# Algorithms, Reductions and Equivalences for Small Weight Variants of All-Pairs Shortest Paths

**Timothy M. Chan**
University of Illinois at Urbana-Champaign, IL, USA

**Virginia Vassilevska Williams**
MIT, CSAIL, Cambridge, MA, USA

**Yinzhan Xu**
MIT, Cambridge, MA, USA

## Abstract

All-Pairs Shortest Paths (APSP) is one of the most well studied problems in graph algorithms. This paper studies several variants of APSP in unweighted graphs or graphs with small integer weights.

APSP with small integer weights in undirected graphs [Seidel'95, Galil and Margalit'97] has an $\tilde{O}(n^\omega)$ time algorithm, where $\omega < 2.373$ is the matrix multiplication exponent. APSP in directed graphs with small weights however, has a much slower running time that would be $\Omega(n^{2.5})$ even if $\omega = 2$ [Zwick'02]. To understand this $n^{2.5}$ bottleneck, we build a web of reductions around directed unweighted APSP. We show that it is *fine-grained equivalent* to computing a rectangular Min-Plus product for matrices with integer entries; the dimensions and entry size of the matrices depend on the value of $\omega$. As a consequence, we establish an equivalence between APSP in directed unweighted graphs, APSP in directed graphs with small ($\tilde{O}(1)$) integer weights, All-Pairs Longest Paths in DAGs with small weights, $c$Red-APSP in undirected graphs with small weights, for any $c \geq 2$ (computing all-pairs shortest path distances among paths that use at most $c$ red edges), $\#_{\leq c}$APSP in directed graphs with small weights (counting the number of shortest paths for each vertex pair, up to $c$), and approximate APSP with additive error $c$ in directed graphs with small weights, for $c \leq \tilde{O}(1)$.

We also provide fine-grained reductions from directed unweighted APSP to All-Pairs Shortest Lightest Paths (APSLP) in undirected graphs with $\{0, 1\}$ weights and $\#_{\bmod c}$APSP in directed unweighted graphs (computing counts mod $c$), thus showing that unless the current algorithms for APSP in directed unweighted graphs can be improved substantially, these problems need at least $\Omega(n^{2.528})$ time.

We complement our hardness results with new algorithms. We improve the known algorithms for APSLP in directed graphs with small integer weights (previously studied by Zwick [STOC'99]) and for approximate APSP with sublinear additive error in directed unweighted graphs (previously studied by Roditty and Shapira [ICALP'08]). Our algorithm for approximate APSP with sublinear additive error is optimal, when viewed as a reduction to Min-Plus product. We also give new algorithms for variants of #APSP (such as $\#_{\leq U}$APSP and $\#_{\bmod U}$APSP for $U \leq n^{\tilde{O}(1)}$) in unweighted graphs, as well as a near-optimal $\tilde{O}(n^3)$-time algorithm for the original #APSP problem in unweighted graphs (when counts may be exponentially large). This also implies an $\tilde{O}(n^3)$-time algorithm for Betweenness Centrality, improving on the previous $\tilde{O}(n^4)$ running time for the problem. Our techniques also lead to a simpler alternative to Shoshan and Zwick's algorithm [FOCS'99] for the original APSP problem in undirected graphs with small integer weights.

**2012 ACM Subject Classification** Theory of computation → Shortest paths

**Keywords and phrases** All-Pairs Shortest Paths, Fine-Grained Complexity, Graph Algorithm

**Digital Object Identifier** 10.4230/LIPIcs.ICALP.2021.47

**Category** Track A: Algorithms, Complexity and Games

**Related Version** *Full Version*: `https://arxiv.org/abs/2102.06181`

**Funding** *Timothy M. Chan*: Supported by NSF Grant CCF-1814026.
*Virginia Vassilevska Williams*: Supported by an NSF CAREER Award, NSF Grants CCF-1528078, CCF-1514339 and CCF-1909429, a BSF Grant BSF:2012338, a Google Research Fellowship and a Sloan Research Fellowship.
*Yinzhan Xu*: Supported by NSF Grant CCF-1528078.

## 1 Introduction

*All-Pairs Shortest Paths (APSP)* is one of the oldest and most studied problems in graph algorithms. The fastest known algorithm for general $n$-node graphs runs in $n^3/2^{\Theta(\sqrt{\log n})}$ [31]. In unweighted graphs, or graphs with small integer weights, faster algorithms are known.

For APSP in *undirected* unweighted graphs (u-APSP), Seidel [21] and Galil and Margalit [12, 13] gave $\tilde{O}(n^\omega)$ time algorithms where $\omega \le 2.373$ is the exponent of matrix multiplication [2, 27, 16]; the latter algorithm works for graphs with small integer weights[1] in $[\pm c_0]$ for $c_0 = \tilde{O}(1)$. The hidden dependence on $c_0$ was improved by Shoshan and Zwick [22].

For *directed* unweighted graphs or graphs with weights in $[\pm c_0]$, the fastest APSP algorithm is by Zwick [34], running in $O(n^{2.529})$ time. This running time is achieved using the best known bounds for rectangular matrix multiplication [17] and would be $\Omega(n^{2.5})$ even if $\omega = 2$.

There is a big discrepancy between the running times for undirected and directed APSP. One might wonder, why is this? Are directed graphs inherently more difficult for APSP, or is there some special graph structure we can uncover and then use it to develop an $\tilde{O}(n^\omega)$ time algorithm for directed APSP as well? (Note that matrix multiplication seems necessary for APSP since APSP is known to capture Boolean matrix multiplication.)

The first contribution in this paper is a fine-grained equivalence between directed unweighted APSP (u-APSP) and a certain rectangular version of the Min-Plus product problem.

The *Min-Plus product* of an $n \times m$ matrix $A$ by an $m \times p$ matrix $B$ is the matrix $C$ with entries $C[i,j] = \min_{k=1}^{m}(A[i,k] + B[k,j])$. Let us denote by $\mathrm{M}^\star(n_1, n_2, n_3 \mid M)$ the problem of computing the Min-Plus product of an $n_1 \times n_2$ matrix by an $n_2 \times n_3$ matrix where both matrices have integer entries in $[M]$. Let $\mathcal{M}^\star(n_1, n_2, n_3 \mid M)$ be the best running time for $\mathrm{M}^\star(n_1, n_2, n_3 \mid M)$.

Zwick's algorithm [34] for u-APSP can be viewed as making a logarithmic number of calls to the Min-Plus product $\mathrm{M}^\star(n, n/L, n \mid L)$ for all $1 \le L \le n$ that are powers of $3/2$. The running time of Zwick's algorithm is thus, within polylogarithmic factors, $\max_L \mathcal{M}^\star(n, n/L, n \mid L)$.

Let $\mathcal{M}(a, b, c)$ denote the running time of the fastest algorithm to multiply an $a \times b$ by a $b \times c$ matrix over the integers. Let $\omega(a, b, c)$ be the smallest real number $r$ such that $\mathcal{M}(n^a, n^b, n^c) \le O(n^{r+\varepsilon})$ for all $\varepsilon > 0$.

The best known upper bound for the Min-Plus product running time $\mathcal{M}^\star(n, n/L, n \mid L)$ is the minimum of $O(n^3/L)$ (the brute force algorithm) and $\tilde{O}(L \cdot \mathcal{M}(n, n/L, n))$ [3]. For $L = n^{1-\ell}$, $\mathcal{M}^\star(n, n/L, n \mid L)$ is thus at most $\tilde{O}(\min\{n^{2+\ell}, n^{1-\ell+\omega(1,\ell,1)}\})$. Over all $\ell \in [0, 1]$, the runtime is maximized at $\tilde{O}(n^{2+\rho})$ where $\rho$ is such that $\omega(1, \rho, 1) = 1 + 2\rho$.

Hence in particular, the running time of Zwick's algorithm is $\tilde{O}(n^{2+\rho})$. This running time has remained unchanged (except for improvements on the bounds on $\rho$) for almost 20 years. The current best known bound on $\rho$ is $\rho < 0.529$, and if $\omega = 2$, then $\rho = 1/2$.

[1] In this paper, $[\pm c_0] = \{-c_0, \dots, c_0\}$ and $[c_0] = \{0, \dots, c_0\}$. The $\tilde{O}$ notation hides polylogarithmic factors (although conditions of the form $c_0 = \tilde{O}(1)$ may be relaxed to $c_0 \le n^{o(1)}$ if we allow extra $n^{o(1)}$ factors in the $\tilde{O}$ time bounds).

Our first result is that u-APSP is sub-$n^{2+\rho}$ fine-grained equivalent to $\mathrm{M}^\star(n, n^\rho, n \mid n^{1-\rho})$:

▶ **Theorem 1.** *If $\mathrm{M}^\star(n, n^\rho, n \mid n^{1-\rho})$ is in $O(n^{2+\rho-\varepsilon})$ time for $\varepsilon > 0$, then u-APSP can also be solved in $O(n^{2+\rho-\varepsilon'})$ time for some $\varepsilon' > 0$. If u-APSP can be solved in $O(n^{2+\rho-\varepsilon})$ time for some $\varepsilon > 0$, then $\mathrm{M}^\star(n, n^\rho, n \mid n^{1-\rho})$ can also be solved in $O(n^{2+\rho-\varepsilon})$ time.*

The Min-Plus product of two $n \times n$ matrices with *arbitrary* integer entries is known to be equivalent to APSP with *arbitrary* integer entries [10], so that their running times are the same, up to constant factors. All known algorithms for directed unweighted APSP (including [34, 3] and others), make calls to Min-Plus product of rectangular matrices with integer entries that can be as large as say $n^{0.4}$. It is completely unclear, however, why a problem in *unweighted* graphs such as u-APSP should require the computation of Min-Plus products of matrices with such large entries. Theorem 1 surprisingly shows that it does. Moreover, it shows that unless we can improve upon the known approaches for Min-Plus product computation, there will be no way to improve upon Zwick's algorithm for u-APSP. The latter is an algebraic problem in disguise.

The main proof of Theorem 1 is simple – what is remarkable are the numerous consequences on equivalences and conditional hardness that follow from this idea. We first use Theorem 1 to build a class of problems that are all equivalent to u-APSP, via $(n^{2+\rho}, n^{2+\rho})$-fine-grained reductions (see [29] for a survey of fine-grained complexity). In particular, if $\omega = 2$ (or more generally when $\omega(1, \frac{1}{2}, 1) = 2$), these are all problems that are $n^{2.5}$-fine-grained equivalent.

Recall that in the *All-Pairs Longest Paths (APLP)* problem, we want to output for every pair of vertices $s, t$ the weight of the longest path from $s$ to $t$. While APLP is NP-hard in general, it is efficiently solvable in DAGs. In the *cRed-APSP* problem, for a given graph in which some edges can be colored red, we want to output for every pair of vertices $s, t$ the weight of the shortest path from $s$ to $t$ that uses at most $c$ red edges. For convenience, we call all non-red edges blue.

