Parallel Reachability and Shortest Paths on Non-Sparse Digraphs: Near-Linear Work and Sub-Square-Root Depth

Vikrant Ashvinkumar
Rutgers University, Piscataway, NJ, USA

Aaron Bernstein
New York University, NY, USA

Maximilian Probst Gutenberg
ETH Zurich, Switzerland

Thatchaphol Saranurak
University of Michigan, Ann Arbor, MI, USA

Abstract

We present parallel algorithms for computing single-source reachability and shortest paths on directed n-vertex m-edge graphs using near-linear $\tilde{O}(m)$ work and $o(\sqrt{n})$ depth whenever $m \geq n^{1+o(1)}$. At the extreme of $m = \Omega(n^2)$, our reachability and shortest path algorithms have depth only $n^{0.136}$ and $n^{0.25+o(1)}$, respectively. The state-of-the-art parallel algorithms with near-linear work for both problems [11, 6, 17, 5, 4] require $\Omega(\sqrt{n})$ depth in all density regimes.

2012 ACM Subject Classification Theory of computation → Graph algorithms analysis

Keywords and phrases shortcut set, parallel reachability, hopset, parallel SSSP

Digital Object Identifier 10.4230/LIPIcs.ICALP.2026.15

Category Track A: Algorithms, Complexity and Games

Related Version *Full Version*: `https://arxiv.org/abs/2605.03892`

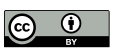
Funding *Aaron Bernstein*: Supported by Sloan Fellowship, Google Research Fellowship, NSF Grant 1942010, and Charles S. Baylis endowment at NYU.
Maximilian Probst Gutenberg: The research leading to these results has received funding from grant no. 200021 204787 of the Swiss National Science Foundation.
Thatchaphol Saranurak: Supported by NSF Grant CCF-2238138 and a Sloan Fellowship.

Acknowledgements We would like to thank Shang-En Huang, who declined co-authorship despite being involved in our discussions.

1 Introduction

Single-source reachability and shortest paths (SSSP) are some of the most fundamental algorithmic problems on digraphs. Given a digraph $G = (V, E)$ on n vertices and $m = \Omega(n)$ edges, and a source $s \in V$, reachability asks for the set of all vertices that s can reach, and shortest paths for the distances from s to v for all $v \in V$. While the solutions to these problems are fairly well understood in the sequential setting – reachability is solvable in $O(m)$ time using a breadth-first search (BFS) and shortest paths in $\widetilde{O}(m)$ time[1] using Dijkstra's algorithm – we are much more limited in our understanding of these problems in other computational models. We focus on these problems under the parallel setting in this paper.

[1] Here and throughout, we use the $\widetilde{O}(\cdot)$ notation to suppress polylogarithmic factors.



The *work* $W(n)$ of a parallel algorithm is the number of primitive operations it performs on an input of size n, which can be interpreted as the running time of the algorithm when sequentialized, i.e., the running time with just one processor. Ideally, $W(n)$ asymptotically equals the best sequential running time in which case we call the algorithm work-efficient. For example, a work-efficient algorithm for reachability would have $W(|E(G)|) = O(m)$, matching the running time of BFS. We call a parallel algorithm nearly work-efficient if $W(n)$ matches the best sequential running time up to polylogarithmic factors. We interchangeably use the terms *depth* or *span* $D(n)$ of a parallel algorithm to refer to the length of the longest chain of sequential dependencies in the algorithm, which can be interpreted as the fastest parallel time the algorithm could possibly run in, i.e., the running time when given an unconstrained number of processors.

In this paper, we give nearly work-efficient algorithms for both single-source reachability and SSSP that have $o(\sqrt{n})$ depth on non-sparse digraphs. Prior to our work, the state-of-the-art nearly work-efficient algorithms for both reachability and SSSP had depth at least $\Omega(\sqrt{n})$ for all graph densities, achieved by [11, 4] and [17, 5, 4] respectively.

1.1 Prior Work and Background

Background on Parallel Reachability

The two most basic algorithms for parallel reachability are (i) using parallel matrix multiplication to compute the transitive closure of G which, while highly parallel with polylog(n) depth, is not remotely work-efficient; and (ii) using parallel BFS which, while work-efficient, unfortunately has span proportional to the depth of the BFS-tree of G, which is $\Omega(n)$ on worst-case inputs. Until somewhat recently, no nearly work-efficient algorithm with sublinear depth had been known. The breakthrough of Fineman [8] gave the first nearly work-efficient algorithm with sublinear depth $\widetilde{O}(n^{2/3})$. This was then built upon by Jambulapati, Liu, and Sidford [11] to give a nearly work-efficient algorithm with $n^{1/2+o(1)}$ depth, which remains the best known upper bound. This algorithm can also be generalized to give a work-span tradeoff [7].

Both [8, 11] utilize a framework based on *shortcut sets*, which dates back to [19]. A β-shortcut set of G is a set $H \subseteq V \times V$ such that the reachability relations of G and $G \cup H$ are the same, and for any $u, v \in V$ such that u reaches v, there is a path from u to v using at most β hops in $G \cup H$. Given a β-shortcut set H, one may compute single-source reachability in $O(m + |H|)$ work and $O(\beta)$ depth by computing a parallel BFS on $G \cup H$ from source s. There is a folklore construction of an $O(\sqrt{n})$-shortcut set H with $|H| = \widetilde{O}(n)$: randomly sample $\widetilde{O}(\sqrt{n})$ vertices and add all shortcut edges uv where u and v are sampled vertices such that u can reach v in G. This was (at the time of [8, 11]) the best known existential bound, but no efficient algorithmic constructions meeting these bounds were known, even in the *sequential* setting. The breakthrough of [8] was a nearly work-efficient construction of an $\widetilde{O}(n^{2/3})$-shortcut set with linear size. [11] then gave the first nearly work-efficient construction of a shortcut set that essentially matches the folklore bound. The key technical contribution to both results was the first near-linear time sequential construction of said shortcut sets, which they then showed how to parallelilize.

Both [8] and [11] construct a shortcut set H with only $\widetilde{O}(n)$ edges. But for many problems, and for parallel reachability in particular, a shortcut set with $\widetilde{O}(m)$ edges would do just as well. For non-sparse graphs, allowing H to contain more edges is conceptually advantageous because the folklore construction can be tuned to achieve a $O(n/\sqrt{m})$-shortcut set H with $|H| = \widetilde{O}(m)$, by simply sampling more vertices. Unfortunately, the constructions of [8, 11] are

not so readily tunable in this way, so we do not have any efficient algorithmic constructions of these denser hopsets, even in the sequential settings. In this paper, we give a near-linear time sequential construction of an $\widetilde{O}(m)$ size β-shortcut set where β scales with the density of the base graph; for any m polynomially larger than n, we achieve β polynomially better than $\sqrt{n}$. We can further improve the tradeoff with fast matrix multiplication: at the extreme case, if $\omega = 2$, our construction essentially matches the bounds of the tuned folklore construction, generalizing [11]. We then show that all our sequential bounds can be easily parallelized, leading to nearly work-efficient parallel reachability algorithms with significantly lower depth in non-sparse graphs.

Background on Parallel Shortest Paths

Rozhoň ⓡ[2] Haeupler ⓡ Martinsson ⓡ Grunau ⓡ Zuzic [17] gave a blackbox reduction from SSSP to single-source $(1+\varepsilon)$-approximate shortest paths, so we focus on the latter.

A similar framework to computing shortcut sets followed by parallel BFS is viable for approximate shortest paths. Instead of BFS, one uses an algorithm of Klein and Subramanian [12] for finding (exact) shortest β hop paths in $\widetilde{O}(m)$ work and $\widetilde{O}(\beta)$ span. And instead of a β-shortcut set, one constructs a so-called (β, ε)-*hopset* of G which is a set $H \subseteq V \times V$ such that $\text{dist}_G(u,v) \leq \text{dist}^{(\beta)}_{G \cup H}(u,v) \leq (1+\varepsilon)\text{dist}_G(u,v)$ for all $u, v \in V$, where $\text{dist}^{(\beta)}$ is the length of the shortest path using at most β hops. The folklore shortcut set is easily adapted to showing the existence of $(O(\sqrt{n}), \varepsilon)$-hopsets H with $|H| = \widetilde{O}(n)$. Analogously, Cao, Fineman, and Russell [6] adapted the shortcut set construction of [11] to give a near-linear time construction of an $(n^{1/2+o(1)}, \varepsilon)$-hopset H with $|H| = \widetilde{O}(n)$, essentially meeting this folklore bound. A work-span tradeoff was then later given in [7].

Just as with shortcut sets, the requirement that $|H| = \widetilde{O}(n)$ is sometimes too stringent. The folklore hopset can in the same way be tuned to give an $(O(n/\sqrt{m}), \varepsilon)$-hopset H with $|H| = \widetilde{O}(m)$ by sampling more vertices, but the hopset of [6] does not immediately generalize in this way. In this paper, we give a near-linear time sequential construction of an $\widetilde{O}(m)$ size (β, ε)-hopset where β scales with the density of the base graph. This is then parallelized and plugged into the abovementioned framework to get faster nearly work-efficient parallel SSSP algorithms.

1.2 Our Results

See Figure 1 for a quick comparison of our main results with the strongest known bounds prior to this work. Note that the parallel SSSP algorithm is purely combinatorial; for reachability, we get a better tradeoff using fast matrix multiplication.

Parallel Reachability and Shortcut Sets

Our main result is on parallel reachability.

▶ **Theorem 1** (Parallel Reachability). *Let ω be the fast matrix multiplication exponent in the work for parallel algorithms using polylogarithmic (in n) span. There is an $\widetilde{O}(m)$ work and $\sqrt[2\omega-2]{n^{\omega+o(1)}/m}$ span randomized parallel algorithm that, given an unweighted digraph G and source $s \in V$, outputs with high probability all vertices that s can reach and all vertices that reach s.*

[2] A randomized author ordering generated in the cited work is delimited by ⓡ separators.

