All-Pairs Shortest Paths for Real-Weighted Undirected Graphs with Small Additive Error

Timothy M. Chan
Department of Computer Science, University of Illinois at Urbana-Champaign, IL, USA

Abstract

Given a graph with n vertices and *real* edge weights in $[0,1]$, we investigate an approximate version of the standard all-pairs shortest paths (APSP) problem where distances are estimated with *additive error* at most ε. Yuster (2012) introduced this natural variant of approximate APSP, and presented an algorithm for directed graphs running in $\tilde{O}(n^{(3+\omega)/2}) \le O(n^{2.687})$ time for an arbitrarily small constant $\varepsilon > 0$, where ω denotes the matrix multiplication exponent. We give a faster algorithm for *undirected* graphs running in $\tilde{O}(n^{(3+\omega^2)/(\omega+1)}) \le O(n^{2.559})$ time for any constant $\varepsilon > 0$. If $\omega = 2$, the time bound is $\tilde{O}(n^{7/3})$, matching a previous result for undirected graphs by Dor, Halperin, and Zwick (2000) which only guaranteed additive error at most 2.

2012 ACM Subject Classification Theory of computation → Design and analysis of algorithms

Keywords and phrases Shortest paths, approximation, matrix multiplication

Digital Object Identifier 10.4230/LIPIcs.ESA.2021.27

1 Introduction

The *all-pairs shortest paths* (APSP) problem is one of the most well-known problems in algorithm design, and plays a central role in the study of fine-grained complexity. Only small (subpolynomial) speedups over the textbook cubic-time algorithms are known for arbitrary real-weighted dense graphs [6, 8, 13, 21], and it has been conjectured that there are no truly subcubic algorithms [20]. If faster running time is desired, one thus turns to *approximation algorithms*.

To this end, Zwick [23] described an $O(n^\omega \log \frac{w_{\max}}{w_{\min}})$-time algorithm with approximation factor $1+\varepsilon$ for any constant $\varepsilon > 0$, for any (directed or undirected) graph with positive real edge weights, where n is the number of vertices, $w_{\min}$ and $w_{\max}$ denote the minimum and maximum edge weight, and $\omega < 2.373$ is the matrix multiplication exponent [2, 11]. For every pair of vertices u and v, the algorithm computes a value $\widetilde{D}[u,v]$ such that $D[u,v] \le \widetilde{D}[u,v] \le (1+\varepsilon)D[u,v]$, where $D[u,v]$ denotes the distance (i.e., the shortest-path weight) from u to v.

While multiplicative approximation is natural, in this paper we are interested in an even stronger form of approximation, where we want small *additive error* bounded by $\varepsilon w_{\max}$. More precisely, for every u and v, we seek a value $\widetilde{D}[u,v]$ such that $D[u,v] \le \widetilde{D}[u,v] \le D[u,v] + \varepsilon w_{\max}$. To see how this can yield a much better estimate, imagine the case when the distance $D[u,v]$ is large; a $(1+\varepsilon)$-factor approximation may differ from the true value by $\varepsilon D[u,v]$, which could be much bigger than ε times the maximum weight of a single edge.[1]

From now on, we assume $w_{\max} = 1$, without loss of generality, by rescaling. In other words, we assume that all edge weights lie in $[0,1]$ and we tolerate additive error at most ε.

[1] In fact, we can achieve additive error at most $O(\varepsilon w_{\max}[u,v])$, where $w_{\max}[u,v]$ denotes the weight of the longest edge in a shortest path from u to v, by guessing a value $w \in [w_{\max}[u,v]/2, w_{\max}[u,v]]$ and removing all edges of weight exceeding w from the graph; $O(\log \frac{w_{\max}}{w_{\min}})$ guesses of w suffice.

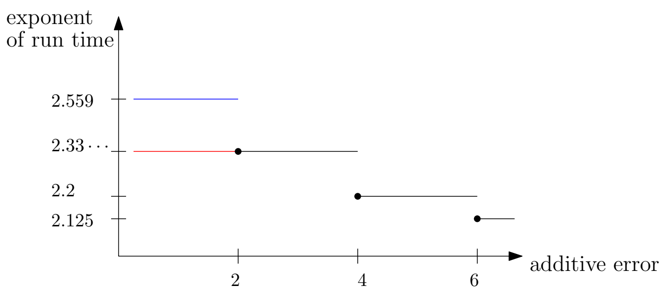

Figure 1 Tradeoffs between additive error and running time for approximate APSP in undirected real-weighted graphs. The blue part indicates the new result using the current matrix multiplication bound $\omega < 2.373$; the red part indicates the new result if $\omega = 2$.