We use the following convention for problem names: the prefix "u-" is for unweighted graphs; the prefix "$[c_0]$-" is for graphs with weights in $[c_0]$ (similarly for "$[\pm c_0]$-" and for other ranges). Input graphs are directed unless stated otherwise.

▶ **Theorem 2.** *The following problems either all have $O(n^{2+\rho-\varepsilon})$ time algorithms for some $\varepsilon > 0$, or none of them do, assuming that $c_0 = \tilde{O}(1)$:*

- $\mathrm{M}^\star(n, n^\rho, n \mid n^{1-\rho})$,
- *u-APSP,*
- *$[\pm c_0]$-APSP for directed graphs without negative cycles,*
- *u-APLP for DAGs,*
- *$[\pm c_0]$-APLP for DAGs,*
- *u-cRed-APSP for* undirected *graphs for any $2 \leq c \leq \tilde{O}(1)$.*

Interestingly, while u-2Red-APSP in undirected graphs above is equivalent to u-APSP and hence improving upon its $\tilde{O}(n^{2+\rho})$ runtime would be difficult, we show that u-1Red-APSP in undirected graphs can be solved in $\tilde{O}(n^\omega)$ time via a modification of Seidel's algorithm, and hence there is a seeming jump in complexity in u-$c$Red-APSP from $c = 1$ to $c = 2$.

Besides the above equivalences we provide some interesting reductions from u-APSP to other well-studied matrix product and shortest paths problems.

Lincoln, Polak and Vassilevska W. [18] reduce u-APSP to some matrix product problems such as All-Edges Monochromatic Triangle and the (min, max)-Product studied in [24, 26] and [25, 9] respectively. Using the equivalence of u-APSP and $\mathrm{M}^\star(n, n/\ell, n \mid \ell^{1-p})$, we can reduce u-APSP to another matrix product called Min Witness Equality Product (MinWitnessEq), where we are given $n \times n$ integer matrices $A$ and $B$, and are required to compute $\min\{k \in [n] : A[i,k] = B[k,j]\}$ for every pair of $(i,j)$. This can be viewed as a merge of the Min Witness product [8] [2] and Equality Product problems [15, 28].

Another natural variant of APSP is the problem of *approximating* shortest path distances. Zwick [34] presented an $\tilde{O}(n^\omega \log M)$ time algorithm for computing a $(1+\varepsilon)$-multiplicative approximation for all pairwise distances in a directed graph with integer weights in $[M]$, for any constant $\varepsilon > 0$.[3] This is essentially optimal assuming no $o(n^\omega)$ time algorithm can multiple $n \times n$ Boolean matrices since any such approximation algorithm can be used to multiply Boolean matrices.

An arguably better notion of approximation is to provide an additive approximation, i.e. outputting for every $u, v$ an estimate $D'[u,v]$ for the distance $D[u,v]$ such that $D[u,v] \leq D'[u,v] \leq D[u,v] + E$, where $E$ is an error that can depend on $u$ and $v$.

At ICALP'08, Roditty and Shapira [20] studied the following variant: given an unweighted directed graph and a constant $p \in [0,1]$, compute for all $u, v$ an estimate $D'[u,v]$ with $D[u,v] \leq D'[u,v] \leq D[u,v] + D[u,v]^p$. They gave an algorithm with running time $\tilde{O}(\max_\ell \min\{n^3/\ell, \mathcal{M}^\star(n, n/\ell^{1-p}, n \mid \ell^{1-p})\})$. For example, for $p = 0$, this matches the time complexity of Zwick's exact algorithm for u-APSP; for $p = 1$, this matches Zwick's $\tilde{O}(n^\omega)$-time algorithm with constant multiplicative approximation factor. For $p = 0.5$, with the current rectangular matrix multiplication bounds [17], the running time is $O(n^{2.447})$.

We obtain an improved running time:

▶ **Theorem 3.** *For any $p \in [0,1]$, given a directed unweighted graph, one can obtain additive $D[u,v]^p$ approximations to all distances $D[u,v]$ in time $\tilde{O}(\max_\ell \mathcal{M}^\star(n, n/\ell, n \mid \ell^{1-p}))$.*

The improvement over Roditty and Shapira's running time is substantial. For example, for all $p \geq 0.415$, the time bound is $O(n^{2.373})$ (the current matrix multiplication running time), whereas their algorithm only achieves $O(n^{2.373})$ for $p = 1$. Our result also answers one of Roditty and Shapira's open question (on whether $\tilde{O}(n^\omega)$ time is possible for any $p < 1$), if $\omega > 2$.

The new algorithm is also *optimal* (ignoring logarithmic factors) in a strong sense, as our reduction technique shows that for all $\ell$, $\mathrm{M}^\star(n, n/\ell, n \mid \ell^{1-p})$ can be tightly reduced to the additive $D[u,v]^p$ approximation of APSP. In particular, u-APSP with constant additive error is fine-grained equivalent to exact u-APSP.

The *All-Pairs Lightest Shortest Paths (APLSP)* problem studied in [6, 33] asks to compute for every pair of vertices $s, t$ the distance from $s$ to $t$ (with respect to the edge weights) and the smallest number of edges over all shortest paths from $s$ and $t$. Traditional shortest-path algorithms can be easily modified to find the lightest shortest paths, but not the faster matrix-multiplication-based algorithms. Our reduction for u-$c$Red-APSP can be easily modified to reduce $\mathrm{M}^\star(n, n^\rho, n \mid n^{1-\rho})$ to $\{0,1\}$-APLSP in undirected graphs, which can be viewed as a conditional lower bound of $n^{2+\rho-o(1)}$ for the latter problem.

▶ **Corollary 4.** *If $\{0,1\}$-APLSP in undirected graphs is in $O(n^{2+\rho-\varepsilon})$ time for $\varepsilon > 0$, then so is $\mathrm{M}^\star(n, n^\rho, n \mid n^{1-\rho})$.*

[2] Recently, there has been renewed interest in studying the Min Witness product, due to a breakthrough [14] on the All-Pairs LCA in DAGs problem, which was one of the original motivations for studying Min Witness.

[3] Bringmann et al. [5] considered the more unusual setting of very large $M$, where the $\log M$ factor is to be avoided.

The fastest known algorithm to date for $\{0,1\}$-APLSP, or more generally, $[c_0]$-APLSP for $c_0 = \tilde{O}(1)$, for directed or undirected graphs is by Zwick [33] from STOC'99 and runs in $O(n^{2.724})$ time with the current best bounds for rectangular matrix multiplication (the running time would be $\tilde{O}(n^{8/3})$ if $\omega = 2$). Chan [6] (STOC'07) improved this running time to $\tilde{O}(n^{(3+\omega)/2}) \le O(n^{2.687})$ but only if the weights are *positive*, i.e., for $([c_0] - \{0\})$-APLSP (and so his result does not hold for $\{0,1\}$-APLSP).

Both Zwick's and Chan's algorithms solve a more general problem, Lex$_2$-APSP, in which one is given a directed graph where each edge $e$ is given two weights $w_1(e), w_2(e)$ and one wants to find for every pair of vertices $u, v$ the lexicographic minimum over all $u$-$v$ paths $\pi$ of $(\sum_{e\in\pi} w_1(e), \sum_{e\in\pi} w_2(e))$. Then APLSP is Lex$_2$-APSP when all $w_2$ weights are 1, and the related *All-Pairs Shortest Lightest Paths (APSLP)* problem is when all $w_1$ weights are 1.

To complement the conditional lower bound for APLSP, and hence Lex$_2$-APSP, we present new algorithms for $[c_0]$-Lex$_2$-APSP for $c_0 = \tilde{O}(1)$, both (slightly) improving Chan's running time and also allowing zero weights, something that Chan's algorithm couldn't support.

▶ **Theorem 5.** *$[c_0]$-Lex$_2$-APSP can be solved in $O(n^{2.66})$ time for any $c_0 = \tilde{O}(1)$.*

If $\omega = 2$, the above running time would be $\tilde{O}(n^{2.5})$, improving Zwick's previous $\tilde{O}(n^{8/3})$ bound [33] and matching our conditional lower bound $n^{2+\rho-o(1)}$. For undirected graphs with *positive* weights in $[c_0] - \{0\}$, we further improve the running time to $O(n^{2.58})$ under the current matrix multiplication bounds.

We next consider the natural problem, *#APSP*, of counting the number of shortest paths for every pair of vertices in a graph. This problem needs to be solved, for example, when computing the so-called *Betweenness Centrality (BC)* of a vertex. BC is a well-studied measure of vertex importance in social networks. If we let $C[s,t]$ be the number of shortest paths between $s$ and $t$, and $C_v[s,t]$ be the number of shortest paths between $s$ and $t$ that go through $v$, then $\text{BC}(v) = \sum_{s,t\neq v} C_v[s,t]/C[s,t]$ and the BC problem is to compute BC$(v)$ for a given graph and a given node $v$.

Prior work [4] showed that #APSP and BC in $m$-edge $n$-node unweighted graphs can be computed in $O(mn)$ time via a modification of Breadth-First Search (BFS).[4] However, all prior algorithms assumed a model of computation where adding two integers of arbitrary size takes constant time. In the more realistic word-RAM model (with $O(\log n)$ bit words), these algorithms would run in $\tilde{\Theta}(mn^2)$ time, as there are explicit examples of graphs with $m$ edges (for any $m$, a function of $n$) for which the shortest paths counts have $\Theta(n)$ bits.[5] In particular, the best running time in terms of $n$ so far has been $\tilde{O}(n^4)$.

We provide the first genuinely $\tilde{O}(n^3)$ time algorithm for #APSP, and thus Betweenness Centrality, in directed unweighted graphs.

▶ **Theorem 6.** *u-#APSP can be solved in $\tilde{O}(n^3)$ time by a combinatorial algorithm.*

This runtime cannot be improved since there are graphs for which the output size is $\Omega(n^3)$.

Since the main difficulty of the #APSP problem comes from the counts being very large, it is interesting to consider variants that mitigate this. Let $U \le n^{\tilde{O}(1)}$. Let $\#_{\text{mod}\,U}$APSP be the problem of computing all pairwise counts modulo $U$. Let $\#_{\le U}$APSP be the problem of

[4] Brandes presented further practical improvements as well.

[5] One example is an $(n/3+2)$-layered graph where the first $n/3$ layers have 2 vertices each and the last 2 layers have $n/6$ vertices each. The $i$-th layer and the $(i+1)$-th layer are connected by a complete bipartite graph for each $1 \le i \le n/3$, while the last two layers are connected by $O(m)$ edges.

computing for every pair of nodes $u, v$ the minimum of their count and $U$ (think of $U$ as a "cap"). Finally, let $\#_{\text{approx-}U}$APSP be the problem of computing a $(1 + 1/U)$-approximation of all pairwise counts (think of keeping the $\log U$ most significant bits of each count).