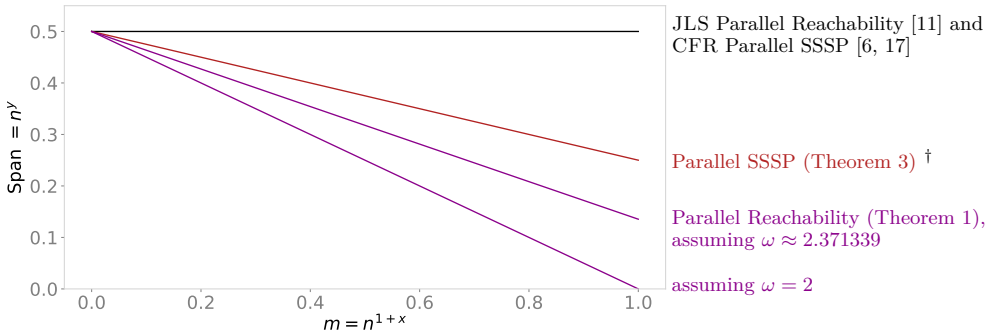


Figure 1 Comparison of spans of $\widetilde{O}(m)$ work parallel algorithms for reachability and SSSP on digraphs. Each plotted curve corresponds to such a parallel algorithm. The x-axis is the density of the input digraph, and the y-axis is the span of the algorithm. †: The bound for parallel SSSP also holds for *combinatorial* parallel reachability (i.e. $\omega = 3$).

Using the fact that $\omega < 2.371339$ (by [1] and Proposition 5), we can compute reachability in $\widetilde{O}(m)$ work and $n^{0.86461}/m^{0.36461}$ span ($n^{0.136}$ span when $m = \Omega(n^2)$). The previous best span for nearly work-efficient algorithms for reachability was at least $\Omega(\sqrt{n})$, regardless of digraph density: [11] achieves depth $n^{1/2+o(1)}$, while the min-cost flow algorithm of [4] reduces the depth to $\widetilde{O}(\sqrt{n})$ when $m \gtrsim n^{1.5}$.

The technical developments used to get the parallel reachability algorithm of Theorem 1 is largely a near-linear time *sequential* construction for a $\sqrt[2\omega-2]{n^{\omega+o(1)}/m}$-shortcut set with size $\widetilde{O}(m)$.

▶ **Theorem 2** (Sequential Shortcut Set). *Let ω be the fast matrix multiplication exponent. There is an $\widetilde{O}(m)$ time randomized algorithm that, given an unweighted digraph G, outputs with high probability a $\sqrt[2\omega-2]{n^{\omega+o(1)}/m}$ -shortcut set H with size $\widetilde{O}(m)$.*

More generally, we show the following tradeoff which the above is a special case of. Let $\varrho \in [\sqrt{n}]$. There is an $\widetilde{O}\left(m + n\varrho^{2\omega-2}\right)$ time randomized algorithm that, given an unweighted digraph G, outputs with high probability an $n^{1/2+o(1)}/\varrho$-shortcut set H with size $\widetilde{O}(n\varrho^2)$.

Refer to Table 1 for the spans of our algorithms given for different values of ω shown in a tabular format, summarizing Theorem 1 and Theorem 2. Of particular interest are the extremes where Theorem 2 outputs, if $\omega = 3$, a combinatorially constructed $n^{3/4+o(1)}/m^{1/4}$-shortcut set and, if $\omega = 2$ an $n^{1+o(1)}/\sqrt{m}$-shortcut set almost matching the tunable folklore shortcut set in its parameters. Also of note: the tradeoff given by Theorem 2 is strictly better than that given by [7], which is a randomized $\widetilde{O}\left(m\varrho^2 + n\varrho^4\right)$ time construction of an $n^{1/2+o(1)}/\varrho$-shortcut set H with size $\widetilde{O}(n\varrho^2)$; most crucially, because of the $m\varrho^2$ term, their tradeoff could not achieve near-linear work for any graph density.

Parallelizing the previous state-of-the-art near-linear size $n^{1/2+o(1)}$-shortcut set of [11] was, while important, somewhat tedious. Using the new technology of [9], we can easily parallelize Theorem 2 – see Theorem 16 for details – and consequently use the parallel construction to immediately get our main result Theorem 1.

Table 1 Span of our near-linear work parallel reachability algorithms (or β value of our β-shortcut sets) for different values of ω.

Value of ω	Span (or β)	Span (or β) when $m = \Theta(n^2)$
(Combinatorial): 3	$n^{3/4+o(1)}/m^{1/4}$	$n^{1/4+o(1)}$
(Ideal): 2	$n^{1+o(1)}/\sqrt{m}$	$n^{o(1)}$
(Current [1]): 2.371339	$n^{0.86461}/m^{0.36461}$	$n^{0.136}$

Parallel SSSP and Hopsets

We then show analogous results for SSSP, albeit with slightly weaker bounds. The details of these results can be found in the full version.

▶ **Theorem 3** (Parallel SSSP). *There is an $\widetilde{O}(m)$ work and $\sqrt[4]{n^{3+o(1)}/m}$ span randomized parallel algorithm that, given a polynomially bounded non-negative integer weighted digraph G and source $s \in V$, outputs with high probability $\mathrm{dist}(s, v)$ for all $v \in V$ along with a shortest path tree rooted at s.*

On dense digraphs, this gives a nearly work-efficient $n^{1/4+o(1)}$ span algorithm for SSSP. The previous best span for nearly work-efficient algorithms for SSSP was at least $\Omega(\sqrt{n})$, similar to the previous state of reachability: [17, 5] achieve depth $n^{1/2+o(1)}$, while the min-cost flow algorithm of [4] reduces the depth to $\widetilde{O}(\sqrt{n})$ when $m \gtrsim n^{1.5}$.

Analogously to parallel reachability, the main driver of Theorem 3 is a near-linear time *sequential* construction of a $\left(\sqrt[4]{n^{3+o(1)}/m}, \varepsilon\right)$-hopset H with size $\widetilde{O}(m/\varepsilon^2)$.

▶ **Theorem 4** (Sequential Hopset). *There is an $\widetilde{O}(m/\varepsilon^2)$ time randomized algorithm that, given a polynomially bounded non-negative integer weighted digraph G, outputs with high probability a $\left(\sqrt[4]{n^{3+o(1)}/m}, \varepsilon\right)$-hopset H with size $\widetilde{O}(m/\varepsilon^2)$.*

More generally, we show the following tradeoff which the above is a special case of. Let $\varrho \in [\sqrt{n}]$. There is an $\widetilde{O}\left(m/\varepsilon^2 + n\varrho^4\right)$ time randomized algorithm that, given a polynomially bounded [3] non-negative integer weighted digraph G, outputs with high probability a $\left(n^{1/2+o(1)}/\varrho, \varepsilon\right)$-hopset H with size $\widetilde{O}\left(n/\varepsilon^2 + n\varrho^2\right)$.

In particular, Theorem 4 on dense digraphs runs in $\widetilde{O}(n^2/\varepsilon^2)$ time and shortcuts $(1+\varepsilon)$-approximate shortest paths to $n^{1/4+o(1)}$ hops, whereas the previous best hop bound was $n^{1/2+o(1)}$ given by [6] for all density regimes. Similarly to shortcut sets, the tradeoff of Theorem 4 is strictly better than (the sequential version) of [7].

We get Theorem 3 by parallelizing Theorem 4, again simply, using the new technology of [9] – see the full version for details on said parallelization.

Due to technical complications described in the full version, not all of the ideas used for shortcut sets (Theorem 2) go through for constructing hopsets (Theorem 4). Nevertheless, the main conceptual ideas do indeed go through, allowing us to match the bound in the combinatorial version of Theorem 2, i.e. when $\omega = 3$ (see Table 1 for more on the combinatorial version).

[3] Since we ultimately intend to compare our SSSP bounds with [17], we assume polynomially bounded edge weights the same way they have. Without this polynomial bound, our running time and hopset size suffers a $\log(nW)$ factor where W is the largest weight in G.

Related Work on Shortcut Sets and Hopsets

There exists a rich literature on shortcut sets and hopsets with stronger existential bounds than the folklore construction [14, 13, 3, 15, 16] though currently no near-linear time construction of is known, even in the sequential setting. In particular, the breakthrough of [14] showed the existence of a $\sqrt[3]{n^2/m}$-shortcut set with size $\widetilde{O}(m)$ which, if constructible in near-linear work, would give a depth $\sqrt[3]{n^2/m} \ll n/\sqrt{m}$ algorithm for reachability. Along another direction, [2] recently showed a deterministic construction of an $O(\sqrt{n})$-shortcut set with size $O(n \log^* n)$ in almost-linear time. We include a more detailed discussion of these related works in the full version.

1.3 Organization

Preliminaries are covered in Section 2. A high-level overview of our main technical ideas is then provided in Section 3. The core of the parallel reachability result is in Section 4 (which goes over a sequential construction of shortcut sets) and finished up in Section 5 where parallelization of the sequential construction is discussed. For the core of the parallel SSSP result (which goes over a sequential construction of hopsets) and open problems, see the full version.

2 Preliminaries

Numbers and Sets

For any positive integer $z \in \mathbb{N}$, we use $[z]$ to denote $\{1, 2, \ldots, z\}$. For integers $a \leq b$, we use $[a, b]$ to denote $\{a, a+1, \ldots, b\}$.