This form of additive approximation was first considered by Yuster [22], who gave an algorithm running in $\tilde{O}(n^{(3+\omega)/2}) \le O(n^{2.687})$ time[2] for any constant additive error $\varepsilon > 0$. If $\omega = 2$, the time bound is $\tilde{O}(n^{2.5})$. For arbitrary (dense) directed graphs, this would be difficult to improve, since approximate APSP with additive error at most c for any constant c is at least as hard as exact unweighted APSP, for which the current best algorithm by Zwick [23] has exponent 2.5 when $\omega = 2$ and is conjectured to be near optimal [9]. However, a question remains as to whether an improved result for approximate APSP with additive error ε is possible for *undirected* graphs (for which $\tilde{O}(n^{\omega})$-time algorithms are known for exact unweighted APSP [15, 19]).

For *unweighted undirected* graphs, the seminal work by Aingworth, Chekuri, Indyk, and Motwani [1] described combinatorial algorithms for approximate APSP achieving $O(1)$ additive error without using fast matrix multiplication. Subsequent improvements were described by Dor, Halperin, and Zwick [12]: in particular, for dense graphs, the best results were an $\tilde{O}(n^{7/3})$-time algorithm with additive error at most 2, and an $\tilde{O}(n^{2+2/(3k-2)})$-time algorithm with additive error at most k for any even constant $k > 2$. As noted in Dor et al.'s paper, these algorithms for unweighted undirected graphs can actually be extended to weighted undirected graphs with arbitrary real edge weights in $[0, 1]$. However, none of these results achieve additive error below 2. In fact, approximate undirected APSP with additive error strictly below 2 is at least as hard as Boolean matrix multiplication (by considering tripartite graphs with unit edge weights), thus ruling out combinatorial algorithms for arbitrarily small ε.

New result. Our main result is a new algorithm for undirected real-weighted graphs with additive error ε. The algorithm uses fast matrix multiplication and achieves running time $\tilde{O}(n^{(3+\omega^2)/(\omega+1)}) \le O(n^{2.559})$. This improves Yuster's $O(n^{2.689})$-time directed-graph algorithm. Furthermore, if $\omega = 2$, our time bound becomes $\tilde{O}(n^{7/3})$, which improves Yuster's $\tilde{O}(n^{2.5})$ bound and also matches Dor, Halperin, and Zwick's time bound for additive error 2 while making the additive error an arbitrarily small constant ε. (See Figure 1.)

Other related work. Other formulations of approximate APSP have been studied in the literature. For example, Bringmann, Künnemann, and Wegrzycki [5] revisited multiplicative approximation, but in a setting where $\frac{w_{\max}}{w_{\min}}$ may be large (Zwick's approximation algorithm [23] works well only when this ratio is bounded). On the other hand, Roditty and

[2] The $\tilde{O}$ notation hides $\log^{O(1)} n$ factors. Factors dependent on ε are also suppressed, for simplicity, but they will all be polynomial in $1/\varepsilon$.

Shapira [18] gave results (recently improved by Chan, Vassilevska Williams, and Xu [9]) on approximate APSP for unweighted directed APSP that achieved *sublinear* additive error bounded by $D[u, v]^p$ for a given constant $p < 1$; the guarantee is stronger than $1 + \varepsilon$ multiplicative approximation as $D[u, v]$ gets larger, but is not as strong as constant additive error.

In the literature on graph spanners, there have also been many results for unweighted undirected graphs with constant additive errors [1, 4, 10].

Our time bound appears unusual among algorithms in the literature that rely on fast matrix multiplication. Coincidentally, Grandoni et al. [16] gave a matrix-multiplication-based algorithm for an entirely different problem (all-pairs lowest common ancestors in DAGs) with a time bound that is also $\tilde{O}(n^{7/3})$ if $\omega = 2$ (though their exponent got below 2.5 under the current matrix multiplication bound). For our result, $n^{7/3}$ is a natural barrier, since even with additive error 2, the current best algorithm by Dor, Halperin, and Zwick [12] still requires $\tilde{O}(n^{7/3})$ time.

Techniques. To see why the additive approximation problem is challenging for real-weighted graphs, consider the standard idea of rounding edge weights and rescaling to turn them into integers. To guarantee additive error at most a constant ε, since a shortest path may require $\Theta(n)$ edges in the worst case, one would need to round edge weights to multiples of ε/n, and the rescaled integers could then be $\Theta(n)$; unfortunately, APSP (or min-plus product) for integer input of that magnitude still requires near cubic time under the current state of the art.

