Faster Approximate All Pairs Shortest Paths

Barna Saha* Christopher Ye*

Abstract

The all pairs shortest path problem (APSP) is one of the foundational problems in computer science. For weighted dense graphs on n vertices, no truly sub-cubic algorithms exist to compute APSP exactly even for undirected graphs. This is popularly known as the APSP conjecture and has played a prominent role in developing the field of fine-grained complexity. The seminal results of Seidel and Zwick show that using fast matrix multiplication (FMM) it is possible to compute APSP on unweighted undirected graphs exactly in $\tilde{O}(n^\omega)$ time, and can be approximated within $(1+\epsilon)$ factor in weighted undirected graphs in time $\tilde{O}(n^\omega)$ respectively. Here ω is the exponent of FMM, which currently stands at $\omega = 2.37188$. Moreover even for unweighted undirected graphs, it is not possible to obtain a $(2-\epsilon)$-multiplicative approximation of APSP for any $\epsilon > 0$ in $o(n^\omega)$ time. Since 2000, a result by Dor, Halperin, and Zwick gave the best 2 approximation algorithm for APSP in unweighted undirected graphs in time $\tilde{O}(n^{7/3})$. This result was recently improved by Deng, Kirkpatrick, Rong, Williams and Zhong to $\tilde{O}(n^{2.2593})$ using fast min-plus product for bounded-difference matrices which uses FMM as a subroutine (the stated bound here uses new results for computing such min-plus products by Durr). In fact both these results obtain a +2-additive approximation. Recently, Roditty (STOC, 2023) improved the previous bounds for multiplicative 2-approximation of APSP in unweighted undirected graphs giving the best known bound of $\tilde{O}(n^{2.25})$. All these algorithms are deterministic. Roditty also considers estimating shortest paths for all paths of length $\geq k$ for $k \geq 4$, and gives improved bounds when the underlying graph is sparse using randomization. Though for dense graphs, the best known bounds still remained at those provided by Dor et al. more than two decades back.

In this paper, we provide a multitude of new results for multiplicative and additive approximations of APSP in undirected graphs for both unweighted and weighted cases. We provide new algorithms for multiplicative 2-approximation of unweighted graphs: a deterministic one that runs in $\tilde{O}(n^{2.072})$ time and a randomized one that runs in $\tilde{O}(n^{2.0318})$ on expectation improving upon the best known bound of $\tilde{O}(n^{2.25})$. The algorithm uses FMM as well as new combinatorial insights. For 2-approximating paths of length $\geq k$, $k \geq 4$, we provide the first improvement after Dor et al. for dense graphs even just using combinatorial methods, and then improve it further using FMM. We next consider additive approximations, and provide improved bounds for all additive β-approximations, $\beta \geq 4$. For example, we achieve a running time of $\tilde{O}(n^{2.155})$ for +4 additive approximation improving over the previously known bound of $\tilde{O}(n^{2.2})$, and for a +6 additive approximation, our algorithm has a running time of $\tilde{O}(n^{2.103})$ as opposed to the $\tilde{O}(n^{2.125})$ time that was previously known. For weighted graphs, we show that by allowing small additive errors along with an $(1+\epsilon)$-multiplicative approximation, it is possible to improve upon Zwick's $\tilde{O}(n^\omega)$ algorithm. For example, it is possible to obtain a bi-criteria $(1+\epsilon, 2w_{u,v})$ approximation in $\tilde{O}(n^{2.152})$ time for the shortest path distance between all vertex pairs u, v where $w_{u,v}$ is the highest weight edge on the u-v shortest path. Additionally, we provide a landscape of such bi-criteria approximations for weighted and unweighted graphs. Our results point out the crucial role that FMM can play even on approximating APSP on unweighted undirected graphs, and reveal new bottlenecks towards achieving a quadratic running time to approximate APSP.

*University of California, San Diego. The authors are partially supported by NSF grants 1652303, 1909046, 2112533, and HDR TRIPODS Phase II grant 2217058.

Contents

1 Introduction

Computing All Pairs Shortest Path (APSP) on graphs is a landmark problem in computer science. It is both one of the foundational problems of fine grained complexity, as well as one that directly or indirectly aids in the computation of many important graph and matrix problems. A large variety of graph and matrix problems can be fine-grained reduced to either unweighted or weighted APSP showing that a better algorithm for APSP will lead to a better algorithm for all those problems [WW10]. The classic approaches like Floyd-Warshall computes APSP on weighted dense graphs, $G = (V, E), |V| = n, |E| = m$, in $O(n^3)$ time; whereas the best result known by Williams has a running time of $O\left(\frac{n^3}{2^{\sqrt{\log n}}}\right)$ [Wil14] which is still not sub-cubic. Indeed the weighted APSP conjecture states that there does not exist any truly subcubic algorithm, that is one running in $O(n^{3-\varepsilon})$ time for some constant $\varepsilon > 0$. However, for unweighted undirected graphs, a seminal result of Seidel showed that APSP can be computed in $O(n^\omega)$ time [Sei95] where ω is the exponent of fast matrix multiplication, and currently stands at $\omega = 2.37188$ [DWZ22]. More generally, when weights are bounded integers in the range $[-M, M]$, APSP can be solved in $O(Mn^\omega)$ time for undirected graphs [SZ99, AGM97] and in subcubic time in directed graphs [Zwi02].

In this paper, we concentrate on undirected graphs, and henceforth all references to graphs indicate undirected graphs if not explicitly mentioned otherwise. Interest in computing APSP has naturally led to the study of approximation algorithms. An estimate $\hat{\delta} : V \times V \to \mathbb{R}$ is an (α, β)-approximation of the actual shortest path metric $\delta : V \times V \to \mathbb{R}$ if $\delta(u, v) \leq \hat{\delta}(u, v) \leq \alpha\delta(u, v) + \beta$ for all $u, v \in V \times V$. Therefore an $(\alpha, 0)$-approximation implies a pure multiplicative approximation whereas a $(1, \beta)$-approximation implies a pure additive approximation. There is a huge body of literature on approximating APSP, from multiplicative and additive approximation in sub-cubic time [ACIM99, DHZ00, CZ01, BK10, BK07, DKR+22, Rod23], computing distance oracle that trades off preprocessing with query time, to developing space efficient data structures [MN07, BK10, PR10, WN12, AG13, Che14, Che15, EP16, ENWN16, Som16, AR20, CZ22]. In a seminal work, Zwick gave an $((1 + \epsilon), 0)$-approximation algorithm for weighted APSP that runs in $\tilde{O}(\frac{n^\omega}{\epsilon} \log W)$ time [Zwi98] where W is the largest edge weight. Dependency on W was later removed to obtain a strong polynomial running time of $O\left(\frac{n^\omega}{\epsilon}\text{polylog}\left(\frac{n}{\epsilon}\right)\right)$ [BKW19]. Moreover, even for unweighted graphs, a better than $(2, 0)$-approximation in $o(n^\omega)$ time is not possible [DHZ00]. Naturally this leads to the question whether a $(2, 0)$ approximation is possible in $o(n^\omega)$ time and even better in $O(n^2)$ time. While designing an $\tilde{O}(n^2)$ time algorithm for a $(2, 0)$-approximation still remains open, it is possible to get a $(3, 0)$-approximation in $\tilde{O}(n^2)$ time [DHZ00, CZ01].

Let us first consider unweighted graphs. So far, the best running time to achieve a $(2, 0)$-approximation is due to Roditty [Rod23]. Roditty gave an algorithm with a running time of $\tilde{O}(n^{2.25})$ for a $(2, 0)$ approximation which improves upon the $\tilde{O}(n^{2.2593})$ running time previously known [DKR+22, Dür23]. In fact, both these results are based on Dor et al.'s work [DHZ00]. Dor, Halperin and Zwick gave an algorithm to achieve a $(1, 2)$-approximation that runs in $\tilde{O}(n^{7/3})$ time and a $(1, 4)$-approximation that runs in $\tilde{O}(n^{9/4})$ time among other results. Roditty utilizes the $\tilde{O}(n^{9/4})$ time algorithm and brings in new ideas to show that on paths of length 3, it is possible to get a $+2$-additive approximation. Moreover, paths of length 1 can trivially be found exactly in $O(m)$ time, and paths of length 2 can be approximated within $+2$-additive errors from Dor et al.'s work [DHZ00]. All, these together imply a $(2, 0)$-approximation in $\tilde{O}(n^{9/4})$ time by Roditty [Rod23]. On the other hand, Deng et al. showed the first step of the $\tilde{O}(n^{7/3})$ time algorithm of Dor et al. can be made faster by utilizing fast algorithms for bounded-difference $(\min, +)$ product [CDXZ22, CDX22, WX20, BGSW19], therefore essentially giving a faster $(1, 2)$-approximation algorithm. These lead to several interesting open questions.

Can we use algebraic methods to get a faster $(2,0)$-approximation?

Using bounded-difference $(\min,+)$ product in the first step of $\tilde{O}(n^{9/4})$ does not help, as that running time itself is quite large. Dor et al. provided an entire trade-off between running time and additive error. They showed for every even β, it is possible to approximate APSP within $+\beta$ additive error in time $\tilde{O}(\min(n^{2-\frac{2}{\beta+2}}m^{\frac{2}{\beta+2}}, n^{2+\frac{2}{3\beta-2}}))$. While Deng et al.'s work [DKR+22] improved the running time for a $(1,2)$-approximation, it left open the scope of improving the running time for algorithms that allow higher additive errors. The best known bounds for those still stand at where they were more than two decades back. Employing the bounded-difference $(\min,+)$-product as the first step in Dor et al.'s algorithm for higher additive errors provide no improvements.

Can we use algebraic methods to get faster $(1,\beta)$-approximation for all $\beta > 0$?

As will become apparent, the challenge in computing a fast $(2,0)$-approximation lies in handling paths of short lengths, for which a multiplicative 2-approximation implies a good additive approximation. Another interesting contribution of Roditty's work [Rod23] is to provide an improved running time when a $(2,0)$-approximation is sought only for path lengths greater than a certain threshold. In particular, they show that a $(2,0)$-approximation can be obtained for vertex pairs at distance at least k in time $\tilde{O}\left(\min\left(n^{2-\frac{2}{k+4}}m^{\frac{2}{k+4}}, n^{2+\frac{2}{3k-2}}\right)\right)$. This improves upon the previous bound of $\tilde{O}\left(\min\left(n^{2-\frac{2}{k+2}}m^{\frac{2}{k+2}}, n^{2+\frac{2}{3k-2}}\right)\right)$ by Dor et al. for sparse graphs while leaving the same bounds for dense graphs. Clearly, this raises the question whether it is possible to get improved bounds for dense graphs.

Can we get a faster $(2,0)$-approximation for vertex pairs at distance at least k for dense graphs?

Moving to weighted graphs, as stated before a $((1+\epsilon),0)$-approximation, $\epsilon > 0$, is possible in $O\left(\frac{n^\omega}{\epsilon}\text{polylog}\left(\frac{n}{\epsilon}\right)\right)$ time [Zwi98, BKW19]. Multiple works have studied a natural question whether a bi-criteria (α,β)-approximation, $\alpha > 0, \beta > 0$, can have better time complexity [BK10, BGS05, BK07, Elk05]. Baswana and Kavitha [BK10] and Berman and Kasiviswanathan [BK07] obtained $(2, w_{u,v})$-approximations in $\tilde{O}(n^2)$ time where $w_{u,v}$ is the largest weight in the shortest path between u, v. Berman and Kasiviswanathan also showed an $(1+\epsilon, 2w(u,v))$-approximation with a running time of $\tilde{O}(\frac{n^\omega}{\epsilon^3}\log\frac{n}{\epsilon})$ [BK07]. This later result improved upon a prior work of Elkin where a $(1+\varepsilon, M\beta(\varepsilon,\rho,\zeta))$-approximation is obtained in time $O(mn^\rho + n^{2+\zeta})$ with M being the ratio between the heaviest and lightest edge in the graph [Elk05]. The constant $\beta(\varepsilon,\rho,\zeta)$ depends on ζ as $(1/\zeta)^{\log 1/\zeta}$, inverse exponentially on ρ, and inverse polynomially on ε. These lead to an interesting question, if we fix $\alpha = (1+\epsilon)$, can we show a running time trade-offs with varying β? A similar question applies for all $\alpha < 2$.

Can we get a faster (α,β)-approximation for weighted graphs where $\alpha = (1+\epsilon)$ and $\beta > 0$?

1.1 Our Contributions

In this paper, we provide multitude of results on approximation APSP on undirected graphs answering all of the above questions. We start with our contributions on unweighted graphs.

1.1.1 Multiplicative Approximation on Unweighted Graphs (Section 3)

We significantly improve upon the current best known bound of $\tilde{O}(n^{2.25})$ [Rod23] for a $(2,0)$ approximation of unweighted APSP. Specifically, we obtain the following theorems. Theorem 3.1 is also obtained in a concurrent work by Dory, Forster, Kirkpatrick, Nazari, Vassilevska Williams, and de Vos [DFK+23].

Theorem 3.1. *Let G be an undirected, unweighted graph with n vertices. Algorithm 1 computes a $(2,0)$-approximate APSP solution in expected time $\tilde{O}\left(n^{2.03184039}\right)$.*

Theorem 3.3. *Let G be an undirected, unweighted graph with n vertices. Algorithm 3 deterministically computes a $(2,0)$-approximate APSP solution in time $\tilde{O}(n^{2.07203166})$.*

Our results use FMM with the current best known bounds [DWZ22, GU18] and several new combinatorial insights to bring down the running time very close to $O(n^2)$. We also observe that a $(7/3,0)$-approximation on unweighted graphs can be computed in $\tilde{O}(n^2)$ time (see Appendix D).

1.1.2 Multiplicative Approximations for Long Paths (Section 4)

We improve the bounds for a $(2,0)$-approximation on paths of length at least k, for all $k \geq 4$ on dense graphs even just using combinatorial techniques and then further using algebraic methods. Combining with Roditty's results [Rod23], these imply an improvement for all cases (sparse and dense) over Dor et al.'s result [DHZ00] for paths of length at least 4.

Below, we state the combinatorial results and the further improvements using FMM are stated in Section 5.

$(2,0)$-Multiplicative Approximation for $\delta(u,v) \geq k$			
k	[DHZ00], [Rod23]	Algorithm 4 (Combinatorial)	Algorithm 5 (uses FMM)
4	$n^{11/5} = n^{2.200}$	$n^{15/7} = n^{2.1429}$ (Algorithm 15)	$n^{2.01973523}$
6	$n^{17/8} = n^{2.125}$	$n^{21/10} = n^{2.1000}$	$n^{2.01084688}$
8	$n^{23/11} = n^{2.091}$	$n^{29/14} = n^{2.0715}$	$n^{2.00745825}$
10	$n^{29/14} = n^{2.072}$	$n^{37/18} = n^{2.0556}$	$n^{2.00573823}$
12	$n^{35/17} = n^{2.059}$	$n^{45/22} = n^{2.0455}$	$n^{2.00462679}$

Table 1: Improvements in computing $(2,0)$-approximate APSP for $\delta(u,v) \geq k$ on undirected, unweighted graphs with n vertices and m edges. For $k = 4$, Algorithm 15 is more efficient than Algorithm 4. While only a few examples are shown above, we obtain improvements for all k. See Proposition C.4 for the derivation of some running times for Algorithm 5.

Theorem 1.1 (Stated as Corollary E.11)**.** *Let $k \geq 4$ be an even integer. Then, we can compute a $(2,0)$-approximation for distances $\delta(u,v) \geq k$ combinatorially in expected time*

$$\tilde{O}\left(\min\left(n^{2-\frac{2}{k+4}} m^{\frac{2}{k+4}}, n^{2+\frac{1}{2(k-1)}}, n^{2+\frac{2}{3k+2}}\right)\right)$$

In particular, we output $\hat{\delta}$ such that $\delta(u,v) \leq \hat{\delta}(u,v)$ for all u, v and $\hat{\delta}(u,v) \leq 2\delta(u,v)$ whenever $\delta(u,v) \geq k$.

In contrast, Roditty achieves a bound of

$$\tilde{O}\left(\min\left(n^{2-\frac{2}{k+4}} m^{\frac{2}{k+4}}, n^{2+\frac{2}{3k-2}}\right)\right)$$

(Corollary 2.6 of Roditty [Rod23]).

Table 1 illustrates the comparison in running time between our algorithms and the previous results of Roditty [Rod23] and Dor et al. [DHZ00] for dense graphs.

1.1.3 Additive Approximation (Section 5)

We show it is possible to get better additive approximations $(1, \beta)$ for all $\beta > 0$. Previously such a result was known only for $\beta = 2$ [DKR+22].

Theorem 5.6. *Let $\beta \geq 4$ be an even integer. Let G be an undirected, unweighted graph with n vertices. Algorithm 6 computes $\hat{\delta}$ such that $\delta(u,v) \leq \hat{\delta}(u,v) \leq \delta(u,v) + \beta$ for all $u, v \in V$ in time, $\tilde{O}\left(n^{2+\frac{2x}{\beta+2}}\right)$ where x is the solution to, $\omega\left(1 - \frac{\beta-2}{\beta+2}x, 1-x, 1-\frac{\beta-4}{\beta+2}x\right) = 1 + \frac{4+2\beta}{\beta+2}x$.*[1]

Table 2 shows the improvement in running time for various $(1, \beta)$-approximation errors beyond the work of Dor, Halperin, and Zwick [DHZ00]. The concurrent work [DFK+23] obtains a $(1+\varepsilon, \beta)$-approximation faster than [DHZ00] for $\beta \leq 9$. We obtain $(1, \beta)$-approximation improving upon [DHZ00] for all β and do so without incurring any multiplicative error. Our algorithm is also faster than the $(1+\varepsilon, \beta)$-approximation of [DFK+23] for $\beta \geq 8$. Due to the additional multiplicative error, we instead compare [DFK+23] with Algorithm 9 in Table 4.

$+\beta$-Additive Approximation		
β	[DHZ00]	Algorithm 6
4	$n^{11/5} = n^{2.2}$	$n^{2.15506251}$
6	$n^{17/8} = n^{2.125}$	$n^{2.10300405}$
8	$n^{23/11} = n^{2.0909}$	$n^{2.07733373}$
10	$n^{29/14} = n^{2.0715}$	$n^{2.06196791}$

Table 2: Improvements in computing $+\beta$ approximation on undirected, unweighted graphs with n vertices. While a few examples are shown above, we obtain improvements for all $\beta \geq 4$. All running times are computed with [Bra].

1.1.4 Weighted Graphs & (α, β)-approximations (Section 6 & Section 7)

We show new results that allow for $(1+\epsilon)$-multiplicative approximation of APSP, and some additive errors to go significantly below n^ω for weighted graphs. We also show interesting new trade-offs between $\alpha < 2$, and β for both unweighted and weighted graphs (see Theorem 7.1 and Theorem 7.4).

[1] $\omega(a,b,c)$ is the minimum value such that the product of a $\lceil n^a \rceil \times \lceil n^b \rceil$ matrix by a $\lceil n^b \rceil \times \lceil n^c \rceil$ matrix can be computed in $O(n^{\omega(a,b,c)+\varepsilon})$ arithmetic operations for any constant $\varepsilon > 0$. Note $\omega = \omega(1,1,1)$.

Approximation Algorithms on Weighted Graphs		
Work	Approximation Factor	Time
[Zwi02]	$(1+\varepsilon, 0)$	n^{ω}
[Elk05]	$(1+\varepsilon, M\beta(\zeta,\rho,\varepsilon))$	$mn^{\rho}+n^{2+\zeta}$
[BK07]	$(1+\varepsilon, 2w_{u,v})$	$n^{2.24}$
Algorithm 8	$(1+\varepsilon, 2w_{u,v})$	$n^{2.1519}$
Algorithm 9	$(1+\varepsilon, 2w_{u,v}(\beta))$	$n^{2+x/(\beta+1)}$

Table 3: Comparison with previous results for (α,β)-approximations on weighted graphs. M denotes the ratio between the heaviest and lightest edge in the graph G. $w_{u,v}(\beta)$ denotes the weight of the β heaviest edges on the shortest path between u, v and $w_{u,v} = w_{u,v}(1)$.

Theorem 6.3. *Let $\beta \geq 2$ be an integer and $\varepsilon > 0$. Let G be an undirected, unweighted graph with n vertices. Algorithm 9 computes $\hat{\delta}$ such that $\delta(u,v) \leq \hat{\delta}(u,v) \leq (1+\varepsilon)\delta(u,v) + 2w_{u,v}(\beta)$ in time $\tilde{O}\left(\frac{n^{2+\frac{x}{\beta+1}}}{\varepsilon}\right)$ where x is the solution to, $\omega\left(1-\frac{\beta-1}{\beta+1}x, 1-x, 1-\frac{\beta-2}{\beta+1}x\right) = 2+\frac{x}{\beta+1}$. Here, $w_{u,v}(\beta)$ denotes the total weight of the β heaviest edges of a shortest path P.*

We compare Algorithm 9 with concurrent work [DFK+23]. In addition to handling weighted graphs, our algorithm is faster than both [DHZ00] and [DFK+23] for all $\beta \geq 4$.