We obtain the following result for u-$\#_{\leq U}$APSP in directed graphs:

▶ **Theorem 7.** *u-$\#_{\leq U}$APSP (in directed graphs) can be solved in $n^{2+\rho}\text{polylog } U \leq n^{2+\rho+o(1)}$ time.*

*Furthermore, for any $U \geq 2$, if u-$\#_{\leq U}$APSP can be solved in $O(n^{2+\rho-\varepsilon})$ time for some $\varepsilon > 0$, then so can u-APSP (with randomization). For any $2 \leq U \leq \tilde{O}(1)$, the converse is true as well.*

Thus, we get a conditionally optimal algorithm for u-$\#_{\leq U}$APSP. For $2 \leq U \leq \tilde{O}(1)$, the theorem above gives a fine-grained equivalence between u-$\#_{\leq U}$APSP and u-APSP; in particular, for $U = 2$, the problem corresponds to testing *uniqueness* for the shortest path of each pair. (For large $U$, however, it is not a fine-grained equivalence since the algorithm for u-$\#_{\leq U}$APSP does not go through Min-Plus product, but rather directly uses fast matrix multiplication.)

Our algorithm from Theorem 7 is based on Zwick's algorithm for u-APSP. We show that one can also modify Seidel's algorithm for u-APSP in undirected graphs to obtain $\tilde{O}(n^{\omega})$ time algorithms for u-$\#_{\leq U}$APSP and u-$\#_{\text{mod}\, U}$APSP in undirected graphs.

▶ **Theorem 8.** *u-$\#_{\leq U}$APSP and u-$\#_{\text{mod}\, U}$APSP in undirected graphs can be solved in $\tilde{O}(n^{\omega} \log U)$ time.*

Furthermore, we show that u-$\#_{\text{approx-}U}$APSP in undirected graphs can be solved in $O(n^{2.58}\text{polylog } U)$ time, somewhat surprisingly, by a slight modification of our undirected Lex$_2$-APSP algorithm (despite the apparent dissimilarity between the two problems).

**Paper Organization and Techniques.** In Section 3, we show the web of reductions around u-APSP, proving Theorem 1, Theorem 2, the hardness of additive $D[u, v]^p$ approximate u-APSP and the hardness of u-$\#_{\leq U}$APSP in Theorem 7.

In Section 4, we give our algorithms for approximating APSP with additive errors, proving Theorem 3. In Section 5, we describe our algorithms for Lex$_2$-APSP. Due to space limitation, we defer our algorithms for various versions of #APSP to the full paper (including an algorithm for u-$\#_{\leq U}$APSP to complete the proof of Theorem 7, the proof of Theorem 8, and the algorithm for u-$\#_{\text{approx-}U}$APSP). An exception is our near-cubic algorithm for exact u-#APSP (proof of Theorem 6), which is simple and is described in Section 6.

For approximating APSP with additive error, we propose an interesting *two-phase* variant of Zwick's algorithm [34]. Zwick's algorithm computes distance products of $n \times (n/\ell)$ with $(n/\ell) \times n$ matrices for $\ell$ in a geometric progression. Our idea is to do less during the first phase, computing products of $(n/\ell) \times (n/\ell)$ with $(n/\ell) \times n$ matrices instead. We complete the work during a second phase. The observation is that for the APSP approximation problem, we can afford to perform the distance computation in the first phase *exactly*, but use approximation to speed up the second phase. The resulting approximation algorithm is even simpler than Roditty and Shapira's previous (slower) algorithm [20].

Our Lex$_2$-APSP algorithm for directed graphs also uses this two-phase approach, but in a more sophisticated way to control the size of the numbers in the rectangular matrix products. A number of interesting new ideas are needed.

To further illustrate the power of this two-phase approach, we also show (in the full version) how the idea can lead to an alternative $\tilde{O}(c_0 n^\omega)$ time algorithm for the standard $[c_0]$-APSP problem for undirected graphs, rederiving Shoshan and Zwick's result [22] in an arguably simpler way. This may be of independent interest (as Shoshan and Zwick's algorithm has complicated details).

Our Lex$_2$-APSP algorithm for *undirected* graphs uses small dominating sets for high-degree vertices, an idea of Aingworth et al. [1]. Originally, this idea was for developing combinatorial algorithms for *approximate* shortest paths that avoid matrix multiplication. Interestingly, we show that this idea can be combined with (rectangular) matrix multiplication to compute *exact* Lex$_2$ shortest paths.

## 2 Preliminaries

The computation model of all algorithms and reductions in this paper is the word-RAM model with $O(\log n)$ bit words.

We let $\mathcal{M}(n_1, n_2, n_3)$ denote the best known running time for multiplying an $n_1 \times n_2$ by an $n_2 \times n_3$ matrix over the integers. We use $\omega(a, b, c)$ to denote the rectangular matrix multiplication exponent, i.e. the smallest real number $z$ such that $\mathcal{M}(n^a, n^b, n^c) \le O(n^{z+\varepsilon})$ for all $\varepsilon > 0$. In particular, let $\omega = \omega(1, 1, 1)$. It is known that $\omega \in [2, 2.373)$ [2, 27, 16]. The best known bounds for $\omega(a, b, c)$ are in [17].

Let $\mathcal{M}^\star(n_1, n_2, n_3 \mid \ell_1, \ell_2)$ be the time to compute the Min-Plus product of an $n_1 \times n_2$ matrix $A$ with an $n_2 \times n_3$ matrix $B$, where all finite entries of $A$ are from $[\ell_1]$ and all finite entries of $B$ are from $[\ell_2]$. Let us also denote $\mathcal{M}^\star(n_1, n_2, n_3 \mid \ell) := \mathcal{M}^\star(n_1, n_2, n_3 \mid \ell, \ell)$. It is known [3] that $\mathcal{M}^\star(n_1, n_2, n_3 \mid \ell) \le \tilde{O}(\ell \cdot \mathcal{M}(n_1, n_2, n_3))$. This algorithm in [3] first replaces each entry $e$ in both matrices $A, B$ by $(n_2 + 1)^e$, then uses fast rectangular matrix multiplication to compute the product of the new matrices $A, B$. Since each arithmetic operation takes $\tilde{O}(\ell)$ time, the running time follows.

More generally, let $\mathcal{M}^\star(n_1, n_2, n_3 \mid m_1, m_2, m_3 \mid \ell_1, \ell_2)$ be the time to compute $m_3$ given entries of the Min-Plus product of an $n_1 \times n_2$ matrix $A$ with an $n_2 \times n_3$ matrix $B$, where $A$ has at most $m_1$ finite entries, all from $[\ell_1]$, and $B$ has at most $m_2$ finite entries, all from $[\ell_2]$.

## 3 Directed APSP and Rectangular Min-Plus with Bounded Weights

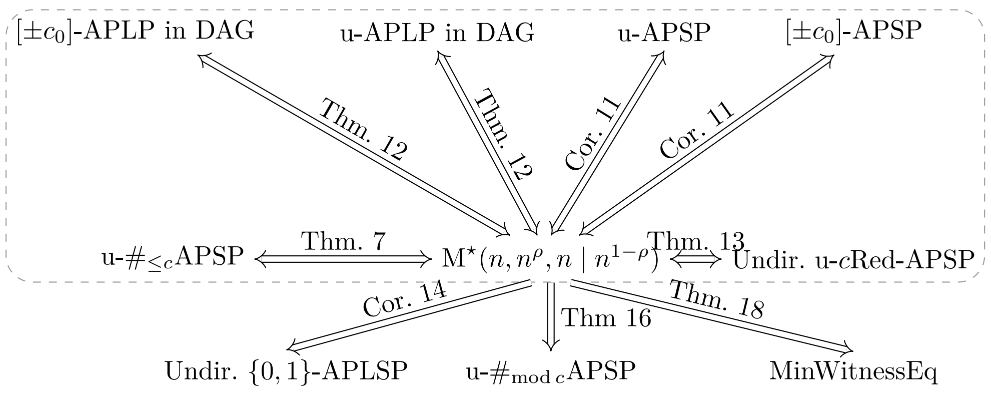

**Figure 1** The web of (a subset of) the reductions in this paper. All reductions are $(n^{2+\rho}, n^{2+\rho})$-fine grained reductions, where $\rho$ is such that $\omega(1, \rho, 1) = 1 + 2\rho$. The problems in the bounding box are sub $n^{2+\rho}$-equivalent. Here, $c_0 = \tilde{O}(1)$, and $2 \le c \le \tilde{O}(1)$. For the current best bounds on rectangular matrix multiplication [17], $\rho$ is roughly 0.529.

Here we consider the All-Pairs Shortest Paths (APSP) problem in unweighted directed graphs, or more generally in directed graphs with integer weights in $[\pm c_0]$ with $c_0 = \tilde{O}(1)$ and no negative cycles. Zwick [34] showed that this problem in $n$-node graphs can be solved in time $\tilde{O}(n^{2+\rho})$ time where $\rho$ is such that $\omega(1,\rho,1) = 1+2\rho$. For the current best bounds on rectangular matrix multiplication [17], $\rho$ is roughly 0.529.

Zwick's algorithm can be viewed as a reduction to rectangular Min-Plus matrix multiplication. The algorithm proceeds in stages, for each $\ell$ from 0 to $\log_{3/2}(n^{1-\rho})$.

In stage $\ell$, up to logarithmic factors, one needs to compute the Min-Plus product of two matrices $A_\ell$ and $B_\ell$ where $A_\ell$ has dimensions $n \times n/(3/2)^\ell$ and $B_\ell$ has dimensions $n/(3/2)^\ell \times n$ and both matrices have entries bounded by $(3/2)^\ell$. Intuitively, this computes the pairwise distances that are roughly $(3/2)^\ell$. After stage $\log_{3/2}(n^{1-\rho})$, the algorithm also runs Dijkstra's algorithm[6] to and from $\tilde{O}(n^\rho)$ nodes $S$ sampled randomly and uses $\tilde{O}(n^{2+\rho})$ extra time to complete the computation of the distances by considering for every $u, v \in V$, $\min_{s\in S}\{D[u,s] + D[s,v]\}$. This can also be viewed as using the brute-force algorithm to compute the Min-Plus products when $(3/2)^\ell \geq n^{1-\rho}$.

The total running time is within logarithmic factors of

$$n^{2+\rho} + \sum_{\ell=0}^{\log_{3/2}(n^{1-\rho})} \mathcal{M}^\star(n, n/(3/2)^\ell, n \mid (3/2)^\ell),$$

where $\mathcal{M}^\star(n_1, n_2, n_3 \mid M)$ is the Min-Plus product running time for matrices with entries in $\{0,\dots,M\}$ and dimensions $n_1 \times n_2$ by $n_2 \times n_3$. With the known bounds for Min-Plus product, $\mathcal{M}^\star(n, n^\tau, n \mid M) \leq \tilde{O}(Mn^{\omega(1,\tau,1)})$, and the running time of Zwick's algorithm becomes $\tilde{O}(n^{2+\rho} + n^{1-\rho+\omega(1,\rho,1)})$, which is $\tilde{O}(n^{2+\rho})$ when $\omega(1,\rho,1) = 1+2\rho$.