Yuster's previous algorithm for directed graphs [22] is based on an approach by Zwick [23] (originally for exact small-edge-weighted APSP), which divides into two cases: paths that use a few edges (i.e., hops) vs. paths that use many edges. For shortest paths with less than L edges, the aforementioned rounding approach reduces the problem to computing min-plus products for small integers bounded by $O(L)$, which reduces to standard matrix multiplication on $\tilde{O}(L)$-bit numbers. On the other hand, for shortest paths with more than L edges, there exists a small hitting set (also called a "bridging set") with $\tilde{O}(n/L)$ vertices, and we can run a single-source/single-sink algorithm from/to each such vertex. Finally, the parameter L is chosen (near $\sqrt{n}$ if $\omega = 2$) to balance cost.

To improve the running time, we combine this approach with Aingworth et al.'s approach [1] (which Dor et al.'s algorithm [12] builds upon). Aingworth et al.'s approach divides into two cases differently: vertices with low degree vs. vertices with high degree. Low-degree vertices are less expensive because the number of incident edges is small. On the other hand, for high-degree vertices, there exists a small dominating set, and so these vertices can be covered by a small number of "clusters"; sources in the same cluster are close together, and so distances from one fixed source s give us a good approximation (with $O(1)$ additive error) to distances from other sources in the same cluster (since the graph is undirected). To reduce the additive error from $O(1)$ to $O(\varepsilon)$, we need a number of further ideas. We will use min-plus products at each layer of the BFS tree from s. To make the error sum to $O(\varepsilon)$, we will set the error tolerance at each layer to be proportional to the size of the layer (roughly $\varepsilon n_i/n$ if the i-th layer has size n_i, as we will describe in Section 4). Bounding the total running time requires some care, since the min-plus products are done to matrices of different dimensions at the different layers.

The general plan shares some similarity with an exact combinatorial APSP algorithm by Chan [7] for sparse undirected unweighted graphs, which also modifies Aingworth et al.'s algorithm and simulates BFS to compute distances from multiple sources in a cluster.

However, what makes our algorithm interesting is the combination of Aingworth et al.'s approach with fast matrix multiplication. (A recent algorithm by Chan, Vassilevska Williams, and Xu [9] for a different problem – all-pairs lightest shortest paths for undirected small-weighted graphs – also combines Aingworth et al.'s approach with matrix multiplication, but does not involve approximation nor real weights. Our algorithm here is more elaborate.)

2 Preliminaries

Given two matrices A and B, we let $A \star B$ denote the min-plus product, i.e., $(A \star B)[u, v] = \min_z(A[u, z] + B[z, v])$.

Let $\mathcal{M}(n_1, n_2, n_3)$ denote the time complexity for computing the standard product of an $n_1 \times n_2$ and an $n_2 \times n_3$ matrix.

Let $\mathcal{M}^\star(n_1, n_2, n_3 \mid \ell)$ denote the time complexity for computing the min-plus product of an $n_1 \times n_2$ and an $n_2 \times n_3$ matrix, where all the matrix entries are from $\{0, 1, \ldots, \ell, \infty\}$. As is well known [3], $\mathcal{M}^\star(n_1, n_2, n_3 \mid \ell) \le \tilde{O}(\ell \cdot \mathcal{M}(n_1, n_2, n_3))$.

3 Small Distances

We begin with a lemma on computing small distances, which will be useful later. Yuster [22] has already observed how to solve the problem for paths with small number of edges, but our lemma is more challenging, since a path with small weight could still have a large number of hops.

▶ **Lemma 1.** *Given a directed or undirected graph $G = (V, E)$ with n vertices and real edge weights in $[0, 1]$, and given $\beta > \varepsilon$, we can approximate all distances that are at most β, with additive error $O(\varepsilon)$, in $\tilde{O}((\beta/\varepsilon)n^\omega)$ time.*

Proof. We use a form of repeated squaring: loosely speaking, we recursively solve the problem for $\beta/2$ and compute the min-plus product of the resulting matrix with itself. The additive error ε needs to be roughly halved in the recursive call, but luckily the ratio β/ε stays roughly the same.

Let $\delta > 0$ be a parameter to be set later. Let $\beta_{\min}$ be the smallest edge weight. Assume that $\beta > \varepsilon$. For every $u, v \in V$, we will compute $\widetilde{D}^{(\beta,\varepsilon)}[u, v]$, an approximation to $D[u, v]$ with additive error at most ε, provided that $D[u, v] \le \beta$. (More precisely, if $\widetilde{D}^{(\beta,\varepsilon)}[u, v] \ne \infty$, it is a valid approximation; and if $D[u, v] \le \beta$, then it is guaranteed that $\widetilde{D}^{(\beta,\varepsilon)}[u, v] \ne \infty$.)