$(1+\varepsilon, \beta)$-Additive Approximation			
β	[DHZ00] (Weighted)	[DFK+23] (Unweighted)	Algorithm 9 (Weighted)
2	$n^{7/3} = n^{2.34}$	$n^{2.152}$	$n^{2.152}$
4	$n^{11/5} = n^{2.200}$	$n^{2.119}$	$n^{2.094}$
6	$n^{17/8} = n^{2.125}$	$n^{2.098}$	$n^{2.058}$
8	$n^{23/11} = n^{2.0909}$	$n^{2.084}$	$n^{2.043}$
10	$n^{29/14} = n^{2.0715}$	[DHZ00]	$n^{2.034}$

Table 4: Comprison with [DFK+23] of computing $(1+\varepsilon, \beta)$ approximation on undirected, unweighted graphs with n vertices. While a few examples are shown above, we obtain improvements for all $\beta \geq 4$. [DHZ00] (with adaptions from [CZ01]) and Algorithm 9 additionally handle weighted graphs.

1.2 Other Related Work

Dor, Halperin and Zwick's results to additively approximate APSP [DHZ00] improves upon an earlier work of Aingworth et al. [ACIM99] where a $+2$ additive approximation was obtained in $O(n^{2.5})$ time. Cohen and Zwick [CZ01] observed that the algorithm of Dor et al. [DHZ00] obtain $2w_{u,v}(\beta)$ additive approximations for weighted graphs where $w_{u,v}(\beta)$ denote the weights of the β heaviest edges on the shortest path. Cohen and Zwick [CZ01] obtain a variety of multiplicative approximation algorithms for weighted graphs with stretch factors 2, $7/3$, 3, which are later improved upon by Baswana and Kavitha [BK10]. For directed graphs with real weights in $[0, M]$, Yuster [Yus12] obtained an additive approximation with error εM in time $O(n^{(3+\omega)/2})$. Building upon this work, Chan [Cha21] improved the running time on undirected graphs to $O(n^{(3+\omega^2)/(\omega+1)})$.

There is a long line of work investigating approximate distance oracles, where the goal is to trade-off the pre-processing time with query time along with further considerations such as space complexity. Thorup and Zwick gave a stretch $2k-1$ distance oracle with $O(k)$ query time, $O(kn^{1+1/k})$ space,

and $O(kmn^{1/k})$ pre-processing time [TZ05]. Since then a rich literature of work has followed with improvements in pre-processing and query time, space complexity and bi-criteria approximations [MN07, BK10, WN12, AG13, Che14, Che15, EP16, ENWN16, PR10, Som16, AR20, CZ22].

Comparisons with the concurrent work of [DFK+23] can be found in Theorem 3.1, Table 2, and Table 4.

2 Preliminaries

Let $G = (V, E)$ be an undirected, unweighted graph with vertices V and edges E. Let n denote the number of vertices and m the number of edges. Given a pair of vertices u, v, let $\delta(G, u, v)$ denote the distance in G between u, v i.e. the length of the shortest path connecting u and v. When the underlying graph G is clear, we omit this parameter and write $\delta(u, v)$. A path P can be denoted by a sequence of vertices $(u, u_2, \ldots, v)$ or by its endpoints $P_{u,v}$. Given two paths P, Q that share an endpoint (and no other vertices), let $P \circ Q$ denote the concatenation of the two paths. Let $N(u) = \{v \in V \text{ s.t. } (u, v) \in E\}$ denote the neighborhood of u and $N(u, d) = \{v \in V \text{ s.t. } \delta(u, v) \leq d\}$ denote the depth d neighborhood of u.

Definition 2.1. *Let G be an graph and $\delta(G, u, v)$ denote the length of the shortest path from u to v. A distance estimate $\hat{\delta}(u, v) : V \times V \to \mathbb{R}$ is,*

1. *an α* **multiplicative approximation** *if $\delta(u, v) \leq \hat{\delta}(u, v) \leq \alpha\delta(u, v)$ for all $u, v \in V$. This can also sometimes referred to as an α* **stretch** *approximation.*
2. *a β* **additive approximation** *if $\delta(u, v) \leq \hat{\delta}(u, v) \leq \delta(u, v) + \beta$ for all $u, v \in V$. This can sometimes be denoted as a $+\beta$* **approximation**.
3. *an (α, β)* **approximation** *if $\delta(u, v) \leq \hat{\delta}(u, v) \leq \alpha\delta(u, v) + \beta$ for all $u, v \in V$.*

On an unweighted graph, let $\textbf{BFS}(G, w)$ denote running breadth-first search on graph G from root node w. When the graph G does not need to be specified, this may also be denoted $\textbf{BFS}(w)$. We also make use of a truncated **BFS**, which is an execution of **BFS** with bounded depth. A depth bounded **BFS** will be denoted $\textbf{BFS}(G, w, k)$ for depth k or $\textbf{BFS}(w, k)$ when the graph G is clear.

Definition 2.2. *Let $G = (V, E)$ be an undirected, unweighted graph. A set of vertices D* **dominates** *$U \subset V$ if every $u \in U$ is either in D or has a neighbor in D.*

For a given vertex $u \in U$, define the **representative of u in D,** *denoted $r(u, D)$, be an arbitrary vertex $z \in D \cap N(u)$.*

For a given vertex $z \in D$, define the **constituency of z in U,** *denoted $q(z, D)$, as the set $\{u \in U \text{ s.t. } r(u, D) = z\}$.*

Definition 2.3. *Let $G = (V, E)$ be an undirected, unweighted graph. Let s be a degree threshold. Define $V_s = \{v \in V \text{ s.t. } \deg(v) \geq s\}$. Define $E_s = \{(u, v) \in E \text{ s.t. } \min(\deg(u), \deg(v)) < s\}$.*

Lemma 2.4. *Let U be a universe of n elements. Let $\mathcal{F} = \{S_1, \ldots, S_n\}$ denote a collection of subsets $S_i \subset U$ such that $|S_i| \geq s$ for all i. Then, there is a deterministic algorithm* **HittingSet** *that computes a hitting set X of size $O\left(\frac{n \log n}{s}\right)$ of $\mathcal{F}$ in time $\tilde{O}(ns)$. There is also a randomized algorithm* **rHittingSet** *that with high probability computes a hitting set X of size $O\left(\frac{n \log n}{s}\right)$ of $\mathcal{F}$ in time $O(n)$.*

We note that we can easily verify that a randomly sampled set is indeed a hitting set in $O(ns)$ time, by checking each set S_i for an element in D. For any vertex $v \in V_s$, we may interpret $N(v)$ as

a subset of V of size at least s. In particular, applying the above lemma we can immediately obtain the following.

Lemma 2.5. *Let G be a graph on n vertices. There is a deterministic algorithm* **Dominate** *and randomized algorithm* **rDominate** *that computes a dominating set D of size $O\left(\frac{n \log n}{s}\right)$ of V_s in time $O(m + ns)$ and $O(n)$ respectively.*

Lemma 2.6. *Let G be a graph on n vertices. Given degree thresholds $s_1 > s_2 > \ldots > s_{k-1}$, there is a deterministic algorithm* **Decompose** *and a randomized algorithm* **rDecompose** *that outputs edge sets $\{E_i\}_{i=1}^k$, edge set E^*, and vertex sets $\{D_i\}_{i=1}^k$ satisfying,*

1. *$E_i = \{(u, v) \in E$ s.t. $\min(\deg(u), \deg(v)) < s_{i-1}\}$ is the set $E_{s_{i-1}}$.*
2. *D_i dominates $V_{s_i} = \{v \in V$ s.t. $\deg(v) \geq s_i\}$ and $|D_i| = \tilde{O}\left(\frac{n}{s_i}\right)$. For convenience, V_{s_i} may also be denoted V_i.*
3. *$D_1 \subset D_2 \subset \ldots \subset D_k = V$ and $E_k \subset E_{k-1} \subset \ldots \subset E_1 = E$.*
4. *$E^* = \bigcup_{i=1}^k E_i^*$ where each $E_i^* \subset E$ has for every $v \in V_i$, at least one edge $(v, w) \in E_i^*$ for $w \in D_i$.*

Furthermore, **Decompose** *runs in $\tilde{O}(kn^2)$ time and* **rDecompose** *runs in $\tilde{O}(kn)$ time. Note that* **rDecompose** *satisfies the above conditions with high probability.*

For a given vertex $v \in V$, define the **level of** *v, denoted $\ell(v)$, as the integer i such that $s_i \leq \deg(v) < s_{i-1}$. For a given edge $e \in E$, define the* **level of** *e, denoted $\ell(e)$, as the integer i such that $e \in E_i \setminus E_{i+1}$.*

For sake of completeness, these proofs are given in Appendix B.

We also quote a result from Baswana and Kavitha [BK10] that we use crucially for our randomized algorithms (Section 3.1 and Section 4).

Theorem 2.7 ([BK10])**.** *There exists a randomized algorithm* **BK2APASP** *that returns a $(2, 0)$-approximation of APSP on undirected unweighted graphs with an expected running time complexity $\tilde{O}\left(m\sqrt{n} + n^2\right)$ and space complexity $O(n^2)$.*

3 Improved $(2, 0)$-Approximation

In this section, we provide improved running time bounds for a $(2, 0)$-approximation of APSP on unweighted, undirected graphs. We begin with a simple randomized algorithm followed by a slightly slower deterministic algorithm.

3.1 A Randomized $(2, 0)$-Approximate APSP

We prove the following theorem

Theorem 3.1. *Let G be an undirected, unweighted graph with n vertices. Algorithm 1 computes a $(2, 0)$-approximate APSP solution in expected time $\tilde{O}\left(n^{2.03184039}\right)$.*

High Level Overview We begin by briefly recapping the powerful idea of degree decomposition in computing APSP approximations, first initiated by the work of Aingworth, Chekuri, Indyk, and Motwani [ACIM99]. The idea is roughly as follows. If we consider all vertices with degree at least $n^{1/2}$, there will be a small set D of size $n^{1/2} \log n$ such that every vertex of high degree has a neighbor

in D or is itself in D. D is known as a dominating set. We compute exact distances from D, and then for each pair of vertices, we can take the minimum length path passing through a vertex in D. On paths with at least one high degree vertex, this gives a $+2$ approximation as a vertex in D is adjacent to some vertex in the shortest path P. On the other hand, for each pair of vertices whose shortest path contains no high degree vertices, we can compute APSP in time $O(n^{5/2})$ by computing **BFS** (or **Dijkstra**) from every vertex on the graph $G' \subset G$ containing only edges adjacent to low degree vertices.

Dor, Halperin and Zwick extended this idea to consider three degree thresholds $s_0 = n^{2/3}, s_1 = n^{1/3}, s_2 = 0$ [DHZ00]. These degree thresholds are used to compute the dominating sets $D_0 \subset D_1 \subset D_2 = V$ according to **Decompose** (Lemma 2.6) where D_i dominates $V_i = \{v \in V \text{ s.t. } \deg(v) \geq s_i\}$ and $E_i = \{(u,v) \in E \text{ s.t. } \min(\deg(u), \deg(v)) < s_{i-1}\}$ for $i = 0, 1, 2$ (take $s_{-1} = n+1$). They use **Dijkstra**(instead of **BFS**) from all vertices at the lowest level, i.e., in $G_2 = (V, E_2)$, resulting in a running time of $\tilde{O}\left(n^{2+1/3}\right) = \tilde{O}\left(n^{7/3}\right)$.

For a $(2,0)$-approximation, it is enough to have a $(1,2)$-approximation on paths of length up to 3. Using this crucial observation, Roditty was able to extend the decomposition to 4 levels, and then call a $(1,4)$-approximation algorithm of [DHZ00] for paths of length at least 4. This reduces the running time from $\tilde{O}\left(n^{7/3}\right)$ to $\tilde{O}\left(n^{2+1/4}\right) = \tilde{O}\left(n^{9/4}\right)$.

To improve the bound, we instead proceed as follows. We consider shortest paths of length $\geq C$, and less than C separately where C is some threshold which we later set to $C = 22$. We also consider another threshold $x \in (0,1)$, and define $V_0 = \{v \text{ s.t. } deg(v) \geq n^x\}$. We will set x appropriately later.

High Degree Vertices: Paths of length $\leq C$. Consider first all shortest paths P with at least one vertex from V_0 that is of degree $\geq n^x$. Moreover, concentrate only on paths of length less than $C = 22$. Decompose G into $(1-x)\log n$ levels with degree thresholds $t_j = \frac{n}{2^j}$ for $1 \leq j \leq (1-x)\log n$ and compute dominating sets C_j of size $\tilde{O}(\frac{n}{t_j})$ and edge sets F_j of size $O\left(nt_{j-1}\right)$. Computing **BFS** on the subgraph (V, F_j) from each vertex of C_j therefore altogether requires $\tilde{O}(n^2)$ time. Moreover $|\cup_j C_j| = \tilde{O}\left(n^{1-x}\right)$. For each pair of vertices, compute the estimate $\hat{\delta}(u,v) = \min_{w \in \cup_j C_j} \delta(w,u) + \delta(w,v)$ with a $(\min,+)$ product. Since, we concentrate on paths of length up to a constant C, we can compute the above efficiently with the **BoundedMinPlus** algorithm (Theorem B.8). For the value j such that $P \subset F_j$ but $P \not\subset F_{j+1}$, there is some vertex $w \in P$ such that $t_j \leq \deg(w) < t_{j-1}$ and $w^* \in C_j$ is a neighbor of w. Thus, the computed estimate is a $+2$ approximation, which is also a $(2,0)$-approximation. Computation of the **BoundedMinPlus** requires time $\tilde{O}\left(\omega(1, 1-x, 1)\right)$.

Paths of length $\geq C$. For paths of lengths at least $C = 22$, we are allowed to obtain estimates with large additive errors, which can be computed efficiently using known results from [DHZ00]. We can choose any value of C large enough so that executing the additive approximation of [DHZ00] does not affect the overall running time.

Lemma 3.2. *([DHZ00]) Let $\beta \geq 2$ be even. Let G be an undirected, unweighted graph with n vertices and m edges. There is an algorithm computing a $+\beta$-approximate APSP solution in time*

$$\tilde{O}(\min(n^{2-\frac{2}{\beta+2}} m^{\frac{2}{\beta+2}}, n^{2+\frac{2}{3\beta-2}}))$$

*Denote **SparseAPASP** the algorithm running in time $\tilde{O}(n^{2-\frac{2}{\beta+2}} m^{\frac{2}{\beta+2}})$ and **DenseAPASP** the algorithm running in time $\tilde{O}(n^{2+\frac{2}{3\beta-2}})$.*

Consider paths P such that at least one vertex on P has degree $\geq n^x$. We run **DenseAPASP** with a running time of $\tilde{O}\left(n^{2+2/(3C-2)}\right) = \tilde{O}\left(2.03125\right)$.

Low Degree Vertices. Consider the graph $G_x = (V, E_{n^x})$ where $E_{n^x} = \{(u,v)\ s.t.\ deg(u) < n^x$ or $deg(v) < n^x\}$. Then $|E_{n^x}| = O(n^{1+x})$. We simply run an algorithm by Baswana and Kavitha that satisfies Theorem 2.7 [BK10]. This computes a $(2,0)$-approximation of paths contained in G_x with an expected time of $\tilde{O}\left(n^{1.5+x} + n^2\right)$.

Time Complexity Balancing $n^{1.5+x}$ with $\omega(1, 1-x, 1)$, we get $x = 0.53184039$, and an overall running time of $\tilde{O}\left(n^{2.03184}\right)$. Note that the running time bound holds on expectation.

Algorithm Algorithm 1 begins by initializing the distance matrix with adjacency matrix and setting parameters x, C which will be chosen to optimize the running time.

Phase 1 (Lines 4-6), uses **BoundedMinPlus** on matrix M_t to estimate distances for paths of length at most C containing at least one vertex of degree $\geq n^x$. To do so, Line 4 considers $\log n$ degree thresholds, each of the form $t = \frac{n}{2^j}$. Within each threshold, a dominating set C_t is computed (Line 5) and a edge set $E_{2n/t}$ such that the degree of each vertex does not exceed $\frac{2n}{t}$. Line 12 of **DominatingSetAPASP** computes a **BFS** within the sub-graph $G_t = (V, E_{2n/t})$ from each node in the dominating set C_t, keeping only distances up to $C+1$. Then, for each pair of vertices, u, v, a bounded $(\min, +)$ product is computed in Line 15 of **DominatingSetAPASP** to compute the shortest path between u, v passing through a vertex $w \in C_t$. We repeat this computation for all t, balancing the thresholds so that each **BFS** search can be computed efficiently. We repeat for all t large enough such that the dominating set C_t is small.

Phase 2 uses **BK2APASP** to compute $(2,0)$ approximations for paths with maximum degree n^x.

Phase 3 uses **DenseAPASP** to compute a $+C$ approximation, which implies a $(2,0)$ approximation for all paths of length at least C.

Next, we give the pseudocode of **Random2ApproxAPSP** and its correctness proof along with a time complexity analysis.

Correctness

Proof. First, $\delta(u,v) \leq \hat{\delta}(u,v)$ for all u, v. For each call to **DominatingSetAPASP**, this follows from Lemma C.1 as $\hat{\delta}(u,v) \geq \delta(G_t, u, v) \geq \delta(u,v)$ as $G_t \subset G$. The claim then follows from the correctness of **BK2APASP** and **DenseAPASP**.

We now show that $\hat{\delta}(u,v) \leq 2\delta(u,v)$ for all u, v. Let P be a shortest path and $\deg(P) = \max_{v \in V} \deg(v)$ be the maximum degree of a vertex in path P.

Algorithm 1 RANDOM2APPROXAPSP(G)

Input : Unweighted, undirected Graph $G = (V, E)$ with n vertices
Output: Distance estimate $\hat{\delta} : U \times V \to \mathbb{Z}$ such that $\delta(u, v) \leq \hat{\delta}(u, v) \leq 2\delta(u, v)$ for all $u, v \in V$

Fix parameters $x \leftarrow 0.53184039$ and $C \leftarrow 22$

$\hat{\delta}(u, v) \leftarrow \begin{cases} 1 & (u, v) \in E \\ \infty & \text{o/w} \end{cases}$

Phase 1: Compute +2 Approximate Distances of High Max Degree Paths

for $t = 2^j$ *for* $0 \leq j \leq \lceil (1 - x) \log n \rceil$ **do**

  $C_t \leftarrow$ DOMINATE $\left(G, \frac{n}{t}\right)$ and $G_t = \left(V, E_{\frac{2n}{t}}\right)$

  $\hat{\delta} \leftarrow \min(\hat{\delta},$ DOMINATINGSETAPASP$(G_t, V, V, C_t, C + 1))$

Phase 2: Compute Approximate Distances of Low Degree Paths

$\hat{\delta} \leftarrow \min(\hat{\delta},$ BK2APASP$(G_x))$ where $G_x = (V, E_{n^x})$

Phase 3: Compute Approximate Distances of Long Paths

$\hat{\delta} \leftarrow \min(\hat{\delta},$ DENSEAPASP$(G, C))$

Algorithm 2 DOMINATINGSETAPASP(G, V_1, V_2, W, C)

Input : Unweighted, undirected Graph $G = (V, E)$, source and target subsets $V_1, V_2 \subset V$ and dominating subset $W \subset V$, and distance bound C
Output: Distance estimate $\hat{\delta} : U \times V \to \mathbb{Z}$ such that $\delta(u, v) \leq \hat{\delta}(u, v) \leq \min_{w \in W} \delta(u, w) + \delta(w, v)$ for all u, v

for $w \in W$ **do**

  $\hat{\delta}(w) \leftarrow$ BFS(G, w)

Construct $V_1 \times W$ matrix A where $A(v, w) = \begin{cases} \hat{\delta}(w, v) & \hat{\delta}(w, v) \leq C \\ \infty & \text{o/w} \end{cases}$

Construct $W \times V_2$ matrix B where $B(w, v) = \begin{cases} \hat{\delta}(w, v) & \hat{\delta}(w, v) \leq C \\ \infty & \text{o/w} \end{cases}$

$\hat{\delta} \leftarrow$ BOUNDEDMINPLUS(A, B, C)

Case 1: $\delta(u, v) \geq C$ In this case, DENSEAPASP computes a $+C$ approximation so that,

$$\hat{\delta}(u, v) \leq \delta(u, v) + C \leq 2\delta(u, v)$$

Case 2: $\deg(P) < n^x$ In this case, $P \subset G_x = (V, E_{n^x})$ so that BK2APASP computes,

$$\hat{\delta}(u, v) \leq 2\delta(u, v)$$

Case 3: $\deg(P) \geq n^x$ **and** $\delta(u, v) \leq C$ We handle this case with Phase 1 and obtain a +2 approximation. Note that if $\delta(u, v) = 1$, then we already computed an exact distance when initializing $\hat{\delta}$.

