New Parameterized Algorithms for APSP in Directed Graphs

Ely Porat[1], Eduard Shahbazian[2], and Roei Tov[3]

1 **Department of Computer Science, Bar-Ilan University, Ramat-Gan, Israel**
porately@gmail.com
2 **Department of Computer Science, Bar-Ilan University, Ramat-Gan, Israel**
shediyo@gmail.com
3 **Department of Computer Science, Bar-Ilan University, Ramat-Gan, Israel**
roei81@gmail.com

Abstract

All Pairs Shortest Path (APSP) is a classic problem in graph theory. While for general weighted graphs there is no algorithm that computes APSP in $O(n^{3-\varepsilon})$ time ($\varepsilon > 0$), by using fast matrix multiplication algorithms, we can compute APSP in $O(n^{\omega} \log n)$ time ($\omega < 2.373$) for undirected unweighted graphs, and in $O(n^{2.5302})$ time for directed unweighted graphs. In the current state of matters, there is a substantial gap between the upper bounds of the problem for undirected and directed graphs, and for a long time, it is remained an important open question whether it is possible to close this gap.

In this paper we introduce a new parameter that measures the *symmetry* of directed graphs (i.e. their closeness to undirected graphs), and obtain a new parameterized APSP algorithm for directed unweighted graphs, that generalizes Seidel's $O(n^{\omega} \log n)$ time algorithm for undirected unweighted graphs. Given a directed unweighted graph G, unless it is highly asymmetric, our algorithms can compute APSP in $o(n^{2.5})$ time for G, providing for such graphs a faster APSP algorithm than the state-of-the-art algorithms for the problem.

1998 ACM Subject Classification G.2.2 Graph Theory

Keywords and phrases Graphs, distances, APSP, fast matrix multiplication

Digital Object Identifier 10.4230/LIPIcs.ESA.2016.72

1 Introduction

All Pairs Shortest Path (APSP) has a long history, emerging from the 1950's to our present days. In APSP our goal is to compute the distances between all pairs of vertices in the graph. For general directed weighted graphs with n vertices, the algorithm of Floyd-Warshall [6] computes APSP in $O(n^3)$ time. For sparse graphs with m edges, we can obtain an improved algorithm that runs in $O(mn + n^2 \log n)$ time, by first finding and eliminating cycles of negative weight, using Johnson's algorithm [15], and then executing Dijkstra algorithm [6] (implemented with Fibonacci heaps [8]) from each vertex in the graph. This classic result was later improved by Pettie [16] to $O(mn + n^2 \log \log n)$, and for undirected graphs with nonnegative edge weights, APSP can be computed in $O(mn)$ time, using Thorup's algorithm for single source shortest path [24].

All the algorithms mentioned above have a running time of $O(n^3)$ for dense graphs. Fredman was the first to break the $O(n^3)$ (cubic) time barrier, by obtaining an algorithm with a running time of $O(n^3/(\log \log n/\log n)^{1/3})$ [7]. Since then, a line of subsequent improvements to Fredman's algorithm followed (e.g. [22, 12, 23, 27, 13, 3], etc.), where

the current best result for the problem was recently obtained by Chan and Williams [4], which showed an algorithm that computes APSP in $O(n^3/2^{\Omega(\log n)^{1/2}})$ time. Nevertheless, these algorithms only provides a slightly improvement to the classic cubic-time algorithm of Floyd and Warshall, and it remains an open question whether an $O(n^{3-\varepsilon})$-time algorithm for APSP exists. The hardness of APSP and other fundamental graph and matrix problems (e.g. minimum weight cycle, replacement paths on directed weighted graphs, etc.) that are all known to have cubic time algorithms but no $O(n^{3-\varepsilon})$-time algorithms, might be explained by the result of V. Vassilevska Williams and R. Williams [25], which proved that these problems are subcubic equivalent, meaning, either all of these problems can be solved in $O(n^{3-\varepsilon})$ time or none of them can be.

By using Fast Matrix Multiplication (FMM) algorithms, truly subcubic time (i.e. $O(n^{3-\varepsilon})$) algorithms can be obtained for unweighted graphs and for graphs with small integer edge-weights. The naïve algorithm for multiplying two $n \times n$ matrices runs in $O(n^3)$ time, however, there exist faster algorithms to compute matrix multiplication (e.g. Strassen's algorithm [6], Coppersmith-Winograd [5]). Denote ω to be the exponent of square matrix multiplication, currently $\omega < 2.373$ ([11]) is the smallest known value for ω. Notice, it is straightforward to reduce Boolean Matrix Multiplication (BMM) to FMM in $O(n^2)$ time, hence BMM can be also computed in $O(n^\omega)$ time. Given this fact, many APSP algorithms use the following basic property: let G be an unweighted graph with adjacency matrix A and let $M = A^k$, it follows $M[i,j] = 1$ if and only if there is a shortest path in G from i to j of length at most k. Alon, Galil and Margalit [2] were the first to obtain a truly subcubic algorithm for APSP. They showed an algorithm that computes APSP in $\widetilde{O}(n^{(\omega+3)/2}) = \widetilde{O}(n^{2.69})$ time[1] for directed graphs with edge-weights from $\{-1, 0, 1\}$. Zwick in [26] improved Alon, Galil, and Margalit's result [2]. He showed that using fast *rectangular* matrix multiplication algorithms (current fastest rectangular matrix multiplication algorithm is due to Le Gall [10]), APSP for directed graphs with edges of weights $\{-M, \ldots, M\}$ can be computed in $O(M^{0.68}n^{2.53})$ time. For undirected graphs with edges of small integers weights $\{1, \ldots, M\}$, Galil and Margalit [9] showed an algorithm with a running time of $\widetilde{O}(M^{(\omega+1)/2}n^\omega)$. This result was later improved by Shoshan and Zwick [21] to $\widetilde{O}(Mn^\omega)$. Considering the case of undirected unweighted graphs, Seidel [20] obtained an algorithm for APSP that runs in $\widetilde{O}(n^\omega)$ time. The advantage of Seidel's $\widetilde{O}(n^\omega)$ time algorithm is that it is much simpler than that of [9].

Currently, for unweighted graphs, there is a large gap in the upper bounds for directed and undirected graphs. Many believe that $\omega = 2+o(1)$, and if this is indeed the case, then the $\widetilde{O}(n^\omega)$ algorithms for undirected (e.g. Seidel's algorithm) match, up to logarithmic factors, the natural $O(n^2)$-time lower bound for the problem. On the other hand, considering this case for directed graphs, both the algorithm of Alon, Galil and Margalit, and the algorithm of Zwick run in $\widetilde{O}(n^{2.5})$ time. Moreover, the improved result of Zwick relies on the fact, that currently there exist faster algorithms for rectangular matrix multiplication (e.g. [10]) than square rectangular matrix multiplication. Actually, if only using square matrix multiplication, the over twenty years old result of Alon, Galil and Margalit is still the best we know so far. Also, currently, the only known way to achieve $\widetilde{O}(n^\omega)$-time APSP algorithm for directed graphs is to settle with an *approximation*: Zwick showed in [26] an algorithm that computes a $(1+\varepsilon)$ approximation to APSP in $\widetilde{O}(\frac{1}{\varepsilon}n^\omega \log \frac{1}{\varepsilon})$ time. This raises an interesting open question, whether an $\Omega(n^{2.5})$-time is the lower bound to compute APSP for directed graphs.