To compute $\widetilde{D}^{(\beta,\varepsilon)}$:

1. First recursively compute $D' = \widetilde{D}^{(\beta/2,(1-\delta)\varepsilon/2)}$. Round the entries in D' (upward) to multiples of $\delta\varepsilon/3$.
2. Let $A'[u, v]$ be $w(u, v)$ rounded (upward) to a multiple of $\delta\varepsilon/3$. If $A'[u, v] > \beta$, reset $A'[u, v] = \infty$.
3. Set $\widetilde{D}^{(\beta,\varepsilon)} = D' \star A' \star D'$. If $\widetilde{D}^{(\beta,\varepsilon)}[u, v] > \beta + \varepsilon$, reset $\widetilde{D}^{(\beta,\varepsilon)}[u, v] = \infty$.

Correctness follows since any path with weight at most β can be expressed as $\pi_1 e \pi_2$, where each of π_1 and π_2 is a subpath with weight at most $\beta/2$, and e is a single edge (the "median"). The additive error is bounded by $2(1 - \delta)\varepsilon/2 + \delta\varepsilon/3 + \delta\varepsilon/3 + \delta\varepsilon/3 = \varepsilon$.

Since the finite entries of D' and A' after rescaling are integers bounded by $O(\frac{\beta}{\delta\varepsilon})$, these min-plus products take $O(\mathcal{M}^\star(n, n, n \mid \frac{\beta}{\delta\varepsilon})) = \tilde{O}(\frac{\beta}{\delta\varepsilon} n^\omega)$ time. The total time satisfies the recurrence

$$T(\beta, \varepsilon) = T(\beta/2, (1 - \delta)\varepsilon/2) + \tilde{O}(\tfrac{\beta}{\delta\varepsilon} n^\omega),$$

which yields $T(\beta, \varepsilon) = \tilde{O}(\frac{\beta}{\delta\varepsilon} (\frac{1}{1-\delta})^{\log(\beta/\beta_{\min})} n^\omega)$.

We may assume that $\beta \leq n$ and $\beta_{\min} \geq \varepsilon/n$, since we can initially round edge weights to multiples of ε/n. We set $\delta = 1/\log(\beta/\beta_{\min}) = \Omega(1/\log(n/\varepsilon))$, so that $(\frac{1}{1-\delta})^{\log(\beta/\beta_{\min})}$ is bounded by a constant. ◀

4 Multiple Sources

Next, we solve the subproblem of approximating shortest paths from multiple sources in a close-knit "cluster" of vertices $S \subseteq V$. (To optimize the final running time, we find it useful to separate out a preprocessing stage that does not depend on S.)

▶ **Lemma 2.** *Given an undirected graph $G = (V, E)$ with n vertices and real edge weights in $[0, 1] \cup \{\infty\}$, and a parameter B, we can preprocess in $\tilde{O}((1/\varepsilon)Bn^{\omega})$ time, so that: given a subset $S \subseteq V$ of size $O(n/t)$ where every pair of vertices in S have distance at most an integer constant c, we can approximate the distances for all pairs in $S \times V$ with additive error $O(\varepsilon)$, in $\tilde{O}((1/\varepsilon)(n^{\omega}/B^{\omega-2} + n^{\omega}/t^{\omega-2}))$ time.*

Proof. Let $\varepsilon' = \varepsilon/\log n$. Fix a vertex $s \in S$. We first compute all distances from s in $O(n^2)$ time by Dijkstra's algorithm. Let $V_i = \{v \in V : D[s, v] \in [i, i + 1)\}$. For any interval I, let $V_I = \bigcup_{i \in I} V_i$. Let $n_i = |V_{[i-2c-3,i+2c+3]}|$; note that $\sum_i n_i = O(n)$.

We describe an algorithm to compute a partial matrix $\widetilde{D}$ of approximate distances. For subsets $S_1, S_2 \subseteq V$, let $\widetilde{D}(S_1, S_2)$ denote the submatrix of $\widetilde{D}$ containing only entries for $(u, v) \in S_1 \times S_2$.

Step 0. For each i, we compute $\widetilde{D}(V_{i-1}, V_i)$ by applying Lemma 1 to approximate distances between all $u \in V_{i-1}$ and all $v \in V_i$ that are bounded by $2c + 1$, in the subgraph induced by $V_{[i-2c-3,i+2c+3]}$ (which has size $O(n_i)$), with additive error $O(\frac{\varepsilon' n_i}{n})$. (More precisely, for $u \in V_{i-1}$ and $v \in V_i$, if $\widetilde{D}[u, v] \neq \infty$, then the computed value $\widetilde{D}[u, v]$ is a valid approximation; and if u and v have distance at most $2c + 1$ in the induced subgraph, then it is guaranteed that $\widetilde{D}[u, v] \neq \infty$.) For all i with $n_i \leq n/B$, the total running time is