Let w be the vertex of maximum degree in P. Let j be the integer such that $\frac{n}{2^j} \leq \deg(w) < \frac{n}{2^{j-1}}$, noting that $j \leq \lceil (1 - x) \log n \rceil$ since $\deg(w) > n^x$. Consider the iteration where $t = 2^j$ of DOMINATINGSETAPASP (Algorithm 2). Then, $P \subset G_t = (V, E_{\frac{2n}{t}})$. Then, since $\deg(w) \geq \frac{n}{t}$ we

have that there is some neighbor of w, say $w^* = r(w, C_t) \in C_t$. Since $\delta(u, v) \leq C$, by the triangle inequality,

$$\delta(G_t, u, w^*) \leq \delta(u, w) + 1 \leq C + 1$$

and similarly $\delta(G_t, w^*, v) \leq C + 1$ so that both entries are finite in matrices A, B.

Then, computing the $(\min, +)$ product,

$$\hat{\delta}_t(u, v) \leq \delta(G_t, u, w^*) + \delta(G_t, w^*, v) \leq \delta(u, w) + \delta(w, v) + 2 = \delta(u, v) + 2$$

Then, we immediately obtain,

$$\hat{\delta}(u, v) \leq \delta(u, v) + 2 \leq 2\delta(u, v)$$

□

Time Complexity

Proof. We analyze the time complexity of our algorithm. x, C are parameters that will be optimized.

Phase 1 Fix some $t = 2^j$. We compute set C_t of size $|C_t| = \tilde{O}(t)$ in time $O(m) = O(n^2)$ by Lemma 2.5. The graph G_t has at most $\frac{2n^2}{t}$ edges, so execution of all $|C_t|$ **BFS** searches requires $\tilde{O}(n^2)$-time. We can also upper bound t as $t = O(n^{1-x})$. Since $C = O(1)$, computing the **BoundedMinPlus**$(M_t, M_t^T, C + 1)$ requires time $Cn^{\omega(1,1-x,1)} \geq n^2$. Then, since there are at most $\log n$ such j, Phase 1 requires overall time,

$$\tilde{O}\left(Cn^{\omega(1,1-x,1)}\right)$$

Phase 2 In Phase 2, we call **BK2APASP** on the graph G_x which requires $\tilde{O}(m\sqrt{n}) = \tilde{O}(n^{1.5+x})$ expected time.

Phase 3 In Phase 3, we call **DenseAPASP** which requires time $\tilde{O}(n^{2+2/(3C-2)})$. We can equivalently use **AdditiveAPASP** from Section 5 but this will not affect the overall running time of our algorithm.

Then, balancing $\omega(1, 1 - x, 1) = 1.5 + x$, we choose $x = 0.53184039$ [Bra], this leads to a time complexity of $\tilde{O}(n^{2.03184039})$.

Then, we simply choose C such that $2 + \frac{2}{3C-2} \leq 2.03184039$, noting that $C = 22$ suffices. □

Note. Note that if we replace the **BoundedMinPlus** computation by vanilla $(\min, +)$-product, and set $x = \frac{3}{4}$, then we get a combinatorial (randomized) algorithm matching $\tilde{O}(n^{2.25})$ time bound obtained by Roditty [Rod23]. However, Roditty's algorithm is deterministic. In the following Section 3.2, we give a deterministic algorithm that significantly improves upon Roditty's bound.

3.2 A Deterministic $(2, 0)$-Approximate APSP

We now move to our deterministic algorithm which gives a slightly weaker running time bound, but brings in many new ideas that will be useful in later sections. We prove the following theorem.

Theorem 3.3. *Let G be an undirected, unweighted graph with n vertices. Algorithm 3 deterministically computes a $(2, 0)$-approximate APSP solution in time $\tilde{O}(n^{2.07203166})$.*

We extend Roditty's deterministic algorithm that uses 4 levels significantly to consider $k+1=8$ levels. We handle paths of length longer than twelve using a $(1,12)$-approximation algorithm of [DHZ00] which requires time $n^{2+1/17}$ and does not dominate the total running time. We consider the following degree thresholds $s_0 = n^x, s_1 = n^{\frac{6}{7}x}, s_2 = n^{\frac{5}{7}x}, \ldots, s_6 = n^{\frac{1}{7}x}$ and $s_7 = 0$. We will leave x as a parameter to be optimized later. Given these degree thresholds, we compute dominating sets $D_0 \subset D_1 \subset \ldots \subset D_6 \subset D_7 = V$ according to **Decompose** (Lemma 2.6) where D_i dominates $V_i = \{v \in V \text{ s.t. } \deg(v) \geq s_i\}$ and $E_i = \{(u,v) \in E \text{ s.t. } \min(\deg(u), \deg(v)) < s_{i-1}\}$ for $i = 0, 1, .., 7$ (take $s_{-1} = n+1$).

High Degree Vertices: Paths of length $\leq C$. We handle high degree vertices in a similar way as we did for our randomized algorithm, that is we compute a dominating set of size $\tilde{O}\left(n^{1-x}\right)$ in $\tilde{O}\left(n^2\right)$ time, and compute **BoundedMinPlus** to obtain a $+2$-additive approximation of all paths of length at most C going through some vertex in V_0. This step requires $\tilde{O}\left(n^{\omega(1,1-x,1)}\right)$ time.

Paths of length $\geq C$. For paths of lengths at least $C = 12$, we will use known results from [DHZ00]. Consider paths P such that all vertices on P have degree less than n^x. Here, we have decomposed the remaining graph into $k = 7$ levels. **SparseAPASP** with $k = 7$ levels achieves a $+12$ approximation. So, we can simply run it. The algorithm **SparseAPASP** executes **Dijkstra**$(G_{i,w}, w, \hat{\delta})$ from all $w \in D_i$ where $G_{i,w} = (V, E_i \cup E^* \cup (w \times V))$ is equipped with the current estimates $\hat{\delta}$ as edge weights, which requires $\tilde{O}(n^{2+\frac{x}{7}})$ time as D_i has size $\tilde{O}(n^{1-\frac{7-i}{7}x})$ and the graph $G_{i,w}$ has $O\left(n^{1+\frac{7-(i-1)}{7}x}\right)$ edges. If path P has a vertex in V_0, then we can run **DenseAPASP** with a running time of $\tilde{O}\left(n^{2+1/17}\right) = \tilde{O}\left(n^{2.0588}\right)$. Again, since we only need **DenseAPASP** to compute good approximations on paths with high degree vertices, we can choose any constant value of C large enough such that executing **DenseAPASP** does not dominate the running time.

Low Degree Vertices: Paths of length $\leq C$.

Paths of length 6 or more For paths of length at least 6, we compute a $+6$ approximation following **BoundedAdditiveAPASP** in Section 5.3. We give a brief overview here with special attention to the specific application required with more details provided in Section 5.3. We use fast rectangular matrix multiplication judiciously to improve upon [DHZ00].

We compute **BoundedAdditiveAPASP** on the original graph G and therefore choose new degree thresholds and use a new degree decomposition of the graph G. We first describe our algorithm combinatorially, and then mention where fast matrix multiplication can be applied. For a full discussion, see Section 5.3. To obtain a $+6$ approximation, choose 7 degree thresholds $r_1, \ldots, r_7$ and decompose the vertex and edge sets of G as $C_1 \subset \ldots \subset C_8 = V$ and $F_8 \subset F_7 \subset \ldots \subset F_1 = E$. At level i, compute **Dijkstra** from each $w \in C_i$ on the graph

$$G_{i,w} = \left(V, F_i \cup \left(\bigcup_{i+j_1+j_2 \leq 17} C_{j_1} \times C_{j_2}\right) \cup F^* \cup (w \times V)\right)$$

noting that both $F_i, C_{j_1} \times C_{j_2}$ have size $O\left(n^{1+\frac{8-(i-1)}{8}}\right)$. Here $F^* = \bigcup_{i=1}^{7} F_i^*$ is output by **Decompose**.

Consider a shortest path P. We use the following definition of blocking vertices and blocking levels.

Definition 3.4. *Let u, v be vertices in a graph G and P a path between u, v. Let $(D_1, \ldots, D_k)$, $(E_1, \ldots, E_k)$ be the outputs of a call to* DECOMPOSE *with degree thresholds $s_1, \ldots, s_{k-1}$. If there is an edge $(a, b) \in P$ such that $(a, b) \notin E_{\ell(u)}$, the* **blocking vertex** *of P from u, denoted $b(u, P)$, is the closest vertex to v that is an endpoint to such an edge. If no such edge exists, then $b(u, P) = v$.*

Then, the **blocking vertices of** *P is the set $B(P) = \{x_0, x_1, \ldots, x_t\}$ defined in the following manner:*

1. $x_0 = u$
2. $x_1 = v$
3. $x_i = b\left(x_{i-1}, P_{x_{i-1}, x_{i-2}}\right)$ *where $P_{x_{i-1}, x_{i-2}}$ is the sub-path of P between x_{i-1}, x_{i-2}.*

For any $j \geq \ell(v)$, **blocking levels of** *P* **at level** *j is the set $L_B(P, j) = \{\ell(x_i)$ s.t. $x_i \in B(P)$ and $\ell(x_i) < j\}$. Denote $L_B(P) = L_B(P, \ell(v))$.*

We prove that the number of blocking levels directly determines the additive error accumulated on a given path.

Lemma 3.5. *Let u, v be vertices in a graph G and P be a shortest path between u, v. Let $B(P)$ be the blocking vertices of P. Then, the distance estimate $\hat{\delta}$ satisfies*

$$\hat{\delta}(v^*, u) \leq \delta(v, u) + 2|L_B(P)| + 1$$

For any level $j \geq \ell(v)$, if $v \in D_j$,

$$\hat{\delta}(v, u) \leq \hat{\delta}_j(v, u) \leq \delta(v, u) + 2|L_B(P, j)|$$

The proof is deferred to Appendix A.1.

If $P \subset F_5$ then we obtain a $+6$ additive approximation as a Corollary to Lemma 3.5 as $|L_B(P)| \leq 3$. Thus, suppose $P \subset F_j$ but $P \not\subset F_{j+1}$ for some $j \leq 4$.

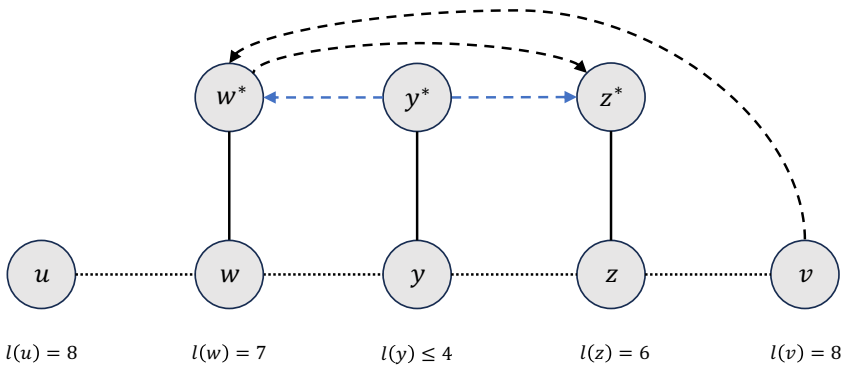


Figure 1: Solid lines denote edges and dotted lines represent paths. Blue dashed arrows denote distance estimates used in the $(\min, +)$ product. Black dashed arrows denote edges $w \times V$ used in $G_{j,w}$.

For any vertex $w \in V_7$ and $w^* = r(w, C_7)$, the graph G_{7,w^*} contains edge sets $F_7 \cup (w^* \times V) \cup \left(\bigcup_{j=1}^{4} C_j \times C_6\right)$. For $v \in V$, let z denote the first vertex on $P_{w,v}$ such that the sub-path $P_{z,v} \subset F_7$ and $y \in P_{w,v}$ be the vertex of maximum degree. See Figure 1. Then $\ell(z) \leq 6$, and $\ell(y) \leq 6$. Suppose $\ell(z) = 6$ (other cases are only easier). Let $y^* = r(y, C_j)$ where $j \leq 6$ and $z^* = r(z, C_6)$.

Then, since $P \subset F_j$, $\hat{\delta}(y^*, z^*) \leq \delta(y, z) + 2$ and $\hat{\delta}(y^*, w^*) \leq \delta(y, w) + 2$. Executing **Dijkstra** from w^*, the path $(w^*, y^*, z^*, z) \circ P_{z,v} \subset G_{7,w^*}$ so that the distance estimate,

$$\hat{\delta}(w^*, v) \leq \hat{\delta}(w^*, y^*) + \hat{\delta}(y^*, z^*) + 1 + \delta(z, v) \leq \delta(w, v) + 5$$

Then, returning to the shortest path P with endpoints u, v, the graph $G_{8,u}$ contains $F_8 \cup (u \times v)$. If w denotes the closest vertex to v on P such that $P_{w,u} \subset F_8$ and $w^* = r(w, C_7)$, then the path $(v, w^*, w) \circ P_{w,u} \subset G_{8,v}$ so that the distance estimate computed is at most,

$$\hat{\delta}(v, u) \leq \hat{\delta}(v, w^*) + 1 + \delta(w, u) \leq \delta(v, u) + 6$$

Note that we have not actually required the use of every $C_{j_1} \times C_{j_2}$ such that $i + j_1 + j_2 \leq 17$ when executing **Dijkstra** from C_j. Instead, we have only used the sets $C_i \times C_6$ for $i \leq 4$ when executing **Dijkstra** from C_7. Instead of adding these edges, we can compute the distances that would have been obtained with these edges with a $(\min, +)$ product and encode these distances into the smaller edge set $w \times V$. We encode the distances $C_i \times C_7$ into a matrix A_i and the distances $C_i \times C_6$ into a matrix B_i and compute $A_i * B_i$ with entries bounded by 14 for $i = 1, 2, 3, 4$. In the example above, this gives the estimate $\hat{\delta}(w^*, z^*) \leq \delta(w, z) + 4$. Then, when executing **Dijkstra** from $w^* \in C_7$, the path $(w^*, z^*, z) \circ P_{z,v} \subset G_{7,w^*}$ so the distance estimate is again at most,

$$\hat{\delta}(w^*, v) \leq \hat{\delta}(w^*, z^*) + 1 + \delta(z, v) \leq \delta(w, v) + 5$$

Paths of length 5 or less We now describe the most challenging cases: obtaining a $(2, 0)$ approximation on shortest paths $P_{u,v}$ of length at most 5. Let $G_x = (V, E_{n^x})$ denote the graph G pruned so that the maximum degree of G_x is strictly less than n^x. We can compute a depth 2 **BFS**(that is **BFS**tree upto two levels) in time n^{2x} from each vertex, with a total time of n^{1+2x}. Our choice of x will ensure this is allowed.

As a warm-up, consider obtaining a $+2$ approximation on paths of length at most 3. For 2 edge paths, a depth 2 **BFS** computes exact distances for all paths of length 2 in G_x. For 3 edge paths $P = (u, u_2, v_2, v)$, assume $\deg(u) \geq \deg(v)$. A depth 2 **BFS** from u implies $\hat{\delta}(u, v_2) = 2$ and if $u^* = r(u, D_{\ell(u)})$ then we can efficiently update $\hat{\delta}(u^*, v_2) \leq 1 + \hat{\delta}(u, v_2) = 3$. Then, in the graph $G_{\ell(u),u^*}$, there is the path (u^*, v_2, v) so **Dijkstra** computes a distance estimate $\hat{\delta}(u^*, v) \leq 4$. Finally, in the graph $G_{k,v}$ (that is at the lowest level) there is the path (v, u^*, u) so **Dijkstra** computes a distance estimate $\hat{\delta}(u, v) \leq 5$, which is a $+2$ approximation.

Finally, we consider paths $P = (u, u_2, u_3, v_3, v_2, v)$ of 5 edges. Paths of length 4 are easier to handle and we will mention the necessary modifications in the full proof. Recall that we have decomposed the graph G_x into $k = 7$ levels with degree thresholds $s_1, s_2, ..., s_6, s_7 = 0$.

Consider the case where both endpoints have low degree $\deg(u), \deg(v) < s_6$, or identically $\ell(u), \ell(v) = 7$. Let w.l.o.g, $\deg(u_2) \geq \deg(v_2)$. Let $u_2^* = r(u_2, D_{\ell(u_2)})$. If $P \subset E_{\ell(u_2)}$, then $\hat{\delta}(u_2^*, v) \leq \delta(u_2, v) + 1$. Otherwise, both $\ell(u_3), \ell(v_3) < \ell(u_2)$ (since even if one of them is at $\ell(u_2)$ or higher, we have $P \subset E_{\ell(u_2)}$). Therefore, $\hat{\delta}(v_3^*, u_2^*) \leq \delta(v_3, u_2) + 2$ so the path $(u_2^*, v_3^*, v_3) \circ P_{v_3,v} \subset G_{\ell(u_2),u_2^*}$ and **Dijkstra** computes the desired distance estimate: $\hat{\delta}(u_2^*, v) \leq \delta(u_2, v) + 3$. Then, as $(v, u_2^*, u_2, u) \subset G_{7,v}$, **Dijkstra** computes a $+4$ additive approximation. Figure 2 gives an illustration.

Then, assume at least one endpoint (say u) has degree $\geq s_{k-1}$. Suppose there is some vertex (say v_3) with $n^x > \deg(v_3) \geq s_1$. Using a bounded rectangular $(\min, +)$ product for matrices of size $D_1 \times V$ and $D_1 \times D_6$, the $(\min, +)$ product yields an estimate $\hat{\delta}(u^*, v) \leq \delta(u*, v) + 2 \leq \delta(u, v) + 3$. Then, as $(v, u^*, u) \subset G_{7,v}$, **Dijkstra** computes a $+4$ additive approximation. Figure 3 gives an illustration.

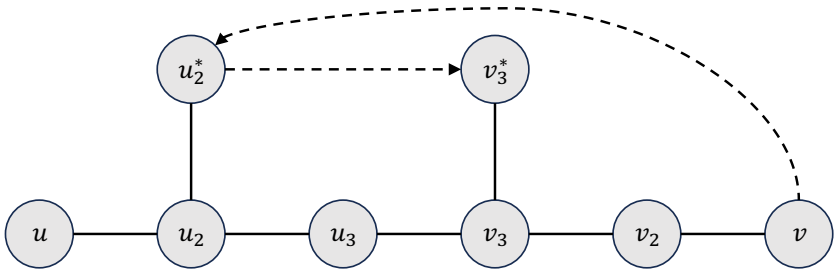


Figure 2: Case 2.3.1: $\ell(u) = \ell(v) = k$. Solid lines denote edges and dotted lines represent paths. Black dashed arrows denote edges $w \times V$ in $G_{j,w}$.

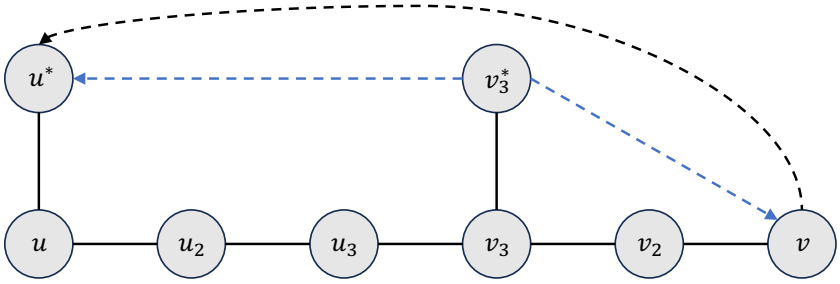


Figure 3: Case 2.3.2: $\deg(u) \geq s_6$ and $\deg(v_3) > s_1$. Solid lines denote edges in G. Blue dashed arrows denote distance estimates used in the $(\min, +)$ product. Black dashed arrows denote edges $w \times V$ in $G_{j,w}$.

We can now assume that every vertex has degree $< s_1$. We now compute depth 3 **BFS** in $G_2 = (V, E_2)$ from every vertex with degree at most s_2. One depth 3 **BFS** requires $s_2 s_1^2 = n^{\frac{3k-4}{k}x}$ time (contrast to n^{3x} if naively applying depth 3 **BFS** on the graph G_x). The correctness depends on extensive case analysis.

In Case 2.3.3, suppose $\deg(u_2), \deg(v_2) \geq s_2$. Suppose $\deg(v) \geq \deg(u)$. From our earlier discussion, since $P \subset E_2$, $\hat{\delta}(u_2^*, v) \leq \delta(u, v) + 1$. Then, since the path $(v^*, u_2^*, u_2, u) \subset G_{\ell(v),v^*}$, **Dijkstra** computes an estimate,

$$\hat{\delta}(v^*, u) \leq \hat{\delta}(v^*, u_2^*) + 2 \leq \delta(u, v) + 3$$

Then, as $(u, v^*, v) \subset G_{8,u}$, **Dijkstra** computes a $+4$ additive approximation.

