Additive, Near-Additive, and Multiplicative Approximations for APSP in Weighted Undirected Graphs: Trade-Offs and Algorithms

Liam Roditty

Department of Computer Science, Bar-Ilan University, Ramat Gan, Israel

Ariel Sapir[1]

Department of Computer Science, Bar-Ilan University, Ramat Gan, Israel

Abstract

We present a $+2\sum_{i=1}^{k+1} W_i$-APASP algorithm for dense weighted graphs with a runtime of $\tilde{O}\left(n^{2+\frac{1}{3k+2}}\right)$, where W_i is the weight of an i^{th} heaviest edge on a shortest path between two vertices. Dor, Halperin and Zwick [FOCS'96 and SICOMP'00] introduced two algorithms for the commensurate unweighted $+2\cdot(k+1)$-APASP problem: one for sparse graphs with a runtime of $\tilde{O}\left(n^{2-\frac{1}{k+2}} m^{\frac{1}{k+2}}\right)$ and one for dense graphs with a runtime of $\tilde{O}\left(n^{2+\frac{1}{3k+2}}\right)$. Subsequently, Cohen and Zwick [SODA'97 and JALG'01] adapted the algorithm for sparse graphs to the weighted setting, namely a $+2\sum_{i=1}^{k+1} W_i$-APASP algorithm with the same $\tilde{O}\left(n^{2-\frac{1}{k+2}} m^{\frac{1}{k+2}}\right)$ runtime. We fill the nearly three decades old gap by providing an algorithm for dense weighted graphs, matching the runtime for the unweighted setting.

In addition, we explore *nearly additive* APASP, where the multiplicative stretch is $1+\varepsilon$. We present a $(1+\varepsilon, \min\{2W_1, 4W_2\})$-APASP algorithm with a runtime of $\tilde{O}\left(\left(\frac{1}{\varepsilon}\right)^{O(1)}\cdot n^{2.15135313}\cdot \log W\right)$. This improves upon Saha and Ye [SODA'24], which had the same runtime, yet $(1+\varepsilon, 2W_1)$-APASP.

For pure multiplicative APASP, we present a $\left(\frac{7}{3}+\varepsilon\right)$-APASP algorithm with a runtime of $\tilde{O}\left(\left(\frac{1}{\varepsilon}\right)^{O(1)}\cdot n^{2.15135313}\cdot \log W\right)$. This improves, for dense graphs, the $\tilde{O}\left(nm^{\frac{2}{3}}+n^2\right)$ runtime of the $\frac{7}{3}$-APASP algorithm by Baswana and Kavitha [FOCS'06 and SICOMP'10], at the cost of introducing an additional ε to the multiplicative stretch.

We further view this result in a broader framework of $\left(\frac{3\ell+4}{\ell+2}+\varepsilon\right)$-APASP algorithms, similarly to the family of $\frac{3\ell+4}{\ell+2}$-APASP algorithms by Akav and Roditty [ESA'21]. This also generalizes the $(2+\varepsilon)$-APASP algorithm by Dory, Forster, Kirkpatrick, Nazari, Vassilevska Williams, and de Vos [SODA'24].

Finally, we show that it is possible to "bypass" an $\tilde{\Omega}(n^{\omega})$ conditional lower bound by Dor, Halperin, and Zwick for α-APASP with $\alpha<2$, by allowing an additive component to the approximation (e.g. a $\left(\frac{6k+3}{3k+2}, \sum_{i=1}^{k+1} W_i\right)$-APASP with $\tilde{O}\left(n^{2+\frac{1}{3k+2}}\right)$ runtime.).

2012 ACM Subject Classification Theory of computation → Shortest paths

Keywords and phrases Graph, Shortest Paths, Weighted Graphs, Approximation, Undirected, Single Source Shortest-Paths, Multi-Source Shortest-Paths, All-Pairs Shortest-Paths, SSSP, MSSP, MSASP, APSP, APASP

Digital Object Identifier 10.4230/LIPIcs.FSTTCS.2025.50

Related Version *Full Version*: http://www.arxiv.org/abs/2509.04640

[1] Corresponding author

45th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2025).
Editors: C. Aiswarya, Ruta Mehta, and Subhajit Roy; Article No. 50; pp. 50:1–50:22
Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

1 Introduction

We study the classic **A**ll-**P**airs **S**hortest **P**aths (*APSP*) problem in a weighted undirected graph $G = (V, E, w)$ with non-negative weights $w : E \to \mathbb{R}^+$. Floyd-Warshall's algorithm [30, 21, 28] solves APSP in $O(n^3)$ time. Johnson's algorithm [16] improves it to $O(n^2 \log n + nm)$, which is still $O(n^3)$ for dense graphs. Despite advances (e.g. [19, 24, 29]), no truly sub-cubic APSP algorithm currently exists. The fastest known one runs in $O\left(\frac{n^3}{2^{\sqrt{\Omega(\log n)}}}\right)$ time [29].

Facing this, an immediate question arises: *Can we approximate APSP in less than $\tilde{O}(n^3)$ time?* To explore this, we properly define the problem of **A**ll-**P**airs **A**pproximated **S**hortest **P**aths (*APASP*). For parameters α, β, the goal of an (α, β)-APASP is to compute, for all $u, v \in V$, an approximate distance $d[u, v]$ s.t. $\delta(u, v) \leq d[u, v] \leq \alpha \cdot \delta(u, v) + \beta$, where $\delta(u, v)$ is the distance between u and v. If $\beta = 0$, the approximation is purely multiplicative, referred to as an α-APASP, where α is the *multiplicative stretch*; if $\alpha = 1$, the approximation is purely additive, known as $+\beta$-APASP, where β is the *additive stretch*; otherwise, the approximation is "mixed".

In weighted graphs, defining additive approximation in a meaningful way is a bit tricky. By scaling edge weights, we can show that any constant $+\beta$-APASP is equivalent to APSP. To overcome this, we explore additive stretches that depend on w. One approach is a $+\beta W$-APASP, where $W = \max_{e \in E}\{w(e)\}$. However, it might hold that most shortest paths use edges whose weight is much smaller than W, hence a $+\beta W$-APASP yields a very "weak" guarantee.

Cohen and Zwick [9] addressed this issue by defining $W_i = W_i(u \rightsquigarrow v)$ to be the weight of an i^{th} heaviest edge on a shortest path $u \rightsquigarrow v$. This approach has since influenced many approximation results for various graphs distance problems (e.g., [22, 18, 1, 14]).

We consider the approach of Cohen and Zwick under a broader spectrum which generalizes the concept of equivalence between weighted and unweighted approximations. Consider an (α, β)-APASP in an unweighted graph for arbitrary α, β and let $f(\beta, G, P)$ be a function whose parameters are: a numerical value β, the weighted graph G and a path P between two vertices V. The problem of an $(\alpha, f(\beta, G, P))$-APASP in weighted graphs is a *commensurate version* of the (α, β)-APASP problem in unweighted graphs if, when the weight function $w(e) = 1$ for all $e \in E$, the additive stretch of the two problems is the same. That is, $f(\beta, G, P) = \beta$.

Two algorithms $\mathcal{A}_1$ and $\mathcal{A}_2$, with runtimes $T_1(n)$ and $T_2(n)$, respectively, are *runtime equivalent* if $T_1(n) \in \tilde{\Theta}_W(T_2(n))$, where $\tilde{\Theta}_W(\cdot)$ hides poly-logarithmic factors of n or W. An $(\alpha, f(\beta, G, P))$-APASP algorithm $\mathcal{A}_1$ for weighted graphs is a *strongly commensurate version* of an (α, β)-APASP algorithm $\mathcal{A}_2$ in unweighted graphs if $(\alpha, f(\beta, G, P))$-APASP is a commensurate version of (α, β)-APASP and $\mathcal{A}_1$ and $\mathcal{A}_2$ are runtime equivalent. Thus, as long as $\mathcal{A}_2$ is not improved, $\mathcal{A}_1$ is optimal. We consider the broad question:

▶ **Question 1.** *For an (α, β)-APASP algorithm in unweighted graphs, what are its strongly commensurate versions $(\alpha, f(\beta, w))$-APASP algorithms in weighted graphs?*

Partial answers are already known. Cohen and Zwick [9] had two $+2W_1$-APASP algorithm with runtimes $\tilde{O}\left(n^{\frac{3}{2}} m^{\frac{1}{2}}\right)$ and $\tilde{O}\left(n^{\frac{7}{3}}\right)$, matching the runtimes of the two $+2$-APASP algorithms in unweighted graphs by Dor, Halperin, and Zwick [11]. Since $+2W_1 = +2$ when $w(e) = 1$ for all $e \in E$, the two pairs are, respectively, strongly commensurate.

Table 1 Our results for combinatorial and algebraic APASP for weighted graphs.

Approximation	Combinatorial	Algebraic	Reference
$+2\sum_{i=1}^{k+1} W_i$	$n^{2+\frac{1}{3k+2}}$	–	Section 4
$\frac{7}{3}+\varepsilon$	–	$mn^{\beta}+n^{2+\gamma}+\left(\frac{1}{\varepsilon}\right)^{O(1)}\cdot n^{\omega(1-\beta-\gamma)}\cdot\log W$ $\approx\left(\frac{1}{\varepsilon}\right)^{O(1)}\cdot n^{2.15135313}\cdot\log W$	Section 5
$\frac{3\ell+4}{\ell+2}+\varepsilon$	–	$mn^{\beta}+n^{2+\gamma}+\left(\frac{1}{\varepsilon}\right)^{O(1)}\cdot n^{\omega(1-\beta-\ell\cdot\gamma)}\cdot\log W$	Section 6
$(1+\varepsilon,\min\{2W_1,4W_2\})$	–	$mn^{\beta}+n^{2+\gamma}+\left(\frac{1}{\varepsilon}\right)^{O(1)}\cdot n^{\omega(1-\beta-\gamma)}\cdot\log W$ $\approx\left(\frac{1}{\varepsilon}\right)^{O(1)}\cdot n^{2.15135313}\cdot\log W$	Section 7

Cohen and Zwick also generalized one of their $+2W_1$-APASP algorithms to a $+2\sum_{i=1}^{k+1} W_i$-APASP algorithm for weighted sparse graphs with $\tilde{O}\left(n^{2-\frac{1}{k+2}}m^{\frac{1}{k+2}}\right)$ runtime, hence finding a strongly commensurate version to the unweighted $+2\cdot(k+1)$-APASP algorithm of Dor, Halperin, and Zwick for sparse graphs. Yet, the $+2\cdot(k+1)$-APASP algorithm for unweighted dense graphs with $\tilde{O}\left(n^{2+\frac{1}{3k+2}}\right)$ runtime of Dor, Halperin and Zwick was left unaddressed, raising the following question:

▶ **Question 2.** *Is there a* $+2\sum_{i=1}^{k+1} W_i$*-APASP algorithm which is a strongly commensurate version of the* $+2\cdot(k+1)$*-APASP algorithm for dense unweighted graphs of [11]?*

Question 2 remained open for nearly three decades since the work of Cohen and Zwick [9], despite extensive progress on (α,β)-APASP algorithms for weighted graphs (e.g., [6, 17, 7, 3, 22, 12]). In this paper, we answer it in the affirmative by presenting a $+2\sum_{i=1}^{k+1} W_i$-APASP algorithm for dense weighted graphs, running in $\tilde{O}\left(n^{2+\frac{1}{3k+2}}\right)$ time – establishing a strongly commensurate version of Dor, Halperin, and Zwick's algorithm.

A meaningful observation is that, in this approximation, no edge should be taken into consideration more than once. That is, any i^{th} heaviest edge can contribute to the additive stretch at most once. Therefore, for $u, v \in V$, if there are less than k edges in a given shortest path $u \rightsquigarrow v$, the approximation which is due to that specfific path $u \rightsquigarrow v$, is bounded by $3\cdot\delta(u,v)$. If **all** shortest paths between u and v have less than k edges, the approximation for this pair is bounded by $3\cdot\delta(u,v)$.