Graphs

In this paper we work with unweighted digraphs $G = (V, E)$ and also weighted digraphs $G = (V, E, w)$. We use $V(G)$ (or when unambiguous, V) to refer to the vertex set of G and $E(G)$ (or when unambiguous, E) to refer to the edge set of G. An edge pointing from vertex u to vertex v is denoted with uv. For any $V' \subseteq V$ we use $G[V']$ to denote the subgraph of G induced on V'. When it is clear, we use $n = |V|$ and $m = |E|$.

Paths and Distances

A path using h hops is a sequence of vertices $P = \langle v_0, v_1, \ldots, v_h \rangle$ where $v_{i-1}v_i \in E$. A subpath of $P = \langle v_0, v_1, \ldots, v_j \rangle$ is a path of the form $\langle v_a, v_{a+1}, \ldots, v_z \rangle$ with $0 \leq a \leq z \leq h$. If we say a path P is split into k subpaths $P_1, P_2, \ldots, P_k$ we mean that the P_i are disjoint and $P = P_1 P_2 \ldots P_k$.

Hopbounds

We say that a digraph G has *reachability hopbound* h if for all $s, t \in V$ such that s reaches t, there is a path using at most h hops from s to t.

Shortcut Sets

$H \subseteq V^2$ is a *β-shortcut set* of a digraph G if **1)** for all $st \in H$, which we call shortcuts, there is an st-path in G and **2)** The reachability hopbound in $G \cup H$ is not more than β.

Reachability Relations and Relevant Vertices

If there is a path from s to t we say s reaches t and also t is reached by s; we say here that s and t are related. $R^+(G,s)$ is the set of all vertices s reaches in G and $R^-(G,t)$ is the set of all vertices which reach t in G. We denote the set of *relevant vertices* to v with $\mathrm{R}(G,v) = R^+(G,v) \cup R^-(G,v)$. These are extended to sets in the natural way; for example, $R^+(G,S) = \cup_{s\in S} R^+(G,s)$.

Path-related Vertices (Ancestors, Descendants, Bridges)

For any path P, we call $R^-(G,P) \setminus R^+(G,P)$ its *ancestors* and $R^+(G,P) \setminus R^-(G,P)$ its *descendants* and finally $R^-(G,P) \cap R^+(G,P)$ its *bridges*.

Fast Matrix Multiplication

We use the following result for parallel matrix multiplication, which states that we can multiply matrices with work matching the current best sequential time complexity of [1] and span only $O(\log n)$.

▶ **Proposition 5** (Paraphrasing (with some modification) of Theorem 5.7 Part 1 in [10]). *Let A and B be two n by n matrices with entries in a ring R with operations $+, \times$. The matrix AB can be computed by a parallel algorithm with $O(\log n)$ span and $O(M(n))$ work, where $M(n)$ is the best known sequential bound for computing AB over R by an algorithm that can be written as an algebraic circuit with $+, \times$ gates.*

That is to say, throughout this paper $\omega < 2.371339$ (even in the parallel setting) which is established by [1], whose (sequential) algorithm satisfies the premise of Proposition 5.

Probability

We use the following one-sided Chernoff bound for sums of independent $\{0,1\}$ random variables X_i with mean $\mu = \mathbb{E}\left[\sum_i X_i\right]$:

$$\Pr\left[X \geq (1+\delta)\,\mu\right] \leq e^{-\delta^2\mu/(2+\delta)} \text{ for } \delta \geq 0.$$

3 High-Level Overview

Here we give a sketch of the main ideas in this paper. To obtain our result, we introduce a new pruning step into the framework of [11, 6], respectively called TC-Pruning and TruncSSSP-Pruning. The main technical contribution is actually in the analysis of this. We sketch a new top-down analysis of [11, 6] which naturally suggests how to apply the pruning steps, leading to the improvements below. Since the core insights can be found within our reachability result, which is much simpler, we omit any discussion of SSSP in this overview.

3.1 Summary of Prior Work

By using standard techniques, the reachability problem on a digraph G is reduced to constructing a shortcut set of G; indeed, if one can construct a β-shortcut set H in $\widetilde{O}(m)$ work and D span, then one gets a parallel reachability algorithm by running a parallel BFS from the source s on $G \cup H$, which takes an additional $\widetilde{O}(m + |H|)$ work and β span (so, in sum, an $\widetilde{O}(m + |H|)$-work and $O(D + \beta)$-span algorithm). We thus focus on the problem of constructing a shortcut set.

[11] refines the breakthrough of [8] to give an $\widetilde{O}(m)$ work algorithm which constructs an $n^{1/2+o(1)}$-shortcut set in span $n^{1/2+o(1)}$. As a precursor to this, they give a sequential $\widetilde{O}(m)$ time algorithm which constructs an $n^{1/2+o(1)}$-shortcut set with $\widetilde{O}(n)$ size, henceforth called the JLS shortcut set. We first turn our attention to this sequential construction.

Let us assume G is a directed acyclic graph (DAG). Suppressing some details (e.g. parameter settings) that are inessential to an overview, the JLS algorithm can be described recursively in the following way, where G is the level 0 recursive instance. On a level r recursive instance $G[V']$, where $V' \subseteq V$, some pivot vertices $S \subseteq V'$ are selected randomly. For each $v \in V'$ and $p \in S$, if v reaches p then add vp to the shortcut set, and if p reaches v then add pv to the shortcut set. Next, we partition $V' = V'_1 \sqcup V'_2 \sqcup \dots$ into an equivalence relation based on the reachability relation to S in $G[V']$; that is, for all $u, v \in V'$ and i, we have $u, v \in V'_i$ iff the set of pivots in S that reach u and v are equal and the set of pivots in S that u and v reach are equal. We then set each $G[V'_i]$ to be a level $r+1$ recursive instance. A complete description of JLS is given later in Section 4.

For appropriately chosen parameters, the above algorithm runs in $\widetilde{O}(m)$ time and yields an $n^{1/2+o(1)}$-shortcut set with near-linear size.

Framework for Bounding the Diameter

To bound the diameter of JLS, we fix an arbitrary path P and show that P is shortcut to length $n^{1/2+o(1)}$. First, observe that at any fixed recursion level, say r, the path P is split into contiguous subpaths $P_1, P_2, \dots$ each belonging to distinct level $r+1$ recursive instances $G_1, G_2, \dots$ respectively. To see this, note that if a pivot vertex reaches (resp. is reached by) any two vertices in P, then it reaches (resp. is reached by) every vertex on the subpath between said two vertices in P; thus, if x and y are in the same subproblem G_i, then so are all the vertices on any path from x to y. (See Observation 8 for the formal proof.)

We use this observation to define the *subproblem tree of* P, which tracks how P is shortcut. (i) The root node[4] of this tree, at level 0, is P; (ii) If a bridge of a level r node P' is selected in the recursion level r instance G' (in which P' is contained) then P' is a leaf node and, otherwise, P' has level $r+1$ children nodes $P'_1, P'_2, \dots$ corresponding to how P' is split. The reader should observe that each leaf P' is shortcut to $O(1)$ hops (moving through the bridge of P') and, hence, the number of nodes in the subproblem tree upper bounds the number of hops P is shortcut to (up to a constant factor).

[11] shows that the number of nodes in the subproblem tree of any path P is bounded above by $n^{1/2+o(1)}$. Crucially, they use the following key lemma within a bottom-up argument by induction:

$$\mathbb{E}\left[\sum_{i=1}^{|S|+1} \left|\mathrm{R}(G'_i, P'_i)\right| \,\middle|\, |S|\right] \leq \frac{2}{|S|+1}\left|\mathrm{R}(G', P')\right|, \qquad \text{(Proposition 11)}$$

where P' (in subproblem G') is an arbitrary internal node in the subproblem tree and S are the pivots selected in $\mathrm{R}(G', P')$ that split P' into subpaths $P'_1, P'_2, \dots, P'_{|S|+1}$ in subproblems $G'_1, G'_2, \dots, G'_{|S|+1}$ respectively. Recall (from Section 2) that $\mathrm{R}(G', P')$ are the vertices in G' that can reach or are reached by some vertex in P'.

[4] Here and throughout, we use "node" to refer to the nodes of subproblem trees, which are analysis tools which our algorithms are not aware of, and "vertex" to refer to the vertices of digraphs, which are the objects our algorithms interface with.

We suggest that Proposition 11 is best interpreted in the following way. Intuitively, splitting a path P' into $P'_1, P'_2, \ldots, P'_{|S|+1}$ is detrimental to shortcutting P' (and hence P) since the algorithm will no longer be able to add any shortcut between P'_i and P'_j for $i \neq j$. For example, if $|S| \gg n^{1/2+o(1)}$, then there would no longer be any hope of shortcutting P to $n^{1/2+o(1)}$ hops. On the other hand, Proposition 11 shows that splitting a path still offers progress: the number of relevant vertices to P' drops significantly. Since $P' \subseteq \bigcup_{i=1}^{|S|+1} \mathrm{R}(G'_i, P'_i)$ and each $v \in P'$ is a bridge of some P'_i, the fraction of bridges increases[5]. It is consequently more likely to select bridges as pivots and shortcut the subpaths that P' is split into to $O(1)$ hops each.

3.2 Summary of Our Main Result

We now sketch a new proof of the diameter bound of JLS, which reveals an opportunity to lessen the diameter of the shortcut set to $\beta \ll n^{1/2+o(1)}$ at the expense of having more edges in the shortcut set. Thereafter, we will show concretely how to exploit this opportunity by adding just one line to JLS.

A New, Top-Down, More General Analysis of the JLS Diameter Bound

For simplicity's sake, let us assume something stronger than Proposition 11: that the typical scenario occurs *deterministically*.

$$\sum_{i=1}^{|S|+1} |\mathrm{R}(G'_i, P'_i)| \leq \frac{2}{|S|+1} |\mathrm{R}(G', P')|. \qquad \text{(Idealized Proposition 11)}$$

Assuming Idealized Proposition 11 above does not change any structure of our argument and only removes clutter related to formalization of probabilistic guarantees.