So what are the obstacles that prevent us from obtaining an $\widetilde{O}(n^\omega)$-time algorithm for directed unweighted graphs as well? Consider Seidel's algorithm. The idea of this algorithm

[1] The notation $\widetilde{O}$-notation suppress polylogarithmic factors from the O-notation.

is fairly simple: let $G = (V, E)$ be a directed graph with adjacency matrix A, and assume we have computed recursively the distance matrix for the graph G' induced by A^2. Recall that in G', an edge (u, v) exists if and only if there is a path of length at most 2 from u to v. This implies that either $d_G(u, v) = 2d_{G'}(u, v)$, if $d_G(u, v)$ is even, or $d_G(u, v) = 2d_{G'}(u, v) - 1$, otherwise. Since in each recursion the distances in the induced graph is cut by half, the depth of the recursion is $O(\log n)$. The only thing that is left in order to complete the algorithm, is to determine for every $u, v \in V$, if $d_G(u, v)$ is even or odd. Let $N(v)$ be the set of neighbors of v in G. For every $w \in N(v)$, by the triangle inequality for unweighted *undirected* graphs,

$$d(u, v) - 1 \leq d(u, w) \leq d(u, v) + 1$$

Using this property, it is not hard to verify that $\sum_{w \in N(v)} d_{G'}(u, w) \geq |N(v)| \cdot d_{G'}(u, v)$ when $d_G(u, v)$ is even and $\sum_{w \in N(v)} d_{G'}(u, w) < |N(v)| \cdot d_{G'}(u, v)$ otherwise. Since the sums $\sum_{w \in N(v)} d_{G'}(u, w)$, for every $u, v \in V$, can be computed at once using a single matrix multiplication, the total computation time of a recursion stage is $O(n^\omega)$ (for more details, see [20]). The sole obstacle that prevents us from implementing Seidel's algorithm for directed graphs is that the triangle inequality in *directed* unweighted graphs holds only for one side: for an incoming edge (w, v), it only holds that $d(u, v) - 1 \leq d(w, u)$ (similarly, for an outgoing edge (v, w), we have $d(u, w) \leq d(u, v) + 1$). For an incoming edge (w, v), we can get two-sides bounds for $d(u, w)$, if we have a path from v to w. Specifically, if for an incoming edge (w, v), it holds that $d(v, w) \leq d$ (for some d), then $d(u, w) \leq d(u, v) + d$ (similar argument can be applied for an outgoing edge as well). In other words, for any $u, v \in V$, the more an edge (w, v) is "symmetric" (i.e. $d(v, w)$ is close to 1), the better bounds we get from the triangle inequality on $d(u, w)$, using $d(u, v)$. As by definition, undirected graphs are fully-symmetric, we have that $d(v, u) = 1$ for every edge (u, v) in an undirected unweighted graph. On the other hand, in a directed graph, for an edge (w, v), it might be the case that the graph does not have any path at all from v to w, thus we cannot guarantee a two-sides bound by the triangle inequality for $d(u, w)$ as in unweighted graphs.

In this paper we introduce a new parameter for directed strongly-connected graphs that measures the closeness of a directed strongly-connected graph to symmetric graph (i.e. undirected). The symmetry parameter $s(G)$ of a directed strongly-connected graph $G = (V, E)$ is defined to be $\max_{(u,v) \in E}\{d(v, u)\}$. Interestingly, the definition of the symmetry parameter is very similar to the definitions of the *girth* and *diameter*, both natural and well-studied (e.g. [1, 14, 17, 18, 19]) distance-related graphs parameters: the girth of directed graphs is $\min_{(u,v) \in E}\{d(v, u)\}$, and the diameter of the graph is $\max_{(u,v) \in V \times V}\{d(u, v)\}$. As we shall see later, similarly to the girth and the diameter, we can also compute the symmetry parameter of the graph in $\widetilde{O}(n^\omega)$ time.

Our main contribution in this paper is an algorithm (Theorem 8) that provides a non-trivial generalization of Seidel's algorithm, and computes APSP for a directed strongly-connected graph G with symmetry parameter $s = s(G)$ in $O(s \cdot n^\omega \log_{s+1} n)$ time. The rough idea of the algorithm is as follows. As discussed above, using the symmetry parameter, we can obtain generalized triangle inequalities. In our algorithm we use these new inequalities, and adapt Seidel's algorithm to find all the pairs $(u, v) \in V \times V$, such that $d(u, v) \equiv 1 \mod (s + 1)$, and their distances. However, it is unclear how to deduce the distances for all other pairs of vertices, i.e., for all $(u, v) \in V \times V$, such that $d(u, v) \not\equiv 1 \mod (s + 1)$. Our approach to tackle this obstacle is to find, using one BMM, all the pairs $(u, v) \in V$ such that $d(u, v) \equiv i \mod (s + 1)$, based on that we already know all the pairs $(u, v) \in V$, such that $d(u, v) \equiv j \mod (s+1)$, for all $0 < j < i$. Surprisingly, here comes the parameter into action again. Using a designated matrix multiplication, the parameter allows us to find all the relevant pairs and

their distances. Continuing this process inductively, we can compute the distances between all pairs of vertices. Our algorithm can also be applied on general directed unweighted graphs, by using an $\widetilde{O}(n^{\omega})$-time reduction from APSP for general directed unweighted graphs to APSP for directed strongly-connected unweighted graphs.

In the definition of the symmetry parameter, we are taking the maximum over all the edges, thus, even a single edge in a graph can cause the symmetry parameter of the graph to be very large. However, this can be relaxed by the concept of *violating-edge*. Let $z < n$ be some threshold value. An edge (u, v) is a z-violating edge if $d(v, u) > z$. By applying Breadth-First Search (BFS) in and out of the endpoints of all z-violating edges in the graph, we can remove these edges, and by that, decrease the symmetry parameter of the new graph to be as small as our threshold value z. Now, the distance from u to v in the original graph is either their distance in the new graph, or otherwise, equals to $d(u, x) + 1 + d(y, v)$, where (x, y) is some z-violating edge. Using this idea, for any graph that has at most $o(n^{0.53})$ violating edges for a $o(n^{0.157})$-threshold, we can compute APSP faster than the state of the art algorithm for the problem. Notice that a larger threshold is allowed as ω decreases. For example, if $\omega = 2 + o(1)$, then our APSP runs in $o(n^{2.5})$ time for any graph that has at most $o(n^{1/2})$ violating edges for any $o(n^{1/2})$-threshold. This may suggest the following strategy to compute APSP for directed unweighted graphs. First compute in $\tilde{O}(n^{\omega})$ time (Lemma 13) the number of violating-edges for any $o(n^{0.157})$-threshold. If there are at most $o(n^{0.53})$ violating edges, use our algorithm, otherwise use the state of the art algorithm for the problem.