In our approach, we utilized only hitting sets and **S**ingle **S**ource **S**hortest **P**aths (*SSSP*) invocations. However, other powerful tools exist. **M**atrix **M**ultiplication (*MM*), particularly **M**in **P**lus **M**atrix **M**ultiplication (*MPMM*), is a key tool for reducing the runtime of shortest path algorithms. Over the years, there have been many improvements (e.g. [5, 13, 27, 4]) on the exponent of n for the runtime of fast MM. However, these fast MM algorithms "hide" large constants in their runtime, which make them inefficient for practical use.

Motivated by this, Aingworth, Chekuri, Indyk and Motwani [2] initiated the study of algorithms for graph approximation problems (not necessarily APASP) that avoid fast MM. Since then, APASP algorithms have been broadly categorized into either: an *algebraic*

Table 2 Known additive (and nearly) combinatorial and algebraic APASP for unweighted graphs.

Approximation	Combinatorial	Algebraic
$+2$	$\min\left\{n^{\frac{7}{3}}, n^{\frac{3}{2}} m^{\frac{1}{2}}\right\}$ [11]	$n^2 t^{\frac{1}{2}} + MPCRBD(n, \frac{n}{t}) \approx n^{2.2867}$ [10]
$(1+\varepsilon, 2)$	–	$n^{2+\frac{1-r}{2}} + \frac{1}{\varepsilon} n^{\omega(r)} \approx \frac{1}{\varepsilon} \cdot n^{2.15135313}$ [12]
$+2 \cdot (k+1)$	$\min\left\{n^{2+\frac{1}{3k+2}}, n^{2-\frac{1}{k+2}} m^{\frac{1}{k+2}}\right\}$ [11]	–

approach – which allows the use of fast MM (e.g., [8, 12, 17, 22, 31, 15]), and a *combinatorial approach* – where such algorithms are not used (e.g., [3, 6, 12, 17, 20, 9]). We summarize[2] results for unweighted graphs in Table 2 and for weighted graphs in Table 3.

One algebraic approach focuses on *nearly additive* approximations. Specifically, we show a $(1+\varepsilon, \min\{2W_1, 4W_2\})$-APASP with a runtime of $\tilde{O}\left(\frac{1}{\varepsilon} \cdot n^{2.15135313} \cdot \log W\right)$. This improves upon Saha and Ye [22], which had the same runtime, yet computed a $(1+\varepsilon, 2W_1)$-APASP.

Another approach considers purely multiplicative stretches. We present a general framework of $\left(\frac{3\ell+4}{\ell+2} + \varepsilon\right)$-APASP algorithms with $\tilde{O}\left(mn^{\beta} + n^{2+\gamma} + \left(\frac{1}{\varepsilon}\right)^{O(1)} \cdot n^{\omega(1-\beta-\ell\cdot\gamma)} \cdot \log W\right)$ runtime, where β and γ are parameters which depend on the graph's density and ℓ is a parameter that indicates the quality of the stretch. When the graph is dense, the runtime becomes $\tilde{O}\left(n^{2+\beta} + \left(\frac{1}{\varepsilon}\right)^{O(1)} \cdot n^{\omega(1-(\ell+1)\cdot\beta)} \cdot \log W\right)$, where $2+\beta = \omega(1-(\ell+1)\cdot\beta)$.

To properly understand the results that lie within this framework, we must first examine purely multiplicative stretches. In this setting, the goal is to compute an α-APASP as efficiently as possible. To do so, we first consider the known upper and lower bounds.

Dor, Halperin and Zwick [11] show that computing a $(2-\varepsilon)$-APASP, for any $\varepsilon > 0$, is as hard as **B**oolean **M**atrix **M**ultiplication (*BMM*), which is only known to be solved in $\tilde{O}(n^{\omega})$[3]. This conditional lower bound means that 2-APASP is the smallest multiplicative stretch s.t. an α-APASP algorithm may require less than $\tilde{O}(n^{\omega})$ time.

Since APASP algorithms output an $n \times n$ matrix, $\Omega(n^2)$ time is unavoidable. For weighted graphs, the only known purely multiplicative stretch that can be computed in quadratic time is 3-APASP [9, 6]. This stands in contrast to the conditional lower bound for $(2-\varepsilon)$-APASP, implying the existence of a threshold $\alpha \in [2,3]$, for which there is an α-APASP algorithm with $\tilde{O}(n^2)$ runtime, while any $(\alpha-\varepsilon)$-APASP algorithm would require more[4] than quadratic time.

▶ **Question 3.** *What is the smallest $\alpha \in [2,3]$, for which there is an α-APASP algorithm that runs in $\tilde{O}(n^2)$ time, while any $(\alpha-\varepsilon)$-APASP algorithm necessitates $\tilde{\Omega}(n^{2+\gamma})$ time?*

---

[2] Some results depend on certain parameters, such as $\omega(\alpha, \beta, \gamma)$ or $MPCRBD(n, m)$, which is the runtime for a MPMM on column and row bounded difference matrices of dimensions $n \times m$ and $m \times n$. We provide the mathematical expression and the numerical exponent with currently known values.

[3] $\omega(\alpha, \beta, \gamma)$ is the infimum of the exponent of the runtime for all algorithms which compute the multiplication of an $n^{\alpha} \times n^{\beta}$ and $n^{\beta} \times n^{\gamma}$ matrices. $\omega(\beta) = \omega(1, \beta, 1)$ and $\omega = \omega(1)$.

[4] Assuming BMM cannot be solved in quadratic time.

Table 3 Known combinatorial and algebraic APASP runtimes for weighted graphs.

Approximation	Combinatorial	Algebraic
$+2W_2$	$n^{\frac{3}{2}}m^{\frac{1}{2}}$ [9]	-
$+\min\{2W_1, 4W_2\}$	$n^{\frac{7}{3}}$ [9]	-
$(1+\varepsilon, 2W_1)$	–	$\frac{1}{\varepsilon} \cdot n^{\omega(1,1-x,1)} \cdot \log W + n^{2+\frac{x}{2}}$ $\approx \frac{1}{\varepsilon} \cdot n^{2.15135313} \cdot \log W$ [22]
$+2\sum_{i=1}^{k+1} W_i$	$n^{2-\frac{1}{k+2}}m^{\frac{1}{k+2}}$ [9]	–
$\left(1+\varepsilon, 2\sum_{i=1}^{k+1} W_i\right)$	–	$\frac{1}{\varepsilon} \cdot n^{\omega\left(1-\frac{k}{k+2}\cdot x, 1-x, 1-\frac{k-1}{k+2}\cdot x\right)} \cdot \log W + n^{2+\frac{x}{k+2}}$ [22]
2	$n^{\frac{1}{2}}m + n^2$ [6, 3]	–
$2+\varepsilon$	–	$n^{1-r}m + \left(\frac{1}{\varepsilon}\right)^{O(1)} n^{\omega(r)} \cdot \log W$ $\approx \left(\frac{1}{\varepsilon}\right)^{O(1)} \cdot n^{2.21235201} \cdot \log W$ [12]
$(2, W_1)$	n^2 [6, 7]	–
$\frac{7}{3}$	$nm^{\frac{2}{3}} + n^2$ [6, 3]	–
$\frac{3\ell+4}{\ell+2}$	$n^{2-\frac{3}{\ell+2}}m^{\frac{2}{\ell+2}} + n^2$ [3]	–
3	n^2 [9, 6]	–

While exact quadratic-time algorithms are rare, there is notable progress for "near quadratic" ones, as summarized in Table 3. This paper contributes in this direction by presenting a $(\frac{7}{3}+\varepsilon)$-APASP algorithm with $\tilde{O}\left(\left(\frac{1}{\varepsilon}\right)^{O(1)} \cdot n^{2.15135313} \cdot \log W\right)$ runtime. For dense graphs, we improve the $\tilde{O}\left(m^{\frac{2}{3}}n + n^2\right)$ runtime of the $\frac{7}{3}$-APASP algorithm by Baswana and Kavitha [6], at the cost of adding an ε to the multiplicative stretch.

Baswana and Kavitha also presented a 2-APASP algorithm with $\tilde{O}\left(n^{\frac{3}{2}}m^{\frac{1}{2}} + n^2\right)$ runtime. Recently, Dory, Forster, Kirkpatrick, Nazari, Vassilevska Williams and de Vos [12] presented a $(2+\varepsilon)$-APASP algorithm with a $\tilde{O}\left(\left(\frac{1}{\varepsilon}\right)^{O(1)} \cdot n^{2.21235201} \cdot \log W\right)$ runtime, improving the runtime for dense graphs at the cost of adding an ε. This algorithm and our own $\left(\frac{7}{3}+\varepsilon\right)$-APASP algorithm belong to a the previously discussed unified framework of $\left(\frac{3\ell+4}{\ell+2}+\varepsilon\right)$-APASP algorithms with $\tilde{O}\left(mn^{\beta} + n^{2+\gamma} + \left(\frac{1}{\varepsilon}\right)^{O(1)} \cdot n^{\omega(1-\beta-\ell\cdot\gamma)} \cdot \log W\right)$ runtime.

While these results primarily explore trade-offs between the runtime and the multiplicative stretch, we would also like to fix the runtime and provide trade-offs between the multiplicative and additive stretches, proceeding in alignment with the direction of Saha and Ye [22].

We begin by focusing on cases where the multiplicative stretch is below 2, offering a way to "bypass" the conditional lower bound of $\Omega(n^{\omega})$ of Dor, Halperin and Zwick [11] for (α, β)-APASP algorithms where $\alpha < 2$.

▶ **Question 4.** *For which values of α, β such that $\alpha < 2$ is it possible to compute an (α, β)-APASP in less than $\tilde{O}(n^{\omega})$ runtime?*

We partially answer this by presenting a family of (α, β)-APASP algorithms with $\tilde{O}\left(n^{2+\frac{1}{3k+2}}\right)$ runtime, such as $\left(\frac{6k+3}{3k+2}, \sum_{i=1}^{k+1} W_i\right)$-APASP. Furthermore, our approach helps bridge a known gap in the literature: no combinatorial algorithms currently exists for purely

Table 4 Runtime equivalence for $\tilde{O}\left(n^{2+\frac{1}{3k+2}}\right)$.

Approximation	Combinatorial	Reference
$+2\sum_{i=1}^{k+1} W_i$	$n^{2+\frac{1}{3k+2}}$	Section 4
$\left(\frac{6k+3}{3k+2}, \sum_{i=1}^{k+1} W_i\right)$	$n^{2+\frac{1}{3k+2}}$	Section 8
$\left(\frac{(9k+4)\cdot\alpha+(3k+2)\cdot\beta}{(\alpha+\beta)\cdot(3k+2)}, \frac{2\beta}{\alpha+\beta}\sum_{i=1}^{k+1} W_i\right)$	$n^{2+\frac{1}{3k+2}}$	Section 8
$\frac{9k+4}{3k+2}$	$\leq n^{2+\frac{1}{3k+2}}$	[3]

multiplicative α-APASP in weighted graphs where $\alpha \in \left(2, \frac{7}{3}\right)$. The lack of such can be partially accommodated by an (α, β)-APASP s.t. $\alpha \in \left(2, \frac{7}{3}\right)$ and β is small. For instance, we obtain a $(\frac{11}{5}, \frac{W_1+W_2}{4})$-APASP in $\tilde{O}\left(n^{\frac{11}{5}}\right)$ time and a $\left(\frac{13}{6}, \frac{2\cdot(W_1+W_2+W_3)}{3}\right)$-APASP in $\tilde{O}\left(n^{\frac{17}{8}}\right)$ time (See: Appendix C). These belong to a wider family of approximations computable in $\tilde{O}(n^{2+\frac{1}{3k+2}})$ time (see Table 4). We broaden the question to any runtime function $T(n)$:

▶ **Question 5.** *Given an (α_1, β_1)-APASP with $T(n)$ runtime, what other (α_2, β_2)-APASP can be computed within $\tilde{O}(T(n))$ runtime?*

1.1 Related Work

Cohen and Zwick [9] had a 2-APASP algorithm with $\tilde{O}\left(n^{\frac{3}{2}}m^{\frac{1}{2}} + n^2\right)$ runtime and a $\frac{7}{3}$-APASP in $\tilde{O}\left(n^{\frac{7}{3}}\right)$ time. Both runtimes were improved by Baswana and Kavitha [6] to $\tilde{O}(m\sqrt{n} + n^2)$ and $\tilde{O}\left(m^{\frac{2}{3}}n + n^2\right)$, respectively. Kavitha [17] showed a $\frac{5}{2}$-APASP algorithm with $\tilde{O}\left(n^{\frac{9}{4}}\right)$ runtime, which was improved by Akav and Roditty [3] to $\tilde{O}\left(n^{\frac{5}{4}}m^{\frac{1}{2}} + n^2\right)$ time.