If $\omega = 2$, then $\rho$ is $1/2$ and the running time of Zwick's algorithm becomes $\tilde{O}(n^{2.5})$. This running time is a seeming barrier for the APSP problem in directed graphs.

In the full version we prove the following technical theorem which rephrases Zwick's algorithm [34] as a reduction.

▶ **Theorem 9.** *Let $\rho$ be the solution to $\omega(1,\rho,1) = 1+2\rho$. If the Min-Plus product of an $n \times n^\rho$ matrix by an $n^\rho \times n$ matrix where both matrices have integer entries bounded by $n^{1-\rho}$ (denoted as $\mathrm{M}^\star(n, n^\rho, n \mid n^{1-\rho})$) can be computed in $O(n^{2+\rho-\epsilon})$ time for some $\epsilon > 0$, then APSP in directed $n$ node graphs with integer edge weights in $[\pm c_0]$ for $c_0 = \tilde{O}(1)$ can be solved in $O(n^{2+\rho-\epsilon'})$ time for $\epsilon' > 0$.*

If $\omega = 2$, the above theorem statement becomes: If the Min-Plus problem of an $n \times \sqrt{n}$ matrix by a $\sqrt{n} \times n$ matrix where both matrices have integer entries bounded by $\sqrt{n}$ can be computed in $O(n^{2.5-\delta})$ time for some $\delta > 0$, then APSP in directed $n$ node graphs with integer edge weights in $[\pm c_0]$ for $c_0 = \tilde{O}(1)$ can be solved in $O(n^{2.5-\delta'})$ time for $\delta' > 0$.

We will show a reduction in the reverse direction as well, showing that rectangular Min-Plus product with suitably bounded entries can be reduced back to unweighted directed APSP.

▶ **Theorem 10.** *For any fixed $k \in (0,1)$, $\mathrm{M}^\star(n, n^k, n \mid n^{1-k})$ can be reduced in $O(n^2)$ time to APSP in a directed unweighted graph with $O(n)$ vertices.*

[6] If there are negative weights, one also needs to run single source shortest paths (SSSP) from a node, as in Johnson's algorithm and then reweight the edges so that they are nonnegative. SSSP can be solved in $\tilde{O}((m + n^{1.5})\log^2(c_0)) = \tilde{O}(n^2)$ time [23].

A consequence of Theorem 9, and the fact that u-APSP is a special case of $[\pm c_0]$-APSP for directed graphs without negative cycles, is the following equivalence.

▶ **Corollary 11.** *Let $\rho$ be such that $\omega(1,\rho,1) = 1+2\rho$. Then u-APSP, $[\pm c_0]$-APSP for directed graphs without negative cycles, and $\mathrm{M}^\star(n, n^\rho, n \mid n^{1-\rho})$ are sub-$n^{2+\rho}$ fine-grained equivalent for $c_0 = \tilde{O}(1)$.*

In particular, if $\omega = 2$, APSP in directed unweighted graphs is sub-$n^{2.5}$ fine-grained equivalent to the Min-Plus problem of an $n \times \sqrt{n}$ matrix by a $\sqrt{n} \times n$ matrix where both entries have integer entries bounded by $\sqrt{n}$.

**Proof of Theorem 10.** Let $A$ be an $n \times n^k$ matrix and let $B$ be an $n^k \times n$ matrix, both with entries in $\{1, \ldots, n^{1-k}\}$.

We will create a directed graph as follows. Let $I$ be a set of $n$ nodes, which represent the rows of $A$. Let $J$ be a set of $n$ nodes, which represent the columns of $B$.

For every $p \in [n^k]$ corresponding to a column of $A$ (or row of $B$), create a path of $2n^{1-k}+1$ nodes:

$$X(p) := x_{p,n^{1-k}} \to x_{p,n^{1-k}-1} \to \ldots \to x_{p,0} \to y_{p,1} \to y_{p,2} \to \ldots \to y_{p,n^{1-k}}.$$

For every $i \in [n]$ and $p \in [n^k]$, consider $t = A[i,p] \in [n^{1-k}]$. Add an edge from $i \in I$ to $x_{p,t}$. Similarly, for every $j \in [n]$ and $p \in [n^k]$, consider $t' = B[p,j] \in [n^{1-k}]$. Add an edge from $y_{p,t'}$ to $j \in J$.

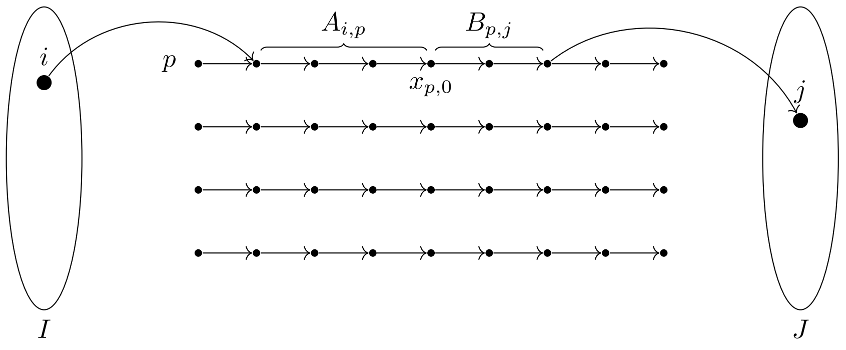

**Figure 2** Sketch of the construction in proof of Theorem 10. For each vertex $i$ and path $p$, we add an edge from $i$ to a vertex on the path $p$ whose distance to the middle point $x_{p,0}$ on the path is $A_{i,p}$. For each path $p$ and vertex $j$, we add an edge from a vertex on the path whose distance from the middle point $x_{p,0}$ on the path is $B_{p,j}$ to vertex $j$.

Now, consider some $i \in [n], p \in [n^k], j \in [n]$ and $A[i,p] + B[p,j]$. If we consider the path consisting of $(i, x_{p,A[i,p]})$, $(y_{p,B[p,j]}, j)$ and the subpath of $X(p)$ between $x_{p,A[i,p]}$ and $y_{p,B[p,j]}$, its length is exactly $2 + A[i,p] + B[p,j]$. Also, any path from $i$ to $j$ is of this form. Thus, the shortest path from $i \in I$ to $j \in J$ in the created graph is exactly of length $2 + \min_p\{A[i,p] + B[p,j]\}$, and thus computing APSP in the directed unweighted graph we have created computes the Min-Plus product of $A$ and $B$.

The number of vertices in the graph is $O(n^k \cdot n^{1-k}) = O(n)$. ◀

One consequence of Corollary 11 is that u-APSP and computing the predecessor matrix in unweighted directed APSP are also sub-$n^{2+\rho}$ fine-grained equivalent. It was known that Zwick's algorithm [34] can compute the predecessor matrix for unweighted directed APSP, which can also be viewed as a sub-$n^{2+\rho}$ time reduction from computing the predecessor matrix to $\mathrm{M}^\star(n, n^\rho, n \mid n^{1-\rho})$. Also, if we can compute the predecessor matrix for the graph constructed in the above proof, we would know which path $X(p)$ the shortest path from $i$ to $j$ uses, which in turn solves $\mathrm{M}^\star(n, n^\rho, n \mid n^{1-\rho})$. Thus, computing the predecessor matrix for unweighted directed APSP is sub-$n^{2+\rho}$ fine-grained equivalent to $\mathrm{M}^\star(n, n^\rho, n \mid n^{1-\rho})$, and thus also equivalent to u-APSP.

Zwick's algorithm is general enough to apply to some variants of APSP. One example is the All-Pairs Longest Paths (APLP) problem in DAGs. To compute APLP in a DAG, we first negate the weight of every edge, then the problem becomes APSP, on which we can directly apply Zwick's algorithm. Therefore, Zwick's algorithm show reductions from u-APLP and $[\pm c_0]$-APLP in DAGs to $\mathrm{M}^\star(n, n^\rho, n \mid n^{1-\rho})$.

Perhaps more surprisingly, the other direction of the reduction also holds. Therefore, APLP in DAG and APSP in graphs with weights bounded by $\tilde{O}(1)$ are sub-$O(n^{2+\rho})$ equivalent.

▶ **Theorem 12.** *Let $\rho$ be such that $\omega(1, \rho, 1) = 1 + 2\rho$. Then u-APLP in DAGs, $[\pm c_0]$-APLP in DAGs and $\mathrm{M}^\star(n, n^\rho, n \mid n^{1-\rho})$ are sub-$n^{2+\rho}$ fine-grained equivalent.*

The proof of Theorem 12 follows from the same approach and is deferred to the full version.

All problems shown equivalent to u-APSP above are problems on directed graphs. One natural question is that whether some problems on undirected graphs are also in this equivalence class, or whether we can show some undirected graph problems require $n^{2+\rho-o(1)}$ time if we assume problems in this equivalence class also require $n^{2+\rho-o(1)}$ time. To answer these questions, we first consider the u-$c$Red-APSP problem.

▶ **Theorem 13.** *Let $\rho$ be such that $\omega(1, \rho, 1) = 1 + 2\rho$. u-cRed-APSP for $2 \le c = \tilde{O}(1)$ and $\mathrm{M}^\star(n, n^\rho, n \mid n^{1-\rho})$ are sub-$n^{2+\rho}$ fine-grained equivalent.*

The proof of Theorem 13 uses a similar graph construction and is in the full version.

By slightly modifying the proof of Theorem 13, we can show conditional hardness for APLSP on undirected graphs where the edge weights are in $\{0, 1\}$. The proof is in the full version.

▶ **Corollary 14.** *Let $\rho$ be such that $\omega(1, \rho, 1) = 1 + 2\rho$. Suppose $\mathrm{M}^\star(n, n^\rho, n \mid n^{1-\rho})$ requires $n^{2+\rho-o(1)}$ time. Then APLSP on undirected graphs where the edge weights can be $\{0, 1\}$ also requires $n^{2+\rho-o(1)}$ time.*

Using similar ideas we also show hardness for *Vertex-Weighted APSP* in undirected graphs, where the vertex weights may be large. (The current best algorithms for Vertex-Weighted APSP for directed graphs [6, 32] had running time about $O(n^{2.85})$; the bound is $\tilde{O}(n^{11/4})$ if $\omega = 2$. No better algorithms were known in the undirected graphs – which our conditional lower bound attempts to explain.) The proof is in the full version.

▶ **Corollary 15.** *Let $\rho$ be such that $\omega(1, \rho, 1) = 1 + 2\rho$. Suppose $\mathrm{M}^\star(n, n^\rho, n \mid n^{1-\rho})$ requires $n^{2+\rho-o(1)}$ time. Then vertex-weighted APSP on undirected graphs where the vertex weights are in $[O(n^{1-\rho})]$ also requires $n^{2+\rho-o(1)}$ time.*

The conditional hardness for u-$\#_{\mathrm{mod}\,U}$APSP and u-$\#_{\le U}$APSP for any $U \ge 2$ can be proved by combining our graph construction with randomized techniques for a unique variant of Min-Plus product; see the proof in the full version.