Another interesting property of our algorithm is that it provides an improvement over a basic parameterized-APSP algorithm for directed unweighted graphs, that is also used sometimes as an ingredient in other algorithms for the problem (e.g. in [2]). For a directed unweighted graph G with adjacency matrix A and *diameter* $D \leq n$, we can easily compute APSP in $O(D \cdot n^{\omega})$ by computing $A, A^2, A^3, \ldots, A^D$. For a large diameter, the running time of this algorithm can be as high as $O(n^{1+\omega})$ (which is even worse than Floyd-Warshall's algorithm), however, it can be turned to be useful for graphs that their diameter is less than $n^{2.5-\omega}$, in this case, the running time of this algorithm will be $o(n^{2.5})$. Our algorithms provide improvements over this basic algorithm in two ways. First they provide a weaker constraint on the graph, as $s(G) = \max_{(u,v)\in E}\{d(v, u)\} \leq \max_{(u,v)\in V\times V}\{d(u, v)\} = D$. Also, using the concept of z-violating edges, we can decrease the value of the symmetry parameter, while, to our best knowledge, there is no such equivalent method for the parameterized-APSP algorithm that is based on the diameter.

The paper is organized as follows. In the next section we provide some preliminaries for our algorithms. Section 3 presents our main algorithm. In Section 4 we show a reduction that allows us to compute APSP on general directed graphs using our parameterized algorithm. In Section 5 we give a hybrid algorithm that allows us to reduce the size of the symmetry parameter of the graph in exchange to additional BFS executions on the graph. We also show in this section how to compute in $\widetilde{O}(n^{\omega})$ time the symmetry parameter of a graph and the violating edges in the graph for some specific threshold.

2 Preliminaries

Let $G = (V, E)$ be a directed graph with n vertices and m edges. Let $u, v \in V$ and let $d_G(u, v)$ be the length of the shortest path from u to v in G. If there is no path from u to v in G, then $d_G(u, v) = \infty$. We use a simplified notation $d(u, v)$, when the referred graph is clear from the context. A directed graph is strongly-connected if for all $u, v \in V$, $d(u, v) < \infty$.

Our goal in this paper is to compute efficiently All Pairs Shortest Path (APSP) for directed unweighted graphs. The general form of the problem is defined as follows.

▶ **Definition 1** (The APSP Problem)**.** Let $G = (V, E)$. Compute a matrix M of size $|V| \times |V|$, such that for every $u, v \in V$, $M(u, v) = d(u, v)$.

The diameter of a graph G is $D(G) = \max_{u,v \in V}\{d(u, v)\}$. Notice that a directed graph G is strongly-connected if and only if $D(G) < \infty$. An extended definition of the diameter that includes non-necessarily strongly-connected graphs (with non-empty edges set) is $\max_{u,v \in V}\{d(u, v) \mid d(u, v) < \infty\}$. Let $v \in V$, $N_{in}(v)$ is the set of all vertices that have an outgoing edge to v, namely, $N_{in}(v) = \{u \mid (u, v) \in E\}$.

The *symmetry parameter* introduced here is a new notion that measures the closeness of a directed strongly-connected graph to a symmetric (undirected) graph. We define the symmetry parameter of an edge (u, v) as $d(v, u)$. The symmetry parameter of a graph is the maximum over the symmetry parameters of all the edges in the graph.

▶ **Definition 2** (Directed Graphs, Symmetry Parameter)**.** Let $G = (V, E)$ be a directed strongly-connected graph. $s(G) = \max_{(u,v) \in E}\{d(v, u)\}$.

Notice that in undirected graphs $s(G) = 1$. Since the definition of the symmetry parameter of the graph takes the maximum over all the edges, we may achieve a smaller value for the parameter if we exclude some of the edges in the graph. This idea is encapsulated in the definition of z-violating edges.

▶ **Definition 3** (z-violating Edges)**.** Let $G = (V, E)$ be a directed strongly-connected graph and let $z < s(G)$ be a threshold value. An edge $(u, v) \in E$ is z-violating if $d(v, u) > z$.

We denote by A_G, the adjacency matrix of a graph G. In A_G, it holds for every $u, v \in V$ that:

$$A_G(u, v) = \begin{cases} 1 & (u, v) \in E \vee u = v \\ 0 & (u, v) \notin E \end{cases}$$

Note that for undirected graphs the adjacency matrix is symmetric.

Let A and B be two $n \times n$ matrices with integral values. We denote $A \cdot B$ as the integer multiplication of A and B. We use the same notation, when A and B are Boolean matrices and the operation is Boolean Matrix Multiplication (BMM).We can multiply two integer/Boolean metrics in $O(n^\omega)$ time, where currently $\omega < 2.373$ [10].

We use the notation A^k $(k > 0)$ for the multiplication of A by itself k times. For the rest of the paper, when using A^k, we assume the multiplication that is taken is BMM. Naïvely, we can compute A^k in $O(k \cdot n^\omega)$, however this can be done in $O(n^\omega \log n)$ time using "repeated squaring" method (e.g. [14]). The matrix that is obtained from multiplying the adjacency matrix of a graph by itself is meaningful in respect to the distances of the graph. This relation is given in the following proposition.

▶ **Proposition 4** (Distances vs. the Adjacency Matrix [14])**.** *Let $D_k = (A_G)^k$. $D_k(u, v) = 1$ if and only if $d(u, v) \leq k$.*

Denote G_k to be the graph induced from the adjacency matrix D_k. The graph G_k is the graph G with some additional new edges. Specifically, G_k has a new edge (u, v), if $(u, v) \notin G$ and there exists a shortest path in G from u to v of length at most k. In other words, in G_k we add "shortcuts edges" to paths of length at most k in G. This leads us to the following definition: let $k > 1$ and $u, v \in V$, we denote r_{uv} to be $(d_G(u, v) \mod k)$, that is, the residues of $d(u, v)/k$.

3 Parameterized Algorithm for APSP

In this section we show in Theorem 8 how to compute APSP for a directed strongly-connected graph G in $O(s \cdot n^{\omega} \log_{s+1} n)$ time, where $s = s(G)$ is the symmetry parameter of G. Before we turn to prove the theorem, we give some lemmas that are needed for our theorem.