$$\begin{aligned}
\tilde{O}\left(\sum_i \frac{n}{\varepsilon' n_i} \cdot n_i^{\omega}\right) &= \tilde{O}\left(\sum_i \frac{n}{\varepsilon'} \cdot n_i^{\omega-1}\right) \\
&\leq \tilde{O}\left(\sum_i \frac{n}{\varepsilon'} \cdot (n/B)^{\omega-2} \cdot n_i\right) \\
&= \tilde{O}(\tfrac{n}{\varepsilon'} \cdot (n/B)^{\omega-2} \cdot n) \;=\; \tilde{O}((1/\varepsilon')n^{\omega}/B^{\omega-2}).
\end{aligned}$$

In the case when $n_i > n/B$, we apply Lemma 1 instead to the original graph with additive error $O(\varepsilon'/B)$, which is at least as good as $O(\frac{\varepsilon' n_i}{n})$, in $\tilde{O}((1/\varepsilon')Bn^{\omega})$ time – note that this can be done just once during preprocessing.

Step 1. For $i = 0, \ldots, c$, we compute $\widetilde{D}(S, V_i)$ by using Lemma 1 to approximate all distances bounded by $O(c)$, with additive error $O(\varepsilon')$.

For each $i = c+1, \ldots, t$, we compute $\widetilde{D}(S, V_i)$ by taking the min-plus product $\widetilde{D}(S, V_{i-1}) \star \widetilde{D}(V_{i-1}, V_i)$, with additive error $O(\frac{\varepsilon' n_i}{n})$. We do the following *filtering* step: for each entry $\widetilde{D}[x, y]$ just computed, if $\widetilde{D}[x, y] \notin D[s, y] - D[s, x] \pm (2c + O(\varepsilon))$, reset $\widetilde{D}[x, y] = \infty$. Because of the filtering step, for all $u \in S$, $z \in V_{i-1}$, and $v \in V_i$, we have $\widetilde{D}[u, z] \in i \pm O(c)$ if it is finite, and $\widetilde{D}[z, v] \in O(c)$ if it is finite. Thus, in computing $\widetilde{D}(S, V_{i-1}) \star \widetilde{D}(V_{i-1}, V_i)$, we can make the matrix entries lie in $O(c)$ by shifting. We can then round entries to multiples of $\frac{n}{\varepsilon' n_i}$. So, the total time to compute the products for all $i = c+1, \ldots, t$ is

$$\begin{aligned}
\tilde{O}\left(\sum_{i=1}^{t} \mathcal{M}^{\star}(n/t, n_i, n_i \mid \tfrac{n}{\varepsilon' n_i})\right) &\le \tilde{O}\left(\sum_{i=1}^{t} \tfrac{n}{\varepsilon' n_i} \cdot \mathcal{M}(n/t, n_i, n_i)\right) \\
&\le \tilde{O}\left(\sum_{i=1}^{t} \tfrac{n}{\varepsilon' n_i} \cdot (\tfrac{n/t}{n_i} n_i^{\omega} + (\tfrac{n_i}{n/t})^2 (n/t)^{\omega})\right) \\
&= \tilde{O}\left(\tfrac{1}{\varepsilon'} \sum_{i=1}^{t} ((n^2/t) n_i^{\omega-2} + (n^{\omega-1}/t^{\omega-2}) n_i)\right) \\
&\le \tilde{O}(\tfrac{1}{\varepsilon'} \cdot ((n^2/t) n^{\omega-2} t^{3-\omega} + (n^{\omega-1}/t^{\omega-2}) n)) \\
&= \tilde{O}((1/\varepsilon') n^{\omega}/t^{\omega-2}).
\end{aligned}$$

Step 2. Now, assume that $\widetilde{D}(S, V_i)$ has been computed for all $i = 0, \ldots, \ell/2$ for a given $\ell \ge 2t$. We will compute $\widetilde{D}(S, V_i)$ for all $i = \ell/2+1, \ldots, \ell$. First pick an $i_0 \le \ell/2$ with $|V_{i_0}| = O(n/\ell)$. For each $i = i_0+1, \ldots, \ell$, we compute $\widetilde{D}(V_{i_0}, V_i)$ by taking the min-plus product $\widetilde{D}(V_{i_0}, V_{i-1}) \star \widetilde{D}(V_{i-1}, V_i)$, with additive error $O(\frac{\varepsilon' n_i}{n})$. Do the filtering step as before. Because of the filtering step, for all $u \in V_{i_0}$, $z \in V_{i-1}$, and $v \in V_i$, we have $\widetilde{D}[u, z] \in i - i_0 \pm O(c)$ if it is finite, and $\widetilde{D}[z, v] \in O(c)$ if it is finite. By a similar analysis, the total time to compute these products for $i = i_0+1, \ldots, \ell$ is upper-bounded by

$$\tilde{O}\left(\sum_{i=1}^{\ell} \mathcal{M}^{\star}(n/\ell, n_i, n_i \mid \tfrac{n}{\varepsilon' n_i})\right) \le \tilde{O}((1/\varepsilon') n^{\omega}/\ell^{\omega-2}) \le \tilde{O}((1/\varepsilon') n^{\omega}/t^{\omega-2}).$$

