/5+o(1)} D^{2/5}\right)\right)$ rounds.

Proof (Proof of Theorem 1.C). Our algorithm is based on the directed unweighted RPaths algorithm described earlier. The key tool is to replace h-hop BFS computation in line 9 of Algorithm 1 with $(1+\epsilon)$-approximate h-hop limited shortest path computation, using an algorithm in ([24], Theorem 3.6), which gives us a $\tilde{O}(k+h)$-round algorithm for k sources.

With this change, approximate distances are computed in the skeleton graph in line 3 of Algorithm 2. Thus, the local detour distances (both short and long) are $(1+\epsilon)$-approximate detour distances in line 5. The final replacement paths add these approximate detours to exact distances (line 15 of Algorithm 1) and are hence $(1+\epsilon)$-approximate. Using the same analysis as Lemma 5, we get an algorithm with round complexity $\tilde{O}\left(n^{2/3} + \sqrt{nh_{st}} + D\right)$.

When h_{st} is small, we can improve the $h_{st} \cdot SSSP$ round algorithm used in the exact unweighted algorithm. A recent result in [23] shows that k-source approximate directed weighted SSSP can be performed in $\tilde{O}(\sqrt{nk} + D)$ rounds if $k \geq n^{1/3}$ and in $\tilde{O}(\sqrt{nk} + D + k^{2/5} n^{2/5+o(1)} D^{2/5})$ rounds if $k < n^{1/3}$. We compute

all detours using an h_{st}-source SSSP computation by treating each $a \in P_{st}$ as a source and computing shortest path distances in $G - P_{st}$. This method is efficient when $h_{st} < n^{1/3}$. Combining the two methods proves our result. □

4 Further Research

We have presented several nontrivial algorithms and lower bounds for RPaths and 2-SiSP in the CONGEST model, with many of our results being near-optimal. A key avenue for further research is to narrow or close the gap between the $\tilde{O}(n^{2/3} + \sqrt{nh_{st}} + D)$ upper bound and $\tilde{\Omega}(\sqrt{n} + D)$ lower bound for directed unweighted RPaths and approximate weighted directed RPaths. Another question is whether the dependence on h_{st} can be improved in our algorithms.

References

1. Abboud, A., Censor-Hillel, K., Khoury, S.: Near-linear lower bounds for distributed distance computations, even in sparse networks. In: Gavoille, C., Ilcinkas, D. (eds.) Proceedings of DISC 2016, pp. 29–42. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-53426-7_3
2. Abboud, A., Grandoni, F., Williams, V.V.: Subcubic equivalences between graph centrality problems, apsp and diameter. In: Proceedings of SODA 2015, pp. 1681–1697. SIAM (2015)
3. Agarwal, U., Ramachandran, V.: Fine-grained complexity for sparse graphs. In: Proceedings of STOC 2018, pp. 239–252 (2018)
4. Agarwal, U., Ramachandran, V.: Faster deterministic all pairs shortest paths in congest model. In: SPAA 2020, pp. 11–21. ACM (2020)
5. Ancona, B., Censor-Hillel, K., Dalirrooyfard, M., Efron, Y., Williams, V.V.: Distributed distance approximation. In: OPODIS 2020, vol. 184, pp. 30:1–30:17. Schloss Dagstuhl - Leibniz-Zentrum für Informatik (2020)
6. Bar-Yossef, Z., Jayram, T.S., Kumar, R., Sivakumar, D.: An information statistics approach to data stream and communication complexity. J. Comput. Syst. Sci. **68**(4), 702–732 (2004)
7. Bernstein, A., Nanongkai, D.: Distributed exact weighted all-pairs shortest paths in near-linear time. In: Proceedings of STOC 2019, pp. 334–342. ACM (2019)
8. Bodwin, G., Parter, M.: Restorable shortest path tiebreaking for edge-faulty graphs. J. ACM **70**(5), 1–24 (2023)
9. Cao, N., Fineman, J.T.: Parallel exact shortest paths in almost linear work and square root depth. In: Proceedings of SODA 2023, pp. 4354–4372. SIAM (2023)
10. Chechik, S., Cohen, S.: Near optimal algorithms for the single source replacement paths problem. In: Proceedings of SODA 2019, pp. 2090–2109. SIAM (2019)
11. Chechik, S., Magen, O.: Near optimal algorithm for the directed single source replacement paths problem. In: ICALP 2020, vol. 168, pp. 81:1–81:17 (2020)
12. Chechik, S., Mukhtar, D.: Single-source shortest paths in the CONGEST model with improved bounds. Distrib. Comput. **35**(4), 357–374 (2022)
13. Dinitz, M., Robelle, C.: Efficient and simple algorithms for fault-tolerant spanners. In: PODC 2020, pp. 493–500. ACM (2020)