In undirected graphs, for vertices u, v and a neighbor w of v, using the triangle inequality, we have $d(u,v) - 1 \leq d(u,w) \leq d(u,v) + 1$. Considering the same situation in directed graphs, however, we can only guarantee either that $d(u,v) - 1 \leq d(u,w)$, where (w,v) is an incoming edge of v, or that $d(u,w) \leq d(u,v) + 1$, where (v,w) is an outgoing edge of v. With the symmetry parameter we can obtain both the following lower and upper bounds on $d(u,w)$.

▶ **Lemma 5.** *Let $G = (V,E)$ be a directed, strongly-connected, unweighted graph. Then:*

1. *For every $u, v \in V$ and every $w \in N_{in}(v)$, it follows that $d(u,v) - 1 \leq d(u,w) \leq d(u,v) + s(G)$.*
2. *For every $u, v \in V$ $(u \neq v)$ there exists a $w \in N_{in}(v)$ such that $d(u,w) = d(u,v) - 1$.*

Proof. Let $u, v \in V$. If $w \in N_{in}(v)$, then $d(w,v) = 1$, and from the definition of the symmetry parameter $d(v,w) \leq s(G)$. From the triangle inequality on the pair (u,v), we get that $d(u,v) \leq d(u,w) + d(w,v) \leq d(u,w) + 1$, that is, $d(u,v) - 1 \leq d(u,w)$. From the triangle inequality on the pair (u,w), we get that $d(u,w) \leq d(u,v) + d(v,w) \leq d(u,v) + s(G)$. This proves the first part of the lemma.

For the second part, consider a shortest path P from u to v, and let w be the vertex that immediately preceding v on P. Since $u \neq v$, we have that $w \neq v$, and therefore $w \in N_{in}(v)$. Since w is on a shortest path from u to v, $d(u,v) = d(u,w) + 1$, and thus $d(u,w) = d(u,v) - 1$. ◀

The next lemma shows the relation between the distances in the original graph G and the graph that is induced by $A' = (A_G)^x$.

▶ **Lemma 6.** *Let $G = (V,E)$ be a directed graph and let $0 < x \leq n$ be an integer. Let G' be the graph induced from $A' = (A_G)^x$. Let $r_{uv} = d_G(u,v) \mod x$, for every $u, v \in V$ it follows that:*

1. $d_{G'}(u,v) = \lceil \frac{d_G(u,v)}{x} \rceil$.
2. *If $r_{uv} = 0$:* $d_G(u,v) = x \cdot d_{G'}(u,v)$.
3. *If $r_{uv} \neq 0$:* $d_G(u,v) = x \cdot (d_{G'}(u,v) - 1) + r_{uv}$.

Proof. Let $u, v \in V$, such that $d = d_G(u,v)$. Notice that $d_G(u,v) = x \cdot \lfloor \frac{d_G(u,v)}{x} \rfloor + r_{uv}$. Proposition 4 implies that $d_G(u,v) \leq x \cdot d_{G'}(u,v)$. Notice first that it cannot be $d_{G'}(u,v) < \lceil \frac{d_G(u,v)}{x} \rceil$, as otherwise we get $d_G(u,v) \leq x \cdot (\lceil \frac{d_G(u,v)}{x} \rceil - 1) < \lfloor \frac{d_G(u,v)}{x} \rfloor + r_{uv}$. Denote $b = \lfloor \frac{d_G(u,v)}{x} \rfloor$. Let $P = \langle u = v_0, \ldots, v_d = v \rangle$ be a shortest path from u to v in G. From Proposition 4 we have for every $0 \leq i \leq b$ the edges $(v_{i \cdot x}, v_{(i+1) \cdot x})$ are in G'. Also, if $r_{uv} \neq 0$, we have the edge $(v_{b \cdot x}, v_d)$, and otherwise (i.e. when $r_{uv} = 0$) we have $v_{b \cdot x} = v_d$. In both cases, we get that there exists a path from u to v in G' of length $\lceil \frac{d_G(u,v)}{x} \rceil$. Claims (2) and (3) of this lemma follow immediately from the construction of the path in G'. ◀

Lemma 7 guarantees that the symmetry parameter of the graph induced by $A' = (A_G)^x$ is not bigger than $s(G)$.

▶ **Lemma 7.** *Let $G = (V,E)$ be a directed strongly-connected graph and let $0 < x \leq n$ be an integer. Let G' be the graph induced from $A' = (A_G)^x$. It follows that G' is also strongly-connected and $s(G') \leq s(G)$.*

Proof. First note that since G is strongly-connected and we have from Lemma 6 that $d_{G'}(u,v) = \lceil \frac{d_G(u,v)}{x} \rceil$ for every $u, v \in V$, it follows that G' is also strongly-connected. Let (u,v) be an edge in G'. From the way we obtained G', $d_G(u,v) \leq x \cdot d_{G'}(u,v) = 1 \cdot x = x$. Let $P = \langle u = v_0, \ldots v_d = v \rangle$ be a shortest path from u to v in G. By the symmetry parameter of G, there exists a path in G of length $s(G)$ at most from every (v_{i+1}, v_i), $0 \leq i \leq d_G(u,v)$. This implies a shortest path in G from v to u of length $x \cdot s(G)$ at most. By Lemma 6, $d_{G'}(v,u) = \lceil \frac{d_G(v,u)}{x} \rceil \leq \lceil \frac{x \cdot s(G)}{x} \rceil = \lceil s(G) \rceil = s(G)$, as required. ◀

We now turn to present our main theorem. Notice that the algorithm receives as an input the parameter $s(G)$ of G. As we shall show later, $s(G)$ can be computed in $O(n^\omega \log n)$ time. A pseudocode of our algorithm is given in Algorithm 2.

▶ **Theorem 8.** *Let $G = (V,E)$ be a directed strongly-connected unweighted graph with a parameter $s = s(G)$. We can compute APSP for G in $O(s \cdot n^\omega \log_{s+1} n)$ time.*

Proof. Notice first that $s = s(G)$ is well defined, since G is strongly-connected. Let $s' = s+1$. Our algorithm will follow a similar approach as done in Seidel's algorithm [20] for undirected graphs.

Let A be the adjacency matrix of the graph G (recall we assume that A always has 1's in its diagonal). If all entries in A are 1's, we return A as the distance matrix for G. If this is not the case, we compute $A' = A^{s'}$ in $O(s' \cdot n^\omega)$ time. Let G' be the graph of A' and let D' be the distance matrix for G', that is, $D'(u,v) = d_{G'}(u,v)$. We obtain D' by invoking our algorithm recursively. Notice that according to Lemma 7 the parameter $s(G)$ does not increase in G' (i.e. $s(G') \leq s(G)$).