These multiplicative stretches were initially computed in an ad-hoc manner, each requiring its own tailored algorithm. The approaches for these APASP algorithms were developed without a unifying framework or a recognition of a broader underlying structure. Only later did Akav and Roditty [3] observe that these seemingly distinct stretches are, in fact, a part of a family of problems characterized by the form of $\frac{3\ell+4}{\ell+2}$-APASP that can be computed in $\tilde{O}\left(n^{2-\frac{3}{\ell+2}}m^{\frac{2}{\ell+2}} + n^2\right)$ time. For example, $\ell = 0$ indicates a 2-APASP algorithm that runs in $\tilde{O}\left(n^{\frac{1}{2}}m + n^2\right)$ and $\ell = 3$ indicates a $\frac{13}{5}$-APASP algorithm that runs in $\tilde{O}\left(n^{\frac{7}{5}}m^{\frac{2}{5}} + n^2\right)$ time.

A related problem is **M**ulti-**S**ource **S**hortest **P**aths (*MSSP*), where instead of computing all pairs of distances, we are given a set $S \subseteq V$ of sources and we need to compute the distances from this set to any $v \in V$. We define an (α, β)-MSASP similarly to an (α, β)-APASP. Let $\varepsilon > 0$, $|S| \in O(n^r)$ and assume edge weights are from $\{0, 1, \ldots, W\} \cup \{\infty\}$. Recently, Elkin and Neiman [15] a $(1+\varepsilon)$-MSASP algorithm with $\tilde{O}\left(m^{1+o(1)} + \left(\frac{1}{\varepsilon}\right)^{O(1)} \cdot n^{\omega(r)} \cdot \log W\right)$ runtime.

Dory, Forster, Kirkpatrick, Nazari, Vassilevska Williams and de Vos [12] utilized the result of Elkin and Neiman [15] and the 2-APASP of Baswana and Kavitha [6]. They achieved a $(2+\varepsilon)$-APASP algorithm with a $\tilde{O}\left(\left(\frac{1}{\varepsilon}\right)^{O(1)} \cdot n^{2.21235201} \cdot \log W\right)$ runtime, which improves the runtime of the 2-APASP of Baswana and Kavitha [6] for dense graphs. We generalize this $(2+\varepsilon)$-APASP algorithm [12] and our own $\left(\frac{7}{3}+\varepsilon\right)$-APASP algorithm, similarly to Akav and Roditty [3], and present a unified framework of $\left(\frac{3\ell+4}{\ell+2}+\varepsilon\right)$-APASP algorithms.

An equivalent approach was taken by Saha and Ye [22]. They computed a $(1+\varepsilon, 2\sum_{i=1}^{k+1} W_i)$-APASP. Their base case was $(1+\varepsilon, 2W_1)$-APASP with $\tilde{O}\left(\frac{1}{\varepsilon}\cdot n^{2.15135313}\cdot \log W\right)$ runtime. We improve the base case and compute a $(1+\varepsilon, \min\{2W_1, 4W_2\})$-APASP, in the same runtime.

2 Toolkit

2.1 Basic graph notions

Unless stated otherwise, we consider an undirected graph $G=(V,E,w)$ with a non-negative weight function $w: E\to\mathbb{R}^{\geq 0}$, where the weight of an edge $(u,v)\in E$ is $w(u,v)$. Denote $|V|=n$ and $|E|=m$. For $k\in\mathbb{N}$, let $[k]=\{1,2,\ldots,k\}$. Each vertex $u\in V$ has its set of neighbours $N(u)$ and the set $\Gamma(u,k)$ of k-nearest neighbours. A *path* from a vertex u to a vertex v is an ordered list $u=y_0,y_1,y_2,\ldots,y_k,y_{k+1}=v$ s.t. for any $i\in[k+1]$: $(y_{i-1},y_i)\in E$. The weight of a path is $w(P)=\sum_{e\in P} w(e)$. A path with the smallest weight is a *shortest path* and its weight is the *distance*, denoted by $\delta(u,v)$. The distance between a vertex u to a set of vertices $S\subseteq V$ is the distance from u to the nearest $s\in S$. That is, $\delta(u,S)=\min_{s\in S}\{\delta(u,s)\}$. The *pivot* of u is a[5] vertex $p_S(u)$ such that $\delta(u,p_S(u))=\delta(u,S)$.

Denote a[6] shortest path between u and v by $u \rightsquigarrow v$: $w(u \rightsquigarrow v)=\delta(u,v)$. If $u \rightsquigarrow v$ is a single edge, we write u—v, hence $w(u,v)=\delta(u,v)$. Let $W_i(u \rightsquigarrow v)$ be the weight of the i^{th} heaviest edge. That is, $W_1(u \rightsquigarrow v)$ for the heaviest, $W_2(u \rightsquigarrow v)$ for the second heaviest, etc. The notion of a $W_i(u \rightsquigarrow v)$ applies to the fixed path $u \rightsquigarrow v$, and so does $\sum_{i\in I} W_i$. Yet, if no specific $u \rightsquigarrow v$ is stated, we consider **any**[7] shortest path, but all edges must be **from the same path**.

We denote by $d[u,v]$ the distance estimate computed by our algorithms. For an (α,β)-APASP algorithm, we prove that $\delta(u,v)\leq d[u,v]\leq \alpha\cdot\delta(u,v)+\beta$. Since G is undirected, we assume that any update to $d[u,v]$ also updates $d[v,u]$. This adds only $O(1)$ per update and does not affect the overall runtime.

2.2 Nearest vertices and pivots

Given several sets $T_1,T_2,\ldots,T_\ell$ of elements from a universe $U=\{u_1,u_2,\ldots,u_n\}$, a *hitting set* is a set $S\subseteq U$ such that for any $i\in[\ell]$: $S\cap T_i\neq\varnothing$. In our context, $U=V$ and $T_u=\Gamma(u,k)$ for $u\in V$ and some k that will be later specified. A hitting set, in this sense, is a set $S\subseteq V$ of small size which contains at least one of the k-nearest vertices of u, for all $u\in V$. Hence its pivot $p_S(u)$ is the nearest vertex between u and S. That is, $\delta(u,p_S(u))\leq\delta(u,s)$ for all $s\in S$.

Let $A\subseteq V$ and $u\in V$. We can consider $\Gamma(u,A,k)=\{a_{1,u},a_{2,u},\ldots,a_{k,u}\}$ where $\delta(u,a_{i-1,u})\leq\delta(u,a_{i,u})$ for $i\in[k]$, and for any $a\in A$: $\delta(a_{k,u})\leq\delta(a,u)$. Hence, the nearest vertices from the set A. These definitions are useful for APASP by distinguishing between

[5] If there are several candidates we can break ties by a consistent criteria.
[6] Several shortest paths may exist. If nothing is specifically mentioned, we consider an arbitrary one.
[7] Or, the smallest sum.

pairs of vertices $u, v \in V$ that are "far" from each other – and those that are "near" – that is, one lies within $\Gamma(\cdot, k)$ of the other. For near vertices, usually the exact distance is computed, by an algorithm [6, 25] similar to Dijkstra, while for "far" vertices, we use an approximation on paths that pass through the pivots.

$E_S(u) = \{(u,v) \mid w(u,v) < \delta(u,S)\}$ is the set of edges near u whose weight is smaller than $\delta(u,S)$, and $E_S = \bigcup_{u\in V} E_S(u)$. Denote the auxiliary edges $H_S = \{(u, p_S(u)) \mid u \in V\}$. If there are multiple hitting sets $H = \bigcup_S H_S$ is their union. When concerning a hierarchy of hitting-sets $S_1 \supseteq S_2 \supseteq \ldots$, we assume that $S_{i+1} \subseteq S_i$, and that each S_i has its set of edges E_{S_i} and pivots $p_i(u) \in S_i$ for each $u \in V$ (See Appendix A).

2.3 Min-plus matrix multiplication

A problem related to APSP is MPMM, which uses a different multiplication operator: given two matrices A and B, their product $C = A \star B$, satisfies $C[i,j] = \min_{k=1}^{n}\{A[i,k] + B[k,j]\}$. MPMM is runtime-equivalent to APSP [26]. However, MM and MPMM are not not known to be runtime-equivalent[8]. Nonetheless, faster MM algorithms can be utilized for MPMM by redefining the matrices (e.g. [23, 31, 8]).

Zwick [31] introduced **A**pproximated **M**in-**P**lus **M**atrix **M**ultiplication (*AMPMM*) in the same sense as APASP and presented a $(1+\varepsilon)$-AMPMM algorithm with $\tilde{O}\big(\frac{1}{\varepsilon}\cdot n^{\omega}\cdot \log W\big)$ runtime. That is, this algorithm computes a matrix $\tilde{C}$, where $C[i,j] \le \tilde{C}[i,j] \le (1+\varepsilon)\cdot C[i,j]$. It was later utilized for a $(1+\varepsilon)$-APASP [8] in $\tilde{O}\big(\frac{1}{\varepsilon}\cdot n^{\omega}\cdot \log \frac{n}{\varepsilon}\big)$ time.

3 Overview

Common to all our algorithms is a hierarchy of hitting sets $V = S_0 \supseteq S_1 \supseteq \ldots \supseteq \varnothing$, where any $S_{i+1} \subseteq S_i$. Each level S_i has its own pivots $p_i(u)$ for every $u \in V$ and an associated edge set E_{S_i} (see: Section 2.2). The number and sizes of the hitting sets differs between the algorithms.

From the combinatorial perspective, our methods consist of selecting pivots through hitting sets and invoking multiple SSSP on graphs with sets of original and auxiliary sets of edges, tailored for the desired approximation. Some of our algorithms are order-sensitive considering the various SSSP invocations, as explicitly stated if so. We also consider distance updates through selected vertices, i.e. we have $d[u,x]$ and $d[x,v]$ and we update $d[u,v]$.

Our main contribution in the combinatorial setting is our $+2\sum_{i=1}^{k+1} W_i$-APASP algorithm for dense weighted graphs with a runtime of $\tilde{O}\left(n^{2+\frac{1}{3k+2}}\right)$. Extending the $+2\cdot(k+1)$-APASP algorithm of Dor, Halperin, and Zwick [11] to the weighted setting gives rise to several challenges. First, we use $\Gamma(u,k)$ instead of $N(u)$. This change is required to ensure that if there exists an edge $(x,y) \in u \rightsquigarrow v$ s.t. $(x,y) \notin E_{S_i}$, then $\delta(x, p_i(x)) \le w(x,y)$, hence a path $u \rightsquigarrow p_i(x) \rightsquigarrow v$ would not "cost" us more than $2\cdot w(x,y)$, which is at most $2W_1(u \rightsquigarrow v)$. Repeating this approach $k+1$ times results in an additive error of at most $2\cdot(k+1)\cdot W_1(u \rightsquigarrow v)$. However, our goal is to avoid reusing the weight W_1 of the heaviest edge repeatedly and to use the lighter edges $W_2, W_3, ..., W_{k+1}$. To obtain this, we must fix a certain shortest path and carefully make sure we do not use any edge more than once in the upper-bound for our

[8] If they would have been, as we already know $\omega < 3$, we could refute the APSP conjecture.

additive error. We achieve that by recursively concatenating auxiliary paths through pivots, ensuring that each additive term corresponds to a distinct edge from the original $u \rightsquigarrow v$. We emphasize that this issue does not exist in the unweighted setting, where the upper-bound is simply $+2 \cdot (k+1)$.

From the algebraic perspective, our methods consist of invoking $(1+\varepsilon)$-MSASP instead of several SSSP invocations, and $(1+\varepsilon)$-AMPMM for designated matrices. Note that rectangular MPMM can be used as a tool to replace distance updates through selected vertices.