▶ **Theorem 16.** *Let $\rho$ be such that $\omega(1, \rho, 1) = 1 + 2\rho$. Suppose $\mathrm{M}^\star(n, n^\rho, n \mid n^{1-\rho})$ requires $n^{2+\rho-o(1)}$ time (with randomization). Then u-$\#_{\mathrm{mod}\,U}$APSP and u-$\#_{\le U}$APSP for any $U \ge 2$ requires $n^{2+\rho-o(1)}$ time.*

In Section 4, we will give an algorithm for approximating APSP with sublinear additive errors. Using the same technique as our reductions from Rectangular Min-Plus product to APSP problems, we can show a conditional lower bound for this problem.

▶ **Theorem 17.** *Given a directed unweighted graph $G = (V, E)$ with $n$ vertices and a function $f > 0$ where $\frac{\ell}{f(\ell)}$ is nondecreasing. Suppose we can approximate the shortest-path distance $D[u, v]$ with additive error $f(D[u, v])$, for all $u, v \in V$ in $T(n)$ time, then* $\max_{1\le\ell\le n} \mathcal{M}^\star\left(n, n/\ell, n \mid \frac{\ell}{f(\ell)}\right) \le O(T(n))$.

**Proof.** Fix any $1 \le \ell \le n$. First, note that $\mathcal{M}^\star\left(n, n/\ell, n \mid \frac{\ell}{f(\ell)}\right) = \Theta\left(\mathcal{M}^\star\left(n, n/\ell, n \mid \frac{\ell}{Cf(\ell)}\right)\right)$ for any constant $C$. Here, we take $C = 12$ to be a large enough constant.

Suppose we are given an $n \times n/\ell$ matrix $A$ and an $n/\ell \times n$ matrix $B$, whose entries are positive integers bounded by $\frac{\ell}{12f(\ell)}$, and we want to compute their Min-Plus product $A \star B$. We use a similar reduction as the one in the proof of Theorem 10, but stretching the length of the middle paths. Specifically, we create vertex set $I$ of size $n$, vertex set $J$ of size $n$, and $n/\ell$ paths of the form $X(p) := x_{p,\frac{\ell}{3f(\ell)}} \rightsquigarrow \cdots \rightsquigarrow x_{p,0} \rightsquigarrow y_{p,0} \rightsquigarrow \cdots \rightsquigarrow y_{p,\frac{\ell}{3f(\ell)}}$. From $x_{p,i}$ to $x_{p,i-1}$ and $y_{p,j}$ to $y_{p,j+1}$, we embed paths of length $6f(\ell)$; from $x_{p,0}$ to $y_{p,0}$, we embed a path of length $\ell - 2$. Similar to previous reductions, for every $i \in [n] = I$ and $p \in [n/\ell]$, we add an edge from $i$ to $x_{p,A[i,p]}$; for every $j \in [n] = J$ and $p \in [n/\ell]$, we add an edge from $j$ to $y_{p,B[p,j]}$. Then the distance from $i \in I$ to $j \in J$ in this graph equals $\ell + 6f(\ell)(A \star B)[i, j]$.

Since $0 \le (A \star B)[i, j] \le \frac{\ell}{6f(\ell)}$, we must have $\ell \le \ell + 6f(\ell)(A \star B)[i, j] \le 2\ell$. Since $\frac{\ell}{f(\ell)}$ is nondecreasing, we must have $f(t\ell) \le tf(\ell)$ for any $t \ge 1$, and thus $f(\ell + 6f(\ell)(A \star B)[i, j]) \le 2f(\ell)$. Therefore, an $f(\ell + 6f(\ell)(A \star B)[i, j])$-additive approximation of APSP can determine that the distance from $i \in I$ to $j \in J$ is in $\ell + 6f(\ell)(A \star B)[i, j] \pm 2f(\ell)$, from which we can compute $(A \star B)[i, j]$ easily since $(A \star B)[i, j]$ must be an integer. ◀

Finally, we give a reduction from u-APSP to Min Witness Equality, where we are given $n \times n$ integer matrices $A$ and $B$, and are required to compute $\min\{k \in [n] : A[i, k] = B[k, j]\}$ for every pair of $(i, j)$. Reductions from u-APSP to matrix product problems are considered by Lincoln et al. [18], where they show reductions from u-APSP to the All-Edges Monochromatic Triangle problem and $(\min, \max)$-product problem, but their techniques do not seem to apply to Min Witness Equality.

The proof of the following theorem is deferred to the full version.

▶ **Theorem 18.** *Let $\rho$ be such that $\omega(1, \rho, 1) = 1 + 2\rho$. Suppose $\mathrm{M}^\star(n, n^\rho, n \mid n^{1-\rho})$ requires $n^{2+\rho-o(1)}$ time. Then Min Witness Equality requires $n^{2+\rho-o(1)}$ time.*

## 4 Additive Approximation Algorithms for APSP

In this section, we give an algorithm for approximate APSP with additive errors in directed unweighted graphs, to match the lower bound that we have just proved in Theorem 17 (ignoring logarithmic factors). Namely, our algorithm achieves running time $\tilde{O}(\max_\ell \mathcal{M}^\star(n, n/\ell, n \mid \ell^{1-p}))$, which improves Roditty and Shapira's previous algorithm [20] with running time $\tilde{O}(\max_\ell \min\{n^3/\ell, \mathcal{M}^\star(n, n/\ell^{1-p}, n \mid \ell^{1-p})\})$.

Let $D[u, v]$ denote the shortest-path distance from $u$ to $v$.

**Overview.** The new algorithm is a variation of Zwick's exact u-APSP algorithm [34], and is actually simpler than Roditty and Shapira's algorithm. The idea is to compute as many as the shortest-path distances *exactly* as we can in $\tilde{O}(n^\omega)$ time in an initial phase. In the second phase, we apply rectangular matrix multiplication to submatrices computed from the first phase, where entries are approximated by rounding and rescaling.

**Preliminaries.** For every $\ell$ that is a power of 3/2, let $R_\ell \subseteq V$ be a subset of $\tilde{O}(n/\ell)$ vertices that hits all shortest paths of length $\ell/2$ [34]. (For example, a random sample works with high probability.) We may assume that $R_{(3/2)^i} \supseteq R_{(3/2)^{i+1}}$ (because otherwise, we can add $R_{(3/2)^j}$ to $R_{(3/2)^i}$ for all $j > i$ and the size bound would still hold). For subsets $S_1, S_2 \subseteq V$, let $D(S_1, S_2)$ denote the submatrix of $D$ containing the entries for $(u,v) \in S_1 \times S_2$.

**Phase 1.** We first solve the following subproblem: compute $D[u,v]$ (exactly) for all $(u,v) \in R_\ell \times V$ with $D[u,v] \leq \ell$, and similarly for all $(u,v) \in V \times R_\ell$ with $D[u,v] \leq \ell$.

Suppose we have already computed $D[u,v]$ for all $(u,v) \in R_{2\ell/3} \times V$ with $D[u,v] \leq 2\ell/3$, and similarly for all $(u,v) \in V \times R_{2\ell/3}$ with $D[u,v] \leq 2\ell/3$.

We take the Min-Plus product $D(R_\ell, R_{2\ell/3}) \star D(R_{2\ell/3}, V)$. For each $(u,v) \in R_\ell \times V$, if its output entry is smaller than the current value of $D[u,v]$, we reset $D[u,v]$ to the smaller value. Similarly, we take the Min-Plus product $D(V, R_{2\ell/3}) \star D(R_{2\ell/3}, R_\ell)$. For each $(u,v) \in V \times R_\ell$, if its output entry is smaller than the current value of $D[u,v]$, we reset $D[u,v]$ to the smaller value. We reset all entries greater than $\ell$ to $\infty$.

To justify correctness, observe that for any shortest path $\pi$ of length between $2\ell/3$ and $\ell$, the middle $(2\ell/3)/2 = \ell/3$ vertices must contain a vertex of $R_{2\ell/3}$, which splits $\pi$ into two subpaths each of length at most $\ell/2 + \ell/6 \leq 2\ell/3$.

We do the above for all $\ell$'s that are powers of 3/2. The total cost is

$$\tilde{O}\left(\max_\ell \mathcal{M}^\star(n/\ell, n/\ell, n \mid \ell)\right) \leq \tilde{O}\left(\max_\ell \ell \cdot \mathcal{M}^\star(n/\ell, n/\ell, n)\right) \leq \tilde{O}\left(\max_\ell \ell^2 (n/\ell)^\omega\right) = \tilde{O}(n^\omega).$$

**Phase 2.** Next we approximate all shortest-path distances $D[u,v]$ where $D[u,v]$ is between $2\ell/3$ and $\ell$, with additive error $O(f(\ell))$ for a given function $f$, as follows:

We compute the Min-Plus product $D(V, R_{2\ell/3}) \star D(R_{2\ell/3}, V)$, keeping only entries bounded by $O(\ell)$. As we allow additive error $O(f(\ell))$, we round entries to multiples of $f(\ell)$. This takes $\tilde{O}(\mathcal{M}^\star(n, n/\ell, n \mid \frac{\ell}{f(\ell)}))$ time.

To justify correctness, observe as before that in any shortest path $\pi$ of length between $2\ell/3$ and $\ell$, some vertex in $R_{2\ell/3}$ splits the path into two subpaths of length at most $2\ell/3$.

We repeat for all $\ell$'s that are powers of 3/2. The total cost is $\tilde{O}\left(\max_\ell \mathcal{M}^\star(n, n/\ell, n \mid \frac{\ell}{f(\ell)})\right)$.

Standard techniques for generating witnesses for matrix products can be applied to recover the shortest paths (e.g., see [11, 34]).

▶ **Theorem 19.** *Given a directed unweighted graph $G = (V, E)$ with $n$ vertices and a function $f$ where $\frac{\ell}{f(\ell)}$ is nondecreasing, we can approximate the shortest-path distance $D[u,v]$ with additive error $O(f(D[u,v]))$ for all $u, v \in V$, in $\tilde{O}\left(\max_\ell \mathcal{M}^\star(n, n/\ell, n \mid \frac{\ell}{f(\ell)})\right)$ time.*

▶ Remark. For $f(\ell) = \ell^p$, we can upper-bound the running time by

$$\begin{aligned}\tilde{O}\left(\max_\ell \mathcal{M}^\star(n, n/\ell, n \mid \ell^{1-p})\right) &\leq \tilde{O}\left(L^{1-p} \cdot \mathcal{M}(n, n/L, n) + n^3/L\right) \\ &\leq \tilde{O}\left(L^{1-p}(n^{2+o(1)} + n^\omega / L^{(\omega-2)/(1-\alpha)}) + n^3/L\right)\end{aligned}$$

for any choice of $L$, where $\alpha$ is the rectangular matrix multiplication exponent (satisfying $\omega(1,1,\alpha)=2$). For example, we can set $L=n^{3-\omega}$, and for $p>1-\min\{\frac{\omega-2}{1-\alpha},\frac{\omega-2}{3-\omega}\}$, get optimal $\tilde{O}(n^{\omega})$ running time. In fact, with the current rectangular matrix multiplication bounds we get $\tilde{O}(n^{2.373})$ time for $p\geq 0.415\geq(\omega(1,0.373,1)-2\cdot 0.373-1)/(1-0.373)$. Roditty and Shapira [20] specifically asked whether there exists $p<1$ for which $\tilde{O}(n^{\omega})$ time is possible; we have thus answered their question affirmatively if $\omega>2$.