Our goal is to compute D, the distance matrix of G, based on D', the distance matrix computed recursively for G'. According to Lemma 6, for every $u, v \in V$ it follows that $d_{G'}(u,v) = \lceil \frac{d_G(u,v)}{s'} \rceil$ and $d_G(u,v) = s' \cdot (d_{G'}(u,v) - 1) + r_{uv}$ (or $d_G(u,v) = s' \cdot d_{G'}(u,v)$, for the case that $r_{uv} = 0$). Since we know $d_{G'}(u,v)$, this implies we only left to compute r_{uv} in order to find the distance from u to v in G.

Our first step is to compute the distances for all $u, v \in V$ such that $d_G(u,v) = d_{G'}(u,v)+1$ (i.e. $r_{uv} = 1$).

Let $u, v \in V$. Denote $k = d_{G'}(u,v)$. Examine first the case where $r_{uv} = 1$. Let $w \in N_{in}(v)$. It follows from Lemma 5 that there exists a $w' \in N_{in}(v)$ such that $d_G(u,w') = d_G(u,v) - 1$. Since $r_{uv} = 1$, it must be that $d_{G'}(u,w') = k-1$, as $r_{uw'} = 0$ and $d_{G'}(u,w') = \lceil \frac{d(u,w')}{s'} \rceil$. From Lemma 5 and from the symmetry parameter s of G it follows that for every $w \in N_{in}(v)$, $d_G(u,w) \leq d_G(u,v) + s$. Therefore, $d_{G'}(u,w) \leq k$, and for $r_{uv} = 1$, we have that $\sum_{w \in N_{in}(v)} d_{G'}(u,w) < |N_{in}(v)|k = |N_{in}(v)| d_{G'}(u,v)$.

For the case that $r_{uv} \neq 1$. We know from Lemma 5 that for every $w \in N_{in}(v)$ we have $d_G(u,v) - 1 \leq d_G(u,w) \leq d_G(u,v) + s$. Since for every $w' \in N_{in}(v)$, such that $d_G(u,w) = d_G(u,v) - 1$, it holds that $d_{G'}(u,w') = k$, it implies that $d_{G'}(u,w) \geq k$ for every $w \in N_{in}(v)$ and we have in this case that $\sum_{w \in N_{in}(v)} d_{G'}(u,w) \geq |N_{in}(v)|k = |N_{in}(v)| d_{G'}(u,v)$.

We conclude that for every $u, v \in V$, $r_{uv} = 1$ if and only if $\sum_{w \in N_{in}(v)} d_{G'}(u,w) < |N_{in}(v)| d_{G'}(u,v)$. We can obtain $\sum_{w \in N_{in}(v)} d_{G'}(u,w)$, for every $u, v \in V$, by computing the integer matrix multiplication $D_1 = D' \cdot A$ in $O(n^\omega)$ time. By computing the integer matrix multiplication $1^{n \times n} \cdot A$, we can obtain $|N_{in}(v)|$ for every $v \in V$. Overall we can check for every $u, v \in V$ if $\sum_{w \in N_{in}(v)} d_{G'}(u,w) < |N_{in}(v)| d_{G'}(u,v)$ in $O(1)$ time. At this stage we set $D(u,v) = s'(D'(u,v) - 1) + 1$, for every $u, v \in V$ such that $r_{uv} = 1$.

It is left to compute the distances for every $u, v \in V$ such that $r_{uv} \neq 1$. We show by induction for $1 \leq i \leq s'$, that we can compute the distances for every $u, v \in V$ such that $r_{uv} = i \mod s'$ in $O(n^\omega)$ time. We already showed the base case of the induction (i.e. for

$i = r_{uv} = 1$). By the hypothesis of the induction, we assume that we have the distances for every $u, v \in V$ such that $r_{uv} = j \mod s'$, $1 \leq j \leq i$, and we show that we can compute the distances for $u, v \in V$ such that $r_{uv} = (i+1) \mod s'$.

Let $N^i_{uv} = \{w \in N_{in}(v) \mid r_{uw} = i \mod s'\}$.

▶ **Claim 9.** *Let $u, v \in V$ $(u \neq v)$, such that $d_{G'}(u, v) = k$. For every $w \in N^i_{uv}$, $d_{G'}(u, w)$ is either $k-1$, k or $k+1$. Moreover, if $r_{uv} \neq 1$, it cannot be that $d_{G'}(u, w) = k-1$.*

Proof. By Lemma 5, $d_G(u, v) - 1 \leq d_G(u, w)$. Now, $d_{G'}(u, w) = \lceil \frac{d_G(u,w)}{s'} \rceil \geq \lceil \frac{d_G(u,v)-1}{s'} \rceil \geq \lceil \frac{d_G(u,v)}{s'} \rceil - 1 = d_{G'}(u, v) - 1 = k-1$. Notice that the only case that $\lceil \frac{d_G(u,v)-1}{s'} \rceil = \lceil \frac{d_G(u,v)}{s'} \rceil - 1$ is when $r_{uv} = 1$. Therefore, $d_{G'}(u, w) \geq k-1$, and $d_{G'}(u, w) \geq k$ for $r_{uv} \neq 1$.

Assume toward contradiction that there exists $w \in N^i_{uv}$ such that $d_{G'}(u, w) > k+1$. For such a w, we have $d_G(u, w) > s'(k+1)$. Now, since $(w, v) \in E$ and the symmetry parameter of G is s, it follows that $d_G(v, w) \leq s$, but then $d_G(u, v) + d_G(v, w) \leq s'k + s' = s'(k+1) < d_G(u, w)$, a contradiction to the minimality of $d_G(u, w)$. ◀

Using Claim 9 we can now provide a criteria we will use to identify every $u, v \in V$ $(u \neq v)$ such that $r_{uv} = (i+1) \mod s'$.

▶ **Claim 10.** *Let $u, v \in V$ $(u \neq v)$, such that $r_{uv} = j \mod s'$. Then:*
1. *If $j = i+1 \Rightarrow \sum_{w \in N^i_{uv}} d_{G'}(u, w) < |N^i_{uv}| \cdot (d_{G'}(u, v) + 1)$*
2. *If $i+2 \leq j \leq s' \Rightarrow \sum_{w \in N^i_{uv}} d_{G'}(u, w) \geq |N^i_{uv}| \cdot (d_{G'}(u, v) + 1)$*

Proof. Denote $d_{G'}(u, v) = k$. Since the claim only considers $j \geq 2$, we have that $r_{uv} \neq 1$. Now, according to Claim 9 it holds that $d_{G'}(u, w) = k$ or $d_{G'}(u, w) = k+1$ for every $w \in N^i_{uv}$. If $j = i+1$ then there exists a $w' \in N^i_{uv}$ with $d_{G'}(u, w') = k$. Therefore, $\sum_{w \in N^i_{uv}} d_{G'}(u, w) < |N^i_{uv}| \cdot (k+1) = |N^i_{uv}| \cdot (d_{G'}(u, v) + 1)$.