Our main contribution in the algebraic setting is our unified framework of $\left(\frac{3\ell+4}{\ell+2}+\varepsilon\right)$-APASP algorithms with $\tilde{O}\left(mn^{\beta}+n^{2+\gamma}+\left(\frac{1}{\varepsilon}\right)^{O(1)} \cdot n^{\omega(1-\beta-\ell\cdot\gamma)} \cdot \log W\right)$ runtime. For dense graphs $m \in \Theta\left(n^2\right)$, setting $\ell=0$ results in the $(2+\varepsilon)$-APASP algorithm of Dory, Forster, Kirkpatrick, Nazari, Vassilevska Williams and de Vos [12], setting $\ell=1$ results in our own $\left(\frac{7}{3}+\varepsilon\right)$-APASP algorithm. For $\ell=2$ we obtain a $\left(\frac{5}{2}+\varepsilon\right)$-APASP algorithm with $\tilde{O}\left(\left(\frac{1}{\varepsilon}\right)^{O(1)} \cdot n^{2.1185119} \cdot \log W\right)$ runtime. These results improve upon the framework of Akav and Roditty [3] for dense graphs, at the cost of an additional ε to the multiplicative stretch.

4 Additive $+2\sum_{i=1}^{k+1} W_i$-APASP

Algorithm 1 uses a parameter k that indicates a trade-off between the runtime and the approximation and parameters $\beta_1, \ldots, \beta_{3k+2} \in (0,1)$, whose values will be determined as part of the runtime analysis. Our algorithm works as follows: for each $i \in [3k+2]$ we set $\alpha_i = \sum_{j=1}^{i} \beta_j$ and compute a hitting set S_i to the sets $\Gamma(u, n^{\alpha_i})$, over all $u \in V$. For each S_i we also compute its corresponding E_{S_i}. We set $S_0 = V$ and $S_{3k+3} = \varnothing$, hence $E_{S_{3k+3}} = E$.

We initialize our distance approximation matrix $d : V \times V \to \mathbb{R}^{\geq 0}$ with edges' weights or ∞ otherwise. For $i \in [3k+2]$ and $u \in V$ we find the pivot $p_i(u)$ and set $d[u, p_i(u)] \leftarrow \delta(u, p_i(u))$ by a single SSSP invocation (See: Observation 14). We proceed with SSSP invocations from all $s \in S_i$ over the weighted graph $G_i(s) = \left(V, E_{S_{i+1}} \cup (\{s\} \times V) \cup H \cup F_i, d\right)$, where:

- $E_{S_{i+1}}$ - The union, over **all** $u \in V$, of edges (u,v) such that $w(u,v) < \delta(u, p_{i+1}(u))$,
- $\{s\} \times V$ - Auxiliary edges from s to V, representing distances previously computed,
- $H = \{(u, p_j(u)) \mid u \in V\}$ - Auxiliary edges from **any** u to its pivots $p_j(u)$ for $j \in [3k+2]$,
- F_i - Auxiliary edges from any $s_j \in S_j$ to any $s_\ell \in S_\ell$, for indices j, ℓ s.t. $i+j+\ell \geq 3k+1$.

The weights of these edges are given by $d : V \times V \to \mathbb{R}^{\geq 0}$, which is a weight function over $G_i(s)$. These SSSP invocations are performed in descending order, from $i \leftarrow 3k+2$ to 0, since they rely on auxiliary edges whose weights were previously computed. To properly understand the reason behind this, consider a $u \in V$ and let $i \in [3k+2]$ be the largest index such that $u \in S_i$. Whenever $d[p_{i+1}(v), u]$ is updated, we also update $d[u, p_{i+1}(v)]$. Therefore, when Algorithm 1 will invoke SSSP from $u \in S_i$, the auxiliary edge $(u, p_{i+1}(v))$ will be considered as well, and its weight is the already computed value $d[u, p_{i+1}(v)]$.

To prove an upper-bound of $+2\sum_{i=1}^{k+1} W_i$, we show $d[u,v] \leq \delta(u,v) + 2 \cdot (w(e_1) + \ldots + w(e_{k+1}))$ for **distinct** $k+1$ edges $e_1, \ldots, e_{k+1} \in u \rightsquigarrow v$. That is, our main argument would be that $e_i \neq e_j$, i.e.: no edge is taken into consideration in the upper-bound more than once. Let:

Algorithm 1 $+2\sum_{i=1}^{k+1} W_i$-APASP$(G = (V, E, w))$.

Input: An undirected non-negative weighted graph $G = (V, E, w)$ and $k \in \mathbb{N}$.
Output: A $+2\sum_{i=1}^{k+1} W_i$-APASP.

Let $\beta_1, \ldots, \beta_{3k+2} \in (0, 1)$ to be fixed later
Initialize d as the edges' weights and ∞ otherwise
for $i \in [3k + 2]$ **do**
  $\alpha_i \leftarrow \sum_{j=1}^{i} \beta_j$
  for $u \in V$ **do**
    Compute $\Gamma(u, V, n^{\alpha_i})$
  Compute S_i to be a hitting set to $\{\Gamma(u, V, n^{\alpha_i}) \mid u \in V\}$
  Compute the distances from any $u \in V$ to its pivot $p_i(u)$
  Construct the set of edges E_{S_i}
for $i \in \langle 3k + 2, \ldots, 1, 0\rangle$ **do**
  $F_i \leftarrow \varnothing$
  for $j \in \{0, 1, \ldots, 3k + 2\}$ **do**
    for $\ell \in \{3k + 1 - i - j, 3k + 2 - i - j, \ldots, 3k + 2\}$ **do**
      $F_i \leftarrow F_i \cup (S_\ell \times S_j)$
  for $s \in S_i$ **do**
    Invoke SSSP from s on $G_i(s) = \big(V, E_{S_{i+1}} \cup (\{s\} \times V) \cup H \cup F_i, d\big)$ and update d accordingly
return d

- $\kappa(u)$ be the largest index $i \in [3k + 2]$ for which $u \in S_i$. As the graph is undirected, we can assume w.l.o.g that $\kappa(v) \leq \kappa(u)$ (See: Figure 1).
- $\lambda(u \rightsquigarrow v)$ be smallest $j \in [3k + 3]$ s.t. $u \rightsquigarrow v \subseteq E_{S_j} \cup H$, $u \rightsquigarrow v \not\subseteq E_{S_{j-1}} \cup H$ (See: Figure 2).
- $\xi(j, u \rightsquigarrow v)$ be the edge $(x, y) \in u \rightsquigarrow v$ for which $y \rightsquigarrow v \subseteq E_{S_{j+1}}$ and $(x, y) \notin E_{S_j} \cup H$. In other words, it is the first edge from v's side that does not belong to $E_{S_{j+1}} \cup H$. Additionally:
  - $\varphi(j, u \rightsquigarrow v) = x$ and $\psi(j, u \rightsquigarrow v) = y$ are the vertices of this edge (See: Figure 3).
  - $\zeta(j, u \rightsquigarrow v)$ is the maximal index $\ell \in [3k + 2]$ for which: $\delta(y, p_\ell(y)) \leq w(x, y)$. The edge $(x, y) \notin E_{S_{j+1}}$, hence $\delta(y, p_{j+1}(y)) \leq w(x, y)$. Hence, such $j + 1 \leq \ell \leq 3k + 2$ exists.
- $\xi(u \rightsquigarrow v)$ be simply $\xi(\lambda(u \rightsquigarrow v) - 1, u \rightsquigarrow v)$. That is, we view the first edge (x, y), from v's side, s.t. $(x, y) \notin E_{S_{\lambda(u \rightsquigarrow v)}} \cup H$ while $y \rightsquigarrow v \subseteq E_{S_{\lambda(u \rightsquigarrow v)}} \cup H$. Note such an edge exists due to the definition of $\lambda(u \rightsquigarrow v)$. We denote the endpoints of this edge by $\varphi(u \rightsquigarrow v)$ $\psi(j, u \rightsquigarrow v)$, and define: $\zeta(u \rightsquigarrow v) = \zeta(\lambda(u \rightsquigarrow v) - 1, u \rightsquigarrow v)$ (Replace j by $\lambda(u \rightsquigarrow v) - 1$ in Figure 3).
- $\Delta(u \rightsquigarrow v)$ be an upper-bound on the difference between the computed distance estimation $d[u, v]$ and the real distance $\delta(u, v)$, where $d[u, v]$ is computed due to the path $u \rightsquigarrow v$.
- $|u \rightsquigarrow v|$ be the number of edges in $u \rightsquigarrow v$.
- $\mu(x \text{—} y, u \rightsquigarrow v)$ be the[9] integer $j \in [|u \rightsquigarrow v|]$ such that $W_j(u \rightsquigarrow v) = w(x, y)$. In other words, it is an index specifying "how heavy" is the edge (x, y).

[9] If the weights of two edges on a certain shortest path are equal, we can break ties.

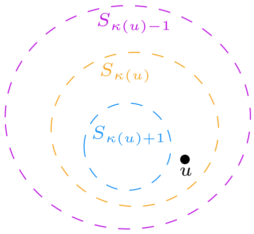

Figure 1 The (smallest hitting set) largest index $i \in [3k+2]$ s.t. $u \in S_i$.

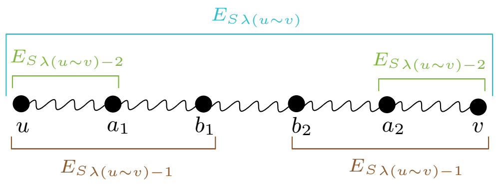

Figure 2 The first index $j \in [3k+2]$ s.t. $u \rightsquigarrow v \subseteq E_{S_j} \cup H$.

- $\mu(j, u \rightsquigarrow v)$ be equal to $\mu(\xi(j, u \rightsquigarrow v), u \rightsquigarrow v)$. That is, we ask "how heavy" is the edge $\xi(j, u \rightsquigarrow v)$, which is the first edge, from v's side, that is not in $E_{S_{j+1}} \cup H$.
- and $\mu(u \rightsquigarrow v)$ be $\mu(\xi(u \rightsquigarrow v), u \rightsquigarrow v)$ – "how heavy" is the edge $\xi(u \rightsquigarrow v)$, which is the first edge, from v's side, that is not in $E_{S_{\lambda(u \rightsquigarrow v)}} \cup H$.

We state a recurrence relation on $\Delta(u \rightsquigarrow v)$, considering three possible conditions:

▶ **Lemma 1.**

$$\Delta(u \rightsquigarrow v) \leq \begin{cases} 0 & \lambda(u \rightsquigarrow v) \leq \kappa(u)+1 & :\text{(A)} \\ 2W_{\mu(u \rightsquigarrow v)} & \begin{array}{l}\lambda(u \rightsquigarrow v) > \kappa(u)+1 \text{ and} \\ \lambda(u \rightsquigarrow v) \geq 3k+1-\kappa(u)-\kappa(v)\end{array} & :\text{(B)} \\ \begin{array}{l} 2W_{\mu(\kappa(u)+1, u \rightsquigarrow v)} + \\ \Delta\big(p_{\zeta(\kappa(u)+1, u \rightsquigarrow v)}(y) \rightsquigarrow y — x \rightsquigarrow u\big)\end{array} & \begin{array}{l}\lambda(u \rightsquigarrow v) > \kappa(u)+1 \text{ and} \\ \lambda(u \rightsquigarrow v) < 3k+1-\kappa(u)-\kappa(v)\end{array} & :\text{(C)} \end{cases}$$

Where $x = \varphi(\kappa(u)+1, u \rightsquigarrow v)$ *and* $y = \psi(\kappa(u)+1, u \rightsquigarrow v)$.

The proof Lemma 1 is in Appendix B. The core of Algorithm 1's correctness lies within Lemma 2, where we utilize Lemma 1 and show two claims: One is that no edge is considered more than once in the upper-bound and the other is that there at most $k+1$ recurrences until we reach a halting condition – either Condition (A) or Condition (B) of Lemma 1. By proving both claims, we conclude that there are at most $k+1$ **distinct** edges $e_1, \ldots, e_{k+1} \in u \rightsquigarrow v$ that are considered in the upper-bound.