Finally, we compute $\widetilde{D}(S, V_{(\ell/2,\ell]})$ by taking the min-plus product $\widetilde{D}(S, V_{i_0})$ and $\widetilde{D}(V_{i_0}, V_{(\ell/2,\ell]})$ with additive error $O(\varepsilon')$. Do the filtering step as before. Because of the filtering step, for all $u \in S$, $z \in V_{i_0}$, and $v \in V_{(\ell/2,\ell]}$, we have $\widetilde{D}[u, z] \in i_0 \pm O(c)$ if it is finite, and $\widetilde{D}[z, v] \in D[s, v] - i_0 \pm O(c)$ if it is finite. We can again make the matrix entries lie in $O(c)$ by shifting. This product takes time

$$\begin{aligned}
\tilde{O}(\mathcal{M}^{\star}(n/t, n/\ell, n \mid 1/\varepsilon')) &\le \tilde{O}((1/\varepsilon')\mathcal{M}(n/t, n/t, n)) \\
&\le \tilde{O}((1/\varepsilon') t (n/t)^{\omega}) = \tilde{O}((1/\varepsilon') n^{\omega}/t^{\omega-1}).
\end{aligned}$$

We repeat the above for all $\ell \ge 2t$ that are powers of 2.

Correctness. For every $u \in S$ and $v \in V$, we claim that $\widetilde{D}[u, v]$ approximates $D[u, v]$ with additive error $O(\varepsilon)$. To see this, let π be the shortest path from u to v. For any subpath of π, say, from x to y, we have $D[x, y] = D[u, y] - D[u, x] \in D[s, y] - D[s, x] \pm 2c$, since $D[s, u] \le c$ and the graph is undirected (this justifies the filtering step). If $u \in S$ and $v \in V_i$ with $i > c$, then π must pass through a vertex $z \in V_{i-1}$. For every node z' in the subpath from z to v, $D[z, z'] \le D[u, v] - D[u, z] \le D[s, v] - D[s, z] + 2c \le 2c + 2$, and $D[s, z'] \in D[s, z] \pm D[z, z'] \in i \pm (2c + 3)$, so the subpath lies in the subgraph induced by $V_{[i-2c-3, i+2c+3]}$. It follows that the total additive error in Step 1 is $O(\sum_i \frac{\varepsilon' n_i}{n}) = O(\varepsilon')$. The analysis of Step 2 is similar: if $u \in S$ and $v \in V_i$ with $i > \ell/2$, then π must pass through a vertex $u' \in V_{i_0}$, and the subpath from u' to v must pass through a vertex $z \in V_{i-1}$. The overall additive error is bounded by $O(\varepsilon' \log n) = O(\varepsilon)$. ◀

5 Overall Algorithm

Our overall algorithm employs a careful combination of Aingworth et al.'s technique [1] involving low- vs. high-degree vertices, and Zwick's technique [23] involving short vs. long paths, and a judicious choice of several parameters to balance cost.

Let B, L, and t be parameters to be set later. Let V_{high} be the set of all vertices of degree more than n/t, and V_{low} be the set of all vertices of degree at most n/t.

Phase 1. We will first compute an approximation $\widetilde{D}[u,v]$ to $D[u,v]$ for all $u \in V_{\text{high}}$ and $v \in V$, as follows:

Let $X \subseteq V$ be a dominating set for V_{high}, such that every vertex in V_{high} is in the (closed) neighborhood of some vertex in X. As noted by Aingworth et al. [1], there exists such a dominating set of size $\tilde{O}(t)$, and it can be found easily by random sampling, or deterministically by a greedy algorithm.

Consequently, we can cover V_{high} by $\tilde{O}(t)$ groups of vertices, such that vertices of each group have distance at most 2 from each other, by taking the neighborhoods of the vertices of X. We may assume that these groups are disjoint (for example, by removing from the i-th group those vertices that appear in the first $i-1$ groups). Furthermore, we may assume that every group has size $O(n/t)$ (by subdividing the groups, which only increases the number of groups by $O(t)$). For each group S, we apply Lemma 2 (with $c=2$). The total time of these $\tilde{O}(t)$ invocations of the lemma is $\tilde{O}(t \cdot (1/\varepsilon)n^{\omega}/B^{\omega-2})$, assuming that $B \le t$, after $\tilde{O}((1/\varepsilon)Bn^{\omega})$-time preprocessing.