If $j \geq i+2$ we cannot have any $w \in N^i_{uv}$ such that $d_{G'}(u, w) = k$, since otherwise we get that $d_G(u, w) + d_G(w, v) = (s' \cdot (k-1) + i) + 1 < s' \cdot (k-1) + (i+2) \leq d_G(u, v)$, a contradiction to the minimality of $d_G(u, v)$. Therefore, $\sum_{w \in W^i_{uv}} (d_{G'}(u, v) + 1) = |N^i_{uv}| \cdot (k+1) = |N^i_{uv}| \cdot (d_{G'}(u, v) + 1)$. ◀

By Claim 10 we can now identify $r_{uv} = j \mod s'$, $j = i+1$ by first checking whether $\sum_{w \in N^i_{uv}} D'_G(u, w) < |N^i_{uv}| \cdot (D'_G(u, v) + 1)$ is satisfied or not. Even if the inequality holds, it still may be that $j < i+1$, but since by the induction hypothesis we already computed the distances for $j \leq i$, we can distinguish between $j = i+1$ and $j < i+1$. We compute the exact distances for $r_{uv} = (i+1) \mod s'$ using Lemma 6. We are left to show how to compute $\sum_{w \in N^i_{uv}} D'_G(u, w)$ and $|N^i_{uv}|$. Notice that at this stage we know the exact distance for every $u, v \in V$ such that $r_{uv} = i \mod s'$. We define $D'_i(u, v)$ to be $D'(u, v)$ if $r_{uv} = i \mod s'$ and 0 otherwise. Similarly, we define $A_i(u, v)$ to be 1 if $r_{uv} = i \mod s'$ and 0 otherwise. We compute by integer matrix multiplications $D_i = D'_i \cdot A$ and $N_i = A_i \cdot A$. Notice that $D_i(u, v) = \sum_{w \in N^i_{uv}} D'_G(u, w)$ and that $N_i(u, v) = |N^i_{uv}|$. This concludes the correctness of the algorithm.

By Lemma 12 the symmetry parameter of the graph can be obtained in $O(n^\omega \log n)$ time. The time to compute $A^{s'}$ is $O(s \cdot n^\omega)$. The time to compute a specific $r_{uv} = i \mod s'$ $(1 \leq i \leq s')$ is also $O(n^\omega)$, therefore, the total running time for a recursive invocation is $(s'+1) \cdot O(n^\omega)$. Since G is an unweighted graph, its diameter is at most $n-1$, and hence we have at most $O(\log_{s'} n)$ recursive invocations. The total time of the algorithm, therefore, is $O(s \cdot n^\omega \log_{s'} n)$. ◀

4 Extending to general directed graphs

As noted earlier, Algorithm 2 works only on strongly-connected graphs, where $s(G)$ is well-defined. Nevertheless, as we shall see here, it is possible to reduce any directed unweighted graph to a directed strongly-connected unweighted graph, where our parameterized-APSP can be applied. Let $G = (V, E)$ be our graph. Let $Diam(G) = \max_{u,v \in V}\{d_G(u, v) \mid d_G(u, v) < \infty\}$ be the diameter of the graph G. In the reduced graph G', we will have $s(G') \leq Diam(G) + 1$.

The reduction is done as follows. Let $G = (V, E)$, and let $d = Diam(G)$. First we build a new directed graph $G' = (V', E')$. The graph G' is initially a copy of G. Next, we add to G' new vertices $v_1, v_2, \ldots, v_d$, and the edges (v_i, v_{i+1}), where $1 \leq i \leq d-1$, and the edge (v_d, v_1). We also add for each vertex $v \in V$ the edges (v, v_1) and (v_d, v).

We can apply on G' our parameterized APSP of Theorem 8.

From our construction $s(G') \leq d + 1$. To see that, first notice that the new vertices $v_1, v_2, \ldots, v_d$ are connected by a cycle of length $d - 1$, therefore, for the edges (v_i, v_{i+1}), where $1 \leq i \leq d - 1$, and the edge (v_d, v_1) the symmetry parameter is $d - 2$. Let $v \in V$, the edges (v, v_1) and (v_d, v) are contained in the cycle $v \to v_1 \to v_2 \to \cdots \to v_d \to v$, thus the symmetry parameter of the edges (v, v_1) and (v_d, v) for every $v \in V$ is d. Also, for an edge $(u, v) \in E$ we have the path $\langle v, v_1, v_2, \ldots, v_d, u\rangle$ of length $d + 1$. The insertion of the new edges to G may shorten the distance between vertices that were originally in G. However, it is easy to see that any path in G' between two vertices from G, that uses at least one of the new edges, must has a length of at least $d + 1$. Since the original diameter of G is d, this implies that for every $u, v \in V$, if $d_{G'}(u, v) \leq d$ then $d_{G'}(u, v) = d_G(u, v)$, otherwise, if $d_{G'}(u, v) > d$ then $d_G(u, v) = \infty$.

The diameter can be computed, using repeated squaring, in $O(n^\omega \log n)$ time. The time to construct G' and to compute the distances of G from the distances of G' is bound by $O(n^2)$. Thus the total time of the reduction is $O(n^\omega \log n)$.

Any algorithm that computes APSP must take $\Omega(n^\omega)$ time (otherwise matrix multiplication can be computed faster), therefore, using the reduction above, we can convert any algorithm that computes APSP for strongly-connected directed graphs in $T(n)$ time into algorithm that computes $T(2n) + O(n^\omega \log(n)) = O(T(n) \log(n))$ time general directed graphs.

5 A hybrid APSP algorithm for directed graphs

As shown, the efficiency of our parameterized-APSP algorithm strongly depends on the symmetry parameter of the input graph. Since by the definition of the parameter, the *maximum* is taken over all the edges of the graph, even a single edge might cause the parameter of the graph to be large.

Recall that for a threshold value $z < s(G)$ of a directed strongly-connected graph G, an edge (u, v) is z-violating edge, if $d(v, u) > z$. We now show that our algorithm can be easily modified such that even if a graph holds to have $o(n^{0.53})$ edges that are $o(n^{0.16})$-violating, the modified algorithm provides a faster algorithm than the state-of-the-art algorithm for the problem [26].

Let $\beta_z(G)$ be the number of z-violating edges in G. The following theorem gives an algorithm to compute APSP for G in a time that depends on z and $\beta_z(G)$. A pseudocode of the algorithm is also given in Algorithm 3.