Fix a $u \rightsquigarrow v$. As G is undirected w.l.o.g. $\kappa(v) \leq \kappa(u)$. Let $(x, y) = \xi(\kappa(u)+1, u \rightsquigarrow v)$ be the first edge, from v's perspective, not in $E_{S_{\kappa(u)+1}}$. If we had an estimate $d[u, y]$, invoking SSSP from $u \in S_{\kappa(u)}$ would result in $d[u, v] \leq d[u, y] + \delta(y, v)$, as $y \rightsquigarrow v \subseteq E_{S_{\kappa(u)+1}}$.

To estimate $d[u, y]$, we would need an index $j \in [3k+2]$ s.t. $u \rightsquigarrow p_j(y) \rightsquigarrow y$ would be a good estimate on $\delta(u, y)$. To avoid having (x, y) considered more than once in the upper-bound, we select the pivot $p_{\zeta(\kappa(u)+1, u \rightsquigarrow v)}(y)$ as in Condition (C) of Lemma 1 (See: Figure 4).

As for the number of recurrences, note that Condition (A) and Condition (B) of Lemma 1 are the halting conditions of the recurrence relation. We prove two upper-bounds on the number of recurrences of Condition (C) of Lemma 1: one until reaching Condition (A) and the other until reaching Condition (B).

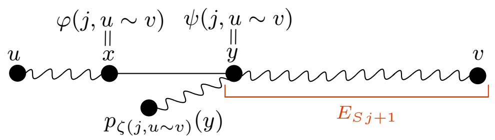

Figure 3 From v's perspective, the subpath $y \rightsquigarrow v \subseteq E_{S_{j+1}} \cup H$ while $(x, y) \notin E_{S_{j+1}} \cup H$. We denote $\varphi(j, u \rightsquigarrow v) = x$, $\psi(j, u \rightsquigarrow v) = y$ and $\xi(j, u \rightsquigarrow v) = (x, y)$. Note that $\delta(y, p_{j+1}(y)) \leq w(x, y)$, and let $\zeta(j, u \rightsquigarrow v)$ be the largest index s.t. $\delta\Big(y, p_{\zeta(j,u \rightsquigarrow v)}(y)\Big) \leq w(x, y)$.

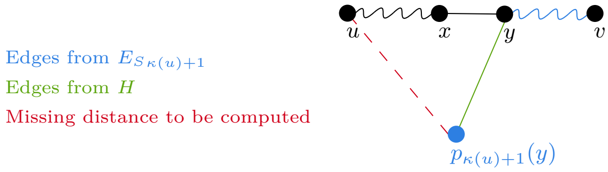

Figure 4 The difference Δ between d and δ. For a simpler, yet inaccurate, understanding, we illustrate $p_{\kappa(u)+1}(y)$. However, the correct pivot to consider is $p_{\zeta\left(\kappa(u)+1, u \rightsquigarrow v\right)}(y)$.

▶ **Lemma 2.** $\Delta(u \rightsquigarrow v) \leq 2 \sum_{i=1}^{k+1} W_i$

Proof. If we are either at Condition (A) or Condition (B) of Lemma 1, the claim is trivial. In Condition (C): $\Delta(u \rightsquigarrow v) = 2W_{\mu(\kappa(u)+1, u \rightsquigarrow v)} + \Delta\big(p_{\zeta(\kappa(u)+1, u \rightsquigarrow v)}(y) \rightsquigarrow y — x \rightsquigarrow u\big)$, where $x = \varphi(\kappa(u) + 1, u \rightsquigarrow v)$ and $y = \psi(\kappa(u) + 1, u \rightsquigarrow v)$. We want to show:

1. No edge is being taken into consideration in $\Delta(u \rightsquigarrow v)$ more than once.
2. The number of recurrences until (including) we reached a halting condition is at most $k + 1$.

We start by showing no edge is considered twice. Let $\hat{u} = p_{\zeta(\kappa(u)+1, u \rightsquigarrow v)}(y)$ and $\hat{v} = u$. Hence, in Condition (C): $\Delta\big(p_{\zeta(\kappa(u)+1, u \rightsquigarrow v)}(y) \rightsquigarrow y — x \rightsquigarrow u\big) = \Delta(\hat{u} \rightsquigarrow y — x \rightsquigarrow \hat{v})$. Observe that $\kappa(\hat{u}) = \kappa\big(p_{\zeta(\kappa(u)+1, u \rightsquigarrow v)}(y)\big) = \zeta(\kappa(u) + 1, u \rightsquigarrow v) \geq \kappa(u) + 1$. By the definition of $\zeta(\kappa(u) + 1, u \rightsquigarrow v)$ we know that $y — x \subseteq E_{S_{\zeta(\kappa(u)+1, u \rightsquigarrow v)+1}} = E_{S_{\kappa(\hat{u})+1}}$. Moreover, the added sub-path $\hat{u} \rightsquigarrow y \subseteq H$, as $\hat{u}$ is pivot of y. Consequently, $\hat{u} \rightsquigarrow y — x \subseteq E_{S_{\kappa(\hat{u})+1}} \cup H$. That is, the edge $\xi(\kappa(\hat{u}) + 1, \hat{u} \rightsquigarrow y — x \rightsquigarrow \hat{v})$ is from the path $x \rightsquigarrow \hat{v}$. Ergo, the recurrence in Condition (C) considers a different edge each time.

We now prove the second claim: for the number of recurrences of Condition (C). As we showed in each recurrence a different edge is considered, all that remains is to bound the number of edges, which is at most one more than the number of recurrences (if we halted due to Condition (B)). We consider two options: $\kappa(u) = \kappa(v)$ and $\kappa(u) > \kappa(v)$. We begin with the first, and by the end briefly consider the second option. We provide two bounds, one when the relation halted due to Condition (B) and the other when it halted due to Condition (A).

For Condition (B), observe that $\kappa(\hat{u}) \geq \kappa(u) + 1$ and $\kappa(\hat{v}) = \kappa(u) = \kappa(v)$. Let $\hat{\hat{u}} = p_{\zeta(\kappa(\hat{u})+1, \hat{u} \rightsquigarrow y — x \rightsquigarrow \hat{v})}(\psi(\kappa(\hat{u}) + 1, \hat{u} \rightsquigarrow y — x \rightsquigarrow \hat{v}))$ and $\hat{\hat{v}} = \hat{u}$. Therefore, $\kappa(\hat{\hat{u}}) \geq \kappa(\hat{u}) + 1$ and $\kappa(\hat{\hat{v}}) = \kappa(\hat{u}) \geq \kappa(u) + 1 > \kappa(u) = \kappa(\hat{v})$. In other words, in the first recurrence of Condition (C), $\kappa(\hat{u})$ increased by at least 1 compared to $\kappa(u)$ and $\kappa(\hat{v})$ is equal to $\kappa(v)$. By the second invocation, both $\kappa(\hat{\hat{u}})$ and $\kappa(\hat{\hat{v}})$ increase by at least 1 compared to $\kappa(\hat{u})$ and $\kappa(\hat{v})$, respectively. The value $\lambda(u \rightsquigarrow v) = \lambda(\hat{u} \rightsquigarrow y — x \rightsquigarrow \hat{v})$ because the only edges we add to the path are from H and we neglect edges that belong to $E_{S_{\kappa(u)+1}} \subseteq E_{S_{\lambda(u \rightsquigarrow v)-1}}$, assuming $\lambda(u \rightsquigarrow v) > \kappa(u) + 1$. Therefore, there would be at most $\left\lceil \frac{3k+1-\lambda(u \rightsquigarrow v)}{2} \right\rceil - \kappa(u)$ recurrences, which means at most $\left\lceil \frac{3k+1-\lambda(u \rightsquigarrow v)}{2} \right\rceil - \kappa(u) + 1 < \frac{3k+1-\lambda(u \rightsquigarrow v)}{2} - \kappa(u) + 2$ distinct edges.

For Condition (A), we bound the number of recurrences by $\lambda(u \rightsquigarrow v) - \kappa(u) - 2$, as $\kappa(u)$ increases by at least 1 during each recurrence. This is also the number of distinct edges.

As both upper-bounds on the number of distinct edges hold, we compare them to find the actual bound: $\lambda(u \rightsquigarrow v) - \kappa(u) - 2 = \frac{3k+1-\lambda(u \rightsquigarrow v)}{2} - \kappa(u) + 2$. This holds for $\lambda(u \rightsquigarrow v) = k+3$. The number of edges is at most $\lambda(u \rightsquigarrow v) - \kappa(u) - 2 = k + 1 - \kappa(u) \leq k + 1$, as $\kappa(u) \geq 0$.

As for the second option, $\kappa(u) \geq \kappa(v)$, the only change is that $\kappa(\hat{u})$ and $\kappa(\hat{v})$ increase by at least 1 from the first iteration rather than the second, which serves in our favor. ◀

▶ **Theorem 3.** *Algorithm 1 computes a* $+2\sum_{i=1}^{k+1} W_i$*-APASP in* $\tilde{O}\left(n^{2+\frac{1}{3k+2}}\right)$ *time.*

The correctness proof and runtime analysis appear in the full version.

5 Multiplicative $\left(\frac{7}{3} + \varepsilon\right)$-APASP

Algorithm 2 uses two parameters $\beta, \gamma \in (0,1)$ whose value will be determined when we will analyze the runtime. Our algorithm works as follows: we compute the hitting set S_1 (resp. S_2) for the sets $\Gamma\left(u, V, n^{\beta}\right)$ (resp. $\Gamma\left(u, V, n^{\beta+\gamma}\right)$), over all $u \in V$. We then compute the set $\Gamma(u, S_1, n^{\gamma})$. We compute E_{S_1} (resp. E_{S_2}). We also set $S_0 = V$ and $S_3 = \varnothing$.

We initialize our distance approximation matrix $d : V \times V \to \mathbb{R}^{\geq 0}$ to be either the weight of the edge or ∞. We find the pivots $p_1(u)$ (resp. $p_2(u)$) for all $u \in V$ and the exact distance $d[u, p_1(u)] \leftarrow \delta(u, p_1(u))$ (resp. $d[u, p_2(u)]$) by a single SSSP invocation (See: Observation 14). We then update distances for $(x, y) \in E$ to the pivots $p_1(v)$ and $p_2(v)$ of all $v \in \Gamma\left(y, V, n^{\beta}\right)$.

Let $i \in \{0, 1\}$. We begin by invoking SSSP from all $s \in S_i$ over the weighted graph $G_i(s) = \left(V, E_{S_{i+1}} \cup (\{s\} \times V) \cup H, d\right)$, where:

- $E_{S_{i+1}}$ - The union, over **all** $u \in V$, of edges (u, v) such that $w(u, v) < \delta(u, p_{i+1}(u))$,
- $\{s\} \times V$ - Auxiliary edges from s to V, representing distances previously computed,
- $H = \{(u, p_j(u)) \mid u \in V\}$ - Auxiliary edges from **any** u to its pivots $p_j(u)$ for $j \in \{1, 2\}$.

The weight of each such edge is given by $d : V \times V \to \mathbb{R}^{\geq 0}$, which is a weight function over $G_i(s)$. We then invoke a $(1+\varepsilon)$-MSASP from S_2. In contrast to Algorithm 1, the order between the invocations of SSSP and MSASP here holds no importance.

Finally, we perform two distance updates. The first is $d[u, v] \leftarrow \min\{d[u, s] + d[s, v], d[u, v]\}$ for $s \in \Gamma(u, S_1, n^{\gamma})$ and the second is simply by all of the pivots of u and v, that is: $d[u, v] \leftarrow \min\{d[u, p_i(u)] + d[p_i(u), v], d[u, v]\}$.

To prove a multiplicative stretch of $\left(\frac{7}{3} + \varepsilon\right)$, we follow the steps of the proof of the $\frac{7}{3}$-APASP algorithm of Baswana and Kavitha [6] by providing a case analysis and proving each case individually.

▶ **Theorem 4.** *Algorithm 2 computes a* $\left(\frac{7}{3} + \varepsilon\right)$*-APASP in* $\tilde{O}\left(\left(\frac{1}{\varepsilon}\right)^{O(1)} \cdot n^{2.15135313} \cdot \log W\right)$ *time, for an arbitrary* $\varepsilon > 0$ *and weights of the set* $[W] \cup \{0, \infty\}$.

Algorithm 2 $\left(\frac{7}{3}+\varepsilon\right)$-APASP$(G=(V,E,w))$.