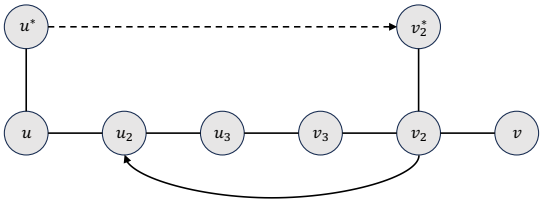
In Case 2.3.4, suppose $\deg(u_2) \geq s_2$ but $\deg(v_2) < s_2$. If $\deg(v) \geq \deg(u)$, then we can proceed exactly as above, so assume $\deg(u) > \deg(v)$. The depth 3 **BFS** from v_2 computes $\hat{\delta}(v_2, u_2) = 3$ and Line 36 updates $\hat{\delta}(v_2^*, u_2) = 1 + \hat{\delta}(v_2, u_2) = 4$. If $\deg(v_2) \geq \deg(u)$, then, the path $(v_2^*, u_2, u, u^*) \subset G_{\ell(v_2),v_2^*}$, **Dijkstra** computes a distance estimate at most,

$$\hat{\delta}(v_2^*, u^*) \leq \hat{\delta}(v_2^*, u_2) + 2 \leq 6$$

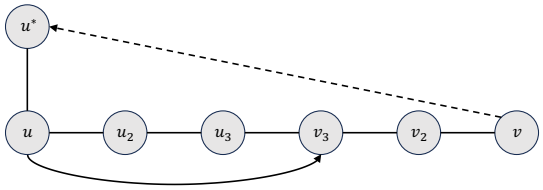
Since $\deg(u) > \deg(v)$, $(u^*, v_2^*, v_2, v) \subset G_{\ell(u),u^*}$, **Dijkstra** computes a distance estimate at most,

$$\hat{\delta}(u^*, v) \leq \hat{\delta}(u^*, v_2^*) + 2 \leq 8$$

Then, as $(v, u^*, u) \subset G_{7,u}$, **Dijkstra** computes a distance estimate at most 9, a $+4$ additive approximation. See Figure 4a for an illustration.



(a) Depth 3 **BFS** search from v_2.



(b) Depth 3 **BFS** search from u.

Figure 4: Case 2.3.4 and 2.3.5: We use depth 3 **BFS** in Line 31 to obtain more accurate distance estimates. Solid lines denote edges in G. Solid arrows denote depth 3 **BFS** executions. Dotted arrows denote edges $w \times V$ used in executing **Dijkstra** on the graph $G_{j,w}$.

On the other hand $\deg(u) > \deg(v_2)$. If $\deg(u) > s_2$, we obtain a $+2$ approximation following the previous discussion of Case 2.3.3. Otherwise, if $\deg(u) < s_2$ then the depth 3 **BFS** from u computes $\hat{\delta}(u, v_3) = 3$ and Line 36 updates $\hat{\delta}(u^*, v_3) = 1 + \hat{\delta}(u, v_3) = 4$. Since $\deg(u) > \deg(v_2)$, $(u^*, v_3, v_2, v) \subset G_{\ell(u),u^*}$ so **Dijkstra** computes a distance estimate at most,

$$\hat{\delta}(u^*, v) \leq \hat{\delta}(u^*, v_3) + 3 \leq 6$$

Then, as $(v, u^*, u) \subset G_{7,u}$, **Dijkstra** computes a distance estimate at most 7, which is a $+2$ additive approximation. See Figure 4b for an illustration.

In the full proof we give a detailed argument on all possible variations of length 5 paths as well as the modifications necessary to compute a $+4$ additive approximation on length 4 paths.

Time Complexity The time-complexity to compute **BoundedMinPlus** to handle paths with at least one vertex in V_0 is $\tilde{O}(\omega(1, 1-x, 1))$. This computation dominates the **BoundedMinPlus** computation in Case 2.3.2 Figure 3 (see Proposition C.3). The time complexity to handle paths with low degree vertices for paths of length at most 5 is

$$\tilde{O}\left(n^{1+2x} + n^{1+\frac{3k-4}{k}x} + n^{2+\frac{x}{k}}\right)$$

where the first term is from running depth 2 **BFS**, the second term is from running depth 3 **BFS** from vertices with degree at most s_2, and the last term is from running **Dijkstra**. If we balance these terms, we get we choose $x = 0.4414248$ and $k = 7$ to obtain the time complexity $\tilde{O}(n^{2.07203166})$. The non-dominating terms include time to run **BoundedAdditiveAPASP**on paths of length $[6, 12]$ which requires $\tilde{O}(n^{2.0638})$ by Corollary 5.14 and **DenseAPASP**(G,12) that requires time $\tilde{O}(n^{35/17}) = \tilde{O}(n^{2.0589})$ by Lemma 3.2.

Algorithm Algorithm 3 computes a deterministic $(2, 0)$-approximation of APSP on unweighted graphs. Phase 1 (Lines 19-21), uses **BoundedMinPlus**on matrix M_t to estimate distances for paths of length at most C containing at least one vertex of degree $\geq n^x$. To do so, Line 19 considers $\log n$ degree thresholds, each of the form $t = \frac{n}{2^j}$. Within each threshold, a dominating set C_t is computed (Line 20) and a edge set $E_{2n/t}$ such that the degree of each vertex does not exceed $\frac{2n}{t}$. Line 12 of **DominatingSetAPASP** computes a **BFS** within the sub-graph $G_t = (V, E_{2n/t})$ from each node in the dominating set C_t, keeping only distances up to $C + 1$. Then, for each pair of vertices, u, v, a bounded $(\min, +)$ product is computed in Line 15 of **DominatingSetAPASP** to compute the shortest path between u, v passing through a vertex $w \in C_t$. We repeat this computation for all t,

balancing the thresholds so that each **BFS** search can be computed efficiently. We repeat for all t large enough such that the dominating set C_t is small.

Phase 2 handles paths with low degree vertices. Line 24 uses **Decompose** on the remaining sparse graph into $k = 7$ degree levels specified in Line 23.

Phase 2.1, computes **BFS** from each vertex in D_1 (Lines 26-27) and use a bounded $(\min, +)$ product to find approximate distances between each pair of vertices $V \times D_{k-1}$ that goes through a vertex in D_1 in Line 15 of **DominatingSetAPASP**.

In Phase 2.2, a depth 2 **BFS** is computed from each vertex in G_x (Line 30), as well as depth 3 **BFS** from each vertex of degree at most s_2 in the graph $G_2 = (V, E_2)$ (Line 31).

Algorithm 3 **2ApproxAPSP**(G)

Input : Unweighted, undirected Graph $G = (V, E)$ with n vertices
Output: Distance estimate $\hat{\delta} : U \times V \to \mathbb{Z}$ such that $\delta(u, v) \leq \hat{\delta}(u, v) \leq 2\delta(u, v)$ for all $u, v \in V$

16 Fix parameters $x \leftarrow 0.45509125$ and $C \leftarrow 12$

17 $\hat{\delta}(u, v) \leftarrow \begin{cases} 1 & (u, v) \in E \\ \infty & \text{o/w} \end{cases}$

18 Phase 1: Compute +2 Approximate Distances of High Max Degree Paths

19 **for** $t = 2^j$ *for* $0 \leq j \leq \lceil (1 - x) \log n \rceil$ **do**

20 $\quad C_t \leftarrow$ **Dominate** $\left(G, \frac{n}{t}\right)$ and $G_t = \left(V, E_{\frac{2n}{t}}\right)$

21 $\quad \hat{\delta} \leftarrow \min(\hat{\delta}, \textbf{DominatingSetAPASP}(G_t, V, V, C_t, C + 1))$

22 Phase 2: Compute Approximate Distances of Low Degree Paths

23 $s_i \leftarrow n^{\frac{k-i}{k}x}$ for all $1 \leq i \leq k - 1 = \frac{C}{2}$

24 $(D_1, D_2, \ldots, D_k), (E_1, E_2, \ldots, E_k), E^* \leftarrow$ **Decompose**$(G_x, (s_1, s_2, \ldots, s_{k-1}))$ where $G_x \leftarrow (V, E_{n^x})$

25 Phase 2.1: Computing Approximate Distances of Paths with Vertex in V_{s_1}

26 $G_{s_1} \leftarrow (V, E_{n^x} \cup E^*)$

27 $\hat{\delta} \leftarrow \min(\hat{\delta}, \textbf{DominatingSetAPASP}(G_{s_1}, V, D_{k-1}, D_1, C + 2))$

28 Phase 2.2: Compute Approximate Distances for $\delta(u, v) \leq 5$

29 **for** $v \in V$ **do**

30 $\quad M_{v,2} \leftarrow \textbf{BFS}(G_x, v, 2)$

31 $\quad$ **if** $\deg(u) \leq s_2$ **then** $M_{u,3} \leftarrow \textbf{BFS}(G_2, u, 3)$ where $G_2 = (V, E_2)$;

32 $\quad \hat{\delta}(v, u) \leftarrow \min(\hat{\delta}(u, v), M_{v,2}(u), M_{v,3}(u))$

33 Phase 2.3: Compute Approximate Distances of Low Degree Paths

34 **for** $1 \leq i \leq k$ **do**

35 $\quad$ **for** $w \in D_i$ **do**

36 $\quad\quad \hat{\delta}(w, v) \leftarrow \min(\hat{\delta}(w, v), \min_{u \in q(w, D_i)} 1 + \hat{\delta}(u, v))$ for all $v \in V$

37 $\quad\quad \hat{\delta} \leftarrow$ **Dijkstra**$(G_{i,w}, w, \hat{\delta})$ where $G_{i,w} \leftarrow (V, E_i \cup (w \times V) \cup E^*)$

38 Phase 3: Compute Approximate Distances of Long Paths

39 $\hat{\delta} \leftarrow \min(\hat{\delta}, \textbf{BoundedAdditiveAPASP}(G, 6, C), \textbf{DenseAPASP}(G, C))$

Phase 2.3 (Lines 26-29), performs **Dijkstra** from each vertex $w \in D_i$ on the graph $(V, E_i \cup E^* \cup (w \times V))$ with the current estimates $\hat{\delta}$ equipped as edge weights. Before **Dijkstra** is executed, the distances from a vertex $w \in D_i$ in the dominating set is updated by examining distances from its constituency (see Definition 2.2).

Finally, Line 31 handles paths of distance between 6 and 12 with a call to **BoundedAdditiveAPASP** and all paths of distance larger than 12 with a call to **DenseAPASP**.

Note that we can also easily use Algorithm 6 in place of **DenseAPASP**, although this will not affect the asymptotic performance of our final algorithm.

Proof of Theorem 3.3

We start with the correctness proof, followed by a run time analysis.

Correctness.

Proof. Consider a vertex pair $u, v \in V \times V$. Let $\delta(u, v)$ denote the length of a shortest path between u, v. From Lemma C.2, we have that $\delta(u, v) \leq \hat{\delta}(u, v)$ for all u, v. We therefore focus on proving $\hat{\delta}(u, v) \leq 2\delta(u, v)$ for all u, v. We can assume $\delta(u, v) \geq 2$ since 1 edge paths can easily be found by examining the adjacency matrix.

Case 1: $\deg(P) \geq n^x$ Suppose $\delta(u, v) \geq C$. By the correctness of **DenseAPASP**, we have $\hat{\delta}(u, v) \leq \delta(u, v) + C \leq 2\delta(u, v)$. Thus, in the following, assume $\delta(u, v) \leq C$

The remaining proof is similar to Case 3 of the randomized algorithm.

Case 2: $\deg(P) < n^x$ We now prove the correctness of Algorithm 3 on low-degree paths.

Case 2.1: $\delta(u, v) \geq C$ Correctness immediately follows from **DenseAPASP**. In particular,

$$\hat{\delta}(u, v) \leq \delta(u, v) + C \leq 2\delta(u, v)$$

Case 2.2: $6 \leq \delta(u, v) \leq C$ Correctness immediately follow from **BoundedAdditiveAPASP**. In particular,

$$\hat{\delta}(u, v) \leq \delta(u, v) + 6 \leq 2\delta(u, v)$$

Case 2.3: $4 \leq \delta(u, v) \leq 5$ Denote the path $P = (u, u_2, u_3, v_3, v_2, v)$. Note that if $\delta(u, v) = 4$, we in fact have $u_3 = v_3$.

Recall that for any vertex, we define the level of vertex v as $\ell(v)$ to be the index i where $s_i \leq \deg(v) < s_{i-1}$. For any path P, define $\ell(P) = \min_{v \in P} \ell(v)$ to be the minimum level of any vertex in P, or alternatively the level of the maximum degree vertex of P. Let $\hat{\delta}_j$ denote the estimate after executing **Dijkstra** from all $w \in D_j$. We now proceed by case analysis on the levels of the vertices comprising shortest path P.

Case 2.3.1: $\ell(u) = \ell(v) = k$ We show a $+4$ additive error in this case. Without loss of generality, assume $\ell(u_2) \leq \ell(v_2)$. Then, for length 4 paths when $u_3 = v_3$, we actually have $P \subset E_{\ell(u_2)}$ so that after the execution of **Dijkstra** at the $\ell(u_2)$ iteration, $\hat{\delta}_{\ell(u_2)}(u_2^*, v) \leq \delta(u_2, v) + 1$ where $u_2^* = r\left(u_2, D_{\ell(u_2)}\right)$. Then, in the k-th iteration, when we execute **Dijkstra** from v, we have access to the edges $(v, u_2^*), (u_2^*, u_2), (u_2, u)$ to find a path of length at most $\delta(u, v) + 2$.

Now, suppose we have a length 5 path so that $u_3 \neq v_3$. If $P \notin E_{\ell(u_2)}$ then $\ell(u_3), \ell(v_3) < \ell(u_2)$. Then, $P \subset E_{\ell(v_3)}$ so that from $v_3^* = r\left(v_3, D_{\ell(v_3)}\right)$ the path $(v_3^*, v_3) \circ P_{v_3, u_2} \circ (u_2, u_2^*) \subset G_{\ell(v_3), v_3^*}$ and **Dijkstra** computes an estimate,

$$\hat{\delta}_{\ell(v_3)}(v_3^*, u_2^*) \leq \delta(v_3, u_2) + 2$$

Then, from u_2^*, since $(u_2^*, v_3^*, v_3) \circ P_{v_3,v} \subset G_{\ell(u_2),u_2^*}$, **Dijkstra** computes,

$$\hat{\delta}_{\ell(u_2)}(u_2^*, v) \leq \hat{\delta}_{\ell(v_3)}(u_2^*, v_3^*) + 1 + \delta(v_3, v) \leq \delta(u_2, v_3) + 3 + \delta(v_3, v) \leq \delta(u_2, v) + 3$$

Finally, in the k-th iteration $(v, u_2^*, u_2, u) \subset G_{k,v}$ so **Dijkstra** finds a path of length at most $\delta(u, v) + 4$.

Case 2.3.2: $\ell(P) = 1$ **and** $\min(\ell(u), \ell(v)) \leq k - 1$ Recall that $P = (u, u_2, u_3, v_3, v_2, v)$. For $\delta(u, v) = 4$, note that $u_3 = v_3$.

This case is handled by Phase 2.1. Without loss of generality, assume $\deg(u) \geq \deg(v)$ so that $\deg(u) \geq s_{k-1}$. Since $\ell(P) = 1$, there is some vertex w such that $\deg(w) \geq s_1$. Then, let $w^* = r(w, D_1) \in D_1$ and $u^* = r(u, D_{k-1}) \in D_{k-1}$. Since $P \subset G_x$,

$$B(w^*, u^*) = \hat{\delta}_{s_1}(w^*, u^*) \leq \delta(w, u) + 2 \leq C + 2$$
$$A(w^*, v) = \hat{\delta}_{s_1}(w^*, v) \leq \delta(w, v) + 1 \leq C + 1$$

Therefore, since both entries are finite, we obtain in the $(\min, +)$ product an estimate at most,

$$\hat{\delta}(u^*, v) \leq \delta(w, u) + \delta(w, v) + 3 \leq \delta(u, v) + 3$$

Then, in Phase 2.3, when we executing **Dijkstra** from $G_{k,v}$, we take the edge (v, u^*) and (u^*, u) to find,

$$\hat{\delta}(v, u) \leq \hat{\delta}(u^*, v) + 1 \leq \delta(u, v) + 4 \leq 2\delta(u, v)$$

Case 2.3.3: $\ell(P) \geq 2$ **and** $\ell(u_2) = \ell(v_2) = 2$ Recall that $P = (u, u_2, u_3, v_3, v_2, v)$. For $\delta(u, v) = 4$, note that $u_3 = v_3$.

We handle this special case by noting that the number of distinct levels in the path P is small. Without loss of generality, suppose $\ell(v) \leq \ell(u)$. Then, after the 2nd iteration,

$$\hat{\delta}_2(u_2^*, v) \leq \delta(u_2, v) + 1$$

as $P \subset E_2$ and $E_2 \cup E^* \subset G_{2,u_2^*}$. Then, after the $\ell(v)$-th iteration, we take the path (v^*, u_2^*, u_2, u) to obtain,

$$\hat{\delta}_{\ell(v)}(v^*, u) \leq \hat{\delta}_2(v^*, u_2^*) + 2 \leq \delta(u_2, v) + 4 \leq \delta(u, v) + 3$$

In the final iteration, we take the path (v, u^*, u) to obtain a $+4$ approximation.

Case 2.3.4: $\ell(P) \geq 2$ **and** $\min(\ell(u_2), \ell(v_2)) = 2, \max(\ell(u_2), \ell(v_2)) \geq 3$ Recall that $P = (u, u_2, u_3, v_3, v_2, v)$. For $\delta(u, v) = 4$, note that $u_3 = v_3$.

Without loss of generality, assume $\ell(u_2) = 2$ and $\ell(v_2) \geq 3$. If $\ell(v) \leq \ell(u)$, then we can proceed in the above case to obtain a $+4$ approximation. Thus, assume $\ell(u) < \ell(v)$. We can now break our analysis into two sub-cases. In the remaining cases, we use the Depth 3 BFS search from Line 31 to ensure correctness.

Suppose $\ell(v_2) \leq \ell(u) < \ell(v)$. This case is illustrated in Figure 4a. Since $\ell(v_2) \geq 3$, we have $\deg(v_2) < s_2$. Then, $\mathbf{BFS}(G_2, v_2, 3)$ finds u_2 so that $\hat{\delta}(v_2, u_2) = \delta(v_2, u_2) = 3$. Then, denote $v_2^* = r(v_2, D_{\ell(v_2)})$. After Line 36 since $v_2 \in q(v_2^*, D_{\ell(v_2)})$, we obtain $\hat{\delta}_{\ell(v_2)}(v_2^*, u^*) \leq \delta(v_2^*, u) + 1$ via the path (v_2^*, u_2, u, u^*). Then in the $\ell(u)$-th iteration, since $\ell(u) < \ell(v)$, we execute **Dijkstra** from u^* and use the path (u^*, v_2^*, v_2, v) to find $\hat{\delta}_{\ell(u)}(u^*, v) \leq \delta(u, v) + 3$. Then, executing **Dijkstra** from v in the k-th iteration, we take the path (v, u^*, u) and find an estimate, $\hat{\delta}(u, v) \leq \delta(u, v) + 4$.

Finally, suppose $\ell(u) < \ell(v_2)$. First, if $\ell(u) = 2 = \ell(P)$, then from Lemma A.3 we have that after the second iteration $\hat{\delta}_2(u^*, v) \leq \delta(u, v)+1$ and obtain a $+2$ approximation in the k-th iteration via the path (v, u^*, u). We can therefore assume $\ell(u) \geq 3$.

Then, we execute a Depth 3 **BFS** from u as illustrated in Figure 4b so that after Line 31 we have $M_{u,3}(v_3) = 3$ (for length 4 paths, we in fact obtain $M_{u,3}(v_2) = 3$). In particular, after Line 36, $\hat{\delta}(u^*, v_3) \leq \delta(u, v_3) + 1$ where $u^* = r(u, D_{\ell(u)})$.

Since $\ell(u) < \ell(v_2), \ell(v)$, in the $\ell(u)$-th iteration, we find v from u^* via the path (u^*, v_3, v_2, v) so that $\hat{\delta}_{\ell(u)}(u^*, v) \leq \delta(u, v) + 1$. Finally, in the k-th iteration, from v we take the path (v, u^*, u) to find,

$$\hat{\delta}(u, v) \leq \delta(u, v) + 2$$

and obtain a $+2$ approximation.

Case 2.3.5: $\ell(P) \geq 2$ **and** $\min(\ell(u_2), \ell(v_2)) \geq 3$ Recall that $P = (u, u_2, u_3, v_3, v_2, v)$. For $\delta(u, v) = 4$, note that $u_3 = v_3$.