▶ **Theorem 11.** *Let $G = (V, E)$ be a directed, strongly-connected graph, with a parameter $s(G)$, and let $z < s(G)$ be an integer. We can compute APSP for G in $O(zn^\omega \log n + n^2\beta_z(G))$ time.*

Proof. Let $E_z(G)$ be the set of edges in G that are z-violating. We first compute $E_z(G)$ in $O(n^\omega \log z)$ time, using Lemma 13. Let $(u, v) \in E_z(G)$, we compute in $O(m) = O(n^2)$ time BFS in and out of u and v. Next we removed from G all the edges in $E_z(G)$. Let H be the new graph after the removal of the edges. For the sake of simplicity we assume H is strongly-connected (otherwise, we can run the parameterized APSP algorithm on each of its strongly-connected components). Notice that $s(H) < z$, since we remove all the edges that violate the threshold, and any non-violating edge must participates in a cycle such that all its edges are non-violating edges (thus removing the violating edges would not affect a non-violating edge). Therefore, we compute in $O(zn^\omega \log n)$ time the distances of H. If a shortest path from u to v in G contains an edge (p, q) from $E_z(G)$, we can find the distance in $|E_z(G)| = \beta_z(G)$ time by taking the minimum from $min_{(p,q)\in E_z(G)}\{d(u, p) + d(q, v) + 1\}$ (the distances $d(u, p)$ and $d(q, v)$ are obtained from the in and out BFS computed for all the endpoints in $E_z(G)$). Otherwise, $d(u, v) = d_H(u, v)$. The time for this step is $O(n^2\beta_z(G))$ and the total time for this algorithm is $O(zn^\omega \log n + n^2\beta_z(G))$. ◀

Since the reduction to strongly-connected graph and the computation of the z-violating edges takes $O(n^\omega \log n)$ time, this leads us to the following hybrid approach. Reduce the graph to strongly-connected and compute the violating edges for z such that it gives faster running time than the current known fastest algorithm for the problem. If the number of violating edges does not exceed the threshold, use Algorithm 3, otherwise use the fastest known algorithm for the problem. Unless an $O(n^\omega \log n)$-time algorithm for the problem is found, this hybrid approach may provide a faster running time for some instances of the problem.

5.1 Fast computation of the parameter and the violating edges

Our algorithms need to know the symmetry parameter of the graph. We show in the next lemma how to compute this parameter in $O(n^\omega \log n)$ time.

▶ **Lemma 12.** *Let $G = (V, E)$ be a directed strongly-connected graph with symmetry parameter $s(G)$. We can compute $s(G)$ in $O(n^\omega \log n)$ time.*

Proof. Denote $D_G = (A_G \bigvee I)$. Compute $D_G, D_G^2, D_G^4, \ldots, D_G^{2^{\log n}}$ using repeated squaring. Let k be the first value in $\{0, \ldots, \lceil \log n \rceil\}$ such that $D_G^{2^k}(v, u) = 1$ for every $(u, v) \in E$. If $k = 0$, then $s(G) = 1$. Otherwise, by Proposition 4 it must be that $s(G) \in [2^{k-1}, 2^k]$. By setting $s'(G) = 2^k$, we have a 2-approximation for $s(G)$. We need only $\lceil \log n \rceil$ matrix multiplication to compute $D_G, D_G^2, D_G^4, \ldots, D_G^{2^{\log n}}$, thus the time for this is $O(n^\omega \log n)$.

Now, we show how to compute $s(G)$ exactly. We have $s(G) \in [2^{k-1}, 2^k]$, we perform a binary search for $s(G)$ in the range $[2^{k-1}, 2^k]$. Let $[a, b]$ $(a < b)$ be the current range, we compute D_G^c for $c = \frac{a+b}{2}$. We check if $D_G^c(v, u) = 1$ for every $(u, v) \in E$. If it is true, we continue the search in the range $[a, c]$, otherwise we continue the search in the range $[c, b]$. There are $O(\log n)$ steps in the binary search. We can compute D_G^c by using only 2 matrix multiplications. To see this, notice first, that since we do our binary search on $[2^{k-1}, 2^k]$, it follows, that at any stage of the binary search, $(b - a)/2$ is a power of 2. Since $c = \frac{a+b}{2} = a + \frac{(b-a)}{2}$, we have that $D_G^c = D_G^a \cdot D_G^{2^j}$ (for some $j \leq k$), where D_G^a and $D_G^{2^j}$ were already computed. This concludes that the total time to compute $s(G)$ exactly is $O(n^\omega \log n)$. ◀

Similarly, we find all z-violating edges for a $z < n$ in time $O(n^\omega \log z)$, by computing D^z_G and returning all the edges $(u, v) \in E$ such that $D^z_G(v, u) = 0$. The statement is given in the following lemma.

▶ **Lemma 13.** *Let $G = (V, E)$ be a directed strongly-connected graph, and let $z < n$ be an integral value. We can find all z-violating edges for a $z < n$ in $O(n^\omega \log z)$ time.*

References

1. Donald Aingworth, Chandra Chekuri, Piotr Indyk, and Rajeev Motwani. Fast estimation of diameter and shortest paths (without matrix multiplication). *SIAM J. Comput.*, 28(4):1167–1181, 1999. doi:10.1137/S0097539796303421.
2. Noga Alon, Zvi Galil, and Oded Margalit. On the exponent of the all pairs shortest path problem. In *32nd Annual Symposium on Foundations of Computer Science, San Juan, Puerto Rico, 1-4 October 1991*, pages 569–575, 1991.
3. Timothy M. Chan. More algorithms for all-pairs shortest paths in weighted graphs. *SIAM J. Comput.*, 39(5):2075–2089, 2010. doi:10.1137/08071990X.
4. Timothy M. Chan and Ryan Williams. Deterministic apsp, orthogonal vectors, and more: Quickly derandomizing razborov-smolensky. In *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2016, Arlington, VA, USA, January 10-12, 2016*, pages 1246–1255, 2016. doi:10.1137/1.9781611974331.ch87.
5. Don Coppersmith and Shmuel Winograd. Matrix multiplication via arithmetic progressions. *J. Symb. Comput.*, 9(3):251–280, 1990. doi:10.1016/S0747-7171(08)80013-2.
6. Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms (3. ed.)*. MIT Press, 2009. URL: http://mitpress.mit.edu/books/introduction-algorithms.
7. Michael L. Fredman. New bounds on the complexity of the shortest path problem. *SIAM J. Comput.*, 5(1):83–89, 1976. doi:10.1137/0205006.
8. Michael L. Fredman and Robert Endre Tarjan. Fibonacci heaps and their uses in improved network optimization algorithms. *J. ACM*, 34(3):596–615, 1987. doi:10.1145/28869.28874.
9. Zvi Galil and Oded Margalit. All pairs shortest distances for graphs with small integer length edges. *Inf. Comput.*, 134(2):103–139, 1997. doi:10.1006/inco.1997.2620.
10. François Le Gall. Faster algorithms for rectangular matrix multiplication. In *53rd Annual IEEE Symposium on Foundations of Computer Science, FOCS 2012, New Brunswick, NJ, USA, October 20-23, 2012*, pages 514–523, 2012. doi:10.1109/FOCS.2012.80.
11. François Le Gall. Powers of tensors and fast matrix multiplication. In *International Symposium on Symbolic and Algebraic Computation, ISSAC '14, Kobe, Japan, July 23-25, 2014*, pages 296–303, 2014. doi:10.1145/2608628.2608664.
12. Yijie Han. Improved algorithm for all pairs shortest paths. *Inf. Process. Lett.*, 91(5):245–250, 2004. doi:10.1016/j.ipl.2004.05.006.
13. Yijie Han. An $O\left(n^3(\log\log n/\log n)^{5/4}\right)$ time algorithm for all pairs shortest path. *Algorithmica*, 51(4):428–434, 2008. doi:10.1007/s00453-007-9063-0.
14. Alon Itai and Michael Rodeh. Finding a minimum circuit in a graph. *SIAM J. Comput.*, 7(4):413–423, 1978. doi:10.1137/0207033.
15. Donald B. Johnson. Efficient algorithms for shortest paths in sparse networks. *J. ACM*, 24(1):1–13, 1977. doi:10.1145/321992.321993.
16. Seth Pettie. A new approach to all-pairs shortest paths on real-weighted graphs. *Theor. Comput. Sci.*, 312(1):47–74, 2004. doi:10.1016/S0304-3975(03)00402-X.
17. Liam Roditty and Roei Tov. Approximating the girth. *ACM Transactions on Algorithms*, 9(2):15, 2013. doi:10.1145/2438645.2438647.