Input: An undirected non-negative weighted graph $G=(V,E,w)$ and an $\varepsilon>0$.
Output: A $\left(\frac{7}{3}+\varepsilon\right)$-APASP.

Let $\beta,\gamma\in(0,1)$ to be fixed later
Initialize d as the edges' weights and ∞ otherwise
Compute $\Gamma\big(u,V,n^{\beta}\big)$ (resp. $\Gamma\big(u,V,n^{\beta+\gamma}\big)$) for all $u\in V$
Compute S_1 (resp. S_2) as a hitting set to $\big\{\Gamma\big(u,V,n^{\beta}\big)\mid u\in V\big\}$ (resp. $\Gamma\big(u,V,n^{\beta+\gamma}\big)$)
Compute the distances from any $u\in V$ to its pivot $p_1(u)$ (resp. $p_2(u)$)
Construct the set of edges E_{S_1} (resp. E_{S_2})
for $(x,y)\in E$ **do**
  for $v\in\Gamma\big(y,V,n^{\beta}\big)$ **do**
    for $i\in\{0,1,2\}$ **do**
      $d[p_i(v),x]\leftarrow\min\{d[p_i(v),v]+d[v,y]+d[x,y],d[p_i(v),x]\}$
for $i\in\{0,1\}$ **do**
  for $s\in S_i$ **do**
    Invoke SSSP from s on $G_i(s)=\big(V,E_{S_{i+1}}\cup(\{s\}\times V)\cup H,d\big)$ and update d accordingly
Compute a $(1+\varepsilon)$-MSASP from S_2 on $G=(V,E,w)$ and update d accordingly
for $u,v\in V$ **do**
  for $s\in\Gamma(u,S_1,n^{\gamma})$ **do**
    $d[u,v]\leftarrow\min\{d[u,s]+d[s,v],d[u,v]\}$
for $u,v\in V$ **do**
  for $i\in[1,2]$ **do**
    $d[u,v]\leftarrow\min\{d[u,p_i(u)]+d[p_i(u),v],d[u,v]\}$
return d

In the full version, we prove the above and show how the runtime can be made to depend on the graph's density, namely:

$$\tilde{O}\left(mn^{\beta}+n^{2+\gamma}+\left(\frac{1}{\varepsilon}\right)^{O(1)}\cdot n^{\omega(1-\beta-\gamma)}\cdot\log W\right)$$

6 Multiplicative $\left(\frac{3\ell+4}{\ell+2}+\varepsilon\right)$-APASP

We extend our $\left(\frac{7}{3}+\varepsilon\right)$-APASP algorithm (Algorithm 2) into a framework of $\frac{3\ell+4}{\ell+2}$-APASP algorithms (Algorithm 3). Let ℓ be a parameter that indicates the trade-off between the runtime and the approximation and $\beta_1,\ldots,\beta_{\ell+1}\in(0,1)$ be parameters, whose values depend on graph's density. The proofs for Algorithm 3 appear in the full version.

Our algorithm works as follows: for $i\in[\ell+1]$ we set $\alpha_i=\sum_{j=1}^{i}\beta_j$ and compute a hitting set S_i to the sets $\Gamma(u,n^{\alpha_i})$, over all $u\in V$. We compute the corresponding E_{S_i}. We also set $S_0=V$ and $S_{\ell+2}=\varnothing$, hence $E_{S_{\ell+2}}=E$. We initialize the distance approximation matrix $d:V\times V\to\mathbb{R}^{\geq 0}$ to hold all edges' weights or ∞ else. We find the pivot $p_i(u)$ for all $u\in V$ and update the distance $d[u,p_i(u)]\leftarrow\delta(u,p_i(u))$ by a single SSSP invocation (See: Observation 14).

Algorithm 3 $\left(\frac{3\ell+4}{\ell+2}+\varepsilon\right)$-APASP$(G=(V,E,w))$.

Input: An undirected graph $G=(V,E)$, a positive weight function $w:E\to\mathbb{R}^+$, an integer $\ell\in\mathbb{N}$ and an arbitrary $\varepsilon>0$.
Output: A matrix d which is a $\left(\frac{3\ell+4}{\ell+2}+\varepsilon\right)$-APASP.

```
Let β_1, ..., β_{ℓ+1} ∈ (0,1) to be fixed later
Initialize d as the edges' weights and ∞ otherwise
for i ∈ [ℓ] do
    α_i ← Σ_{j=1}^{i} β_j
    for u ∈ V do
        Compute Γ(u, V, n^{α_i})
    Compute S_i to be a hitting set to {Γ(u, V, n^{α_i}) | u ∈ V}
    Compute Γ(u, S_i, n^{β_{i+1}})
    Compute the distances from any u ∈ V to its pivot p_i(u)
    Construct the set of edges E_{S_i}
for (x, y) ∈ E do
    for i ∈ [ℓ+1] do
        for v ∈ Γ(y, S_{i-1}, n^{β_i}) do
            d[p_i(v), x] ← min{d[p_i(v), v] + d[v, y] + d[x, y], d[p_i(v), x]}
for i ∈ {0} ∪ [ℓ] do
    for s ∈ S_i do
        Invoke SSSP from s on G_i(s) = (V, E_{S_{i+1}} ∪ ({s} × V) ∪ H, d) and update d
          accordingly
Compute a (1+ε)-MSASP from S_{ℓ+1} on G = (V, E, w) and update d accordingly
for u, v ∈ V do
    for i ∈ [ℓ] do
        for s ∈ Γ(u, S_i, n^{β_{i+1}}) do
            d[u, v] ← min{d[u, s] + d[s, v], d[u, v]}
for u, v ∈ V do
    min_{i=0}^{ℓ+1} {d[u, p_i(u)] + d[p_i(u), v], d[u, v]}
return d
```

Following the initialization, we update the distances, for $i\in[\ell+1]$, from any $(x,y)\in E$ to $p_i(v)$ of $v\in\Gamma\big(y,S_{i-1},n^{\beta_i}\big)$. That is, $d[p_i(v),x]\leftarrow\min\{d[p_i(v),v]+d[v,y]+d[x,y],d[p_i(v),x]\}$. We now consider an $i\in\{0\}\cup[\ell]$ and proceed with an SSSP invocation from all $s\in S_i$ over the weighted graph $G_i(s)=\big(V,E_{S_{i+1}}\cup(\{s\}\times V)\cup H,d\big)$, where:

- $E_{S_{i+1}}$ - The union, over **all** $u\in V$, of edges (u,v) such that $w(u,v)<\delta(u,p_{i+1}(u))$,
- $\{s\}\times V$ - Auxiliary edges from s to V, representing distances previously computed,
- $H=\{(u,p_j(u))\mid u\in V\}$ - Auxiliary edges from **any** u to its pivots $p_j(u)$ for $j\in[\ell+1]$.

The weight of each is given by $d:V\times V\to\mathbb{R}^{\geq 0}$. Observe that d is a weight function over $G_i(s)$. We then proceed by a $(1+\varepsilon)$-MSASP invocation from the set of sources $S_{\ell+1}$ over the entire graph. We stress that, contrary to Algorithm 1 and similarly to Algorithm 2, the order of invocations of SSSP and MSASP here holds no importance.

Finally, we apply two types of distance updates. Let $u, v \in V$ and $i \in [\ell+1]$. First, $d[u,v] \leftarrow \min\{d[u,s]+d[s,v], d[u,v]\}$ through any $s \in \Gamma\big(u, S_i, n^{\beta_{i+1}}\big)$. The second is from all the $2\cdot(\ell+1)$ pivots[10] of both u and v, namely: $d[u,v] \leftarrow \min\{d[u,p_i(u)]+d[p_i(u),v], d[u,v]\}$.

To prove the correctness of our algorithm, we follow an analogous proof to the $\frac{3\ell+4}{\ell+2}$-APASP of Akav and Roditty [3]. As we performed changes to their algorithm, we provide a full proof of Algorithm 3, indicating where the additional ε affects. We begin with two upper-bounds:

▶ **Lemma 5.** *$d[u,v] \le (1+\varepsilon)\cdot\min\{3\cdot\delta(u,v)-2\cdot\delta(b,v), 3\cdot\delta(u,v)-2\cdot\delta(u,a)\}$*

▶ **Lemma 6.** *$d[u,v] \le \delta(u,v)+2\cdot(\ell+1)\cdot\delta(u,a)$ or $d[u,v] \le \delta(u,v)+2\cdot(\ell+1)\cdot\delta(b,v)$*

The proof of Lemma 5 and Lemma 6 appear in the full version. We follow with our main theorem:

▶ **Theorem 7.** *Let $\alpha_j = \sum_{i=1}^{j}\beta_i$, $\varepsilon > 0$ with weights from $[W]\cup\{0,\infty\}$. Algorithm 3 computes a $\left(\frac{3\ell+4}{\ell+2}+\varepsilon\right)$-APASP in $\tilde{O}\left(\left(\frac{1}{\varepsilon}\right)^{O(1)}\cdot n^{\omega(1-\alpha_{\ell+1})}\cdot\log W + mn^{\beta_1} + \sum_{i=2}^{\ell+1} n^{2+\beta_i}\right)$ time.*

The proof for Theorem 7 appears in the full version. We observe that this runtime can be written as $\tilde{O}\Big(mn^{\beta}+n^{2+\gamma}+\big(\frac{1}{\varepsilon}\big)^{O(1)}\cdot n^{\omega(1-\beta-\ell\cdot\gamma)}\cdot\log W\Big)$, by selecting $\beta_1=\beta$ and $\beta_2=\ldots=\beta_{\ell+1}=\gamma$.

To perceive the wide range of our results, assume the graph is dense $m \in \tilde{\Theta}\big(n^2\big)$. When $\ell=0$ we compute a $(2+\varepsilon)$-APASP in $\tilde{O}\Big(\big(\frac{1}{\varepsilon}\big)^{O(1)}\cdot n^{2.21235201}\cdot\log W\Big)$ time, which is the same as Dory, Forster, Kirkpatrick, Nazari, Vassilevska Williams, and de Vos [12]; when $\ell=1$ our algorithm computes a $\big(\frac{7}{3}+\varepsilon\big)$-APASP in $\tilde{O}\Big(\big(\frac{1}{\varepsilon}\big)^{O(1)}\cdot n^{2.15135313}\cdot\log W\Big)$ time – as in Algorithm 2; when $\ell=2$ our algorithm computes a $\big(\frac{5}{2}+\varepsilon\big)$-APASP in $\tilde{O}\Big(\big(\frac{1}{\varepsilon}\big)^{O(1)}\cdot n^{2.1185119}\cdot\log W\Big)$ time, etc.

7 Near additive $(1+\varepsilon, \min\{2W_1, 4W_2\})$-APASP

Algorithm 4 has two parameters $\beta, \gamma \in (0,1)$ whose values depend on the graph's density and will be determined when we will analyze the runtime. It works as follows: we compute the hitting set S_1 (resp. S_2) for the sets $\Gamma\big(u, V, n^{\beta}\big)$ (resp. $\Gamma\big(u, V, n^{\beta+\gamma}\big)$), over all $u \in V$. We then compute the set $\Gamma(u, S_1, n^{\gamma})$. We compute E_{S_1} (resp. E_{S_2}). We also set $S_0 = V$ and $S_3 = \varnothing$.

The distance approximation matrix $d : V\times V \to \mathbb{R}^{\ge 0}$ is initialized to the weight of the edges or ∞ else-wise. We find the pivots $p_1(u)$ (resp. $p_2(u)$) for all $u \in V$ and the exact distance $d[u,p_1(u)] \leftarrow \delta(u,p_1(u))$ (resp. $d[u,p_2(u)]$) by a single SSSP invocation (See: Observation 14).

We proceed by invoking a $(1+\varepsilon)$-MSASP from S_2 over the entire graph. We then compute a $(1+\varepsilon)$-AMPMM of $d_0^2 \star d_2^0$, where d_i^j is a rectangular matrix containing all values of d, its rows narrowed down to include only the vertices of S_i and its columns to those of S_j. We note that similarly to Algorithm 1, the order is of importance here.