▶ Remark. For directed graphs with weights from $[c_0]$, the running time is

$$\tilde{O}\left(c_0n^{\omega}+\max_{\ell}\mathcal{M}^{\star}(n,n/\ell,n\mid c_0\tfrac{\ell}{f(\ell)})\right).$$

## 5 Algorithms for All-Pairs Lightest Shortest Paths

In this section, we describe algorithms for the following problem, which includes both All-Pairs Lightest Shortest Paths (APLSP) and Shortest Lightest Paths (APSLP) as special cases:

▶ **Problem 20. (Lex$_2$-APSP)** *We are given a graph $G=(V,E)$ with $n$ vertices, where each edge $(u,v)\in E$ has a "primary" weight $w_1(u,v)$ and a "secondary" weight $w_2(u,v)$. For every pair of vertices $u,v\in V$, we want to find a path $\pi$ from $u$ to $v$ that minimizes $(\sum_{e\in\pi}w_1(e),\sum_{e\in\pi}w_2(e))$ lexicographically.*

Let $D[u,v]$ be the lexicographical minimum of $(\sum_{e\in\pi}w_1(e),\sum_{e\in\pi}w_2(e))$. Let $D_1[u,v]$ be the minimum of $\sum_{e\in\pi}w_1(e)$ (the shortest-path distance) and let $D_2[u,v]$ be the second coordinate of $D[u,v]$. APLSP corresponds to the case when all secondary edge weights are 1, whereas APSLP corresponds to the case when all primary edge weights are 1.

The following lemma, which will be important in the analysis of our Lex$_2$-APSP algorithm, bounds the complexity of Min-Plus product of an $n_1\times n_2$ matrix $A$ and an $n_2\times n_3$ matrix $B$ in the case when the finite entries of $A$ come from a small range $[\ell_1]$ (but the finite entries of $B$ may come from a large range $[\ell_2]$). The bound can be made sensitive to the number $m_2$ of finite entries of $B$ and the number $m_3$ of output entries we want. The lemma is a variant of [6, Theorem 3.5] (the basic approach originates from Matoušek's dominance algorithm [19], but this variant requires some extra ideas). It also generalizes and improves (using rectangular matrix multiplication) Theorem 1.2 in [30].

▶ **Lemma 21.** $\mathcal{M}^{\star}(n_1,n_2,n_3\mid\ell_1,\ell_2)=\tilde{O}\left(\min_t(\mathcal{M}^{\star}(n_1,n_2,n_2n_3/t\mid\ell_1)+tn_1n_3)\right)$. *More generally,* $\mathcal{M}^{\star}(n_1,n_2,n_3\mid m_1,m_2,m_3\mid\ell_1,\ell_2)=\tilde{O}\left(\min_t(\mathcal{M}^{\star}(n_1,n_2,m_2/t\mid\ell_1)+tm_3)\right)$.

**Proof.** Divide each column of $B$ into groups of $t$ entries by rank: the first group contains the $t$ smallest elements, the second group contains the next $t$ smallest, etc. (ties in ranks can be broken arbitrarily). Each column may have at most $t$ leftover entries. The total number of groups is at most $m_2/t$.

For each $i\in[n_1]$ and $j'\in[m_2/t]$, let $C[i,j']$ be true iff there exists $k\in[n_2]$ such that $A[i,k]<\infty$ and group $j'$ contains an element with row index $k$. Computing $C$ reduces to taking a Boolean matrix product and has cost $O(\mathcal{M}(n_1,n_2,m_2/t))$.

For each $i\in[n_1]$ and $j'\in[m_2/t]$, suppose that group $j'$ is part of column $j$ and the maximum element in group $j'$ is $x$; let $\widehat{C}[i,j']=\min_{k:B[k,j]\in[x,x+\ell_1]}(A[i,k]+B[k,j])$. Since entries in $A$ are from the range $[\ell_1]\cup\{\infty\}$, and we only keep a size $\ell_1+1$ range of values for matrix $B$, computing $\widehat{C}$ reduces to taking a Min-Plus product with entries in $[\ell_1]$ (after shifting) and has cost $O(\mathcal{M}^{\star}(n_1,n_2,m_2/t\mid\ell_1))$.

To compute the output entry at each of the $m_3$ positions $(i,j)$, we find the group $j'$ in column $j$ with the smallest rank such that $C[i,j']$ is true. Let $x$ be the maximum element in group $j'$. The answer $\min_k(A[i,k]+B[k,j])$ is at most $x+\ell_1$. Thus, the answer is defined by an index $k$ that (i) corresponds to an element in group $j'$, or (ii) corresponds to a leftover element in column $j$, or (iii) has $B[k,j]\in[x,x+\ell_1]$. Cases (i) and (ii) can be handled by linear search in $O(t)$ time; case (iii) is handled by looking up $\widehat{C}[i,j']$. The total time to compute $m_3$ output entries is $O(tm_3)$. ◀

## 5.1 $[c_0]$-Lex$_2$-APSP

Let $c_0=\tilde{O}(1)$. For directed graphs, Zwick [33] presented a variant of his u-APSP algorithm that solves $[c_0]$-Lex$_2$-APSP (and thus $[c_0]$-APLSP and $[c_0]$-ALPSP) in time $\tilde{O}(\max_\ell \mathcal{M}^\star(n,n/\ell,n\mid \ell^2))\le \tilde{O}(\min_L(L^2\mathcal{M}(n,n/L,n)+n^3/L))$. This is $O(n^{2.724})$ by the current bounds on rectangular matrix multiplication [17] (and is $\tilde{O}(n^{8/3})$ if $\omega=2$).

Chan [6] gave a faster algorithm for $([c_0]-\{0\})$-Lex$_2$-APSP (and in fact a special case of Vertex-Weighted APSP that includes $([c_0]-\{0\})$-Lex$_k$-APSP for an arbitrary constant $k$) in time $\tilde{O}(n^{(3+\omega)/2})$, which is $O(n^{2.687})$ by the current matrix multiplication exponent (and is $\tilde{O}(n^{2.5})$ if $\omega=2$). Zwick's algorithm works even when zero primary weights are allowed, but Chan's algorithm does not (part of the difficulty is that the secondary distance of a path may be much larger than the primary distance). A more general version of Chan's algorithm [6] can handle zero primary weights (and $[c_0]$-Lex$_k$-APSP for constant $k$) but has a worse time bound of $\tilde{O}(n^{(9+\omega)/4})$, which can be slightly reduced using rectangular matrix mutiplication [32].

We describe an $O(n^{2.6581})$-time algorithm to solve $[c_0]$-Lex$_2$-APSP for directed graphs, which can handle zero weights and is faster than Zwick's $O(n^{2.724})$-time algorithm; it is also slightly faster than Chan's algorithm. The algorithm uses rectangular matrix multiplication (without which the running time would be $\tilde{O}(n^{(\omega+3)/2})$). It should be noted that Chan's previous algorithm can't be easily sped up using rectangular matrix multiplication, besides being inapplicable when there are zero primary weights.

**Overview.** The new algorithm can be viewed as an interesting variant of Zwick's u-APSP algorithm [34]. Zwick's algorithm uses rectangular Min-Plus products of dimensions around $n\times n/\ell$ and $n/\ell\times n$, in geometrically increasing parameter $\ell$. Our algorithm proceeds in two phases. In both phases, we use the rectangular products of dimensions around $n/\ell\times n/\ell$ and $n/\ell\times n$. In the first phase, we consider $\ell$ in increasing order; in the second, we consider $\ell$ in decreasing order. In these Min-Plus products, entries of the first matrix in each product come from a small range; this enables us to use Lemma 21.

**Preliminaries.** Let $L$ be a parameter to be set later. Let $\lambda[u,v]$ denote the length of a lexicographical shortest path between $u$ and $v$. In this section, the *length* of a path refers to the number of edges in the path.

For every $\ell$ that is a power of 3/2, as in Section 4, let $R_\ell\subseteq V$ be a subset of $\tilde{O}(n/\ell)$ vertices that hits all shortest paths of length $\ell/2$ [33, 34]. We may assume that $R_{(3/2)^i}\supseteq R_{(3/2)^{i+1}}$ (as before). Set $R_1=V$.

For $S_1,S_2\subseteq V$, let $D(S_1,S_2)$ denote the submatrix of $D$ containing the entries for $(u,v)\in S_1\times S_2$.

**Phase 1.** We first solve the following subproblem for a given $\ell \le L$: compute $D[u,v]$ for all $(u,v) \in R_\ell \times V$ with $\lambda[u,v] \le \ell$, and similarly for all $(u,v) \in V \times R_\ell$ with $\lambda[u,v] \le \ell$. (We don't know $\lambda[u,v]$ in advance. More precisely, if $\lambda[u,v] \le \ell$, the computed value should be correct; otherwise, the computed value is only guaranteed to be an upper bound.)

Suppose we have already computed $D[u,v]$ for all $(u,v) \in R_{2\ell/3} \times V$ with $\lambda[u,v] \le 2\ell/3$, and similarly for all $(u,v) \in V \times R_{2\ell/3}$ with $\lambda[u,v] \le 2\ell/3$.

We take the Min-Plus product $D(R_\ell, R_{2\ell/3}) \star D(R_{2\ell/3}, V)$ (where elements are compared lexicographically). For each $(u,v) \in R_\ell \times V$, if its output entry is smaller than the current value of $D[u,v]$, we reset $D[u,v]$ to the smaller value. Similarly, we take the Min-Plus product $D(V, R_{2\ell/3}) \star D(R_{2\ell/3}, R_\ell)$. For each $(u,v) \in V \times R_\ell$, if its output entry is smaller than the current value of $D[u,v]$, we reset $D[u,v]$ to the smaller value. We reset all entries greater than $c_0\ell$ to $\infty$.

To justify correctness, observe that for any shortest path $\pi$ of length between $2\ell/3$ and $\ell$, the middle $(2\ell/3)/2 = \ell/3$ vertices must contain a vertex of $R_{2\ell/3}$, which splits $\pi$ into two subpaths each of length at most $\ell/2 + \ell/6 \le 2\ell/3$.