Given this, we first bound the number of nodes in the subproblem trees.

▷ Claim 6. For all x, there are at most $n^{1/2+o(1)}/\sqrt{x}$ nodes P' in the subproblem tree where $|\mathrm{R}(G', P')| \geq x$.

Proof Sketch. Define the function $\phi(P') = \sqrt{|\mathrm{R}(G', P')|}$. We will show $\sum_{\text{tree nodes } P'} \phi(P') = n^{1/2+o(1)}$ and from there the bound follows since each P' with $|\mathrm{R}(G', P')| \geq x$ contributes $\phi(P') \geq \sqrt{x}$ to the sum.

To see that $\sum_{P'} \phi(P') = n^{1/2+o(1)}$, observe that $\phi(P) = O(\sqrt{n})$; the subproblem tree has $O(\log n / \log\log n)$ levels (we have not justified this in the overview, see Section 4); and

$$\sum_{i}^{|S|+1} \phi(P'_i) = \sum_{i}^{|S|+1} \sqrt{|\mathrm{R}(G'_i, P'_i)|} \leq \sqrt{(|S|+1) \sum_{i}^{|S|+1} |\mathrm{R}(G'_i, P'_i)|} \leq \sqrt{2}\phi(P')$$

where the first inequality is by Cauchy-Schwarz and the second is by Idealized Proposition 11. ◁

By plugging in $x = 1$, we recover the $n^{1/2+o(1)}$ diameter bound of JLS. Even more, we may plug in other values of x to get more refined upper bounds. Noting that the fanout of the subproblem tree is polylogarithmic (another detail omitted here), this suggests the following type of strategy: shortcut all nodes P' with $|\mathrm{R}(G', P')| < x$ to $O(1)$ hops. We

[5] This is not entirely true, since bridges of P' may cease to be bridges of P'_i for any i.

can then consider the *pruned subproblem tree of* P with all nodes P' where $|\mathrm{R}(G', P')| < x$ removed. By Claim 6, the number of nodes in the pruned tree is at most $n^{1/2+o(1)}/\sqrt{x}$ up to polylogarithmic factors, which upper bounds the number of hops to which P is shortcut.

A Simple Combinatorial Improvement (Warmup)

Assume the input digraph is dense – that is, it has $\Omega(n^2)$ edges. Denote $\mathrm{BFS}_{\sqrt{n}}(G', v)$ as the first $\sqrt{n}$ vertices found in a BFS in G' starting from $v \in V(G')$.

Here is a simple modification to the JLS algorithm. For each recursive subproblem G' and each $v \in V(G')$, add all the following edges to the shortcut set: edges from v to $\mathrm{BFS}_{\sqrt{n}}(G', v)$.

Note that $\mathrm{BFS}_{\sqrt{n}}(G', v)$ runs in $O(n)$ time, totaling up to $\widetilde{O}(n^2)$ time over all the calls made in all recursive subproblems. This additional step thus gives a negligible overhead to the running time of JLS.

Crucially, for any node $P' = \langle v_0, v_1, \ldots, v_h \rangle$ in the subproblem tree with $|\mathrm{R}(G', P')| < \sqrt{n} = x$, the edge $v_0 v_h$ will be added to the shortcut set since $v_h \in \mathrm{BFS}_{\sqrt{n}}(G', v_0)$. Namely, P' is shortcut to $O(1)$ hops and is removed from the pruned subproblem tree which, accordingly, has $O(n^{1/4+o(1)})$ nodes by Claim 6 (which, recall, upper bounds the length P is shortcut to).

In summary, this simple modification to JLS gives a near-linear time construction of an $O(n^{1/4+o(1)})$-shortcut set on dense digraphs G.

A Stronger but Non-Combinatorial Improvement

Our main result for reachability does not use the combinatorial improvement above, but instead utilizes fast matrix multiplication. Here we just briefly summarize the idea and leave the details to Section 4. Instead of computing $\mathrm{BFS}_{\sqrt{n}}(G', v)$ from each $v \in V(G')$ in every recursive subproblem G' as in our warmup, we get even more of a speed up from loosely speaking computing shortcuts in an "all-pairs" fashion (i.e. transitive closures). This is easier said than done since one call to a transitive closure algorithm is more expensive than one BFS call; we must thus be judicious of when and on what vertices to call the transitive closures (instead of from every vertex like we did with BFS).

When $|\mathrm{R}(G', P')|$ is small (at the threshold where we wish to prune away the node P' from the subproblem tree of P), one might try to call a transitive closure on a ball centered at v for some $v \in P'$. The flaw in this idea is that the choices made by the algorithm must be oblivious to P'. A more oblivious approach would be to call a transitive closure on $\mathrm{R}(G', p)$ for some pivot $p \in \mathrm{R}(G', P')$. However, $|\mathrm{R}(G', p)|$ is not necessarily the same as $|\mathrm{R}(G', P')|$ since p may not be a vertex in P'. In fact, p may reach or be reached by many vertices that have no reachability relation to P', and hence $|\mathrm{R}(G', p)|$ might be very large despite $|\mathrm{R}(G', P')|$ being very small; calling a transitive closure on $|\mathrm{R}(G', p)|$ may then be too expensive!

Instead, we do the following. For every pivot p, if $|\mathrm{R}(G', p)|$ is small enough, we add all edges in the transitive closure of $\mathrm{R}(G', p)$ to the shortcut set. This can be done in $\widetilde{O}(|\mathrm{R}(G', p)|^{\omega})$ time by repeatedly squaring the adjacency matrix of $G'[\mathrm{R}(G', p)]$. By virtue of $|\mathrm{R}(G', p)|$ being small, we have control over the running time of this improvement. Moreover, since all edges in the transitive closure of $\mathrm{R}(G', p)$ are added to the shortcut set, p would not contribute any children to the node P'. But would it allow us to prune the node P' completely when $|\mathrm{R}(G', P')|$ is small? Not quite. We show in Section 4 that it allows us to prune nodes in the subtree rooted at P' so that the pruned subtree has $O(\log n)$ nodes; loosely, the argument is as follows: for any node P'' in this subtree, if all $p \in \mathrm{R}(G'', P'')$ have

small $|\mathrm{R}(G'', p)|$ then P'' has at most one child and, otherwise, there is some $p \in \mathrm{R}(G'', P'')$ with large $|\mathrm{R}(G'', p)|$, certifying that it is very unlikely for any vertex in $\mathrm{R}(G'', P'')$ to be sampled as a pivot. This yields the $n^{0.136}$-shortcut set in near-linear time for dense graphs.

Parallelization

Finally, we parallelize the shortcut set construction by using a blackbox framework provided by [9]. Morally, the framework says that (up to some fudging) we need only guarantee that a parallel algorithm for computing a β-shortcut set runs in $\widetilde{O}(\beta)$ span on DAGs with reachability hopbound $\widetilde{O}(\beta)$; this is enough to show that a β-shortcut set runs in $\widetilde{O}(\beta)$ span on any digraph, regardless of its diameter. It is easy to show that our sequential construction for β-shortcut sets run in β span on DAGs with reachability hopbound $\widetilde{O}(\beta)$, since BFS calls will run in span β.

4 Sequential Shortcut Set Construction

In this section we prove Theorem 2 which pertains to a sequential construction of shortcut sets. Section 4.1 gives an overview of the JLS shortcut set and outlines a strategy for how to improve upon it. We then go over the details in Section 4.2 and finish things up in Section 4.3.

4.1 The JLS Shortcut Set, and a Strategy for Improved Bounds

The JLS shortcut set, which is a refinement of the construction of Fineman [8], can be summarized at a high level as follows. We first assume the input G is a DAG (since it is easy to compute strongly connected components (SCCs) in a digraph in linear time and, adding $O(n)$ edges to the shortcut set, shortcut each SCC to two hops). There is a global parameter k, which we can think of as $\log n$, that controls a sampling rate and recursion depth $\log_k(n)$. The construction is found via a recursive algorithm where at recursion level r, around k^{r+1} pivot vertices from the base graph G are selected uniformly at random (these pivots are divided up possibly unequally among all level r subproblems on graphs $G[V_1], G[V_2], \ldots$). Let us focus on one level r subproblem, say, $G[V_1]$. Reachability within $G[V_1]$ is computed for the pivots belonging to V_1 and based on this the following actions are made:

- For each pivot p, add the edges vp (resp. pv) to the shortcut set if v reaches (resp. is reached by) p.
- For each $v \in V_1$, give it a set of labels based on its reachability relation to the pivots. That is, for all pivots $p \in V_1$ and $v \in V_1$ (i) if p reaches v, give v the label "p reaches me"; (ii) if v reaches p, give v the label "I reach p"; (iii) otherwise give v the label "I have no relation to p".
- Recurse into $V_{1,1}, V_{1,2}, \ldots \subseteq V_1$ where the $V_{1,i}$'s are an equivalence class in V_1 using the labeling above. That is, $x, y \in V_{1,i}$ iff x and y have the exact same labels.

The aggregate of added shortcuts forms the JLS shortcut set. Below we give a formal description of the algorithm.

JLS

Global Parameters: k is a global parameter to be fixed later. n is the number of vertices in the base input graph (it thus remains fixed through all recursive calls).
Input: A DAG $G = (V, E)$ and a recursion level r.
Output: A shortcut set $H \subseteq V^2$.