14. Elkin, M.: An unconditional lower bound on the time-approximation trade-off for the distributed minimum spanning tree problem. SIAM J. Comput. **36**(2), 433–456 (2006)
15. Ghaffari, M., Parter, M.: Near-optimal distributed algorithms for fault-tolerant tree structures. In: Proceedings of SPAA 2016, pp. 387–396. ACM (2016)
16. Ghaffari, M., Udwani, R.: Brief announcement: distributed single-source reachability. In: Proceedings of PODC 2015, pp. 163–165. ACM (2015)
17. Hoang, L., et al.: A round-efficient distributed betweenness centrality algorithm. In: Proceedings of PPoPP 2019, pp. 272–286. ACM (2019)
18. Katoh, N., Ibaraki, T., Mine, H.: An efficient algorithm for k shortest simple paths. Networks **12**(4), 411–427 (1982)
19. Kushilevitz, E., Nisan, N.: Communication Complexity. Cambridge University Press, Cambridge (1996)
20. Lawler, E.L.: A procedure for computing the k best solutions to discrete optimization problems and its application to the shortest path problem. Manag. Sci. **18**(7), 401–405 (1972)
21. Lenzen, C., Patt-Shamir, B., Peleg, D.: Distributed distance computation and routing with small messages. Distrib. Comput. **32**(2), 133–157 (2019)
22. Manoharan, V., Ramachandran, V.: Near optimal bounds for replacement paths and related problems in the congest model. arXiv preprint arXiv:2205.14797 (2022)
23. Manoharan, V., Ramachandran, V.: Improved approximation bounds for minimum weight cycle in the congest model. In: Proceedings of PODC 2024, ACM (2024, to appear)
24. Nanongkai, D.: Distributed approximation algorithms for weighted shortest paths. In: Proceedings of STOC 2014, pp. 565–573. ACM (2014)
25. Parter, M.: Distributed constructions of dual-failure fault-tolerant distance preservers. In: DISC 2020. Schloss Dagstuhl-Leibniz-Zentrum für Informatik (2020)
26. Parter, M.: Nearly optimal vertex fault-tolerant spanners in optimal time: sequential, distributed, and parallel. In: Proceedings of STOC 2022, pp. 1080–1092. ACM (2022)
27. Peleg, D.: Distributed Computing: A Locality-Sensitive Approach. SIAM (2000)
28. Peleg, D., Rubinovich, V.: A near-tight lower bound on the time complexity of distributed minimum-weight spanning tree construction. SIAM J. Comput. **30**(5), 1427–1442 (2000)
29. Razborov, A.A.: On the distributional complexity of disjointness. Theor. Comput. Sci. **106**(2), 385–390 (1992)
30. Roditty, L., Zwick, U.: Replacement paths and k simple shortest paths in unweighted directed graphs. ACM Trans. Algor. (TALG) **8**(4), 1–11 (2012)
31. Rozhoň, V., Haeupler, B., Martinsson, A., Grunau, C., Zuzic, G.: Parallel breadth-first search and exact shortest paths and stronger notions for approximate distances. In: Proceedings of STOC 2023, pp. 321-334. ACM (2023)
32. Sarma, A.D., et al.: Distributed verification and hardness of distributed approximation. SIAM J. Comput. **41**(5), 1235–1265 (2012)
33. Williams, V.V., Williams, R.R.: Subcubic equivalences between path, matrix, and triangle problems. J. ACM **65**(5), 27:1–27:38 (2018)
34. Yen, J.Y.: Finding the k shortest loopless paths in a network. Manag. Sci. **17**(11), 712–716 (1971)