Without loss of generality, assume $\ell(u_2) \leq \ell(v_2)$. Since u, u_2, v_2, v all have degree at most s_2, we can freely afford Depth 3 **BFS** from any of these vertices in the graph G_2.

Without loss of generality, suppose $\ell(u) \leq \ell(v)$. For length 4 paths, we have computed a $\hat{\delta}(u, v_2) = 3$ after Line 31 so that after $\ell(u)$-th iteration, $\hat{\delta}(u^*, v) \leq \delta(u, v) + 1$ and we obtain a $+2$ approximation in the k-th iteration via the path (v, u^*, u). Similarly, if $\ell(u) \leq \ell(v_2)$, we obtain a $+2$ approximation for length 5 paths.

Thus, we have $\ell(v_2) < \ell(u)$. Then, as in the above case, we compute a Depth 3 **BFS** from v_2 so that after the $\ell(v_2)$-th iteration, $\hat{\delta}_{\ell(v_2)}(v_2^*, u) \leq \delta(v_2, u) + 1$. Then, after the $\ell(u)$-th iteration, $\hat{\delta}_{\ell(u)}(u^*, v) \leq \delta(u, v) + 3$ and we obtain a $+4$ approximation in the k-th iteration.

Case 2.4: $\delta(u, v) = 3$ For length 3 paths, we prove that we find a $+2$ approximation. Denote the path $P = (u, u_2, v_2, v)$.

We can assume that the maximum degree vertex of P is either u_2, v_2. Otherwise, let u be the endpoint of maximum degree. We compute $\hat{\delta}_{\ell(u)}(u^*, v) \leq \delta(u, v)+1$ and obtain a $+2$ approximation in the k-th iteration via path (v, u^*, u).

Suppose without loss of generality $\ell(u) \leq \ell(v)$. Then, $M_{u,2}(v_2) = 2$ so that after Line 36,

$$\hat{\delta}(u^*, v_2) \leq 3$$

where $u^* = r(u, D_{\ell(u)})$. Executing **Dijkstra** from u^*, we can take the path (u^*, v_2, v) and obtain $\hat{\delta}_{\ell(u)}(u^*, v) \leq \delta(u, v) + 1$. Then, in the k-th iteration, we take the path (v, u^*, u) and obtain a $+2$ approximation.

Case 2.5: $\delta(u, v) = 2$ For length 2 paths, we in fact find the exact distance. Since $P \subset G_x$, in Line 30 we compute depth 2 **BFS** from each vertex, and therefore find $\hat{\delta}(u, v) = \delta(u, v) = 2$. □

Time Complexity

Proof. We analyze the time complexity of our algorithm. x, C are parameters that will be optimized.

Phase 1 Following the same arguments as Phase 1 of the randomized algorithm, we can bound the running time of Phase 1 as,

$$\tilde{O}\left(n^{\omega(1,1-x,1)}\right)$$

Phase 2 In Phase 2, we have a graph G_x of maximum degree n^x and therefore $m \leq n^{1+x}$. From Lemma 2.6, the execution of **Decompose** requires $O(m) = O(n^{1+x})$ time.

In Phase 2.1, we can upper bound $|D_1| = \tilde{O}(n^{1-\frac{(k-1)}{k}x})$ by Lemma 2.6. Likewise, $|D_{k-1}| = \tilde{O}(n^{1-\frac{x}{k}})$. Executing **BFS** on graph G_{s_1} with $O(n^{1+x})$ edges from each $w \in D_1$ requires $\tilde{O}(n^{2+\frac{x}{k}})$ time.

Now, computing the $(\min, +)$ product of A, B requires time,

$$\tilde{O}\left(n^{\omega\left(1,1-\frac{k-1}{k}x,1-\frac{x}{k}\right)}\right) = \tilde{O}\left(n^{\omega(1,1-x,1)}\right)$$

by Proposition C.3.

In Phase 2.2, computing Line 30 requires time $O(n^{2x})$ while computing Line 31 requires time $O\left(n^{\frac{3k-4}{k}x}\right)$. Overall, Phase 2.2 requires time,

$$O\left(n^{1+2x} + n^{1+\frac{3k-4}{k}x}\right)$$

Finally, we turn our attention to Phase 2.3. Fix some iteration i. Since the sets $q(w, D_i)$ are disjoint, $\sum_{w \in D_i} |q(w, D_i)| \leq n$ so that Line 36 requires time $O(n^2)$ over all $w \in D_i$. We bound $|D_i| \leq \tilde{O}(n^{1-\frac{k-i}{k}x})$ by Lemma 2.6. Each invocation of **Dijkstra** requires time $|E_i| = O(n^{1+\frac{k-i+1}{k}x})$ so that across all $w \in D_i$, all invocations of **Dijkstra** can be completed in time $\tilde{O}(n^{2+\frac{x}{k}})$.

We ignore Phase 3 for now as we will find that it is not the computational bottleneck. Phases 1 and 2 require time,

$$\tilde{O}\left(n^{\omega(1,1-x,1)} + n^{1+2x} + n^{1+\frac{3k-4}{k}x} + n^{2+\frac{x}{k}}\right)$$

Using [Bra], we choose $x = 0.4414248$ and $k = 7$ to obtain the time complexity $\tilde{O}(n^{2.07203166})$ for Phase 1 and 2. Finally, we conclude the proof by noting that **BoundedAdditiveAPASP**$(G, 6, C)$ requires time $\tilde{O}(n^{2.05794292})$ by Corollary 5.14 and **DenseAPASP**(G, C) requires time $\tilde{O}(n^{35/17}) = \tilde{O}(n^{2.0589})$ by Lemma 3.2. □

4 $(2, 0)$-Multiplicative Approximations for Long Paths

In this section, we present algorithms for computing $(2, 0)$ approximations for paths of at least some length. We begin with a combinatorial result that gives a $(2, 0)$ multiplicative approximation for paths of length at least k for even values of $k \geq 4$.

4.1 Multiplicative Approximation for Long Paths

Theorem 4.1. *Let $k \geq 4$ be an even integer. Algorithm 4 computes a $(2, 0)$ approximation for all paths of length $\delta(u, v) \geq k$ in expected time $\tilde{O}\left(n^{2+\frac{1}{2(k-1)}}\right)$.*

In Appendix E, we improve the above bound for $k = 4$ and 5.

High Level Overview As a simple example, we begin with by showing that we can obtain a $(2, 0)$ approximation for $\delta(u, v) \geq 4$ in time $\tilde{O}(n^{13/6})$. Decompose the graph with degree thresholds $s_1 = n^{5/6}, s_2 = n^{4/6}, \ldots, s_5 = n^{1/6}$. Let P be a shortest path of length at least 4. If $P \subset E_3$, then we execute **BK2APASP**(G_3) in time $n^{13/6}$ where $G_3 = (V, E_3)$ has maximum degree $s_2 = n^{4/6}$. Otherwise, $P \not\subset E_3$ and vertex $w \in P$ of maximum degree has $\deg(w) \geq s_2$ and is dominated by

$w^* = r(w, D_{\ell(w)})$. Suppose $\deg(v) \geq \deg(u)$. Let $z = b(v, P)$ be the blocking vertex so $\deg(z) \geq s_5$ and $z^* = r(z, D_{\ell(z)}) \in D_5$. From w^*, since $P \subset E_{\ell(w)}$,

$$\hat{\delta}(w^*, z^*) \leq \delta(w, z) + 2$$
$$\hat{\delta}(w^*, v) \leq \delta(w, v) + 1$$

Since $D_{\ell(w)} \subset D_2$, and $2 + 5 + 6 \leq 13$, the path $(v, w^*, z^*, z) \circ P_{z,u}$ exists in $G_{6,v}$ (see Line 49 of Algorithm 4) and **Dijkstra** computes a $+4$ approximation.

For the remaining overview, assume $k \geq 6$. We want to obtain a $(2, 0)$ approximation for paths of length at least k. Of course, it suffices to compute a $+k$ approximation. Let P be a path. We will choose l levels with degree thresholds $s_1, s_2, \ldots, s_{l-1}$ (note that $s_l = 0$). In order to compute a $+k$ approximation, it suffices to have for all w with $\deg(w) \geq s_{l-\frac{k-4}{2}}$ and for all $v \in V$, an estimate

$$\hat{\delta}(w^*, v) \leq \delta(w, v) + 5$$

Then, following a similar argument to Lemma A.2, we obtain a $+k$ additive approximation at the l-th level. In order to guarantee a $+5$ error at the $(l - \frac{k-4}{2})$ level, we require the edge set $D_{l_0} \times D_{l-\frac{k-2}{2}}$ for some choice of l_0.

Suppose $P \subset E_{l_0+1}$. Then, we run **BK2APASP**(G_{l_0+1}) and compute a $(2, 0)$ approximation on P. Otherwise, $P \not\subset E_{l_0+1}$ so the maximum degree vertex $z \in P$ has $\deg(z) \geq s_{l_0}$ and there is some vertex $z^* = r(z, D_{\ell(z)})$. Let $P_{w,v}$ be the sub-path from w to v and let $y = b(w, P_{w,v})$ be the blocking vertex so that $\ell(y) \leq l - \frac{k-2}{2}$ and there is $y^* = r(y, D_{l-\frac{k-2}{2}})$. Since $D_{\ell(z)} \subset D_{l_0}$ and $P \subset E_{\ell(z)}$,

$$\hat{\delta}(w^*, z^*) \leq \delta(w, z) + 2$$
$$\hat{\delta}(z^*, y^*) \leq \delta(z, y) + 2$$

If $D_{l_0} \times D_{l-\frac{k-2}{2}} \subset G_{l-\frac{k-4}{2}, w^*}$, the path $(w^*, z^*, y^*, y) \circ P_{y,v} \subset G_{l-\frac{k-4}{2}, w^*}$ and **Dijkstra** computes an estimate at most,

$$\hat{\delta}(w^*, v) \leq \delta(w, v) + 5$$

The only restriction on l_0 therefore is that

$$l_0 + \left(l - \frac{k-4}{2}\right) + \left(l - \frac{k-2}{2}\right) \leq 2l + 1$$

or

$$l_0 \leq k - 2$$

which justifies our choice of $l_0 = k - 2$. We then choose l to optimize the running time trade-off between the executions of **Dijkstra** which requires time $\tilde{O}(n^{2+1/l})$ and **BK2APASP** which requires time $\tilde{O}\left(n^{1.5} s_{l_0}\right) = \tilde{O}\left(n^{1.5} s_{k-2}\right) = \tilde{O}\left(n^{2.5-\frac{k-2}{l}}\right)$.

Algorithm Our algorithm begins by initializing the distance estimates with the adjacency matrix and setting parameters l_0, l. Next, Line 45 decomposes the graph with $l - 1$ degree thresholds.

In Phase 1, at each level j, we compute **Dijkstra** from $w \in D_j$ on the graph $G_{j,w}$ in Line 50. The graph $G_{j,w}$ consists of edge set $E_j \cup E^* \cup (w \times V)$ and $D_{j_1} \times D_{j_2}$ for all $j + j_1 + j_2 \leq 2l - 1$.

Finally, we take the minimum between the estimate computed above and the estimate given by **BK2APASP** on the graph $G_{k_0+1} = (V, E_{k_0+1})$.

Algorithm 4 LongMultiplicativeAPASP(G, k)

Input : Unweighted, undirected graph $G = (V, E)$ with n vertices; even integer k

Output: Distance estimate $\hat{\delta} : U \times V \to \mathbb{Z}$ such that $\delta(u, v) \leq \hat{\delta}(u, v)$ for all $u, v \in V$ and $\hat{\delta}(u, v) \leq 2\delta(u, v)$ whenever $\delta(u, v) \geq k$

Phase 0: Set up and Decompose Graph

$\hat{\delta}(u, v) \leftarrow \begin{cases} 1 & (u, v) \in E \\ \infty & \text{o/w} \end{cases}$

$l \leftarrow 2(k-1)$

$l_0 \leftarrow k - 2$

$s_i \leftarrow n^{1-\frac{i}{l}}$ for all $1 \leq i \leq l-1$

$(D_1, D_2, \ldots, D_l), (E_1, E_2, \ldots, E_l), E^* \leftarrow$ Decompose$(G, (s_1, s_2, \ldots, s_{l-1}))$

Phase 1: Compute Distance Estimates on High Degree Paths

for $1 \leq j \leq l$ **do**

  for $w \in D_j$ **do**

    $G_{j,w} \leftarrow \left(V, E_j \cup \left(\bigcup_{j+j_1+j_2 \leq 2l+1} D_{j_1} \times D_{j_2}\right) \cup E^* \cup (w \times V)\right)$

    $\hat{\delta} \leftarrow$ Dijkstra$(G_{j,w}, w, \hat{\delta})$

$\hat{\delta} \leftarrow \min\left(\hat{\delta}, \textbf{BK2APASP}(G_{l_0+1})\right)$ where $G_{l_0+1} \leftarrow (V, E_{l_0+1})$

Warm-Up: $k = 4$ We first show that we can obtain a $\tilde{O}(n^{13/6})$ algorithm for computing $(2, 0)$ approximation for paths of length at least 4.

Proof. Since $k = 4$, we have $l = 6$ and $l_0 = 2$. We decompose the graph with degree thresholds $s_1 = n^{5/6}, s_2 = n^{4/6}, \ldots, s_5 = n^{1/6}$.

We execute **BK2APASP** on $G_3 = (V, E_3)$. Therefore, if $P \subset E_3$, we obtain a $(2, 0)$ approximation via **BK2APASP**.

Otherwise, $P \not\subset E_3$. In this case, we claim to compute a $+4$ approximation, which implies a $(2, 0)$ approximation for $\delta(u, v) \geq 4$. The vertex $w \in P$ of maximum degree has $\deg(w) \geq s_2$ and is dominated by $w^* = r(w, D_{\ell(w)})$.

Without loss of generality, suppose $\deg(v) \geq \deg(u)$. Let $z = b(v, P)$ be the blocking vertex from v. Since $P \not\subset E_3$, $z \neq u$ and $\deg(z) \geq s_5$.

If $w = z$ then since $P \subset E_{\ell(w)}$, $B(P) = \{u, v, w\}$ (Definition 3.4) and we obtain a $+4$ approximation. In particular,

$$\hat{\delta}_{\ell(w)}(w^*, v) \leq \delta(w, v) + 1$$

where $w^* = r(w, D_{\ell(w)})$. If $\ell(v) = 6$, then,

$$\hat{\delta}(v, u) \leq \hat{\delta}_{\ell(w)}(v, w^*) + 1 + \delta(w, u) \leq \delta(u, v) + 2$$

Otherwise, if $v^* = r(v, D_{\ell(v)})$,

$$\hat{\delta}_{\ell(v)}(v^*, u) \leq \hat{\delta}_{\ell(w)}(v^*, w^*) + 1 + \delta(w, u) \leq \delta(v, u) + 3$$

Of course, this implies a $+4$ approximation when executing Dijkstra on $G_{6,u}$ by taking the path (u, v^*, v).

Therefore, $w \neq z$. From w^*, since $P \subset E_{\ell(w)}$,

$$\hat{\delta}(w^*, z^*) \leq \delta(w, z) + 2$$

$$\hat{\delta}(w^*, v) \leq \delta(w, v) + 1$$

Since $D_{\ell(w)} \subset D_2$ and $z^* \in D_5$, and $2 + 5 + 6 \leq 13$, the path $(v, w^*, z^*, z) \circ P_{z,u}$ exists in $G_{6,v}$ and **Dijkstra** computes a $+4$ approximation.

$$\hat{\delta}(v, u) \leq \hat{\delta}(v, w^*) + \hat{\delta}(w^*, z^*) + 1 + \delta(z, u) = \delta(v, u) + 4$$

To analyze the running time, it suffices to observe that for each i, $|D_i| = \tilde{O}(n^{i/6})$ while $G_{i,w}$ has $O(n^{2-(i-1)/6})$ edges so that the computation of **Dijkstra** requires $\tilde{O}(n^{13/6})$. Finally, computing **BK2APASP** on $G_3 = (V, E_3)$ requires $\tilde{O}(m\sqrt{n}) = \tilde{O}(n^{13/6})$. □

Correctness Finally, we prove Theorem 4.1 for general $k \geq 6$.

Proof. First, we note that $\delta(u, v) \leq \hat{\delta}(u, v)$ since **BK2APASP** is correct and every edge weight in $G_{j,w}$ is a distance estimate computed by some path in the original graph G.

Suppose $P \subset E_{l_0+1}$. By the correctness of **BK2APASP**, $\hat{\delta}(u, v) \leq 2\delta(u, v)$.

Then, suppose $P \not\subset E_{l_0+1}$. Suppose $\ell(v) \leq \ell(u)$. Recall from Lemma A.2 the blocking vertices $B(P) = \{x_0, x_1, \ldots, x_t\}$ and levels $L_B(P)$ of path P.

Let $a = \min_{\ell(x_i) \leq l - \frac{l_0}{2} + 1} i$ be the minimum index of an element in the blocking set such that $\ell(x_i) \leq l - \frac{l_0}{2} + 1$. Since P is not contained in E_{l_0+1}, we can assume a exists and $a \geq 1$. Otherwise, every edge is in $E_{l - \frac{l_0}{2} + 2} = E_{(3k+2)/2} \subset E_{k-1} = E_{l_0+1}$. Since $l \geq \ell(x_1) > \ell(x_2) > \ldots > \ell(x_t) \geq 1$ (Lemma A.1), we can upper bound,

$$a \leq l - \left(l - \frac{l_0}{2} + 1\right) = \frac{l_0}{2} - 1 = \frac{k}{2} - 2$$

Let $x_a \in B(P)$ be the corresponding vertex in $B(P)$. Since P has an edge not in E_{l_0+1}, the last blocking vertex of minimum level must have $\ell(x_t) \leq l_0$. Recall that we denote $v^* = r\left(v, D_{\ell(v)}\right)$ for any vertex v. Since $P \subset E_{\ell(x_t)}$, we again have $\hat{\delta}_{\ell(x_t)}(x_t^*, x_a^*) \leq \delta(x_t, x_a) + 2$ and $\hat{\delta}_{\ell(x_t)}(x_t^*, x_{a+1}^*) \leq \delta(x_t, x_{a+1}) + 2$.

Consider the $\ell(x_a)$-th iteration. The edges $D_{\ell(x_t)} \times D_{\ell(x_{a+1})}$ are in $G_{\ell(x_a), x_a}$ as,

$$\ell(z) + \ell(x_{a+1}) + \ell(x_a) \leq l_0 + \left(l - \frac{l_0}{2}\right) + \left(l - \frac{l_0}{2} + 1\right) = 2l + 1$$

Next observe that $x_t \in P_{x_a, x_{a+1}}$ (Lemma A.1), as each blocking vertex $x_{i+1} = b(x_i, P_{x_i, x_{i-1}})$ is by definition in the sub-path $P_{x_i, x_{i-1}}$. Therefore we have the path $(x_a^*, x_t^*, x_{a+1}^*, x_{a+1}) \circ P_{x_{a+1}, x_{a-1}}$ so **Dijkstra** computes an estimate at most,

$$\begin{aligned}
\hat{\delta}_{\ell(x_a)}(x_a^*, x_{a-1}) &\leq \hat{\delta}_{\ell(x_t)}(x_a^*, x_t^*) + \hat{\delta}_{\ell(x_t)}(x_t^*, x_{a+1}^*) + 1 + \delta(x_{a+1}, x_{a-1}) \\
&\leq \delta(x_a, x_t) + \delta(x_t, x_{a+1}) + \delta(x_{a+1}, x_{a-1}) + 5 \\
&\leq \delta(x_a, x_{a-1}) + 5
\end{aligned}$$

Then, following a similar argument to the inductive step of Lemma A.2, we claim the following for all $1 \leq j \leq a$.

$$\hat{\delta}_{\ell(x_j)}(x_j^*, x_{j-1}) \leq \delta(x_j, x_{j-1}) + 2(2 + (a - j)) + 1$$

where we have established the base case $j = a$ above. We now proceed by induction for $j < a$. Consider an execution of **Dijkstra** from x_j^* in $G_{\ell(x_j), x_j^*}$. Let x_{j+1} be the blocking vertex from the

previous iteration. We take the edges (x_j^*, x_{j+1}^*), $(x_{j+1}^*, x_{j+1}) \in E^*$, and the remaining edges in $E_{\ell(x_j)}$. By induction, we have,

$$\begin{aligned}\hat{\delta}_{\ell(x_j)}(x_j^*, x_{j-1}) &\leq \hat{\delta}_{\ell(x_{j+1})}(x_j^*, x_{j+1}^*) + 1 + \delta(x_{j+1}, x_{j-1}) \\ &\leq \delta(x_j, x_{j+1}) + 2(2 + (a - (j+1))) + 3 + \delta(x_{j+1}, x_{j-1}) \\ &\leq \delta(x_j, x_{j-1}) + 2(2 + (a - j)) + 1\end{aligned}$$

Thus, we have,

$$\hat{\delta}_{\ell(v)}(v^*, u) = \hat{\delta}_{\ell(x_1)}(x_1^*, x_0) \leq \delta(v, u) + 2(a+1) + 1$$

From u, we take the edge (u, v^*), followed by $(v^*, v) \in E^*$ so that,

$$\begin{aligned}\hat{\delta}(u, v) &\leq \hat{\delta}_{\ell(v)}(u, v^*) + 1 \\ &\leq \delta(u, v) + 2(a+1) + 2 \\ &\leq \delta(u, v) + k\end{aligned}$$

□

Time Complexity

Proof. We set $l_0 = k - 2$ and leave l to be optimized.