To take the product, we map each entry $D[u,v]$ of $D(R_{2\ell/3}, V)$ to a number $D_1[u,v] \cdot c_0\ell + D_2[u,v] \in [\tilde{O}(\ell^2)]$. It is more efficient to break the product into $\ell$ separate products, by putting entries of $D(R_\ell, R_{2\ell/3})$ with a common $D_1$ value into one matrix. Then after shifting, the finite entries of each such matrix are in $[\tilde{O}(\ell)]$. (The entries of $D(R_{2\ell/3}, V)$ are still in $[\tilde{O}(\ell^2)]$.) Hence, the computation takes time $\tilde{O}(\ell \cdot \mathcal{M}^\star(n/\ell, n/\ell, n \mid \ell, \ell^2))$.

We do the above for all $\ell \le L$ that are powers of $3/2$ (in increasing order).

**Phase 2.** Next we solve the following subproblem for a given $\ell \le L$: compute $D[u,v]$ for all $(u,v) \in R_{2\ell/3} \times V$ with $\lambda[u,v] \le L$.

Suppose we have already computed $D[u,v]$ for all $(u,v) \in R_\ell \times V$ with $\lambda[u,v] \le L$.

We take the Min-Plus product $D(R_{2\ell/3}, R_\ell) \star D(R_\ell, V)$, keeping only entries bounded by $\tilde{O}(\ell)$ in the first matrix and $\tilde{O}(L)$ in the second matrix. For each $(u,v) \in V \times R_\ell$, if its output entry is smaller than the current value of $D[u,v]$, we reset $D[u,v]$ to the smaller value.

To justify correctness, recall that for $(u,v) \in R_{2\ell/3} \times V$, if $\lambda[u,v] < 2\ell/3$, then $D[u,v]$ is already computed in Phase 1. On the other hand, in any shortest path $\pi$ of length between $2\ell/3$ and $L$, the first $\ell/2$ vertices of the path must contain a vertex of $R_\ell$.

To take the product, we map each entry $D[u,v]$ of $D(R_{2\ell/3}, V)$ to a number $D_1[u,v] \cdot c_0 L + D_2[u,v] \in [\tilde{O}(\ell L)]$. As before, it is better to perform $\ell$ separate products, by putting entries of $D(R_{2\ell/3}, R_\ell)$ with a common $D_1$ value into one matrix. Then after shifting, the finite entries of each such matrix are in $[\tilde{O}(\ell)]$. (The entries of $D(R_{2\ell/3}, V)$ are still in $[\tilde{O}(\ell L)]$.) Hence, the computation takes time $\tilde{O}(\ell \cdot \mathcal{M}^\star(n/\ell, n/\ell, n \mid \ell, \ell L))$.

We do the above for all $\ell \le L$ that are powers of $3/2$ (in decreasing order).

**Last step.** By the end of Phase 2 (when $\ell$ reaches 1), we have computed $D[u,v]$ for all $(u,v)$ with $\lambda[u,v] \le L$. To finish, we compute $D[u,v]$ for all $(u,v)$ with $\lambda[u,v] > L$, as follows:

We run Dijkstra's algorithm $O(|R_L|)$ times to compute $D[u,v]$ for all $(u,v) \in R_L \times V$ and for all $(u,v) \in V \times R_L$. This takes $O(|R_L|n^2) = \tilde{O}(n^3/L)$ time. We then compute $D(V, R_L) \star D(R_L, V)$ by brute force in $O(|R_L|n^2) = \tilde{O}(n^3/L)$ time.

Correctness follows since every shortest path of length more than $L$ must pass through a vertex in $R_L$.

As before, standard techniques for generating witnesses for matrix products can be applied to recover the shortest paths [11, 34].

**Total time.** The cost of Phase 2 dominates the cost of Phase 1. By Lemma 21, the total cost is

$$\tilde{O}\left(\max_{\ell\leq L}\ell\cdot\mathcal{M}^\star(n/\ell,n/\ell,n\mid\ell,\ell L)\;+\;n^3/L\right)$$
$$\leq\;\tilde{O}\left(\max_{\ell\leq L}\ell\cdot\min_t\left(\mathcal{M}^\star(n/\ell,n/\ell,n^2/(\ell t)\mid\ell)+tn^2/\ell\right)\;+\;n^3/L\right).$$

We set $t=n/L$ and obtain

$$\tilde{O}(\max_{\ell\leq L}\ell^2\cdot\mathcal{M}(n/\ell,n/\ell,Ln/\ell)\;+\;n^3/L).$$

Intuitively, the maximum occurs when $\ell=1$, and so we should choose $L$ to minimize $\tilde{O}(\mathcal{M}(n,n,Ln)+n^3/L)$. With the current bounds on rectangular matrix multiplication [17], we choose $L=n^{0.342}$ and get running time $O(n^{2.6581})$. (Formally, we can verify this time bound using the convexity of the function $2x+\omega(1-x,1-x,1.342-x)$.)

▶ **Theorem 22.** *$[c_0]$-Lex$_2$-APSP (and thus $[c_0]$-APLSP and $[c_0]$-APSLP) can be solved in $O(n^{2.6581})$ time for any $c_0=\tilde{O}(1)$.*

**Remarks.** Without rectangular matrix multiplication, the above still gives a time bound of $\tilde{O}(Ln^\omega+n^3/L)$, yielding $\tilde{O}(n^{(3+\omega)/2})$.

The same algorithm works even with negative weights (i.e., for $[\pm c_0]$-Lex$_2$-APSP), like Zwick's previous algorithm [33], assuming no negative cycles.

In the full version, we describe an alternative algorithm that has the same running time, though it does not allow zero primary edge weights (or negative weights).

## 5.2 Undirected ($[c_0]-\{0\}$)-Lex$_2$-APSP

A natural question is whether APLSP or APSLP is easier for undirected graphs. We now describe a faster $O(n^{2.58})$-time algorithm for ($[c_0]-\{0\}$)-Lex$_2$-APSP for undirected graphs. Zero primary weights are not allowed, but zero secondary weights are. (In particular, the algorithm can solve $[c_0]$-APSLP, when all primary weights are 1.)

**Overview.** We follow an idea of Aingworth et al. [1], to divide into two cases: when the source vertex has high degree or low degree. 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 give us good approximation to distances from other sources in the same cluster, by the triangle inequality (since the graph is undirected). On the other hand, for low-degree vertices, the relevant subgraph is sparse, which enables faster algorithms. Originally, Aingworth et al.'s approach was intended for the design of approximation algorithms (with $O(1)$ additive error for unweighted graphs). We will adapt it to find *exact* shortest paths. (Chan [7] previously had also applied Aingworth et al.'s approach to exact APSP, but the goal there was in logarithmic-factor speedup, which was quite different.) In order to handle the high-degree case for Lex$_2$-APSP, we need further ideas to use approximate primary shortest-path distances to compute exact lexicographical shortest-path distances; in particular, we will need Min-Plus products on secondary distances (as revealed in the proof of Lemma 23 below). The combination of Aingworth et al.'s approach with matrix multiplication appears new, and interesting in our opinion.

**Preliminaries.** We first compute $D_1[u,v]$ for all $(u,v)$ by running a known $[c_0]$-APSP algorithm on the primary distances in $O(n^\omega)$ time [3, 21].

Assume that we have already computed $D[u,v]$ for all $(u,v)$ with $D_1[u,v] \le 2\ell/3$ for a given $\ell$. We want to compute $D[u,v]$ for all $(u,v)$ with $D_1[u,v] \le \ell$.

Define $D_2^{(\ell)}[u,v] = D_2[u,v]$ if $D_1[u,v] = \ell$, and $D_2^{(\ell)}[u,v] = \infty$ otherwise. For subsets $S_1, S_2 \subseteq V$, let $D_2^{(\ell)}(S_1,S_2)$ denote the submatrix of $D_2^{(\ell)}$ containing the entries for $(u,v) \in S_1 \times S_2$.

▶ **Lemma 23.** *Let $G = (V,E)$ be an undirected graph with edge weights in $[c_0] - \{0\}$. Assume that we have already computed $D[u,v]$ for all $(u,v)$ with $D_1[u,v] \le 2\ell/3$. Given a set $S$ of vertices that are within primary distance $c = \tilde{O}(1)$ from each other, we can compute $D[u,v]$ for all $u \in S$ and $v \in V$ with $D_1[u,v] \le \ell$ in $O(\mathcal{M}^\star(|S|, n/\ell, n \mid \ell))$ total time.*

**Proof.** Fix $s \in S$. Let $V_i = \{v \in V : D_1[s,v] \in i \pm c\}$. Note that $\sum_i |V_i| = \tilde{O}(n)$. Also note that if $u \in S$ and $D_1[u,v] = i$, then we must have $v \in V_i$ (by the triangle inequality, because the graph is undirected).

Pick an index $m \in [0.4\ell, 0.6\ell]$ with $|V_{m-c_0} \cup \cdots \cup V_m| = \tilde{O}(n/\ell)$.

For $i \le m$, we have already computed $D_2^{(i)}(S, V_i)$.

For $i = m+1, \ldots, \ell$, we will compute $D_2^{(i)}(S, V_i)$ as follows: For each $\Delta \in [c_0]$, we take the Min-Plus product $D_2^{(m-\Delta)}(S, V_{m-\Delta}) \star D_2^{(i-m+\Delta)}(V_{m-\Delta}, V_i)$. Note that $D_2^{(i-m+\Delta)}(V_{m-\Delta}, V_i)$ is already known, since $i - m + \Delta < 2\ell/3$. We take the minimum over all $\Delta \in [c_0]$ for those $(u,v) \in S \times V_i$ with $D_1[u,v] = i$.

Instead of doing the product individually for each $i$, it is more efficient to combine all the matrices $D_2^{(i-m+\Delta)}(V_{m-\Delta}, V_i)$ over all $i > m$. This gives a single matrix (per $\Delta$) with $|V_{m-\Delta}| = \tilde{O}(n/\ell)$ rows and $\sum_{i>m} |V_i| = \tilde{O}(n)$ columns. So, the entire product can be computed in $O(\mathcal{M}^\star(|S|, n/\ell, n \mid \ell))$ time. ◀

Let $L$ be a parameter to be set later. Let $V_{\text{high}}$ be the set of all vertices of degree more than $n/L$, and $V_{\text{low}}$ be the set of all vertices of degree at most $n/L$.

**Phase 1.** We will first compute $D[u,v]$ for all $u \in V_{\text{high}}$ and $v \in V$ with $D_1[u,v] \le \ell$, as follows:

Let $X \subseteq V$ be a dominating set for $V_{\text{high}}$ of size $\tilde{O}(L)$, such that every vertex in $V_{\text{high}}$ is in the (closed) neighborhood of some vertex in $X$. Such a dominating set can be constructed (for example, by the standard greedy algorithm) in $\tilde{O}(n^2)$ time [1].

Let $X = \{x_1, x_2, \ldots, x_{\tilde{O}(L)}\}$. For each $x_i \in X$, we divide $N(x_i) \setminus \left(\bigcup_{j<i} N(x_j)\right)$ – its neighborhood excluding previous neighborhoods – into groups of size $O(n/L)$. The total number of groups is $\tilde{O}(L)$, and the groups cover all vertices in $V_{\text{high}}$. For each such group, we apply Lemma 23 (with $c = 2c_0$). The total time is $\tilde{O}(L \cdot \mathcal{M}^\star(n/L, n/\ell, n \mid \ell))$.