1. Randomly and independently sample, with probability $p_r \leftarrow \frac{100k^{r+1}\log n}{n}$, each $v \in V$ as a pivot. Let S be the set of pivots.
2. For each $p \in S$, compute $\overset{\text{reach } p}{R^-}(G,p)$, $\overset{p \text{ reaches}}{R^+}(G,p)$ and:
   - Add vp to H for all $v \in R^-(G,p)$.
   - Add pv to H for all $v \in R^+(G,p)$.
   - Add p^{Anc} label to all $v \in R^-(G,p)$.
   - Add p^{Des} label to all $v \in R^+(G,p)$.
3. $V_1, V_2, \ldots, V_t \leftarrow$ Partition of all $v \in V$ such that $x, y \in V_i$ iff x and y have the exact same labels.
4. Output $H \cup \left(\bigcup_{i\in[t]} \text{JLS on } G[V_i], r+1\right)$.

▶ **Proposition 7** (Paraphrasing Theorem 5 from [11]). *JLS runs in time $\widetilde{O}(mk)$ and produces an $n^{1/2+O(1/\log k)}$-shortcut set of size $\widetilde{O}(nk)$ with probability at least $1 - O(n^{-10})$.*

The above is proved in [11], with the running time and size bounds following quite easily from Proposition 9 (stated later) and Chernoff bounds. Modulo a few key statements, we will give an alternate (and more extendable) proof of the $n^{1/2+O(1/\log k)}$ diameter bound in Section 4.2. For now, let us try to better understand the JLS shortcut set at a high level and, from this, outline a strategy for how to get a better shortcut set.

The Subproblem Tree

The way the diameter of the shortcut set is upper bounded comes from [8]. There, an arbitrary path P is selected for the sake of analysis. Notice first that P is split into contiguous subpaths in the recursive calls and the same is true of its subpaths, and so on. More specifically, observe the following (shown in [11]).

▶ **Observation 8.** *Let P' be a subpath of P, contained in a level r recursive instance G'. If $z-1$ of the non-bridge vertices in $\mathrm{R}(G', P')$ are chosen as pivots at level r, then P' is split into at most z disjoint subpaths $P'_1, \ldots, P'_z$ belonging to distinct level $r+1$ recursive instances $G'_1, \ldots, G'_z$.*

Proof. Let $P' = \langle v_0, v_1, \ldots, v_h\rangle$ and S be the set of pivots selected from $\mathrm{R}(G', P')$. If $p \in S$ reaches v_i, then it reaches v_j for $j > i$. Similarly, if $p \in S$ is reached by v_i, then it is reached by v_j for $j < i$. Thus, if v_i and v_j are related to S in the same way, then $v_i, v_{i+1}, \ldots, v_j$ all have the same relation to S. Since there are at most $z-1$ locations where the reachability relation can change, there are at most z subpaths. ◀

We will examine how P evolves (i.e. is split and shortcutted) through the execution of JLS more carefully. To do this, we think of the *subproblem tree* of P. Each node of the subproblem tree is associated with some subpath of P, and the tree can be described as:

- Root node P. This is the 0th level of the tree.

- For each node P' in level r of the tree (i.e. P' is contained in a level r recursive instance G'), if a bridge of P' in G' is sampled at level r, then P' is a leaf. Note that in this case P' has been shortcut to 2 hops since, denoting $P' = \langle v_1, v_2, \ldots, v_h \rangle$ and the sampled bridge b, the edges $v_0 b$ and $b v_h$ are added to the shortcut set.
- Otherwise, the path P' is split in the instance G' into subpaths $P'_1, \ldots, P'_z$ belonging to level $r+1$ recursive instances $G'_1, \ldots, G'_z$. The node P' will have z children $P'_1, \ldots, P'_z$ at level $r+1$ of the tree.

It is important to note that the algorithm is not aware of this subproblem tree and it is merely a tool for our analysis. Crucially, the number of nodes in this subproblem tree is (up to a constant factor) an upper bound on the number of hops that P is shortcutted to since we can traverse from the start of P to the end along the leaves (which have been shortcutted to 2 hops) and the edges joining the leaves.

Our Strategy

[11] shows that the subproblem tree of any path P has at most $n^{1/2+O(1/\log k)}$ nodes. We employ algorithmic tactics to prune away nodes from this tree in our analysis, to the extent that the tree we analyze has a substantially smaller number of nodes. For example, a node P' in recursive instance G' may have a large subtree in P's subproblem tree. If we are able to identify some structure in G' so that P' is immediately shortcutted to $O(1)$ hops, we are then permitted to ignore the subtree rooted at P' in our analysis.

In view of this, we will use a pruning strategy (called TC-Pruning) which prunes away all nodes P' where $|\mathrm{R}(G', P')|$ is small. Using our new analysis of the diameter of the JLS shortcut set, we will be able to say that the number of nodes P' where $|\mathrm{R}(G', p')|$ is large is much less than $n^{1/2+O(1/\log k)}$, which yields our improvement. See Section 4.2 for details.

We close this subsection with the following crucial lemma from [11] which says that the size of balls around vertices are exponentially decreasing in r, the recursion level.

▶ **Proposition 9** (Paraphrasing Lemma 4.1 from [11]). *With probability at least $1 - O(n^{-10})$, the following event $\mathcal{E}_9$ holds. For all recursion levels r, for all level r recursive instances G',*

$$\left|R^-(G', v)\right| \leq nk^{-r}$$
$$\left|R^+(G', v)\right| \leq nk^{-r}$$

for all $v \in V(G')$.

We omit rewriting a formal proof of Proposition 9, but this follows from a simple induction on the recursion level: if $|R^-(G', v)| \leq nk^{-r}$ at level $r-1$, then it is true also for level r and, otherwise, one of the first nk^{-r} ancestors of v (say p) will be sampled as a pivot with high probability, precluding any of the later ancestors from being retained in $R^-(G', v)$ at level r since they receive the label "I have no relation to p" while v receives the label "p reaches me".

4.2 Bounding the Diameter Achieved From Using TC-Pruning on JLS

In this subsection, we first focus only on the reachability hopbound guarantee for Theorem 2. We will later analyze the size of the shortcut sets and construction time in Section 4.3.

▶ **Theorem 10.** *Let $\varrho \in \mathbb{N}$. The union of $\Theta(\log n)$ independent calls to JLS with TC-Pruning outputs an $n^{1/2+o(1)}/\varrho$-shortcut set with high probability.*

To begin, TC-Pruning uses the *transitive closure* of subgraphs as a subroutine call. We use $\mathrm{TC}(G)$ to denote all edges in the transitive closure of a digraph G; that is, $st \in \mathrm{TC}(G)$ if and only if s can reach t in G. Note that $\mathrm{TC}(G)$ can be computed in $\widetilde{O}(|V(G)|^{\omega})$ time using repeated squaring of the adjacency matrix of G.

The very simple modification to JLS is then described below (the text is mostly JLS from the previous section, with the only substantial change being the addition of the ✏ line); we add edges from the transitive closure of the ball of each pivot if said balls are small.

JLS with TC-Pruning

Global Parameters: k and ϱ are global parameters to be fixed later. n is the number of vertices in the base input graph (it thus remains fixed through all recursive calls).
Input: A DAG $G = (V, E)$ and a recursion level r.
Output: A shortcut set $H \subseteq V^2$.

1. Randomly and independently sample, with probability $p_r \leftarrow \frac{100k^{r+1}\log n}{n}$, each $v \in V$ as a pivot. Let S be the set of pivots.
2. For each $p \in S$, compute $R^-(G,p)$ (reach p), $R^+(G,p)$ (p reaches), $\mathrm{R}(G,p) = R^-(G,p) \cup R^+(G,p)$ and:
   - Add vp to H for all $v \in R^-(G,p)$.
   - Add pv to H for all $v \in R^+(G,p)$.
   - Add p^{Anc} label to all $v \in R^-(G,p)$.
   - Add p^{Des} label to all $v \in R^+(G,p)$.
   - ✏ TC-Pruning: If $|\mathrm{R}(G,p)| \leq \left(k^2 \log^2 n\right)\varrho^2$, add all edges in $\mathrm{TC}(G[\mathrm{R}(G,p)])$ to H.
3. $V_1, V_2, \ldots, V_t \leftarrow$ Partition of all $v \in V$ such that $x, y \in V_i$ iff x and y have the exact same labels.
4. Output $H \cup \left(\bigcup_{i\in[t]} \text{JLS with TC-Pruning on } G[V_i], r+1\right)$.

As before, we select an arbitrary path P and count the nodes of its now pruned subproblem tree. Below, we describe how TC-Pruning allows us to prune the subproblem tree.

TC-Pruning on the Subproblem Tree

Consider a level r node P' in the original subproblem tree satisfying $|\mathrm{R}(G', P')| \leq \left(k^2 \log^2 n\right)\varrho^2$, and which also has children $P'_1, \ldots, P'_z$ arranged in the order so that $P' = P'_1 \ldots P'_z$. Let P'_x be the last subpath that is touched by a call to $\mathrm{TC}(G'[\mathrm{R}(G', p)])$ for some level r pivot p that is a descendant of P', and let P'_y be the first subpath that is touched by a call to $\mathrm{TC}(G'[\mathrm{R}(G', q)])$ for some level r pivot q that is an ancestor of P'. We remove $P'_1, \ldots, P'_x$ and $P'_y, \ldots, P'_z$ (along with the subtrees rooted at them) from the subproblem tree of P, so that P' now has children $P'_{x+1}, \ldots, P'_{y-1}$.