Decompose requires $\tilde{O}(ln^2)$ time. Over all D_j, the invocations of **Dijkstra** requires $\tilde{O}(n^{2+1/l})$ time, as for each i, $|D_i| = \tilde{O}(n/s_i)$ and $|E_i| = O(ns_{i-1})$ while $|D_{j_1} \times D_{j_2}| = \tilde{O}\left(\frac{n}{s_{j_1}} \times \frac{n}{s_{j_2}}\right) = \tilde{O}\left(n^{\frac{j_1+j_2}{l}}\right) = \tilde{O}\left(n^{2-\frac{j-1}{l}}\right)$ as $j_1 + j_2 \leq 2l + 1 - j$.

BK2APASP on G_{l_0+1} requires $s_{l_0} n^{1.5} = n^{1.5+1-\frac{k-2}{l}}$ time. Balancing,

$$2 + \frac{1}{l} = 2.5 - \frac{k-2}{l}$$

we solve to obtain,

$$\begin{aligned}2l + 1 &= \frac{5}{2}l - k + 2 \\ \frac{l}{2} &= k - 1 \\ l &= 2(k-1)\end{aligned}$$

Thus, the algorithm runs in time $\tilde{O}(n^{2+1/l}) = \tilde{O}(n^{2+\frac{1}{2k-1}})$ □

4.2 Faster Multiplicative Approximations for Long Paths with FMM

In this section, we improve upon Theorem 4.1 using fast matrix multiplication.

Theorem 4.2. *Let $k \geq 4$ be an even integer. Algorithm 4 computes a $(2, 0)$ approximation for all paths of length $\delta(u, v) \geq k$ in expected time $\tilde{O}(n^{2+\frac{x}{l-k+2}})$ where x, l are chosen to minimize,*

$$\max\left(\omega\left(1 - \frac{k-2}{2(l-k+2)}x, 1-x, 1 - \frac{k-4}{2(l-k+2)}x\right), 2 + \frac{x}{l-k+2}, 1.5 + x\right)$$

In Table 1 and Proposition C.4, we exhibit a few running times for $6 \leq k \leq 10$.

High Level Overview We begin by showing that we can obtain a $(2,0)$ approximation for $\delta(u,v) \geq 4$ in time $\tilde{O}(n^{2.01973523})$. Set parameters $x = 0.51973523, M = 35$ and $l = 29$ so that $l_0 = 2$. Decompose the graph with degree thresholds $s_2 = n^x, s_3 = n^{26x/27}, \ldots, s_{28} = n^{x/27}$. Let P be a shortest path of length at least 4. If $P \subset E_3$, then we execute $\textbf{BK2APASP}(G_3)$ in time $n^{1.5+x} = n^{2.01973523}$ where $G_3 = (V, E_3)$ has maximum degree $s_2 = n^x$. If $\delta(u,v) \geq M$, then $\textbf{DenseAPASP}$ computes a $+M$ approximation and we obtain a $(2,0)$ approximation. Otherwise, $P \not\subset E_3$ is of length at most M and vertex $w \in P$ of maximum degree has $\deg(w) \geq s_2$. Suppose $\deg(v) \geq \deg(u)$. Let $z = b(v,P)$ be the blocking vertex so $\deg(z) \geq s_{28}$ and $z^* = r(z, D_{\ell(z)}) \in D_{28}$. Let $\frac{n}{2^i} \leq \deg(w) \leq \frac{n}{2^{i-1}}$. Since $P \subset F_i$, we obtain the estimates from w^*,

$$\hat{\delta}(w^*, z^*) \leq \delta(w,z) + 2 \leq M + 2$$
$$\hat{\delta}(w^*, v) \leq \delta(w,v) + 1 \leq M + 1$$

Then, the bounded $(\min, +)$ product computes,

$$\hat{\delta}(v, z^*) \leq \delta(v,z) + 3$$

as $v \in V, z^* \in D_{28}, w^* \in C_i$. Finally, the path $(v, z^*, z) \circ P_{z,u}$ exists in $G_{6,v}$ and $\textbf{Dijkstra}$ computes a $+4$ approximation.

For the remaining overview, assume $k \geq 6$. In Algorithm 4, we obtained a $+5$ approximation after the $\left(l - \frac{k-4}{2}\right)$ iteration by including edge sets $D_i \times D_{l-\frac{k-2}{2}}$ for all $i \leq l_0$ in the edges of $G_{l-\frac{k-4}{2},w}$ for $w \in D_{l-\frac{k-4}{2}}$. Using fast matrix multiplication, we will pre-compute the distances obtained from these edges and instead encode them in smaller edge set $w \times V$.

Consider a shortest path P. If $P \subset E_{l_0+1}$, then we again invoke $\textbf{BK2APASP}$ to compute a $(2,0)$ approximation. Therefore, suppose $P \not\subset E_{l_0+1}$. For a large enough constant M, we can invoke $\textbf{DenseAPASP}(G, M)$ (Lemma 3.2) or $\textbf{AdditiveAPASP}$ (Theorem 5.6) to compute a $+M$ approximation without affecting the overall running time. Therefore, we can also assume that $\delta(u,v) \leq M$.

We decompose the graph G into $(1-x)\log n$ levels with each degree threshold defined as $t_i = \frac{n}{2^i}$. We compute dominating sets C_i of size $\tilde{O}(n/t_i)$ and edge sets F_i of size at most $O(nt_{i-1}) = O(nt_i)$. Again, to obtain a $+k$ additive approximation, it suffices to have a $+5$ approximation after the $l - \frac{k-4}{2}$ iteration of computing $\textbf{Dijkstra}$. Consider the blocking vertices $B(P) = \{x_0, x_1, \ldots, x_t\}$ (Definition 3.4). Let a be the minimum index such that $\ell(x_a) \leq l - \frac{k-4}{2}$ so that $\ell(x_{a+1}) \leq l - \frac{k-2}{2}$ and $\ell(x_t) \leq l_0$ as we have $P \not\subset E_{k_0+1}$. Then, if $\frac{n}{2^{i^*}} \leq \deg(x_t) \leq \frac{n}{2^{i^*-1}}$, we have $P \subset F_{i^*}$ and in the i^*-th call to $\textbf{DominatingSetAPASP}$,

$$\hat{\delta}(x_t^*, x_a^*) \leq \delta(x_t, x_a) + 2$$
$$\hat{\delta}(x_t^*, x_{a+1}^*) \leq \delta(x_t, x_{a+1}) + 2$$

By our assumption $\delta(u,v) \leq M$ both entries in the matrices constructed by $\textbf{DominatingSetAPASP}$ are finite. Furthermore, $x_t \in P_{x_a, x_{a+1}}$ (Lemma A.1) so the bounded $(\min, +)$ product computes,

$$\hat{\delta}(x_a^*, x_{a+1}^*) \leq \delta(x_a, x_{a+1}) + 4$$

Then, in the graph $G_{\ell(x_a), x_a^*}$ there exists the path $(x_a^*, x_{a+1}^*, x_{a+1}) \circ P_{x_{a+1}, x_{a-1}}$ and we obtain an estimate at most,

$$\hat{\delta}(x_a^*, x_{a-1}^*) \leq \delta(x_a, x_{a-1}) + 5$$

Algorithm Algorithm 4 and Algorithm 5 are similar overall, so we only highlight the differences here. In Phase 1, we use a bounded $(\min, +)$ product to replace the edge sets $D_{j_1} \times D_{j_2}$.

In Line 61, we decompose the graph G into $(1-x)\log n$ degree thresholds and at each threshold invoke **DominatingSetAPASP** to compute approximate distances between $D_{l-\frac{l_0}{2}}$ and $D_{l-\frac{l_0}{2}+1}$ in Line 63. In Phase 2, for each level $j \geq l_0 + 1$, we execute **Dijkstra** from each vertex $w \in D_j$ on the graph $G_{j,w} = (V, E_j \cup E^* \cup (w \times V))$ in Line 68.

Finally, we take the minimum of our estimate with the estimate produced by **BK2APASP** and **DenseAPASP**.

Analysis We briefly argue that Algorithm 5 equipped with fast matrix multiplication should outperform the combinatorial Algorithm 4. Indeed, suppose we use naive matrix multiplication, so that the largest matrix multiplication contains matrices of size $D_{l-\frac{l_0}{2}} \times D_{l_0}$ and $D_{l_0} \times D_{l-\frac{l_0}{2}+1}$ which requires time,

$$\begin{aligned}\tilde{O}\left(\frac{n^3}{s_{l_0} s_{l-\frac{l_0}{2}} s_{l-\frac{l_0}{2}+1}}\right) &= \tilde{O}\left(n^{3-\left(1+\frac{l_0}{2(l-l_0))}+\frac{l_0-2}{2(l-l_0}\right)x}\right) \\ &= \tilde{O}\left(n^{3-\left(\frac{2l-2}{2(l-l_0)}\right)x}\right)\end{aligned}$$

Balancing the running times,

$$3 - \frac{l-1}{l-l_0}x = 2 + \frac{x}{l-l_0}$$

we obtain,

$$x = \frac{l-l_0}{l}$$

so that the overall running time is $2+\frac{1}{l}$, which matches the combinatoiral algorithm Algorithm 4.

Warm-Up: $k = 4$ We begin by showing that we can obtain a $(2,0)$ approximation for $\delta(u,v) \geq 4$ in time $\tilde{O}(n^{2.01973523})$.

Proof. Using [Bra], we set parameters $x = 0.51973523$ and $l = 29$ so that $l_0 = 2$. Choosing $M = 35$ suffices to ensure **DenseAPASP** requires time $\tilde{O}\left(n^{2.0195}\right)$ (Lemma 3.2).

Decompose the graph with degree thresholds $s_2 = n^x, s_3 = n^{26x/27}, \ldots, s_{28} = n^{x/27}$. Let P be a shortest path of length at least 4. If $P \subset E_3$, then we execute **BK2APASP**(G_3) in time $n^{1.5+x} = n^{2.01973523}$ where $G_3 = (V, E_3)$ has maximum degree $s_2 = n^x$. If $\delta(u,v) \geq M$, then **DenseAPASP** computes a $+M$ approximation and we obtain a $(2,0)$ approximation.

Then, assume $P \not\subset E_3$ is of length at most M and vertex $w \in P$ of maximum degree has $\deg(w) \geq s_2$. Suppose $\deg(v) \geq \deg(u)$. Let $z = b(v,P)$ be the blocking vertex so $\deg(z) \geq s_{28}$ and $z^* = r(z, D_{\ell(z)}) \in D_{28}$. If $z = w$, we obtain a $+4$ approximation as in Algorithm 4.

Assume $z \neq w$. Let $\frac{n}{2^i} \leq \deg(w) \leq \frac{n}{2^{i-1}}$. Since $P \subset F_i$, we obtain the estimates,

$$\begin{aligned}\hat{\delta}(w^*, z^*) &\leq \delta(w,z) + 2 \leq M + 2 \\ \hat{\delta}(w^*, v) &\leq \delta(w,v) + 1 \leq M + 1\end{aligned}$$

Then, the bounded $(\min, +)$ product computes,

$$\hat{\delta}(v, z^*) \leq \delta(v,z) + 3$$

Algorithm 5 FMMLongMultiplicativeAPASP(G, k)

Input : Unweighted, undirected graph $G = (V, E)$ with n vertices; approximation parameter k

Output: Distance estimate $\hat{\delta} : U \times V \to \mathbb{Z}$ such that $\delta(u, v) \leq \hat{\delta}(u, v)$ for all $u, v \in V$ and $\hat{\delta}(u, v) \leq 2\delta(u, v)$ whenever $\delta(u, v) \geq k$

Phase 0: Set up and Decompose Graph

$$\hat{\delta}(u, v) \leftarrow \begin{cases} 1 & (u, v) \in E \\ \infty & \text{o/w} \end{cases}$$

Parameters x, l chosen to minimize

$$t_{\max} = \max\left(\omega\left(1 - \frac{k-2}{2(l-k+2)}x, 1-x, 1 - \frac{k-4}{2(l-k+2)}x\right), 2 + \frac{x}{l-k+2}, 1.5 + x\right)$$

M chosen so that $2 + \frac{2}{3M-2} \leq t_{\max}$

$l_0 \leftarrow k - 2$

$s_{l-i} \leftarrow n^{\frac{i}{l-l_0}x}$ for all $1 \leq i \leq l - l_0$

$(D_{l_0}, D_{l_0+1}, \ldots, D_l), (E_{l_0}, E_{l_0+1}, \ldots, E_l), E^* \leftarrow$ **Decompose**$(G, (s_{l_0+1}, s_{l_0+2}, \ldots, s_{l-1}))$

Phase 1: Compute Intermediate Distance Estimates on High Degree Paths

$t_i \leftarrow \frac{n}{2^i}$ for all $1 \leq i \leq b - 1 = \lceil (1-x)\log n \rceil$

$(C_1, C_2, \ldots, C_b), (F_1, F_2, \ldots, F_b), F^* \leftarrow$ **Decompose**$(G, (t_1, t_2, \ldots, t_{b-1}))$

for $1 \leq i \leq b$ **do**

  $\hat{\delta} \leftarrow \min\left(\hat{\delta}, \textbf{DominatingSetAPASP}\left(G_i, D_{l-\frac{l_0}{2}}, D_{l-\frac{l_0}{2}+1}, C_i, M+2\right)\right)$ where $G_i = (V, F_i)$.

Phase 2: Compute Distance Estimates

for $l_0 + 1 \leq j \leq l$ **do**

  for $w \in D_j$ **do**

    $G_{j,w} \leftarrow (V, E_j \cup E^* \cup (w \times V))$

    $\hat{\delta} \leftarrow$ **Dijkstra**$(G_{j,w}, w, \hat{\delta})$

$\hat{\delta} \leftarrow \min\left(\hat{\delta}, \textbf{BK2APASP}(G_{l_0+1})\right)$ where $G_{l_0+1} \leftarrow (V, E_{l_0+1})$

$\hat{\delta} \leftarrow \min\left(\hat{\delta}, \textbf{DenseAPASP}(G, M)\right)$

as $v \in V, z^* \in D_{28}, w^* \in C_i$. Finally, the path $(v, z^*, z) \circ P_{z,u}$ exists in $G_{6,v}$ and **Dijkstra** computes a $+4$ approximation.

The computation of every invocation of **Dijkstra** requires time $\tilde{O}(n^{2+x/27})$, since $l - l_0 = 27$. The computation of **BK2APASP** requires time $\tilde{O}(n^{1.5+x})$. The overall complexity of **DominatingSetAPASP** is dominated by the computation of bounded $(\min, +)$ products, which requires time,

$$\tilde{O}\left(n^{\omega\left(1-\frac{x}{27}, 1-x, 1\right)}\right)$$

where we have again plugged in $l = 29, l_0 = 2$. Plugging in $x = 0.51973523$ yields the desired complexity. □

Correctness We prove Theorem 4.1 for general $k \geq 6$.

Proof. Let u, v be vertices and P a shortest path of length $\delta(u, v)$.

First, note $\delta(u,v) \le \hat{\delta}(u,v)$ for all u, v as **DominatingSetAPASP** returns $\hat{\delta}(u,v) \ge \delta(G_i, u, v) \ge \delta(u,v)$ by Lemma C.1. In Phase 2, every edge weight in $G_{j,w}$ is the length of some path previously found in the original graph G. Finally, **BK2APASP** and **DenseAPASP** return estimates $\hat{\delta}(u,v) \ge \delta(u,v)$.

Now, we only need to show $\hat{\delta}(u,v) \le 2\delta(u,v)$ If $P \subset E_{l_0+1}$, this immediately follows from the correctness of **BK2APASP**. If $\delta(u,v) \ge M$, then we have correctness by **DenseAPASP**, since,

$$\hat{\delta}(u,v) \le \delta(u,v) + M \le 2\delta(u,v)$$

Therefore, assume $P \not\subset E_{l_0+1}$ and $\delta(u,v) \le M$. Without loss of generality, suppose $\ell(v) \le \ell(u)$. Recall from Lemma A.2 the blocking vertices $B(P) = \{x_0, x_1, \ldots, x_t\}$ and levels $L_B(P)$ of path P.

Let $a = \min_{\ell(x_i) \le l - \frac{l_0}{2}+1} i$ be the minimum index of an element in the blocking set such that $\ell(x_i) \le l - \frac{l_0}{2} + 1$. Since $l \ge \ell(x_1) > \ell(x_2) > \ldots > \ell(x_t) \ge 1$, we can upper bound,

$$a \le l - \left(l - \frac{l_0}{2} + 1\right) = \frac{l_0}{2} - 1 = \frac{k}{2} - 2$$

Since P is not contained in E_{l_0+1}, we can assume a exists and $a \ge 1$.

Let $x_a \in B(P)$ be the corresponding vertex in $B(P)$. Since P has an edge not in E_{l_0+1}, the last blocking vertex of minimum level must have $\ell(x_t) \le l_0$. Recall that we denote $v^* = r\left(v, D_{\ell(v)}\right)$ for any vertex v.

Let $\frac{n}{2^{i^*}} \le \deg(x_t) < \frac{n}{2^{i^*-1}}$ and $x_t^* = r(x_t, C_{i^*})$. Since $P \subset F_{i^*}$, the i^*-th invocation of **DominatingSetAPASP** computes

$$\hat{\delta}_{i^*}(x_t^*, x_a^*) \le \delta(x_t, x_a) + 2 \le M + 2$$
$$\hat{\delta}_{i^*}(x_t^*, x_{a+1}^*) \le \delta(x_t, x_{a+1}) + 2 \le M + 2$$

These entries are finite in the matrices A_{i^*}, B_{i^*} constructed by **DominatingSetAPASP**.

Note that $x_t \in P_{x_a, x_{a+1}}$ (Lemma A.1). Furthermore, $\ell(x_a) \le l - \frac{l_0}{2} + 1$ implies $x_a^* \in D_{l - \frac{l_0}{2}+1}$ and $\ell(x_{a+1}) < \ell(x_a)$ implies $x_{a+1}^* \in D_{l-\frac{l_0}{2}}$. Then, the bounded $(\min, +)$ product computes,

$$\begin{aligned}\hat{\delta}_{i^*}(x_a^*, x_{a+1}^*) &\le \hat{\delta}_{i^*}(x_a^*, x_t^*) + \hat{\delta}_{i^*}(x_t^*, x_{a+1}^*) \\ &\le \delta(x_a, x_t) + \delta(x_t, x_{a+1}) + 4 \\ &\le \delta(x_a, x_{a+1}) + 4\end{aligned}$$

Then, in the $\ell(x_a)$-th iteration, we can take the edge sequence $(x_a^*, x_{a+1}^*, x_{a+1})$ and the remaining edges in $E_{\ell(x_a)}$ to compute a distance estimate at most,

$$\hat{\delta}(x_a^*, x_{a-1}^*) \le \hat{\delta}_{i^*}(x_a^*, x_{a+1}^*) + 1 + \delta(x_{a+1}, x_{a-1}) \le \delta(x_a, x_{a-1}) + 5$$

The remaining proof follows exactly as in Theorem 4.1.

□

Time Complexity

Proof. We set $l_0 \leftarrow k - 2$ and leave x, l, M to be optimized. Phase 0 requires $\tilde{O}(ln^2)$ time overall as we only invoke **Decompose**. Invoking **Dijkstra** requires $\tilde{O}(n^{2+\frac{x}{l-l_0}})$ time as $|D_j| = \tilde{O}(n/s_j)$ and $|E_j| \le O(ns_{j-1})$. **BK2APASP** requires $\tilde{O}(m\sqrt{n}) = \tilde{O}(n^{1.5}s_{l_0}) = \tilde{O}(n^{1.5+x})$ time as we call **BK2APASP** on graph G_{l_0+1}, a graph with maximum degree $s_{l_0} = n^x$.