**Phase 2.** Next, for each $u \in V_{\text{low}}$, we will compute $D[u,v]$ for all $v \in V$ with $D_1[u,v] \le \ell$, as follows:

Define a graph $G_u$ containing all edges $(x,y)$ with $x \in V_{\text{low}}$ or $y \in V_{\text{low}}$; for each $z \in V_{\text{high}}$, we add an extra edge $(u,z)$ with weight $D[u,z]$, which has been computed in Phase 1. Then the lexicographical shortest-path distance from $u$ to $v$ in $G_u$ matches the lexicographical shortest-path distance in $G$, because if $\langle u_1, \ldots, u_k \rangle$ is a lexicographical shortest path in $G$ with $u_1 = u$, and $i$ is the largest index with $u_i \in V_{\text{high}}$ (set $i = 1$ if none exists), 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 $O(n^2/L)$ edges, this takes $\tilde{O}(n^2/L)$ time per $u$. The total over all $u$ is $\tilde{O}(n^3/L)$.

As before, standard techniques for generating witnesses for matrix products can be applied to recover the shortest paths [11, 34].

**Total time.** We do the above for all $\ell$'s that are powers of $3/2$. The overall cost is

$$\begin{aligned}
&\tilde{O}\left(\max_{\ell} L \cdot \mathcal{M}^{\star}(n/L, n/\ell, n \mid \ell) + n^3/L\right)\\
&\leq \tilde{O}\left(\max_{\ell} L \cdot \min\left\{n^3/(L\ell),\ \ell \cdot \mathcal{M}(n/L, n/\ell, n)\right\} + n^3/L\right)\\
&= \tilde{O}\left(\max_{\ell \leq L} L\ell \cdot \mathcal{M}(n/L, n/\ell, n) + n^3/L\right) = \tilde{O}(L^2 \cdot \mathcal{M}(n/L, n/L, n) + n^3/L).
\end{aligned}$$

With the current bounds on rectangular matrix multiplication, we choose $L = n^{0.4206}$ and get running time $O(n^{2.5794})$.

▶ **Theorem 24.** *$([c_0] - \{0\})$-Lex$_2$-APSP (and thus -APLSP and -APSLP) for undirected graphs can be solved in $O(n^{2.5794})$ time for any $c_0 = \tilde{O}(1)$.*

▶ Remarks. Without rectangular matrix multiplication, the above still gives a time bound of $\tilde{O}(L^3(n/L)^\omega + n^3/L)$, yielding $\tilde{O}(n^{2+1/(4-\omega)})$.

One could adapt the algorithm to solve Undirected $([c_0] - \{0\})$-Lex$_k$-APSP for a larger constant $k$, but the running time appears worse than the bound $\tilde{O}(n^{(3+\omega)/2})$ by Chan [6] (because of the need to compute a Min-Plus product between matrices with larger entries in Lemma 23).

## 6 Exact u-#APSP

We defer most of our algorithms for #APSP to the full paper. An exception is our algorithm for exact u-#APSP, which is simple and is described below. Interestingly, some of our #APSP algorithms are obtained by modifying our Lex$_2$-APSP algorithms, even though the #APSP and Lex$_2$-APSP problems appear very different.

For exact counts that could be exponentially large, we will describe a combinatorial $\tilde{O}(n^3)$-time algorithm to solve u-#APSP for directed unweighted graphs, in the standard word RAM model (with $(\log n)$-bit words). The idea behind the algorithm is actually related to the Lex$_2$-APSP algorithm in Section 5.2, but simplified with $L = 1$ and without matrix multiplication and dominating sets.

Recall that the goal is to compute the number $C[u, v]$ of shortest paths from $u$ to $v$, for all $u, v \in V$ for a given directed unweighted graph $G = (V, E)$.

We first compute $D[u, v]$ for all $u, v \in V$ in $O(n^3)$ time by known APSP algorithms. There are of course faster APSP algorithms for directed unweighted graphs, but we use the slower $O(n^3)$ time algorithm to keep the whole algorithm combinatorial.

Assume we have already computed $C[u, v]$ for all $u, v$ with $D[u, v] \leq 2\ell/3$ for a given $\ell$. Fix a source vertex $s \in V$. We will compute $C[s, v]$ for all $v$ with $D[s, v] \leq \ell$, as follows:

Let $V_i = \{v \in V : D[s, v] = i\}$. Note that $\sum_i |V_i| = n$, so there exist an index $m \in [0.4\ell, 0.6\ell]$ with $|V_m| = O(n/\ell)$.

For $i \leq m$, we have already computed $C[s, v]$ for all $v \in V_i$.

For $i = m + 1, \ldots, \ell$, by setting $C[s, v] = \sum_{u \in V_m : D[u,v] = i-m} C[s, u] \cdot C[u, v]$, we compute $C[s, v]$ for all $v \in V_i$. Note that $C[s, u]$ and $C[u, v]$ have been computed from the previous iteration, since $i - m < 2\ell/3$. The total number of arithmetic operations is $O(\sum_i |V_i| \cdot |V_m|) = O(n^2/\ell)$. Since the counts are bounded by $O(n^\ell)$ and are $\tilde{O}(\ell)$-bit numbers, the total cost is $\tilde{O}(n^2/\ell \cdot \ell) = \tilde{O}(n^2)$.

We do this for every source $s \in V$. The overall cost is $\tilde{O}(n^3)$.

We do the above for all $\ell$'s that are powers of $3/2$. The final time bound is $\tilde{O}(n^3)$.

▶ **Theorem 25.** *u-#APSP can be solved in $\tilde{O}(n^3)$ time.*

▶ Remarks. This is worst-case optimal up to polylogarithmic factors, as the total number of bits in the answers could be $\Omega(n^3)$.

Recall the Betweenness Centrality of a vertex $v$ is defined as $\mathrm{BC}(v) = \sum_{s,t\neq v} C_v[s,t]/C[s,t]$ where $C_v[s,t]$ is the number of shortest paths between $s$ and $t$ that go through $v$. As an immediate corollary, we can compute the Betweenness Centrality of a given vertex exactly in a directed unweighted graph in $\tilde{O}(n^3)$ time.

▶ **Corollary 26.** *The betweenness centrality of a vertex can be computed in $\tilde{O}(n^3)$ time in a directed unweighted graph.*

## 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)*, page to appear, 2021.

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 Ulrik Brandes. A faster algorithm for betweenness centrality. *The Journal of Mathematical Sociology*, 25(2):163–177, 2001.

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 Artur Czumaj, Miroslaw Kowaluk, and Andrzej Lingas. Faster algorithms for finding lowest common ancestors in directed acyclic graphs. *Theor. Comput. Sci.*, 380(1-2):37–46, 2007.

9 Ran Duan and Seth Pettie. Fast algorithms for (max, min)-matrix multiplication and bottleneck shortest paths. In *Proceedings of the 20th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 384–391, 2009.

10 Michael J Fischer and Albert R Meyer. Boolean matrix multiplication and transitive closure. In *Proceedings of the 12th Annual Symposium on Switching and Automata Theory (SWAT)*, pages 129–131, 1971.

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

12 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.

13 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.

14 Fabrizio Grandoni, Giuseppe F Italian, Aleksander Łukasiewicz, Nikos Parotsidis, and Przemysław Uznański. All-pairs lca in dags: Breaking through the $o(n^{2.5})$ barrier. In *Proceedings of the 2021 ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 273–289. SIAM, 2021.

15 Karim Labib, Przemysław Uznański, and Daniel Wolleb-Graf. Hamming distance completeness. In *Proceedings of the 30th Annual Symposium on Combinatorial Pattern Matching (CPM)*, pages 14:1–14:17, 2019.

16 François Le Gall. Powers of tensors and fast matrix multiplication. In *Proceedings of the 39th International Symposium on Symbolic and Algebraic Computation (ISSAC)*, pages 296–303, 2014.

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 Andrea Lincoln, Adam Polak, and Virginia Vassilevska Williams. Monochromatic triangles, intermediate matrix products, and convolutions. In *Proceedings of the 11th Innovations in Theoretical Computer Science Conference (ITCS)*, pages 53:1–53:18, 2020.

19 Jiří Matoušek. Computing dominances in $E^n$. *Inf. Process. Lett.*, 38(5):277–278, 1991.

20 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*, volume 5125 of *Lecture Notes in Computer Science*, pages 622–633. Springer, 2008.

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

22 Avi Shoshan and Uri Zwick. All pairs shortest paths in undirected graphs with integer weights. In *Proceedings of the 40th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 605–614, 1999.

23 Jan van den Brand, Yin-Tat Lee, Danupon Nanongkai, Richard Peng, Thatchaphol Saranurak, Aaron Sidford, Zhao Song, and Di Wang. Bipartite matching in nearly-linear time on moderately dense graphs. In *2020 IEEE 61st Annual Symposium on Foundations of Computer Science (FOCS)*, pages 919–930. IEEE, 2020.

24 Virginia Vassilevska, Ryan Williams, and Raphael Yuster. Finding the smallest $H$-subgraph in real weighted graphs and related problems. In *Proceedings of the 33rd International Colloquium on Automata, Languages and Programming (ICALP), Part I*, volume 4051 of *Lecture Notes in Computer Science*, pages 262–273, 2006.

25 Virginia Vassilevska, Ryan Williams, and Raphael Yuster. All-pairs bottleneck paths for general graphs in truly sub-cubic time. In *Proceedings of the 39th Annual ACM Symposium on Theory of Computing (STOC)*, pages 585–589, 2007.

26 Virginia Vassilevska, Ryan Williams, and Raphael Yuster. Finding heaviest $H$-subgraphs in real weighted graphs, with applications. *ACM Trans. Algorithms*, 6(3):44:1–44:23, 2010.

27 Virginia Vassilevska Williams. Multiplying matrices faster than Coppersmith-Winograd. In *Proceedings of the 44th ACM Symposium on Theory of Computing (STOC)*, pages 887–898, 2012.

28 Virginia Vassilevska Williams. Problem 2 on problem set 2 of CS367, October 15, 2015. URL: `http://theory.stanford.edu/~virgi/cs367/hw2.pdf`.

29 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.

30 Virginia Vassilevska Williams and Yinzhan Xu. Truly subcubic min-plus product for less structured matrices, with applications. In *Proceedings of the 31st Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 12–29, 2020.

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

32 Raphael Yuster. Efficient algorithms on sets of permutations, dominance, and real-weighted APSP. In *Proceedings of the 20th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 950–957, 2009. URL: `http://dl.acm.org/citation.cfm?id=1496770.1496873`.

**33** Uri Zwick. All pairs lightest shortest paths. In *Proceedings of the 31st Annual ACM Symposium on Theory of Computing (STOC)*, pages 61–69, 1999.

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