In the above event, $P'_1, \ldots, P'_x$ (resp. $P'_y, \ldots, P'_z$) has been shortcut to 1 hop from the call to $\mathrm{TC}(G'[\mathrm{R}(G', p)])$ (resp. $\mathrm{TC}(G'[\mathrm{R}(G', q)])$) from which a shortcut is added from the start of P'_1 to the end of P'_x (resp. start of P'_y to the end of P'_z). The number of nodes in the pruned subproblem tree is thus, up to a constant factor, an upper bound on the number

of hops P is shortcut to. Moving forward, we will call nodes P' in subproblem G' *small* if $|\mathrm{R}(G', P')| \leq \varrho^2$, and otherwise we call them *large*. There are three basic steps to count the number of nodes in the pruned subproblem tree:

- We show that there is at most $n^{1/2+O(1/\log k)}/\varrho$ large nodes. See Section 4.2.1.
- We show that the number of children each node has is $O(k \log n)$. We can use this to charge the maximal subtrees rooted at small nodes to their parent (a large node). See Section 4.2.2.
- We show that for every small node, its subtree has at most $O(\log n)$ nodes. See Section 4.2.3.

In all, letting L be the number of large nodes, T be an upper bound on the size of subtrees rooted at small nodes, and Δ upper bound the number of children each node has, then for $k = \Theta(\log n)$

$$(\#\text{ of nodes}) \leq L + L\Delta T = n^{1/2+o(1)}/\varrho.$$

See Section 4.2.4.

4.2.1 There Can't be Many Large Nodes (Alternate Proof of the JLS Diameter)

We finally provide the proof of the $n^{1/2+O(1/\log k)}$ diameter bound of JLS. To proceed, we will need the following key lemma as a blackbox.

▶ **Proposition 11** (Paraphrasing of Lemma 4.4 from [11]). *Let P' be an arbitrary path in G', and S be the set of pivots selected from $\mathrm{R}(G', P')$. Suppose S does not contain bridges and S splits P' into (possibly empty) subpaths $P'_1, \ldots P'_{|S|+1}$ belonging respectively to recursive instances $G'_1 \ldots G'_{|S|+1}$. Then*

$$\mathbb{E}\left[\sum_{i=1}^{|S|+1} |\mathrm{R}(G'_i, P'_i)| \;\middle|\; |S|\right] \leq \frac{2}{|S|+1} |\mathrm{R}(G', P')|.$$

While splitting a path P' into $P'_1, P'_2, \ldots, P'_{|S|+1}$ precludes shortcutting P' (hence P) to $o(|S|)$ hops, Proposition 11 says that, on average, it becomes $\Omega(|S|)$ times more likely to select pivots on P' and thus shortcut its descendants in the subproblem tree to $O(1)$ hops each. So even if the algorithm fails to resolve P' by selecting a bridge, it makes progress towards resolving $P'_1, P'_2, \ldots, P'_{|S|+1}$ and it is this advantage that leads to the diameter bound of [11].

Henceforth, we will use the notation $G_{P'}$ to refer to the recursive instance a subpath P' belongs to.

▶ **Lemma 12.** *For any x, let X be the random variable counting the number of nodes P' in the (unpruned) subproblem tree such that $|\mathrm{R}(G_{P'}, P')| \geq x$. Then:*

$$\mathbb{E}[X] \leq \sqrt{\frac{n^{1+O(1/\log k)}}{x}}.$$

Proof. Let T be the (unpruned) subproblem tree.

Potential Function. We use the potential function $\phi(P') = \sqrt{|\mathrm{R}(G_{P'}, P')|}$, and will show that $\mathbb{E}\left[\sum_{\text{tree nodes } P'} \phi(P')\right] \leq n^{1/2+O(1/\log k)}$. If we can show this, we are done since nodes P' with $|\mathrm{R}(G_{P'}, P')| \geq x$ contribute at least $\sqrt{x}$ each to the aforementioned sum; there can thus be at most $X \leq n^{1/2+O(1/\log k)}/\sqrt{x}$ such nodes in expectation.

Local Step. We first show that for any P'

$$\mathbb{E}\left[\sum_{P'' \in \text{children}(P')} \phi(P'')\right] \leq \sqrt{2}\,\mathbb{E}\left[\phi(P')\right]. \tag{❀}$$

The expectations in the following chain of inequalities are conditioned on the value of $\phi(P')$.

$$\begin{aligned}
\mathbb{E}\left[\sum_{P'' \in \text{children}(P')} \phi(P'')\right] &= \sum_{C \in [n]} \Pr\left[|\text{children}\,(P')| = C\right] \mathbb{E}\left[\sum_{P'' \in \text{children}(P')} \phi(P'') \;\middle|\; |\text{children}\,(P')| = C\right] \\
&\leq \sum_{C \in [n]} \Pr\left[|\text{children}\,(P')| = C\right] \sqrt{C} \cdot \mathbb{E}\left[\sqrt{\sum_{P'' \in \text{children}(P')} |\mathrm{R}\,(G_{P''}, P'')|} \;\middle|\; |\text{children}\,(P')| = C\right] \\
&\qquad \text{(Cauchy-Schwarz)} \\
&\leq \sum_{C \in [n]} \Pr\left[|\text{children}\,(P')| = C\right] \sqrt{C \cdot \mathbb{E}\left[\sum_{P'' \in \text{children}(P')} |\mathrm{R}\,(G_{P''}, P'')| \;\middle|\; |\text{children}\,(P')| = C\right]} \\
&\qquad \text{(Jensen's Inequality)} \\
&\leq \sum_{C \in [n]} \Pr\left[|\text{children}\,(P')| = C\right] \sqrt{C \cdot \frac{2}{C} |\mathrm{R}\,(G', P')|} \qquad \text{(Key JLS Lemma: Proposition 11)} \\
&= \sqrt{2}\phi(P').
\end{aligned}$$

Using $\mathbb{E}\left[\mathbb{E}\left[X \mid \phi(P')\right]\right] = \mathbb{E}\left[X\right]$ on both sides gives $\mathbb{E}\left[\sum_{P'' \in \text{children}(P')} \phi(P'')\right] \leq \sqrt{2}\,\mathbb{E}\left[\phi(P')\right]$, establishing ❀.

Summing the Pieces Up. The proof of Lemma 12 is then easily completed by summing up over the nodes of T by levels, and using induction on the level to compute (# expected nodes in level r) $\leq \sqrt{2}^r\,\mathbb{E}\left[\phi(P)\right]$. Note that T (deterministically) has $O(\log n / \log k)$ levels, since the sampling probability of being a pivot at recursion level $\Omega(\log n / \log k)$ is 1 after which the algorithm halts.

$$\begin{aligned}
\mathbb{E}\left[\sum_{P' \in T} \phi(P')\right] &= \sum_{r \in [O(\log n / \log k)]} \mathbb{E}\left[\sum_{P' \text{ in level } r \text{ of } T} \phi(P')\right] \quad (T \text{ has } O(\log n / \log k) \text{ levels}) \\
&\leq \sum_{r \in [O(\log n / \log k)]} \sqrt{2}^r\,\mathbb{E}\left[\phi(P)\right] \qquad (❀) \\
&\leq \sqrt{n} \sum_{r \in [O(\log n / \log k)]} \sqrt{2}^r \qquad (\phi(P) \leq \sqrt{n}) \\
&= n^{1/2 + O(1/\log k)}. \qquad \blacktriangleleft
\end{aligned}$$

As a special case when $x = 1$, we recover the diameter bound of JLS from Lemma 12 since every node P' must have $|\mathrm{R}\,(G_{P'}, P')| \geq 1$. More importantly, for our proof, we will use $x = \varrho^2$, the threshold which separates small nodes from large. By Lemma 12, there are no more than $n^{1/2+o(1)}/\varrho$ large nodes in expectation when $k = \Theta(\log n)$.

4.2.2 Nodes Have Few Children

▶ **Observation 13.** *With probability at least $1 - O(n^{-10})$, the event $\mathcal{E}_{13}$ where every node in the subproblem tree has $O(k \log n)$ children holds.*

Proof. We will condition on $\mathcal{E}_9$ holding, which occurs with probability $1 - O(n^{-10})$ by Proposition 9. Let $P' = \langle v_0, v_1, \ldots, v_h \rangle$ be any level r node in the subproblem tree, contained in some level r recursive graph G'. Since $\mathrm{R}\,(G', P') = R^+(G', v_0) \cup R^-(G', v_h)$ and by $\mathcal{E}_9$, it follows that $|\mathrm{R}\,(G', P')| \leq 2nk^{-r}$.

Let $X_{P'}$ be the number of pivots chosen from $\mathrm{R}(G', P')$ at level r. Since the sampling rate at level r is $100k^{r+1}\log n/n$, we have $\mathbb{E}[X_{P'}] \leq 100k\log n$. By a Chernoff bound, $\Pr[X_{P'} > 200k\log n] \leq O(n^{-12})$. Therefore, $\Pr[\bigcup_{\text{nodes } P'}\{X_{P'} > 200k\log n\}] \leq O(n^{-10})$. This implies, using Observation 8 which bounds the number of pieces P' is split to by the number of pivots, that $\Pr[\mathcal{E}_{13}] \geq 1 - O(n^{-10})$. ◀

4.2.3 Subtrees Rooted at Small Nodes are Heavily Pruned

▶ **Lemma 14.** *Let P' be a small node in the (pruned) subproblem tree, and let $T_{P'}$ be the subtree rooted at it.*

$$\mathbb{E}[|V(T_{P'})|] = O(\log n).$$

Proof. Assume $\mathcal{E}_9$ holds.[6] For any small node P'' at level r, we will call it *bad* if there is some $v \in \mathrm{R}(G_{P''}, P'')$ such that $|\mathrm{R}(G_{P''}, v)| > (k^2\log^2 n)\varrho^2$; otherwise P'' is *good*.

Property of good nodes. Notice that if P'' is good, then it has at most 1 child in the pruned subproblem tree since we run $\mathrm{TC}(G[\mathrm{R}(G_{P''}, p)])$ for any pivot $p \in \mathrm{R}(G_{P''}, P'')$.