Phase 1 calls **DominatingSetAPASP** $b = \tilde{O}(1)$ times. In each call, we call **BFS** on the graph (V, F_i) from dominating set C_i in time $\tilde{O}(|C_i||F_i|) = \tilde{O}\left(\frac{n}{t_i} 2nt_i\right) = \tilde{O}\left(n^2\right)$. Then, we compute a **BoundedMinPlus** on matrices of size at most $\left|D_{l-\frac{l_0}{2}}\right| \times |C_i|$ and $|C_i| \times \left|D_{l-\frac{l_0}{2}+1}\right|$. Since $|C_i| = O(|C_b|) = \tilde{O}\left(\frac{n}{t_b}\right) = \tilde{O}\left(2^b\right) = \tilde{O}\left(n^{1-x}\right)$, all computations of **BoundedMinPlus** require time at most,

$$\tilde{O}\left(n^{\omega\left(1-\frac{\frac{l_0}{2}}{l-l_0}x, 1-x, 1-\frac{\frac{l_0}{2}-1}{l-l_0}x\right)}\right) = \tilde{O}\left(n^{\omega\left(1-\frac{l_0}{2(l-l_0)}x, 1-x, 1-\frac{l_0-2}{2(l-l_0)}x\right)}\right)$$
$$= \tilde{O}\left(n^{\omega\left(1-\frac{k-2}{2(l-k+2)}x, 1-x, 1-\frac{k-4}{2(l-k+2)}x\right)}\right)$$

To optimize, we use [Bra] and choose x, l minimizing,

$$t_{\max} = \max\left(\omega\left(1-\frac{k-2}{2(l-k+2)}x, 1-x, 1-\frac{k-4}{2(l-k+2)}\right), 2+\frac{x}{l-k+2}, 1.5+x\right)$$

Finally, note that **DenseAPASP** requires $\tilde{O}(n^{2+\frac{2}{3M-2}})$ time. Given our choice of x, l, we simply choose M to be a large enough constant such that the running time of **DenseAPASP** does not exceed $2+\frac{2}{3M-2} \leq T_{\max}$. We give a few example running times in Proposition C.4 and Table 1. □

5 Additive Approximation via Monotone $(\min, +)$ Product

In the following section, we will use the fast monotone $(\min, +)$ product of Chi, Duan, Xie, and Zhang [CDXZ22] to the additive approximation framework of Dor et al. [DHZ00]. In particular, this will generalize the result of Deng et al. [DKR+22] to all distances. Whereas Deng et al. [DKR+22] use fast rectangular matrix multiplication for bounded difference matrices, we will require a more general fast rectangular matrix multiplication. To do so, let us begin by designing a fast algorithm for multiplying rectangular matrices, of which only one is monotone.

5.1 Rectangular Monotone $(\min, +)$ Product

Following the algorithm of [CDXZ22], we give a simple extension of their result for monotone matrices to rectangular monotone matrices. This result slightly extends the work of Durr [Dür23] to handle the most general case of rectangular matrices, which is required in our application.

Theorem 5.1. *Let A be a $n^a \times n^b$ integer matrix with non-negative entries bounded by $O(n^\mu)$. Let B be a $n^b \times n^c$ row-monotone integer matrix with non-negative entries bounded by $O(n^\mu)$. Then, there is an algorithm* **MonotoneMinPlus** *computing $C = A * B$ in time,*

$$\tilde{O}\left(n^{\frac{(a+b+\mu)+\omega(a,b,c)}{2}}\right)$$

Since the algorithm and proof remain largely unchanged, we only provide a sketch of the proof.

Let $\alpha \in [0, 1]$ be a parameter to be fixed, and let p be a uniformly random prime chosen in $[40n^\alpha, 80n^\alpha]$. We make the same simplifying assumption as in [CDXZ22]. By Lemma 3.4 of [CDXZ22], this assumption is justified.

Assumption 5.2. *(Assumption 3.1 of [CDXZ22]) Let $i \in [n^a], k \in [n^b], j \in [n^c]$. For all i, k, A_{ik} is either ∞ or $(A_{ik} \mod p) < \frac{p}{3}$. For all k, j, B_{kj} is either ∞ or $(B_{kj} \mod p) < \frac{p}{3}$. Each row of B is monotone.*

Lemma 5.3. *(Lemma 3.4 of [CDXZ22]) Let A be a $n^a \times n^b$ integer matrix with non-negative entries bounded by $O(n)$. Let B be a $n^b \times n^c$ row-monotone integer matrix with non-negative entries bounded by $O(n)$. The computation $A * B$ can be reduced to a constant number of $A^i * B^i$ where A^i, B^i satisfy Assumption 5.2.*

Proof. We only define the relevant matrices, leaving the proof to [CDXZ22]. Define for $i \in \{1, 2, 3\}$,

$$A_{ik}^{((i)} = \begin{cases} A_{ik} & \frac{(i-1)p}{3} < (A_{ik} \mod p) < \frac{ip}{3} \\ \infty & \text{o/w} \end{cases}$$

$$B_{kj}^{(1)} = \begin{cases} B_{kj} & (B_{kj} \mod p) < \frac{p}{3} \\ p \lfloor B_{kj}/p + 1 \rfloor & \frac{p}{3} \le (B_{kj} \mod p) < \frac{2p}{3} \\ p \lfloor B_{kj}/p + 1 \rfloor & \frac{2p}{3} \le (B_{kj} \mod p) \end{cases}$$

$$B_{kj}^{(2)} = \begin{cases} p \lfloor B_{kj}/p \rfloor + \lceil p/3 \rceil & (B_{kj} \mod p) < \frac{p}{3} \\ B_{kj} & \frac{p}{3} \le (B_{kj} \mod p) < \frac{2p}{3} \\ p \lfloor B_{kj}/p + 1 \rfloor & \frac{2p}{3} \le (B_{kj} \mod p) \end{cases}$$

$$B_{kj}^{(3)} = \begin{cases} p \lfloor B_{kj}/p \rfloor + \lceil 2p/3 \rceil & (B_{kj} \mod p) < \frac{p}{3} \\ p \lfloor B_{kj}/p \rfloor + \lceil 2p/3 \rceil & \frac{p}{3} \le (B_{kj} \mod p) < \frac{2p}{3} \\ B_{kj} & \frac{2p}{3} \le (B_{kj} \mod p) \end{cases}$$

Then, the matrices,

$$A^{(1)}, A^{(2)} - \lceil p/3 \rceil, A^{(3)} - \lceil 2p/3 \rceil, B^{(1)}, B^{(2)} - \lceil p/3 \rceil, B^{(3)} - \lceil 2p/3 \rceil$$

all satisfy Assumption 5.2. □

Define h such that $2^{h-1} \le p < 2^h$. For all $0 \le l \le h$, define $A_{ik}^{(l)} = \left\lfloor \frac{A_{ik} \mod p}{2^l} \right\rfloor$ if A_{ik} is finite and ∞ otherwise. Define $B_{kj}^{(l)} = \left\lfloor \frac{B_{kj} \mod p}{2^l} \right\rfloor$ if B_{kj} is finite and ∞ otherwise. Define $A_{ik}^* = \lfloor A_{ik}/p \rfloor$ if A_{ik} is finite and ∞ otherwise. Define $B_{kj}^* = \lfloor B_{kj}/p \rfloor$ if B_{kj} is finite and ∞ otherwise. Since A^*, B^* are monotone, $C^* = A^* * B^*$ can be computed in $\tilde{O}(n^{(a+b+\mu)-\alpha})$ time using segment trees as described in [CDXZ22]. If C_{ij} is finite, then by Assumption 5.2 $C_{ij} \mod p < \frac{2p}{3}$ so that $C_{ij}^* = \lfloor C_{ij}/p \rfloor$.

In the next phase, we compute $C^{(l)}$ for $l = h, h-1, \ldots, 0$. Note that $C^{(l)}$ is not necessarily $A^{(l)} * B^{(l)}$. We require $C^{(l)}$ to satisfying the following properties if C_{ij} is finite:

1. $\left\lfloor \frac{(C_{ij} \mod p) - 2(2^l - 1)}{2^l} \right\rfloor \le C_{ij}^{(l)} \le \left\lfloor \frac{(C_{ij} \mod p) + 2(2^l - 1)}{2^l} \right\rfloor$
2. If $C_{i,j_0}^* = C_{i,j_1}^*$ for $j_0 < j_1$, the elements $C_{i,j_0}^{(l)}, C_{i,j_0+1}^{(l)}, \ldots, C_{i,j_1}^{(l)}$ are monotonically non-decreasing.

Notice that when $l = 0$, we have $C_{ij}^{(0)} = C_{ij} \mod p$, which combined with C_{ij}^* allows us to recover C_{ij}.

The entries of $A^{(l)}, B^{(l)}, C^{(l)}$ are therefore non-negative integers at most $O\left(\frac{n^\alpha}{2^l}\right)$ or ∞. Since $B^{(l)}$ is monotone, every row is composed of at most $O\left(\frac{n^\alpha}{2^l}\right)$ intervals, where all the values in each

interval are identical. Now, C^* has $O(n^{\mu-\alpha})$ intervals in each row. Within each interval, $C^{(l)}$ has $O\left(\frac{n^\alpha}{2^l}\right)$ intervals, so that $C^{(l)}$ has $O\left(\frac{n^\mu}{2^l}\right)$ such intervals in each row.

Definition 5.4. *(Definition 3.1 of [CDXZ22]) A segment $(i,k,(j_0,j_1))$ with respect to $B^{(l)}, C^{(l)}$ where $i \in [n^a], k \in [n^b], j_0, j_1 \in [n^c]$ and $j_0 \le j_1$ satisfies that for all $j_0 \le j \le j_1$, $B^{(l)}_{kj} = B^{(l)}_{k,j_0}$, $B^*_{kj} = B^*_{k,j_0}$, $C^{(l)}_{ij} = C^{(l)}_{i,j_0}$, and $C^*_{ij} = C^*_{i,j_0}$.*

We will also maintain sets $T^{(l)}_b$ for $-10 \le b \le 10$ where $T^{(l)}_b$ consists of segments $(i,k,(j_0,j_1))$ satisfying $A_{ik} < \infty$ and $A^*_{ik} + B^*_{kj} \neq C^*_{ij}$ and $A^{(l)}_{ik} + B^{(l)}_{kj} = C^{(l)}_{ij} + b$ for all $j \in [j_0, j_1]$. Finally, we turn to the computation of each $C^{(l)}$ matrix.

Computing $C^{(l)}$

1. Since $p < 2^h$, $A^{(h)}, B^{(h)}$ are zero-matrices. Likewise, $C^{(h)} = 0$ satisfies the required properties. For $b \neq 0$, $T^{(h)}_b = \emptyset$ and $T^{(h)}_0$ includes all segments $(i,k,(j_0,j_1))$ where $A_{ik} < \infty$ and $A^*_{i,k} + B^*_{k,j_0} \neq C^*_{i,j_0}$.
2. Let $l < h$. We will construct $C^{(l)}, T^{(l)}_b$ from $C^{(l+1)}, T^{(l+1)}_b$.
   (a) **Polynomial Matrix Multiplication**
   Construct $A^p_{ik} = x^{A^{(l)}_{ik} - 2A^{(l+1)}_{ik}} y^{A^{(l+1)}_{ik}}$ if A_{ik} finite and 0 otherwise. Define B^p_{kj} analogously. Compute $C^p = A^p B^p$. Since the x degree is either 0 or 1, and the y degree is at most n^α, this phase requires $\tilde{O}(n^{\omega(a,b,c)+\alpha})$ time.
   (b) **Subtracting Error Terms**
   If $C^p_{ij} = 0$, then $C^{(l)}_{ij} = \infty$. Otherwise, for all b, we collect all monomials $\lambda x^c y^d$ where $d = C^{(l+1)}_{ij} + b$ and let $C^p_{ijb}(x)$ be the sum of all such λx^c. Next, we compute,

$$R^p_{ijb}(x) = \sum_{(i,k,(j_0,j_1)) \in T^{(l+1)}_b, j \in [j_0,j_1]} x^{A^{(l)}_{ik} - 2A^{(l+1)}_{ik} + B^{(l)}_{kj} - 2B^{(l+1)}_{kj}}$$

   Let s_{ijb} be the minimum degree of x in $C^p_{ijb}(x) - R^p_{ijb}(x)$ and compute $c_{ijb} = 2d + s_{ijb}$ (or $c_{ijb} = \infty$ if $s_{ijb} = 0$). Finally, output $C^{(l)}_{ij} = \min_b c_{ijb}$.
   Constructing $C^p_{ijb}(x)$ requires $\tilde{O}(n^{a+c+\alpha})$ time. Notice each $T^{(l+1)}_b$ has at most 2 different $B^{(l)}_{kj}$ and thus two $R^p_{ijb}(x)$. Since each interval in $B^{(l+1)}_{kj}$ is broken into at most 2 intervals, we can compute $C^p_{ijb}(x) - R^p_{ijb}(x)$ using segment trees in $\tilde{O}(|T^{(l+1)}_b|)$-time. The overall time required in this stage is therefore $\tilde{O}(n^{a+c+\alpha} + |T^{(l+1)}_b|)$.
   (c) **Computing Triples $T^{(l)}_b$**
   It is shown in [CDXZ22] that each segment with respect to $B^{(l+1)}, C^{(l+1)}$ splits into $O(1)$ segments with respect to $B^{(l)}, C^{(l)}$ and furthermore $\bigcup_b T^{(l)}_b \subset \bigcup_b T^{(l+1)}_b$, so it suffices to enumerate $T^{(l+1)}_b$, breaking each segment into $O(1)$ sub-segments. The time required is $\tilde{O}(|T^{(l+1)}_b|)$.

It then suffices to bound the size of the sets $T^{(l)}_b$, which we do below in Lemma 5.5. Finally, we can optimize parameter α and compute the overall running time. Note that the overall running time is,

$$\tilde{O}\left(n^{(a+b+\mu)-\alpha} + n^{\omega(a,b,c)+\alpha} + n^{a+c+\alpha}\right) = \tilde{O}\left(n^{(a+b+\mu)-\alpha} + n^{\omega(a,b,c)+\alpha}\right)$$

giving the desired result. The proof of correctness follows exactly as in [CDXZ22].

Lemma 5.5. *The expected number of segments in $T_b^{(l)}$ is $\tilde{O}(n^{(a+b+\mu)-\alpha})$*

Proof. Suppose first $2^l > \frac{p}{100}$. Then, $B^{(l)}, C^{(l)}$ have at most $O\left(\frac{n^\mu}{2^l}\right) = O\left(\frac{n^\mu}{p}\right) = O\left(n^{\mu-\alpha}\right)$ intervals in a given row. The overall number of intervals is therefore bounded by $O\left(n^{(a+b+\mu)-\alpha}\right)$

Consider a segment $(i, k, (j_0, j_1))$ and arbitrarily pick $j \in [j_0, j_1]$ where A_{ik} finite and $A^*_{ik} + B^*_{kj} \neq C^*_{ij}$. Then, by Assumption 5.2, we have $|A_{ik} + B_{kj} - C_{ij}| \geq \frac{p}{3}$. We now bound the probability that $(i, k, (j_0, j_1)) \in T_b^{(l)}$. By definition, this implies that,

$$\left\lfloor \frac{A_{ik} \mod p}{2^l} \right\rfloor + \left\lfloor \frac{B_{kj} \mod p}{2^l} \right\rfloor = C_{ij}^{(l)} + b$$

Since $\left\lfloor \frac{(C_{ij} \mod p) - 2(2^l-1)}{2^l} \right\rfloor \leq C_{ij}^{(l)} \leq \left\lfloor \frac{(C_{ij} \mod p) + 2(2^l-1)}{2^l} \right\rfloor$

$$-4 \leq \frac{A_{ik} \mod p}{2^l} + \frac{B_{kj} \mod p}{2^l} - \frac{C_{ij} \mod p}{2^l} - b \leq 4$$

Let $C_{ij} = A_{iq} + B_{qj}$, so that,

$$-4 \cdot 2^l \leq (A_{ik} + B_{kj} - A_{iq} - B_{qj}) \mod p - b \cdot 2^l \leq 4 \cdot 2^l$$

so that $(A_{ik} + B_{kj} - A_{iq} - B_{qj}) \mod p$ takes one of $O(2^l)$ values r in the range $[2^l(b-4), 2^l(b+4)]$. Since $|b| \leq 10$, the largest such value is bounded by $|r| \leq 14 \cdot 2^l < \frac{p}{6} \leq \frac{1}{2}|A_{ik} + B_{kj} - A_{iq} - B_{qj}|$ where we used $2^l < \frac{p}{100}$.

Then, if B_{kj}, B_{qj} are from B in Lemma 5.3, then,

$$|(A_{ik} + B_{kj} - A_{iq} - B_{qj}) - r| = O(n^\mu)$$

The number of primes larger than $40n^\alpha$ that divide this quantity is therefore at most $\frac{\mu}{\alpha} = O(1)$. In particular, such a p is chosen with probability $n^{-\alpha}$.

On the other hand, if B_{kj}, B_{qj} may have been set artificially to some number congruent to $0, \lceil p/3 \rceil, \lceil 2p/3 \rceil$ in Lemma 5.3. We can still bound the probability that p divides $|(A_{ik} + B_{kj} - A_{iq} - B_{qj}) - r|$, as in [CDXZ22].

Finally, we have $O(2^l)$ such remainders r, and in total $O\left(\frac{n^{a+b+\mu}}{2^l}\right)$ segments, so that in expectation we can upper bound $|T_b^{(l)}| = \tilde{O}(n^{(a+b+\mu)-\alpha})$. □

5.2 Framework for Additive Approximations via Monotone $(\min, +)$ Product

We now give an improvement on the generalized algorithm of Dor et al. [DHZ00], using fast matrix multiplication to give a better trade-off between additive approximation error and running time. This extends the work of Deng et al. [DKR+22] to additive error beyond $+2$.

Theorem 5.6. *Let $\beta \geq 4$ be an even integer. Let G be an undirected, unweighted graph with n vertices. Algorithm 6 computes $\hat{\delta}$ such that $\delta(u,v) \leq \hat{\delta}(u,v) \leq \delta(u,v) + \beta$ for all $u, v \in V$ in time, $\tilde{O}\left(n^{2+\frac{2x}{\beta+2}}\right)$ where x is the solution to, $\omega\left(1 - \frac{\beta-2}{\beta+2}x, 1-x, 1-\frac{\beta-4}{\beta+2}x\right) = 1 + \frac{4+2\beta}{\beta+2}x$.*[2]

[2] $\omega(a,b,c)$ is the minimum value such that the product of a $\lceil n^a \rceil \times \lceil n^b \rceil$ matrix by a $\lceil n^b \rceil \times \lceil n^c \rceil$ matrix can be computed in $O(n^{\omega(a,b,c)+\varepsilon})$ arithmetic operations for any constant $\varepsilon > 0$. Note $\omega = \omega(1,1,1)$.

Before we present a high level overview of Theorem 5.6, let us recall some useful definitions.

Definition 5.7. *For a spanning tree $T \subset E$ of a connected graph $G = (V, E)$ on n vertices, an* **Euler Tour** *of T is a sequence of vertices $v_1, v_2, \ldots, v_{2n-1}$ where each vertex of G appears at least once and the edges $(v_i, v_{i+1}) \in T$ for all $1 \leq i \leq 2n-2$.*

Given a spanning tree T, an Euler tour of T can be found in $O(|T|)$ time by conducting depth first search on T.

Definition 5.8. *Let $G = (V, E)$ be a graph on n vertices and $D \subset V$. Consider an arbitrary Euler Tour on G, denoted $v_1, \ldots, v_{2n-1}$. X is an* **Euler Tour Distance Matrix** *of D on V if A is a $|D| \times (2n-1)$ matrix where $X(w, i) = \delta(w, v_i)$ for $w \in D$ and v_i as ordered in the Euler Tour.*

It was first observed by Deng, Kirkpatrick, Rong, Vassilevska-Williams, and Zhong [DKR+22] that it is possible to encode the results of a **BFS** search in a row bounded difference matrix by ordering the vertices according to an Euler tour.

We now present a high level description of our algorithm.

High Level Overview First, we recap the algorithm of Deng et al. [DKR+22] and explain why it does not generalize to additive approximations beyond +2. Consider a +4 approximation. The natural approach is to use a bounded difference matrix multiplication to compute a +2 additive approximation on paths with some vertex of degree at least n^x and to invoke $\textsc{SparseAPASP}(G, 4)$ on paths where all vertices have degree at most n^x. If $\omega_{BD}(1, 1-x, 1)$ denotes the exponent required to compute a rectangular bounded difference $(\min, +)$ matrix product with input $n \times n^{1-x}$ and $n^{1-x} \times n$, then this algorithm requires time $n^{2+\frac{x}{3}}$ where x is the solution to $\omega_{BD}(1, 1-x, 1) = 2 + \frac{x}{3}$. For current bounds on ω_{BD}, this is larger than the $\frac{11}{5}$ given by the combinatorial algorithm of Dor, Halperin, and Zwick [DHZ00]. Why? The $\textsc{DenseAPASP}$ algorithm decomposes the graph G into 5 levels, whereas the above algorithm has only decomposed G into 4 levels.