Phase 2. Let $R \subseteq V$ be a subset of vertices that hits all shortest paths with at least L edges. As shown by Zwick [23], there exists such a hitting set of size $\tilde{O}(n/L)$, and it can be found by random sampling, or deterministically. We will next compute an approximation $\widehat{D}[u,v]$ to $D[u,v]$ for all $u \in R$ and $v \in V$ as follows:

Fix $u \in R$. Define a graph G_u containing all edges xy with $x \in V_{\text{low}}$ or $y \in V_{\text{low}}$; for each $z \in V_{\text{high}}$, we add an extra edge uz with weight $\widetilde{D}[u,z]$, which has been computed in Phase 1. Then the distance from u to v in G_u approximates the distance in G (because if $\langle u_1,\ldots,u_k\rangle$ is a shortest path in G with $u_1=u$, and i is the largest index with $u_i \in V_{\text{high}}$, then $\langle u_1,u_i,\ldots,u_k\rangle$ is a path in G_u). We run Dijkstra's algorithm on G_u from the source u. Since G_u has $O(n^2/t)$ edges, this takes $\tilde{O}(n^2/t)$ time per u. The total over all $u \in R$ is $\tilde{O}((n/L)\cdot(n^2/t)) = \tilde{O}(n^3/(tL))$.

Phase 3. We now approximate $D[u,v]$ for all $u,v \in V$ as follows.

For (u,v) with $D[u,v] \le L$, we use Lemma 1, which takes $\tilde{O}((1/\varepsilon)Ln^{\omega})$ time.

For (u,v) with $D[u,v] > L$, recall that we have computed $\widehat{D}(V,R)$ from Phase 2. For $u \in V$ and $z \in R$, let $D'[u,z] = \widehat{D}[u,z]$ if $\widehat{D}[u,z] \le L + O(\varepsilon)$, and $D'[u,z] = \infty$ otherwise. For $z \in R$ and $v \in V$, let $D''_a[z,v] = (\widehat{D}[z,v] + a) \bmod 10L$. Compute the min-plus product $D'(V,R) \star D''_a(R,V)$ with additive error $O(\varepsilon)$ for $a=0$ and for $a=5L$. Since the finite entries after rescaling are integers bounded by $O(L/\varepsilon)$, this takes $\tilde{O}(\mathcal{M}^{\star}(n,n/L,n \mid L/\varepsilon)) \le \tilde{O}((1/\varepsilon)Ln^{\omega})$ time. For each (u,v), we remember which z gives the minimum for the two products, and take the one with the smaller $\widehat{D}[u,z] + \widehat{D}[z,v]$ among the two.

To justify correctness, consider a pair u,v with $D[u,v] \ge L$. Consider a shortest path π from u to v. There exists a vertex $z^* \in R$ among the first L vertices in π. Thus, $D[u,z^*] \le L$. Furthermore, among all $z \in R$ with $D[u,z] \le L + O(\varepsilon)$, we have $D[z,v]$ lying in an interval I of length $2L + O(\varepsilon)$, since the graph is undirected. For either $a=0$ or $a=5L$, the shifted interval $I+a$ would be completely contained in $[L,9L]$ modulo $10L$, and so the minimum of $\widehat{D}[u,z] + \widehat{D}[z,v]$ would be correctly computed, with additive error $O(\varepsilon)+O(\varepsilon)=O(\varepsilon)$.

Total time. The overall running time (ignoring poly$(1/\varepsilon)$ factors) is

$$\tilde{O}(Bn^\omega + tn^\omega/B^{\omega-2} + n^3/(tL) + Ln^\omega).$$

Choosing $t = B^{\omega-1}$ and $L = B$ (noting that indeed $B \le t$) gives $\tilde{O}(Bn^\omega + n^3/B^\omega)$. Finally, choosing $B = n^{(3-\omega)/(\omega+1)}$ gives $\tilde{O}(n^{\omega+(3-\omega)/(\omega+1)}) = O(n^{(3+\omega^2)/(\omega+1)}) = O(n^{2.559})$.

Standard techniques for generating witnesses for matrix products can be applied to recover the approximate shortest paths [14, 23].