Property of bad nodes. If P'' is bad, there is some $v \in \mathrm{R}(G_{P''}, P'')$ such that $|\mathrm{R}(G_{P''}, v)| > (k^2\log^2 n)\varrho^2$. Using

$$(k^2\log^2 n)\,|\mathrm{R}(G_{P''}, P'')| \underset{P'' \text{ small}}{\leq} (k^2\log^2 n)\varrho^2 \underset{P'' \text{ bad}}{<} |\mathrm{R}(G_{P''}, v)| \underset{\mathcal{E}_9}{\leq} nk^{-r},$$

we get $|\mathrm{R}(G_{P''}, P'')| < nk^{-r-2}\log^{-2} n$. Consequently, the expected number of pivots selected from $\mathrm{R}(G_{P''}, P'')$ is at most $O(k^{-1}\log^{-1} n)$. Letting $X_{P''}$ be the number of children P'' has, and using Observation 8 with the expected number of pivots, $\mathbb{E}[X_{P''}] = 1 + O(k^{-1}\log^{-1} n)$.

Total number of nodes. Let Z_r for $r \leq \log n/\log k$ be the number of nodes in level r of T_P. Conditioning on Z_r, the above bound $\mathbb{E}[X_{P''} \mid Z_r] = 1 + O(k^{-1}\log^{-1} n)$ still holds and hence it follows that

$$\begin{aligned}
\mathbb{E}[Z_{r+1}] &= \mathop{\mathbb{E}}_{Z_r}\left[\mathop{\mathbb{E}}_{Z_{r+1}}[Z_{r+1} \mid Z_r]\right] && \text{(Law of iterated expectations)}\\
&= \mathop{\mathbb{E}}_{Z_r}\left[\mathop{\mathbb{E}}_{Z_{r+1}}\left[\sum_{i\in[Z_r]} X_{P''_i} \,\middle|\, Z_r\right]\right] && \text{(where } P''_i \text{ is the } i\text{th node in level } r)\\
&= \mathbb{E}\left[\left(1 + O(k^{-1}\log^{-1} n)\right)\cdot Z_r\right] && (\mathbb{E}[X_{P''} \mid Z_r] = 1 + O(k^{-1}\log^{-1} n))\\
&= \left(1 + O(k^{-1}\log^{-1} n)\right)^{\log n/\log k} && (Z_0 = 1 \text{ and } r \leq \log n/\log k)\\
&= O(1).
\end{aligned}$$

We conclude that $\mathbb{E}[|V(T_{P'})|] = \mathbb{E}\left[\sum_{r\in[\log n/\log k]} Z_r\right] \leq \frac{\log n}{\log k}\cdot O(1) = O(\log n)$. ◀

4.2.4 Putting the Pieces Together

We now have all the components to prove Theorem 10.

[6] The contribution from $\mathcal{E}_9$ not holding is negligible since $\Pr[\text{not } \mathcal{E}_9]\,\mathbb{E}[|V(T_{P'})| \mid \text{not } \mathcal{E}_9] < 1$. This follows from Proposition 9 and $|V(T_{P'})| \leq n$.

Proof of Theorem 10. Let P be any path in G, and T_P be its subproblem tree, and $k = \Theta(\log n)$. We will show later that $\mathbb{E}[|V(T_P)|] \le n^{1/2+o(1)}/\varrho$. Then, by Markov's inequality $\Pr\left[|V(T_P)| \le n^{1/2+o(1)}/\varrho\right] > 1/2$ so that repeating JLS with TC-Pruning $\Theta(\log n)$ times shortcuts P with probability $\Omega(n^{-3})$. Union bounding over $O(n^2)$ paths (one chosen for each pair $s, t \in V$), the algorithm shortcuts all paths with probability $\Omega(n^{-1})$. Let us hence return to showing that $\mathbb{E}[|V(T_P)|] \le n^{1/2+o(1)}/\varrho$.

Assume $\mathcal{E}_{13}$ holds[7]. Let L be the number of large nodes in T_P. We may upper bound $|V(T_P)|$ by

$$\begin{aligned}
\mathbb{E}[|V(T_P)|] &\le \mathbb{E}\left[L + \sum_{\text{large } P'} \sum_{\substack{P'' \in \text{children}(P') \\ : P'' \text{ small}}} |V(T_{P''})|\right] \\
&\le \mathbb{E}[L] + \mathbb{E}\left[L \cdot O(\log^2 n) O(\log n)\right] && \text{(Observation 13 and Lemma 14)} \\
&= \left(1 + O(\log^3 n)\right) \mathbb{E}[L] = n^{1/2+o(1)}/\varrho. && \text{(Lemma 12)}
\end{aligned}$$

◀

4.3 Remaining Analysis of the Shortcut Set

Here we finally prove Theorem 2 in full.

▶ **Theorem 2** (Sequential Shortcut Set). *Let ω be the fast matrix multiplication exponent. There is an $\widetilde{O}(m)$ time randomized algorithm that, given an unweighted digraph G, outputs with high probability a $\sqrt[2\omega-2]{n^{\omega+o(1)}/m}$ -shortcut set H with size $\widetilde{O}(m)$.*

More generally, we show the following tradeoff which the above is a special case of. Let $\varrho \in [\sqrt{n}]$. There is an $\widetilde{O}\left(m + n\varrho^{2\omega-2}\right)$ time randomized algorithm that, given an unweighted digraph G, outputs with high probability an $n^{1/2+o(1)}/\varrho$-shortcut set H with size $\widetilde{O}(n\varrho^2)$.

Proof. With Theorem 10, it only remains to bound the running time of JLS with TC-Pruning and the size of the shortcut set it produces. We bound the contribution from TC-Pruning since the contribution from JLS is taken care of by Proposition 7. Set $k = \Theta(\log n)$. We will condition on the following event:

$$\left\{\forall r \text{ there are } \widetilde{O}(k^{r+1}) \text{ pivots sampled at level } r\right\} \cap \mathcal{E}_9,$$

which occurs with probability at least $1 - O(n^{-10})$ by a Chernoff bound, Proposition 9, and a union bound.

Time. JLS takes $\widetilde{O}(m)$ time by Proposition 7. We next show that the transitive closure calls, from TC-Pruning, takes $\widetilde{O}(n\varrho^{2\omega-2})$ time. Recall that each level r transitive closure call is made in a subgraph induced on $\mathrm{R}(G, p)$ for each level r pivot p, so long as $|\mathrm{R}(G, p)| = \widetilde{O}(\varrho^2)$. This takes $\widetilde{O}(|\mathrm{R}(G, p)|^\omega)$ time per call, using matrix multiplication and repeated squaring of the adjacency matrix. We will break these calls into two cases, based on the level of recursion r.

[7] The contribution from $\mathcal{E}_{13}$ not holding is negligible since $\Pr[\text{not } \mathcal{E}_{13}]\,\mathbb{E}[|V(T_P)| \mid \text{not } \mathcal{E}_{13}] < 1$. This follows from Observation 13 and $|V(T_P)| \le n$.

- Case 1: $k^r < n/\varrho^2$. Since there are $\widetilde{O}(k^r)$ pivots sampled at recursion level r and below, the time at that level is, up to polylogarithmic factors, $k^r(\varrho^2)^\omega < n\varrho^{2\omega-2}$. Since there are $O(\log n/\log\log n)$ levels, the bound follows.
- Case 2: $k^r \geq n/\varrho^2$. Since there are $\widetilde{O}(k^r)$ pivots sampled at recursion level r and $\mathcal{E}_9$ holds, the time at that level is, up to polylogarithmic factors,

$$k^r\left(\frac{n}{k^r}\right)^\omega = \frac{n^\omega}{k^{r(\omega-1)}} \leq \frac{n^\omega}{(n/\varrho^2)^{\omega-1}} = n\varrho^{2\omega-2}.$$

Since there are $O(\log n/\log\log n)$ levels, the bound follows.

Size. The argument for this is similar to the time bound.

- Case 1: $k^r < n/\varrho^2$. Since there are $\widetilde{O}(k^r)$ pivots sampled at recursion level r, the number of shortcuts added at that level is, up to polylogarithmic factors, $k^r(\varrho^2)^2 < n\varrho^2$. Since there are $O(\log n/\log\log n)$ levels, the bound follows.
- Case 2: $k^r \geq n/\varrho^2$. Since there are $\widetilde{O}(k^r)$ pivots sampled at recursion level r and $\mathcal{E}_9$ holds, the number of shortcuts added at that level is, up to polylogarithmic factors,

$$k^r\left(\frac{n}{k^r}\right)^2 = \frac{n^2}{k^r} \leq \frac{n^2}{n/\varrho^2} = n\varrho^2.$$

Since there are $O(\log n/\log\log n)$ levels, the bound follows.

Main case ($\widetilde{O}(m)$ size shortcut set). Set $\varrho = (m/n)^{1/(2\omega-2)}$. ◀

5 Parallel Reachability

In this short section, we parallelize JLS with TC-Pruning, the shortcut set construction shown in Section 4, proving Theorem 16. We then use the shortcut set to prove Theorem 1, our main result for parallel reachability.

We use the following result from [9] to reduce the construction of shortcut sets on digraphs to that on so-called shallow digraphs.

▶ **Proposition 15** (Paraphrasing Corollary 3.2 from [9]). *Suppose $\lambda > c_0 \log^3 n$ and $\alpha < 1/(c_0 \log^2_\lambda n)$ where c_0 is a sufficiently large constant.*

Suppose there is a parallel algorithm $\mathcal{A}_0$ that, given a digraph G_0 with n vertices and m_0 edges and reachability hopbound $h_0 = \lambda\beta$, returns a β-shortcut set of size $\alpha m_0 + f(n)$.

Then there is a randomized parallel algorithm $\mathcal{A}$ that, given a digraph G with n vertices and m edges, returns a β-shortcut set of size $S(m) = \widetilde{O}(\alpha m + f(n))$. $\mathcal{A}$ makes a polylogarithmic number of sequential calls to $\mathcal{A}_0$ on digraphs with at most $S(m) + m$ edges, and takes an additional $\widetilde{O}(m)$ work and $\widetilde{O}(h_0)$ span.