18 Liam Roditty and Virginia Vassilevska Williams. Minimum weight cycles and triangles: Equivalences and algorithms. In *IEEE 52nd Annual Symposium on Foundations of Computer Science, FOCS 2011, Palm Springs, CA, USA, October 22-25, 2011*, pages 180–189, 2011. doi:10.1109/FOCS.2011.27.

19 Liam Roditty and Virginia Vassilevska Williams. Fast approximation algorithms for the diameter and radius of sparse graphs. In *Symposium on Theory of Computing Conference, STOC'13, Palo Alto, CA, USA, June 1-4, 2013*, pages 515–524, 2013. doi:10.1145/2488608.2488673.

20 Raimund Seidel. On the all-pairs-shortest-path problem in unweighted undirected graphs. *J. Comput. Syst. Sci.*, 51(3):400–403, 1995. doi:10.1006/jcss.1995.1078.

21 Avi Shoshan and Uri Zwick. All pairs shortest paths in undirected graphs with integer weights. In *40th Annual Symposium on Foundations of Computer Science, FOCS'99, 17-18 October, 1999, New York, NY, USA*, pages 605–615, 1999. doi:10.1109/SFFCS.1999.814635.

22 Tadao Takaoka. Subcubic cost algorithms for the all pairs shortest path problem. *Algorithmica*, 20(3):309–318, 1998. doi:10.1007/PL00009198.

23 Tadao Takaoka. A faster algorithm for the all-pairs shortest path problem and its application. In *Computing and Combinatorics, 10th Annual International Conference, COCOON 2004, Jeju Island, Korea, August 17-20, 2004, Proceedings*, pages 278–289, 2004. doi:10.1007/978-3-540-27798-9_31.

24 Mikkel Thorup. Floats, integers, and single source shortest paths. *J. Algorithms*, 35(2):189–201, 2000. doi:10.1006/jagm.2000.1080.

25 Virginia Vassilevska Williams and Ryan Williams. Subcubic equivalences between path, matrix and triangle problems. In *51th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2010, October 23-26, 2010, Las Vegas, Nevada, USA*, pages 645–654, 2010. doi:10.1109/FOCS.2010.67.

26 Uri Zwick. All pairs shortest paths using bridging sets and rectangular matrix multiplication. *J. ACM*, 49(3):289–317, 2002. doi:10.1145/567112.567114.

27 Uri Zwick. A slightly improved sub-cubic algorithm for the all pairs shortest paths problem with real edge lengths. In *Algorithms and Computation, 15th International Symposium, ISAAC 2004, Hong Kong, China, December 20-22, 2004, Proceedings*, pages 921–932, 2004. doi:10.1007/978-3-540-30551-4_78.

A Algorithms

Algorithm 1: Directed-APSP($G = (V, E)$)

if *G is not strongly-connected* **then**
  Reduce G to a strongly-connected graph G';
  $G \leftarrow G'$;
Compute $s = s(G)$;
return Param-Directed-APSP(A_G, s);

Algorithm 2: Param-Directed-APSP(A, s)

$A \leftarrow A \vee I$;
if $A = 1^{n \times n}$ **then**
  return A
$D' \leftarrow$ Param-Directed-APSP(A^{s+1}, s);
$D_1 \leftarrow D' \cdot A$;
foreach $(u, v) \in V^2$ **do** $D(u, v) \leftarrow -\infty$;
foreach $(u, v) \in V^2$ **do**
  if $D_1(u, v) < |N_{in}(v)| \cdot D'(u, v)$ **then**
    $D(u, v) \leftarrow (s + 1)(D'(u, v) - 1) + 1$;
foreach $i \in \{2, \ldots, s + 1\}$ **do**
  foreach $(u, v) \in V^2$ **do**
    if $D(u, v) = (s + 1)(D'(u, v) - 1) + i - 1$ **then**
      $D'_i(u, v) \leftarrow D'(u, v)$; $A_i(u, v) \leftarrow 1$;
    else
      $D'_i(u, v) \leftarrow 0$; $A_i(u, v) \leftarrow 0$;
  $D_i \leftarrow D'_i \cdot A$; $N_i \leftarrow A_i \cdot A$;
  foreach $(u, v) \in V^2$ **do**
    if $D_i(u, v) < |N_i(u, v)| \cdot D'(u, v) \wedge D(u, v) = -\infty$ **then**
      $D(u, v) = (s + 1)(D'(u, v) - 1) + i$;
return D

Algorithm 3: Directed-APSP-z($G = (V, E)$, z)

Compute $E_z(G)$, the z-violating edges of G;
foreach $(u, v) \in E_z(G)$ **do**
  Compute in and out BFS for u and v, and store the obtained distances;
Let $H = (V, E \setminus E_z(G))$;
$D_H \leftarrow$ Param-Directed-APSP($A_H, s(H)$);
foreach $(u, v) \in V^2$ **do**
  $D(u, v) = min\{D_H(u, v), min_{(p,q)\in E_z(G)}\{d(u, p) + d(q, v) + 1\}\}$;
return D;