▶ **Theorem 3.** *Given an undirected graph with n vertices and real edge weights in $[0,1]$, we can solve the approximate APSP problem with additive error $O(\varepsilon)$ in $\tilde{O}(n^{(3+\omega^2)/(\omega+1)}) = O(n^{2.559})$ time, for any constant $\varepsilon > 0$.*

6 Final Remarks

It remains open whether an $\tilde{O}(n^2)$-time algorithm for undirected real-weighted graphs is possible if $\omega = 2$, even with a large constant additive error.

Under the current bounds on matrix multiplication, could our $O(n^{2.559})$ result be further improved? At the moment, we don't know how to use rectangular matrix multiplication to speed up our algorithm. And we don't know either how to use rectangular matrix multiplication to speed up Yuster's $O(n^{2.687})$-time algorithm for directed graphs [22] (ideally, to match up with Zwick's unweighted exact APSP algorithm [23], which has running time $O(n^{2.529})$ via the latest bounds on rectangular matrix multiplication [17]).

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.

2 Josh Alman and Virginia Vassilevska Williams. A refined laser method and faster matrix multiplication. In *Proceedings of the 32nd Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 522–539, 2021. doi:10.1137/1.9781611976465.32.

3 Noga Alon, Zvi Galil, and Oded Margalit. On the exponent of the all pairs shortest path problem. *J. Comput. Syst. Sci.*, 54(2):255–262, 1997.

4 Surender Baswana, Telikepalli Kavitha, Kurt Mehlhorn, and Seth Pettie. Additive spanners and (α,β)-spanners. *ACM Trans. Algorithms*, 7(1):5:1–5:26, 2010. doi:10.1145/1868237.1868242.

5 Karl Bringmann, Marvin Künnemann, and Karol Wegrzycki. Approximating APSP without scaling: equivalence of approximate min-plus and exact min-max. In *Proceedings of the 51st Annual ACM Symposium on Theory of Computing (STOC)*, pages 943–954, 2019.

6 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.

7 Timothy M. Chan. All-pairs shortest paths for unweighted undirected graphs in $o(mn)$ time. *ACM Trans. Algorithms*, 8(4):34:1–34:17, 2012. doi:10.1145/2344422.2344424.

8 Timothy M. Chan and R. Ryan Williams. Deterministic APSP, orthogonal vectors, and more: Quickly derandomizing Razborov-Smolensky. *ACM Trans. Algorithms*, 17(1):2:1–2:14, 2021. doi:10.1145/3402926.

9 Timothy M. Chan, Virginia Vassilevska Williams, and Yinzhan Xu. Algorithms, reductions and equivalences for small weight variants of all-pairs shortest paths. *CoRR*, abs/2102.06181, 2021. To appear in ICALP'21. arXiv:2102.06181.

10 Shiri Chechik. New additive spanners. In *Proceedings of the 24th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 498–512, 2013. doi:10.1137/1.9781611973105.36.

11 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.

12 Dorit Dor, Shay Halperin, and Uri Zwick. All-pairs almost shortest paths. *SIAM J. Comput.*, 29(5):1740–1759, 2000.

13 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.

14 Zvi Galil and Oded Margalit. Witnesses for Boolean matrix multiplication and for transitive closure. *J. Complex.*, 9(2):201–221, 1993.

15 Zvi Galil and Oded Margalit. All pairs shortest paths for graphs with small integer length edges. *J. Comput. Syst. Sci.*, 54(2):243–254, 1997. doi:10.1006/jcss.1997.1385.

16 Fabrizio Grandoni, Giuseppe F. Italiano, Aleksander Lukasiewicz, Nikos Parotsidis, and Przemyslaw Uznanski. All-pairs LCA in DAGs: Breaking through the $O(n^{2.5})$ barrier. In *Proceedings of the 32nd Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 273–289, 2021. doi:10.1137/1.9781611976465.18.

17 Francois Le Gall and Florent Urrutia. Improved rectangular matrix multiplication using powers of the Coppersmith-Winograd tensor. In *Proceedings of the 29th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1029–1046, 2018.

18 Liam Roditty and Asaf Shapira. All-pairs shortest paths with a sublinear additive error. In *Proceedings of the 35th International Colloquium on Automata, Languages and Programming (ICALP), Part I*, pages 622–633, 2008.

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

20 Virginia Vassilevska Williams. On some fine-grained questions in algorithms and complexity. In *Proceedings of the International Congress of Mathematicians (ICM)*, pages 3447–3487, 2018.

21 R. Ryan Williams. Faster all-pairs shortest paths via circuit complexity. *SIAM J. Comput.*, 47(5):1965–1985, 2018. doi:10.1137/15M1024524.

22 Raphael Yuster. Approximate shortest paths in weighted graphs. *J. Comput. Syst. Sci.*, 78(2):632–637, 2012.

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