We are now ready to parallelize JLS with TC-Pruning.

▶ **Theorem 16** (Parallel Near Linear Work Shortcut Set Construction). *Let ω be the fast matrix multiplication exponent in the work for parallel algorithms using polylogarithmic (in n) span. There is an $\widetilde{O}(m)$ work and $\sqrt[2\omega-2]{n^{\omega+o(1)}/m}$ span randomized parallel algorithm that, given an unweighted digraph G, outputs with high probability a $\sqrt[2\omega-2]{n^{\omega+o(1)}/m}$-shortcut set H with size $\widetilde{O}(m)$.*

Proof.

Implementing $\mathcal{A}_0$ (Part 1) – Parallel reduction to shallow DAGs.[8] Observe that we can use the algorithm of [18] with a parallel BFS oracle to find the SCCs of G_0 in $\widetilde{O}(m)$ work and $\widetilde{O}(h_0)$ span. To see this, note that the algorithm of [18] recurses into graphs induced on intervals of the topological order of the SCCs of G_0, hence each recursive instance maintains a h_0 reachability hopbound.

By adding a bidirected star in each SCC of G_0 to our shortcut set, it then suffices to construct a $\beta/2$-shortcut set for the DAG where the SCCs of G_0 are contracted. This incurs at most a factor of two dilation to give a β-shortcut set for G_0. We henceforth assume G_0 is a DAG with reachability hopbound h_0.

Implementing $\mathcal{A}_0$ (Part 2) – Parallel JLS with TC-Pruning on shallow DAGs. Observe that for any $h_0 \gg n^{1/2+o(1)}/\varrho$, JLS with TC-Pruning runs in work $\widetilde{O}\left(m + n\varrho^{2\omega-2}\right)$ and span $\widetilde{O}(h_0)$ for DAGs with reachability hopbound h_0.

To see this, first note that the h_0 reachability hopbound is preserved in all recursive subgraphs: if an arbitrary pair $u \preceq v$ are sent to the same recursive instance G', then every vertex between u and v have the same reachability relations (to the pivots) as u, v and are thus also sent to G'. In particular, the h_0 hop path from u to v is sent to G'. The reachability relations, which are computed by parallel BFS calls, are therefore done in $\widetilde{O}(|E(G')|)$ work and $\widetilde{O}(h_0)$ span in each recursive instance G'. Next, note that each transitive closure call is done in $\widetilde{O}(\varrho^{2\omega})$ work and polylogarithmic in n span.

Setting parameters. Fix $\varrho = (m_0/n)^{1/(2\omega-2)}/\mathrm{polylog}(n)$ for a sufficiently large power in the polylogarithm term. Let the output of JLS with TC-Pruning on shallow DAGs be a β-shortcut set H with $\beta = n^{1/2+o(1)}/\varrho = \sqrt[2\omega-2]{n^{\omega+o(1)}/m_0}$. Note that for this setting of ϱ, we have $|H| = \widetilde{O}(n\varrho^2) = \alpha m_0$ where $\alpha < 1/(c_0 \log_\lambda^2 n)$. Then, pulling back to shallow digraphs, note that $f(n)$ accounts for the edges added by the bidirected stars and hence $f(n) \le n$ and $S(m) = \widetilde{O}(m)$.

Putting things together. Invoking Proposition 15, $\mathcal{A}$ outputs with high probability a $\sqrt[2\omega-2]{n^{\omega+o(1)}/m}$ -shortcut set with size $\widetilde{O}(m)$ in $\widetilde{O}(m)$ work and $\sqrt[2\omega-2]{n^{\omega+o(1)}/m}$ span. ◀

▶ **Theorem 1** (Parallel Reachability). *Let ω be the fast matrix multiplication exponent in the work for parallel algorithms using polylogarithmic (in n) span. There is an $\widetilde{O}(m)$ work and $\sqrt[2\omega-2]{n^{\omega+o(1)}/m}$ span randomized parallel algorithm that, given an unweighted digraph G and source $s \in V$, outputs with high probability all vertices that s can reach and all vertices that reach s.*

Proof. This follows from using Theorem 16 to extract a $\sqrt[2\omega-2]{n^{\omega+o(1)}/m}$ -shortcut set H of G with size $\widetilde{O}(m)$ in $\widetilde{O}(m)$ work and $\sqrt[2\omega-2]{n^{\omega+o(1)}/m}$ span. Then, run a parallel BFS on $G \cup H$ from s in $\widetilde{O}(m)$ work and $\sqrt[2\omega-2]{n^{\omega+o(1)}/m}$ span. ◀

References

1 Josh Alman, Ran Duan, Virginia Vassilevska Williams, Yinzhan Xu, Zixuan Xu, and Renfei Zhou. More asymmetry yields faster matrix multiplication. In *Proceedings of the 2025 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 2005–2039. SIAM, 2025. doi:10.1137/1.9781611978322.63.

[8] This part may be used for any shortcut set algorithm, hence one may assume that $\mathcal{A}_0$ is given a shallow DAG (as opposed to a shallow digraph).

2 Ben Bals, Joakim Blikstad, Greg Bodwin, Daniel Dadush, Sebastian Forster, and Yasamin Nazari. Greedy algorithms for shortcut sets and hopsets. *arXiv preprint arXiv:2511.20111*, 2025. doi:10.48550/arXiv.2511.20111.

3 Aaron Bernstein and Nicole Wein. Closing the gap between directed hopsets and shortcut sets. In *Proceedings of the 2023 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 163–182. SIAM, 2023. doi:10.1137/1.9781611977554.CH7.

4 Jan van den Brand, Hossein Gholizadeh, Yonggang Jiang, and Tijn de Vos. Parallel minimum cost flow in near-linear work and square root depth for dense instances. *arXiv preprint arXiv:2503.13274*, 2025. doi:10.48550/arXiv.2503.13274.

5 Nairen Cao and Jeremy T Fineman. Parallel exact shortest paths in almost linear work and square root depth. In *Proceedings of the 2023 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 4354–4372. SIAM, 2023. doi:10.1137/1.9781611977554.CH166.

6 Nairen Cao, Jeremy T Fineman, and Katina Russell. Efficient construction of directed hopsets and parallel approximate shortest paths. In *Proceedings of the 52nd Annual ACM SIGACT Symposium on Theory of Computing*, pages 336–349, 2020. doi:10.1145/3357713.3384270.

7 Nairen Cao, Jeremy T Fineman, and Katina Russell. Brief announcement: An improved distributed approximate single source shortest paths algorithm. In *Proceedings of the 2021 ACM Symposium on Principles of Distributed Computing*, pages 493–496, 2021. doi:10.1145/3465084.3467945.

8 Jeremy T Fineman. Nearly work-efficient parallel algorithm for digraph reachability. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing*, pages 457–470, 2018. doi:10.1145/3188745.3188926.

9 Bernhard Haeupler, Yonggang Jiang, and Thatchaphol Saranurak. Reducing shortcut and hopset constructions to shallow graphs. In *2026 SIAM Symposium on Simplicity in Algorithms (SOSA)*, pages 385–393. SIAM, 2026. doi:10.1137/1.9781611978964.30.

10 Joseph JáJá. *Parallel algorithms*. Addison Wesley, 1992.

11 Arun Jambulapati, Yang P Liu, and Aaron Sidford. Parallel reachability in almost linear work and square root depth. In *2019 IEEE 60th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 1664–1686. IEEE, 2019. doi:10.1109/FOCS.2019.00098.

12 Philip N Klein and Sairam Subramanian. A randomized parallel algorithm for single-source shortest paths. *Journal of Algorithms*, 25(2):205–220, 1997. doi:10.1006/JAGM.1997.0888.

13 Shimon Kogan and Merav Parter. Beating matrix multiplication for $n^{1/3}$-directed shortcuts. In *49th International Colloquium on Automata, Languages, and Programming (ICALP 2022)*, volume 229 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 82:1–82:20. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022. doi:10.4230/LIPIcs.ICALP.2022.82.

14 Shimon Kogan and Merav Parter. New diameter-reducing shortcuts and directed hopsets: Breaking the barrier. In *Proceedings of the 2022 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1326–1341. SIAM, 2022. doi:10.1137/1.9781611977073.55.

15 Shimon Kogan and Merav Parter. Faster and unified algorithms for diameter reducing shortcuts and minimum chain covers. In *Proceedings of the 2023 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 212–239. SIAM, 2023. doi:10.1137/1.9781611977554.CH9.

16 Shimon Kogan and Merav Parter. Towards Bypassing Lower Bounds for Graph Shortcuts. In *31st Annual European Symposium on Algorithms (ESA 2023)*, volume 274 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 73:1–73:16. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2023. doi:10.4230/LIPIcs.ESA.2023.73.

17 Václav Rozhoň, Bernhard Haeupler, Anders Martinsson, Christoph Grunau, and Goran Zuzic. Parallel breadth-first search and exact shortest paths and stronger notions for approximate distances. In *Proceedings of the 55th Annual ACM Symposium on Theory of Computing*, pages 321–334, 2023. doi:10.1145/3564246.3585235.

18 Warren Schudy. Finding strongly connected components in parallel using $O(\log^2 n)$ reachability queries. In *Proceedings of the twentieth annual symposium on Parallelism in algorithms and architectures*, pages 146–151, 2008. doi:10.1145/1378533.1378560.

19 JD Ullman and M Yannakakis. High-probability parallel transitive-closure algorithms. *SIAM Journal on Computing (Society for Industrial and Applied Mathematics);(United States)*, 20(1), 1991. doi:10.1137/0220006.