The following, in decreasing order $i \leftarrow 1$ to 0 (only two hitting sets), occurs twice: we invoke SSSP from all $s \in S_i$ on the graph $G_i(s) = \big(V, E_{S_{i+1}} \cup (\{s\}\times V) \cup H, d\big)$, where:

[10] Recall that $d[u,v]$ is updated whenever $d[v,u]$ is updated.

Algorithm 4 $(1+\varepsilon, \min\{2W_1, 4W_2\})$-APASP$(G=(V,E,w))$.

Input: An undirected non-negative weighted graph $G=(V,E,w)$ and an $\varepsilon>0$.
Output: A $(1+\varepsilon, \min\{2W_1, 4W_2\})$-APASP.

Let $\beta, \gamma \in (0,1)$ to be fixed later
Initialize d as the edges' weights and ∞ otherwise
Compute $\Gamma\big(u, V, n^{\beta}\big)$ (respectively, $\Gamma\big(u, V, n^{\beta+\gamma}\big)$) for all $u \in V$
Compute S_1 (respectively, S_2) to be a hitting set to $\big\{\Gamma\big(u, V, n^{\beta}\big) \mid u \in V\big\}$ (respectively, $\Gamma\big(u, V, n^{\beta+\gamma}\big)$)
Compute the distances from any $u \in V$ to its pivot $p_1(u)$ (respectively, $p_2(u)$)
Construct the set of edges E_{S_1} (respectively, E_{S_2})
Compute $(1+\varepsilon)$-MSASP from S_2 on $G=(V,E,w)$ and update d accordingly
Compute a $(1+\varepsilon)$-AMPMM $d_0^2 \star d_2^0$ and update d accordingly
for $j \in [2]$ **do**
  for $i \in \langle 1, 0\rangle$ **do**
    for $s \in S_i$ **do**
      Invoke SSSP from s on $G_i(s) = \big(V, E_{S_{i+1}} \cup (\{s\} \times V) \cup H, d\big)$ and update d accordingly
Compute a $(1+\varepsilon)$-AMPMM $d_0^1 \star d_1^1$ and update d accordingly
for $u \in S_0 = V$ **do**
  Invoke SSSP from u on $G_0(u) = (V, E_{S_1} \cup (\{u\} \times V) \cup H, d)$ and update d accordingly
return d

- $E_{S_{i+1}}$ - The union, over **all** $u \in V$, of edges (u,v) such that $w(u,v) < \delta(u, p_{i+1}(u))$,
- $\{s\} \times V$ - Auxiliary edges from s to V, representing distances previously computed,
- $H = \{(u, p_j(u)) \mid u \in V\}$ - Auxiliary edges from **any** u to its pivots $p_j(u)$ for $j \in \{1,2\}$.

In other words, after we performed an SSSP invocation from members of S_1, then $S_0 = V$, then S_1 and then again $S_0 = V$. Observe that $d : V \times V \to \mathbb{R}^{\geq 0}$ is a weight function over $G_i(s)$.

We then compute another $(1+\varepsilon)$-AMPMM, this time for the rectangular matrices $d_0^1 \star d_1^1$. Finally, we invoke one more SSSP from all $u \in V = S_0$ over the graph $G_i(u) = (V, E_{S_1} \cup (\{u\} \times V) \cup H, d)$.

To prove the *near additive* approximation of $(1+\varepsilon, \min\{2W_1, 4W_2\})$, we provide a case analysis and prove each case individually in the full version, where we also show the runtime analysis. We then conclude:

▶ **Theorem 8.** *Algorithm 4 runs in $\tilde{O}\left(\left(\frac{1}{\varepsilon}\right)^{O(1)} \cdot n^{2.15135313} \cdot \log W\right)$ time and computes a $(1+\varepsilon, \min\{2W_1, 4W_2\})$-APASP, for weights of the range $[W] \cup \{0, \infty\}$.*

We observe that while Algorithm 4 computes an equal or tighter approximation than that of Algorithm 2, the latter's runtime can be faster, as it depends on the graph's density.

8 Multiplicative and additive trade-offs

Recall Algorithm 1 which computed a $+2\sum_{i=1}^{k+1} W_i$-APASP and the $\frac{3\ell+4}{\ell+2}$-APASP algorithm by Akav and Roditty [3]. For dense graphs $m \in \tilde{\Theta}(n^2)$, the latter is a $\frac{9k+4}{3k+2}$-APASP with $\tilde{O}\left(n^{2+\frac{1}{3k+2}}\right)$ runtime, by setting $\ell = 3k$.

▶ **Claim 9.** *Let $\alpha, \beta \in \mathbb{R}^+$. It is possible to compute a* $\left(\frac{(9k+4)\cdot\alpha+(3k+2)\cdot\beta}{(\alpha+\beta)\cdot(3k+2)}, \frac{2\beta}{\alpha+\beta}\sum_{i=1}^{k+1} W_i\right)$*-APASP in* $\tilde{O}\left(n^{2+\frac{1}{3k+2}}\right)$ *time.*

The proof is straightforward by running both algorithms (it appears in the full version). We observe that Claim 9 partially addresses Question 5 by presenting a family of APASP algorithms computable in $\tilde{O}(n^{2+\frac{1}{3k+2}})$ time. Claim 9 also partially answers Question 4, as, when $\alpha = \beta = 1$, we get a $\left(\frac{6k+3}{3k+2}, \sum_{i=1}^{k+1} W_i\right)$-APASP, whose multiplicative stretch is smaller than 2. For instance, $k = 1$ indicates that a $\left(\frac{9}{5}, W_1 + W_2\right)$-APASP can be computed in $\tilde{O}\left(n^{\frac{11}{5}}\right)$.

The same idea can be applied as well to Algorithm 3 that computes a $\left(\frac{3\ell+4}{\ell+2} + \varepsilon\right)$-APASP and the $\left(1 + \varepsilon, 2\sum_{i=1}^{k+1} W_i\right)$-APASP algorithm of Saha and Ye [22], e.g. a $\left(\frac{5}{3} + \varepsilon, \min\{W_1, 2W_2\}\right)$-APASP computed in $\tilde{O}\left(\left(\frac{1}{\varepsilon}\right)^{O(1)} \cdot n^{2.15135313} \cdot \log W\right)$ time.

References

1 Reyan Ahmed, Greg Bodwin, Faryad Darabi Sahneh, Stephen Kobourov, and Richard Spence. Weighted additive spanners. *arXiv*, 2021. arXiv:2002.07152.

2 Donald Aingworth, Chandra Chekuri, Piotr Indyk, and Rajeev Motwani. Fast estimation of diameter and shortest paths (without matrix multiplication). *SIAM Journal on Computing*, 28(4):1167–1181, 1999. doi:10.1137/S0097539796303421.

3 Maor Akav and Liam Roditty. A unified approach for all pairs approximate shortest paths in weighted undirected graphs. In *29th Annual European Symposium on Algorithms (ESA 2021)*, volume 204 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 4:1–4:18, Dagstuhl, Germany, 2021. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.ESA.2021.4.

4 Josh Alman, Ran Duan, Virginia Vassilevska Williams, Yinzhan Xu, Zixuan Xu, and Renfei Zhou. More asymmetry yields faster matrix multiplication. *arXiv*, abs/2404.16349, 2024. doi:10.48550/arXiv.2404.16349.

5 Andris Ambainis, Yuval Filmus, and François Le Gall. Fast matrix multiplication: Limitations of the coppersmith-winograd method. *Proceedings of the forty-seventh annual ACM symposium on Theory of Computing*, 2014. URL: https://api.semanticscholar.org/CorpusID:8332797.

6 Surender Baswana and Telikepalli Kavitha. Faster algorithms for all-pairs approximate shortest paths in undirected graphs. *SIAM Journal on Computing*, 39(7):2865–2896, 2010. doi:10.1137/080737174.

7 Piotr Berman and Shiva Prasad Kasiviswanathan. Faster approximation of distances in graphs. In *Algorithms and Data Structures*, pages 541–552, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg. doi:10.1007/978-3-540-73951-7_47.

8 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 SIGACT Symposium on Theory of Computing*, STOC 2019, pages 943–954, New York, NY, USA, 2019. Association for Computing Machinery. doi:10.1145/3313276.3316373.

9 Edith Cohen and Uri Zwick. All-pairs small-stretch paths. *Journal of Algorithms*, 38(2):335–353, 1997. doi:http://doi.org/10.1006/jagm.2000.1117.

10 Mingyang Deng, Yael Kirkpatrick, Victor Rong, Virginia Vassilevska Williams, and Ziqian Zhong. New additive approximations for shortest paths and cycles. In *49th International Colloquium on Automata, Languages, and Programming (ICALP 2022)*, volume 229 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 50:1–50:10, Dagstuhl, Germany, 2022. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. `doi:10.4230/LIPIcs.ICALP.2022.50`.

11 Dorit Dor, Shay Halperin, and Uri Zwick. All-pairs almost shortest paths. *SIAM Journal on Computing*, 29(5):1740–1759, 2000. `doi:10.1137/S0097539797327908`.

12 Michal Dory, Sebastian Forster, Yael Kirkpatrick, Yasamin Nazari, Virginia Vassilevska Williams, and Tijn De Vos. Fast 2-approximate all-pairs shortest paths. *arXiv*, 2023. `doi:10.1137/1.9781611977912.169`.

13 Ran Duan, Hongxun Wu, and Renfei Zhou. Faster matrix multiplication via asymmetric hashing. In *2023 IEEE 64th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 2129–2138, Los Alamitos, CA, USA, 2023. IEEE Computer Society. `doi:10.1109/FOCS57990.2023.00130`.

14 Michael Elkin, Yuval Gitlitz, and Ofer Neiman. Improved weighted additive spanners. In *35th International Symposium on Distributed Computing (DISC 2021)*, volume 209 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 21:1–21:15, Dagstuhl, Germany, 2021. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. `doi:10.4230/LIPIcs.DISC.2021.21`.

15 Michael Elkin and Ofer Neiman. Centralized, parallel, and distributed multi-source shortest paths via hopsets and rectangular matrix multiplication. In *39th International Symposium on Theoretical Aspects of Computer Science (STACS 2022)*, volume 219 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 27:1–27:22, Dagstuhl, Germany, 2022. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. `doi:10.4230/LIPIcs.STACS.2022.27`.

16 Donald Bruce Johnson. Efficient algorithms for shortest paths in sparse networks. *J. ACM*, 24(1):1–13, January 1977. `doi:10.1145/321992.321993`.

17 Telikepalli Kavitha. Faster algorithms for all-pairs small stretch distances in weighted graphs. *Algorithmica*, 63:224–245, 2012. `doi:10.1007/s00453-011-9529-y`.

18 An La and Hung Le. New weighted additive spanners. *arXiv*, 2024. `doi:10.48550/arXiv.2408.14638`.

19 Michael Lawrence Fredman. New bounds on the complexity of the shortest path problem. *SIAM Journal on Computing*, 5(1):83–89, 1976. `doi:10.1137/0205006`.

20 Liam Roditty. New algorithms for all pairs approximate shortest paths. In *Proceedings of the 55th Annual ACM Symposium on Theory of Computing*, STOC 2023, pages 309–320, New York, NY, USA, 2023. Association for Computing Machinery. `doi:10.1145/3564246.3585197`.

21 Bernard Roy. Transitivité et connexité. *Comptes rendus de l'Académie des Sciences*, 249:216–218, 1959. URL: `https://gallica.bnf.fr/ark:/12148/bpt6k3201c/f222.image`.

22 Barna Saha and Christopher Ye. *Faster Approximate All Pairs Shortest Paths*, pages 4758–4827. siam, 2023. `doi:10.1137/1.9781611977912.170`.

23 Avi Shoshan and Uri Zwick. All pairs shortest paths in undirected graphs with integer weights. FOCS '99, page 605, USA, 1999. IEEE Computer Society.

24 Tadao Takaoka. A new upper bound on the complexity of the all pairs shortest path problem. *Information Processing Letters*, 43(4):195–199, 1992. `doi:http://doi.org/10.1016/0020-0190(92)90200-F`.

25 Mikkel Thorup and Uri Zwick. Approximate distance oracles. *Journal of the ACM*, 52:1–24, 2005. `doi:10.1145/1044731.1044732`.

26 Alfred Vaino Aho, John Edward Hopcroft, and Jeffrey David Ullmand. *The Design and Analysis of Computer Algorithms*. Always learning. Pearson Education, 1974. URL: `https://books.google.co.il/books?id=0ipKsZbwSAoC`.

27 Virginia Vassilevska Williams, Yinzhan Xu, Zixuan Xu, and Renfei Zhou. *New Bounds for Matrix Multiplication: from Alpha to Omega*, pages 3792–3835. SIAM, 2024. `doi:10.1137/1.9781611977912.134`.

28 Stephen Warshall. A theorem on boolean matrices. *J. ACM*, 9(1):11–12, January 1962. doi:10.1145/321105.321107.

29 Richard Ryan Williams. Faster All-Pairs Shortest Paths via Circuit Complexity. *SIAM Journal on Computing*, 47(5):1965–1985, 2018. doi:10.1137/15M1024524.

30 Robert Willoughby Floyd. Algorithm 97: Shortest path. *Commun. ACM*, 5(6):345, June 1962. doi:10.1145/367766.368168.

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

A Detailed observations

In this section we provide several observations that are widely common for APASP algorithms that use hitting sets and pivots (e.g. [6, 25, 2]).

▶ **Observation 10** ([2]). *Let $r \in \mathbb{N}$ such that $|T_i| \geq r$ for any $i \in [\ell]$. A hitting set $S \subseteq U$ of size $\tilde{O}\left(\frac{n}{r}\right)$ can be deterministically computed in $O(nr)$ time.*

Recall the notions mentioned in Section 2.1. While the following observations were proven for a very similar concept to $\Gamma(u, k)$, their proofs work for our case well.

▶ **Observation 11.** *Let $k \in [n]$. The sets $\Gamma(u, k)$ and the exact distances $d[u, x] = \delta(u, x)$ for any $x \in \Gamma(u, k)$ can be computed deterministically in $\tilde{O}(kn)$ time, for all $u \in V$.*

▶ **Observation 12** ([6]). *Let $u \in V$, $S \subseteq V$ and $v \in \Gamma(u, k)$. The edges of $u \rightsquigarrow v$ are contained within E_S.*

▶ **Observation 13** ([25]). *Let $u \in V$. Then: $|E_S(u)| \leq k$. Additionally, $|E_S| \leq nk$.*

It should be observed that finding the pivots themselves and the exact distances from each vertex to its pivot, can be accomplished by a single SSSP invocation with an auxiliary vertex connected to S with edges of weight 0.

▶ **Observation 14.** *Let $k \in [n]$. The pivot $p_S(u)$ of each vertex $u \in V$ and the exact distance $d[u, p_S(u)] = \delta(u, p_S(u))$ can be computed in $\tilde{O}(m)$ time.*

▶ **Observation 15.** *The set E_S can be computed in $\tilde{O}(nk)$ time.*

▶ **Observation 16.** *If $S_2 \subseteq S_1$ then $E_{S_1} \subseteq E_{S_2}$.*

B Proof of Lemma 1

In this section, we provide the missing proofs of Lemma 1 from Section 4. We recall:

▶ **Lemma 1.**

$$\Delta(u \rightsquigarrow v) \leq \begin{cases} 0 & \lambda(u \rightsquigarrow v) \leq \kappa(u) + 1 & :(\mathrm{A}) \\ 2W_{\mu(u \rightsquigarrow v)} & \lambda(u \rightsquigarrow v) > \kappa(u) + 1 \text{ and } \lambda(u \rightsquigarrow v) \geq 3k + 1 - \kappa(u) - \kappa(v) & :(\mathrm{B}) \\ 2W_{\mu(\kappa(u)+1, u \rightsquigarrow v)} + \Delta\left(p_{\zeta(\kappa(u)+1, u \rightsquigarrow v)}(y) \rightsquigarrow y — x \rightsquigarrow u\right) & \lambda(u \rightsquigarrow v) > \kappa(u) + 1 \text{ and } \lambda(u \rightsquigarrow v) < 3k + 1 - \kappa(u) - \kappa(v) & :(\mathrm{C}) \end{cases}$$

Where $x = \varphi(\kappa(u) + 1, u \rightsquigarrow v)$ and $y = \psi(\kappa(u) + 1, u \rightsquigarrow v)$.

Proof. Consider Condition (A), where $\lambda(u\rightsquigarrow v) \leq \kappa(u)+1$. Hence, $S_{\kappa(u)+1} \subseteq S_{\lambda(u\rightsquigarrow v)}$. As $u \in S_{\kappa(u)}$, we invoke SSSP on $G_{\kappa(u)}(u)$ which includes the set of edges $E_{S_{\kappa(u)+1}}$. By Observation 16 we know that $E_{S_{\lambda(u\rightsquigarrow v)}} \subseteq E_{S_{\kappa(u)+1}}$. By the definition of $\lambda(u\rightsquigarrow v)$ we know that $u\rightsquigarrow v \subseteq E_{S_{\kappa(u)+1}}$. We conclude that the exact distance $d[u,v]=\delta(u,v)$ will be computed, resulting in $\Delta(u\rightsquigarrow v)=0$.

If Condition (A) does not hold we know that $\kappa(u) < \lambda(u\rightsquigarrow v)-1$. Recall that the SSSP invocations are performed in a decreasing order. Hence, when considering the SSSP invocation from either u or v, the distance estimations from $S_{\lambda(u\rightsquigarrow v)-1}$ to all the vertices have already been computed in a previous iteration.

We now move on to Condition (B), where additionally: $\lambda(u\rightsquigarrow v) \geq 3k+1-\kappa(u)-\kappa(v)$. Let us consider $i=\kappa(u)$, $j=\kappa(v)$ and $\ell=\lambda(u\rightsquigarrow v)$. Let $w=\varphi(u\rightsquigarrow v)$ and $z=\psi(u\rightsquigarrow v)$. Note that $w\rightsquigarrow p_{\lambda(u\rightsquigarrow v)-1}(w) \subseteq E_{S_{\lambda u\rightsquigarrow v}}$ by definition, hence the concatenated path $p_{\lambda(u\rightsquigarrow v)-1}(w)\rightsquigarrow w — z\rightsquigarrow u, p_{\lambda(u\rightsquigarrow v)}(w)\rightsquigarrow w\rightsquigarrow v \subseteq E_{S_{\lambda u\rightsquigarrow v}}$. Thus, the difference remains $\Delta\big(p_{\lambda(u\rightsquigarrow v)-1}(w)\rightsquigarrow w — z\rightsquigarrow u\big) = \Delta\big(p_{\lambda(u\rightsquigarrow v)-1}(w)\rightsquigarrow w\rightsquigarrow v\big) = 0$. In other words, $d\big[p_{\lambda(u\rightsquigarrow v)-1}(w),u\big] = \delta\big(p_{\lambda(u\rightsquigarrow v)-1}(w),u\big)$ and $d\big[p_{\lambda(u\rightsquigarrow v)-1}(w),v\big] = \delta\big(p_{\lambda(u\rightsquigarrow v)-1}(w),v\big)$. Recall that $\ell \geq 3k+1-i-j$, which means that when we invoke SSSP from u the edges $\{u\}\times S_\ell$ and $S_\ell \times S_j$ will be considered as well. Hence:

$$
\begin{aligned}
d[u,v] &\leq \underbrace{d\big[u,p_{\lambda(u\rightsquigarrow v)-1}(w)\big] + d\big[p_{\lambda(u\rightsquigarrow v)-1}(w),v\big]}_{\text{Due to SSSP from } u} = \underbrace{\delta\big(u,p_{\lambda(u\rightsquigarrow v)-1}(w)\big) + \delta\big(p_{\lambda(u\rightsquigarrow v)-1}(w),v\big)}_{\text{Our preceding discussion}} \\
&\leq \underbrace{\delta(u,w) + 2\cdot\delta\big(w,p_{\lambda(u\rightsquigarrow v)-1}(w)\big) + \delta(w,v)}_{\text{Triangle inequality}} = \underbrace{\delta(u,v) + 2\cdot\delta\big(w,p_{\lambda(u\rightsquigarrow v)-1}(w)\big)}_{u\rightsquigarrow w\rightsquigarrow v \text{ is a shortest path}} \\
&\leq \underbrace{\delta(u,v) + 2\cdot w(z,w)}_{\text{By definition of } \lambda(u\rightsquigarrow v)} = \underbrace{\delta(u,v) + 2W_{\mu(u\rightsquigarrow v)}}_{\text{By definition of } \mu(u\rightsquigarrow v)}
\end{aligned}
$$

It follows that $\Delta(u\rightsquigarrow v) = d[u,v]-\delta(u,v) \leq 2W_{\mu(u\rightsquigarrow v)}$.

Finally, we are left with Condition (C), where we assume that $\lambda(u\rightsquigarrow v) < 3k+1-\kappa(u)-\kappa(v)$. In other words, we cannot use the edges through $p_{\lambda(u\rightsquigarrow v)-1}$, as invoking SSSP with these edges will simply take too much time. Let $x=\varphi(\kappa(u)+1,u\rightsquigarrow v)$ and $y=\psi(\kappa(u)+1,u\rightsquigarrow v)$, as stated in the lemma. Here, we note that if we had already previously computed $d\big[p_{\kappa(u)+1}(y),u\big]$, then an SSSP invocation from u can consider the following path:

1. Utilize the auxiliary edge $\big(u,p_{\kappa(u)+1}(y)\big)$, whose weight is simply $d\big[p_{\kappa(u)+1}(y),u\big]$.
2. Use the auxiliary edge $\big(p_{\kappa(u)+1}(y),y\big)$ whose weight is exactly $d\big[y,p_{\kappa(u)+1}(y)\big] = \delta\big(y,p_{\kappa(u)+1}(y)\big)$.
3. Continue on $y\rightsquigarrow v$, as $y=\psi(\kappa(u)+1,u\rightsquigarrow v)$.

The weight of such a path is upper-bounded $d[u,v] \leq d\big[u,p_{\kappa(u)+1}(y)\big] + \delta\big(p_{\kappa(u)+1}(y),y\big) + \delta(y,v)$. Hence, $\Delta(u\rightsquigarrow v) \leq \Delta\big(p_{\kappa(u)+1}(y)\rightsquigarrow y — x\rightsquigarrow u\big) + 2\cdot w(x,y)$. Recall that $\zeta(\kappa(u)+1,u\rightsquigarrow v) \geq \kappa(u)+1$ is defined to be the largest index for which the same edge (x,y) is at least as heavy as the distance to y's pivot. As a consequence, the above holds as well when instead of selecting $p_{\kappa(u)+1}(y)$, we select the pivot $p_{\zeta(\kappa(u)+1,u\rightsquigarrow v)}(y)$. Thus: $\Delta(u\rightsquigarrow v) \leq \Delta\big(p_{\zeta(\kappa(u)+1,u\rightsquigarrow v)}(y)\rightsquigarrow y — x\rightsquigarrow u\big) + 2\cdot w(x,y)$.

The edge (x,y) is the first edge from v's side on $u\rightsquigarrow v$ that is not in $E_{S_{\kappa(u)+1}}$, which means that its the $\mu(\kappa(u)+1,u\rightsquigarrow v)$ heaviest edge on the path $u\rightsquigarrow v$. In other words: $\Delta(u\rightsquigarrow v) \leq \Delta\big(p_{\zeta(\kappa(u)+1,u\rightsquigarrow v)}(y)\rightsquigarrow y — x\rightsquigarrow u\big) + 2\cdot W_{\mu(\kappa(u)+1,u\rightsquigarrow v)}$. ◀

C Values for specified trade-offs

For completeness, we provide, in this section, values for the (α, β)-APASP examples where $\alpha \in \left(2, \frac{7}{3}\right)$ that were mentioned in Section 1 between Question 4 and Question 5. Consider the following:

- $\alpha = 3$, $\beta = 1$ and $k = 1$ results in $\left(\frac{11}{5}, \frac{W_1+W_2}{2}\right)$-APASP in $\tilde{O}\left(n^{\frac{11}{5}}\right)$ time;
- $\alpha = 2$, $\beta = 1$ and $k = 2$ results in $\left(\frac{13}{6}, \frac{2\cdot(W_1+W_2+W_3)}{2}\right)$-APASP in $\tilde{O}\left(n^{\frac{17}{8}}\right)$ time.