Indeed, only using rectangular matrix multiplication $\omega_{BD}(1, y, 1)$ for some $y \in [0, 1]$, we will not be able to obtain any improvement on $\textsc{DenseAPASP}(G, 4)$. For example, recall that the degree thresholds are $s_1 = n^{4/5}, s_2 = n^{3/5}, s_3 = n^{2/5}, s_4 = n^{1/5}$. In order to improve the overall running time, we must decrease s_4 so that the final iteration of computing $\textsc{Dijkstra}$ on E_5 from all $D_5 = V$ does not require $\tilde{O}(n^{11/5})$ time. If we wish to decrease s_4, then the size of D_4 increases. To limit the overall time of running $\textsc{Dijkstra}$ from D_4 on E_4, we must therefore decrease s_3. Following the same reasoning, we must decrease s_2 and s_1. However, the combinatorial algorithm $\textsc{DenseAPASP}$ also includes the edge set $D_2 \times D_4$ when executing $\textsc{Dijkstra}$ from $D_5 = V$. In particular, either D_2 or D_4 needs to reduce in size, which implies that at least one of s_2, s_4 needs to increase, contradicting our previous conclusions.

We cannot hope to succeed simply by using fast $(\min, +)$ matrix multiplication to replace the first level D_1. However, we use fast $(\min, +)$ matrix multiplication to replace the edge set $D_2 \times D_4$ and more generally $D_{j_1} \times D_{j_2}$ for $j + j_1 + j_2 \leq 2k + 1$. Consider +4 approximation as a basic example. Let P be a shortest path between u, v. If $P \subset E_3$, then we compute a +4 approximation as in $\textsc{SparseAPASP}$. Therefore, suppose $P \subset E_j$ and $P \not\subset E_{j+1}$ for some $j \in \{1, 2\}$. Let y be the vertex closest to v such that $P_{u,y} \subset E_5$. Let z be the vertex of maximum degree in P. Let $y^* = r(y, D_4)$ and $z^* = r(z, D_{\ell(z)})$. In the combinatorial algorithm, since $D_2 \times D_4 \subset G_{5,v}$, we find the path $(v, z^*, y^*, y) \circ P_{y,u}$ and compute a +4 approximation. Instead, we use a fast $(\min, +)$ product to find,

$$\hat{\delta}(v, y^*) \leq \min_{w \in D_2} \hat{\delta}(v, w) + \hat{\delta}(w, y^*) \leq \hat{\delta}(v, z^*) + \hat{\delta}(z^*, y^*) \leq \hat{\delta}(v, y) + 3$$

to obtain the same approximation. To utilize fast matrix multiplication, let x be some parameter to be optimized. Since $P \subset E_3$ is handled by **SparseAPASP**, we decompose the graph $G_x = (V, E_{n^x})$ into 3 levels. Then, for $1 \leq j \leq (1-x)\log n$ thresholds $t_j = \frac{n}{2^j}$, we compute a dominating set C_j and edge set F_j. For each j, we construct matrix A_j of dimension $D_4 \times C_j$ and matrix B_j of dimension $C_j \times V$ where we ensure B_j is a monotone matrix by using the Euler tour distance matrix construction and the transformation **BDtoMonotone**. Filling in these matrices with the results computed by **BFS** from C_j on the graph (V, F_j), we compute the correct distance estimate required in Section 5.2. Since overall the **BFS** require time $\tilde{O}(n^2)$, the running time of $(\min, +)$ matrix multiplication dominates.

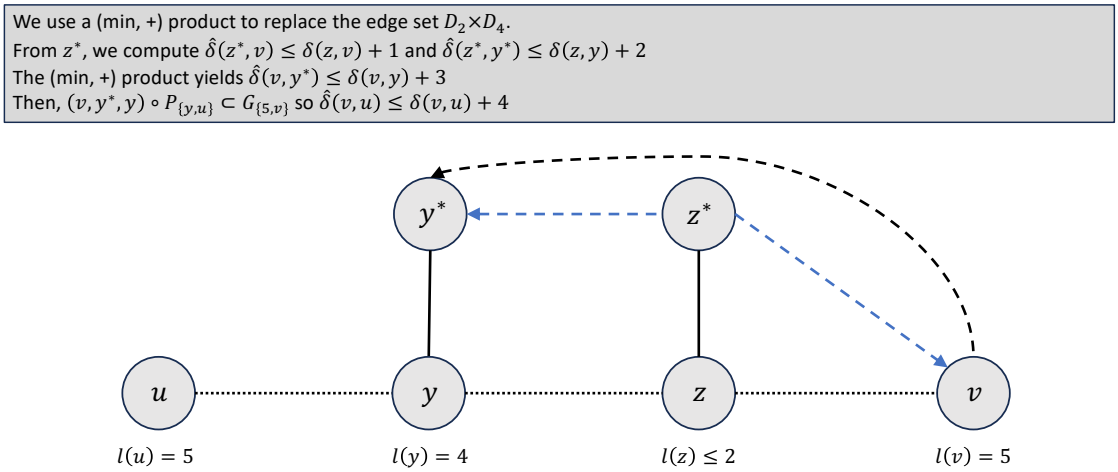


Figure 5: Using $(\min, +)$ product to replace edge set $D_2 \times D_4$. Solid lines represent eges in G. Dotted black lines represent paths in G. Dashed blue arrows represent distance estimates used in the $(\min, +)$ product. Dashed black arrows represent edges $w \times V$ in the graph $G_{j,w}$.

We have already discussed $+6$ approximation in Section 3.2.

For general $+\beta$ approximation, let $k = \frac{3\beta-2}{2}$ and $k_0 = k - \lfloor k/3 \rfloor - 2$. By Lemma A.3, if any path $P \subset E_{k_0+1}$, then we already obtain a $+\beta$-approximation as in **SparseAPASP**. Then, to ensure a $+\beta$ approximation for paths containing a higher degree vertex, we need to ensure that we have a $+5$ approximation after the (k_0+3)-th iteration, we can follow the arguments of Lemma A.3 to obtain a $+\beta$ additive approximation from this point.

We demonstrate this case in Figure 6. In this example, a is a vertex on the shortest path P between u, v such that $\ell(a) = k_0 + 3$. While the diagram only shows $\hat{\delta}(a^*, v) \leq \delta(a, v) + 5$, a similar argument holds for all $x \in P$.

Let b be the vertex closest to a such that $P_{b,v} \subset E_{k_0+3}$. Let z be the highest degree vertex in $P_{a,b}$. Suppose $\frac{n}{2^j} \leq \deg(z) < \frac{n}{2^{j-1}}$. Then, after computing the $(\min, +)$ product $A_j * B_j$, we have an estimate

$$\hat{\delta}(a^*, b^*) \leq \delta(a, b) + 4$$

given by the blue arrows. In particular from a^*, the path $(a^*, b^*, b) \circ P_{b,v} \subset G_{k_0+3,a^*}$ so that

$$\hat{\delta}(a^*, v) \leq \delta(a, v) + 5$$

We next argue that our algorithm has an improved running time. Suppose we use naive matrix multiplication. Then, the time exponent to compute a **MonotoneMinPlus** combinatorially is,

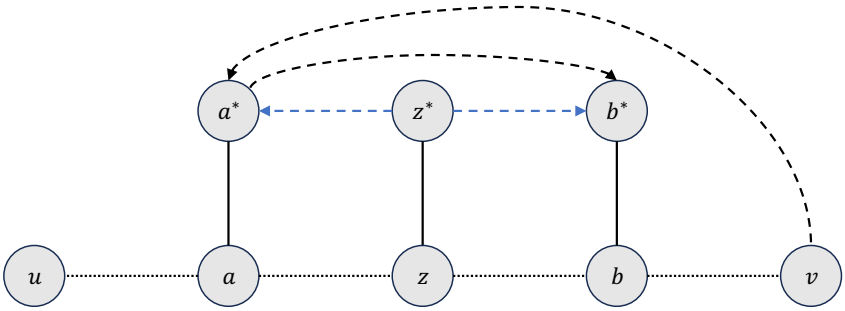


Figure 6: Computing a distance estimate between u, v. Solid black lines denote edges in G. Dotted black lines represent paths in G. Blue dashed arrows denote distance estimates used in the $(\min, +)$ product (Phase 1). Black dashed arrows denote edges $w \times V$ in the graph $G_{j,w}$ (Phase 2).

$$\left(1 - \frac{\beta - 2}{\beta + 2}x\right) + (1 - x) + \left(1 - \frac{\beta - 4}{\beta + 2}x\right) = 3 - \frac{3\beta - 4}{\beta + 2} \cdot x$$

Then, balancing this with the $2 + \frac{2x}{\beta+2}$ exponent required to execute **SparseAPASP**,

$$3 - \frac{3\beta - 4}{\beta + 2} \cdot x = 2 + \frac{2x}{\beta + 2}$$

so that,

$$x = \frac{\beta + 2}{3\beta - 2}$$

and obtain running time exponent $2 + \frac{2}{\beta+2}\frac{\beta+2}{3\beta-2} = 2 + \frac{2}{3\beta-2}$, thus recovering the approximation algorithm of Dor, Halperin, and Zwick [DHZ00]. In particular, by using fast matrix multiplication instead, we obtain a faster running time for all values of β.

Algorithm We are now ready to describe our algorithm. We begin our algorithm by initializing the distance estimate matrix as the adjacency matrix of the graph G, as well as setting parameters x, k. Then, we decompose our graph into $\lfloor k/3 \rfloor + 3$ levels according to the degree thresholds $s_{k_0}, s_{k_0+1}, \ldots, s_{k-1}$.

In Phase 1, we further decompose the top level into roughly $(1 - x) \log n$ levels. At each level, we execute **BFS** from every vertex w in the dominating set C_i on the graph (V, F_i). We construct two matrices A_i, B_i encoding the computed distances between D_{k_0+2}, C_i and C_i, D_{k_0+3} respectively.

In order to apply the **MonotoneMinPlus** algorithm of [CDXZ22], we define B_i' to be the Euler tour distance matrix of C_i on G_i (see Definition 5.8) observing that this is a row-bounded difference matrix. Then, we modify this matrix into a row-monotone matrix by adding j to each column j and taking a sub-matrix of the columns corresponding to D_{k_0+3} to obtain a monotone matrix, as discussed in Section B.2. In the pseudocode, this is contained in the invocation of **BDtoMonotone**. Then, we compute the $(\min, +)$ product of A_i, B_i using **MonotoneMinPlus**.

In Phase 2, we execute **Dijkstra** from each D_i for $i \geq k_0 + 1$. In each iteration, we search the graph $G_{j,w}$ consisting of $E_j \cup E^* \cup (w \times V)$.

We now present the algorithm and proof of Theorem 5.6.

Algorithm 6 $\textsc{AdditiveAPASP}(G, \beta)$

Input : Unweighted, undirected Graph $G = (V, E)$ with n vertices; approximation parameter β
Output: Distance estimate $\hat{\delta} : U \times V \to \mathbb{Z}$ such that $\delta(u, v) \le \hat{\delta}(u, v) \le \delta(u, v) + \beta$ for all $u, v \in V$

71 Phase 0: Set up and Decompose Graph
72 $\hat{\delta}(u, v) \leftarrow \begin{cases} 1 & (u, v) \in E \\ \infty & \text{o/w} \end{cases}$
73 $k \leftarrow \frac{3\beta-2}{2}$
74 $x \leftarrow$ solution to $\omega\left(1 - \frac{\beta-2}{\beta+2}x, 1 - x, 1 - \frac{\beta-4}{\beta+2}x\right) = 1 + \frac{4+2\beta}{\beta+2}x$
75 $s_{k-i} \leftarrow n^{i \cdot \frac{x}{\lfloor k/3 \rfloor + 2}}$ for all $1 \le i \le \lfloor k/3 \rfloor + 2$
76 $k_0 \leftarrow k - (\lfloor k/3 \rfloor + 2) = k - \lfloor k/3 \rfloor - 2$
77 $(D_{k_0}, D_{k_0+1}, \ldots, D_k), (E_{k_0}, E_{k_0+1}, \ldots, E_k), E^* \leftarrow \textsc{Decompose}(G, (s_{k_0}, s_{k_0+1}, \ldots, s_{k-1}))$
78 Phase 1: Estimate Distances on High-Degree Paths
79 $t_i \leftarrow \frac{n}{2^i}$ for all $1 \le i \le l - 1 = \lceil (1 - x) \log n \rceil$
80 $(C_1, C_2, \ldots, C_l), (F_1, F_2, \ldots, F_l), F^* \leftarrow \textsc{Decompose}(G, (t_1, t_2, \ldots, t_{l-1}))$
81 **for** $0 \le i \le l$ **do**
82 **for** $w \in C_i$ **do**
83 $X_i(w) \leftarrow \textbf{BFS}(w)$ on $G_i = (V, F_i)$
84 $\hat{\delta}(w, v) \leftarrow \min(\hat{\delta}, X_i(w, v))$ for all $v \in V$
85 Construct $|D_{k_0+2}| \times |C_i|$ matrix A_i where $A_i(v, w) = X_i(w, v)$
86 Construct $|C_i| \times (2n - 1)$ Euler tour distance matrix B_i' of C_i on G_i
87 $B_i \leftarrow \textsc{BDtoMonotone}(B_i')[C_i, D_{k_0+3}]$
88 $\hat{\delta} \leftarrow \min(\hat{\delta}(w, v), \textsc{MonotoneMinPlus}(A_i, B_i))$
89 Phase 2: Estimate Distances on Low-Degree Paths
90 **for** $k_0 + 1 \le j \le k$ **do**
91 **for** $w \in D_j$ **do**
92 $G_{j,w} \leftarrow (V, E_j \cup (u \times V) \cup E^*)$
93 $\hat{\delta} \leftarrow \textsc{Dijkstra}(G_{j,w}, w, \hat{\delta})$

Warm-Up: $\beta = 4$

We begin by showing a proof of correctness for the case of $\beta = 4$. The somewhat simplified argument in this case will be generalized to arbitrary $\beta \ge 6$ below.

Lemma 5.9. *Algorithm* $\textsc{AdditiveAPASP}(G, 4)$ *returns an estimate* $\hat{\delta}$ *such that* $\delta(u, v) \le \hat{\delta}(u, v) \le \delta(u, v) + 4$ *for all* u, v.

Proof. Let P be a shortest path between u, v. Note that $k = 5, k_0 = 2$.

Suppose $P \subset E_3 = E_{k_0+1}$. Then, $L(P, 5) \subset \{3, 4\}$ and $|L(P, 5)| \le 2$. By Lemma A.3, since $D_5 = D_k = V$, we obtain a $+4$ approximation in Phase 2.

Thus, P has some vertex with degree at least $s_2 = n^x$. Let y denote the vertex in P with degree at least s_4 closest to v. Let z be a vertex of maximum degree on P and i^* be the integer $1 \le i \le l$ such that $\frac{n}{2^{i^*}} \le \deg(z) < \frac{n}{2^{i^*-1}}$. Since $P \subset F_{i^*}$, we have in the i^*-th iteration of Phase 1 that $X_{i^*}(z^*, u) \le \delta(z, u) + 1$ and $X_{i^*}(z^*, y^*) \le \delta(z, y) + 2$ where $z^* = r(z, C_{i^*})$, $y^* = r(y, D_4)$. Therefore, after Line 88,

$$\hat{\delta}_{i^*}(u, y^*) \le X_{i^*}(u, z^*) + X_{i^*}(z^*, y^*) \le \delta(u, z) + \delta(z, y) + 3 = \delta(u, y) + 3$$

Now, in the 5-th iteration in Phase 2, when we execute Dijkstra from u, we have an edge (u, y^*) with weight at most $\delta(u, y) + 3$, an edge $(y^*, y) \in E$ and the remaining edges in $E_5 = E_k$, so that,

$$\hat{\delta}(u, v) \leq \hat{\delta}_{i^*}(u, y^*) + 1 + \delta(y, v) \leq \delta(u, y) + \delta(y, v) + 4 \leq \delta(u, v) + 4$$

□

General Case: $\beta \geq 6$

We now present the proof of Theorem 5.6.

Proof. (Correctness) Let $u, v \in V$ be a pair of vertices and P be a shortest path between u, v. Let $\deg(P) = \max_{v \in P} \deg(v)$ be the maximum degree of any vertex on P.

Note by the definition of $k \leftarrow \frac{3\beta-2}{2}$ and β is an even integer,

$$2\left(\lfloor k/3 \rfloor + 1\right) = 2\left(\left\lfloor \frac{\beta}{2} - \frac{1}{3} \right\rfloor + 1\right) \leq \beta$$

By Lemma 5.10, we only need to show $\hat{\delta}(u, v) \leq \delta(u, v) + 2(\lfloor k/3 \rfloor + 1)$.

Recall that $V_s = \{v \in V \text{ s.t. } \deg(v) \geq s\}$ from Definition 2.3.

Case 1: $\deg(P) < s_{k_0}$ In this case, we have $P \subset E_{k_0+1}$, and we may follow the proof of **SparseAPASP** by Dor, Halperin, and Zwick [DHZ00]. By assumption $L(P, k) \subset \{k_0 + 1, k_0 + 2, \ldots, k - 1\}$ so that $|L(P, k)| \leq \lfloor k/3 \rfloor + 1$. From Lemma A.3, it immediately follows that $\hat{\delta}(u, v) \leq 2(\lfloor k/3 \rfloor + 1)$.

Case 2: $\deg(P) \geq s_{k_0}$ By assumption, there is a vertex $w \in P$ with $\deg(w) \geq s_{k_0+3}$. Then, with Lemma 5.11, we have that for all $x \in P$,

$$\hat{\delta}_{k_0+3}(w^*, x) \leq \delta(w, x) + 5$$

Then, following the same argument as the inductive step of Lemma A.2, consider any vertex $w \in P$ such that $\ell(w) \geq j$ for $j > k_0 + 3$. We claim the following for all j,

$$\hat{\delta}_j(w^*, x) \leq \delta(w, x) + 2(j - (k_0 + 1)) + 1$$

where above we have shown the base case for $j = k_0 + 3$. We now proceed by induction for $j > k_0 + 3$. Consider an execution of **Dijkstra** from w^* in $G_{j,w}$. For any vertex $x \in P$, let y be the last vertex on the sub-path $P_{w,x}$ with $\ell(y) \leq j - 1$.

If no such vertex exists, then $P_{w,x} \subset E_j$ and we compute an exact distance from w^* so that $\hat{\delta}(w^*, x) \leq \delta(w, x) + 1$.

In particular, $\deg(y) \geq s_{j-1}$ so let $y^* = r(y, D_{j-1})$. We take the edges (w^*, y^*), $(y^*, y) \in E^*$, and the remaining edges in E_j. By induction, we have,

$$\begin{aligned}
\hat{\delta}_j(w^*, x) &\leq \hat{\delta}_{j-1}(w^*, y^*) + 1 + \delta(y, x) \\
&\leq \delta(w, y) + 2((j - 1) - (k_0 + 1)) + 3 + \delta(y, x) \\
&\leq \delta(w, x) + 2(j - (k_0 + 1)) + 1
\end{aligned}$$

Then, on the final iteration, let w be the closest vertex to v on P such that $\deg(w) \geq s_{k-1}$. From u, we take the edge (u, w^*), followed by $(w, w^*) \in E^*$ and the remaining path in E_k, so that,

$$\hat{\delta}(u, v) \leq \hat{\delta}_{k-1}(u, w^*) + 1 + \delta(w, v)$$

$$\leq \delta(u,w) + 2(k-1-(k_0+1)) + 2 + \delta(w,v)$$
$$\leq \delta(u,v) + 2(\lfloor k/3 \rfloor + 1)$$

completing the proof. □

Proof. (Time Complexity)

Recall that $k = O(1)$ is a constant.

Phase 0 requires $O(m)$ time, as **Decompose** requires $O(k(m+n))$-time.

We examine Phase 1. Constructing the Euler Tours and **BFS** trees require $\tilde{O}(n^2)$-time over all iterations, as $|C_i||F_i| = \tilde{O}(n^2)$. The expensive step therefore is the $(\min, +)$ product, which we compute with an invocation to **MonotoneMinPlus**. By Theorem 5.1, this requires,

$$\tilde{O}\left(n^{\frac{(a+b+\mu)+\omega(a,b,c)}{2}}\right)$$

where we apply

$$a = 1 - \frac{\lfloor k/3 \rfloor}{\lfloor k/3 \rfloor + 2}x$$
$$b = 1 - x$$
$$c = 1 - \frac{\lfloor k/3 \rfloor - 1}{\lfloor k/3 \rfloor + 2}x$$
$$\mu = 1$$

since all distances in G are bounded by n.

In Phase 2, each $|D_j| = \tilde{O}\left(n^{1-(k-j)\frac{x}{\lfloor k/3 \rfloor+2}}\right)$ and $|E_j| \leq O\left(n^{1+(k-j+1)\frac{x}{\lfloor k/3 \rfloor+2}}\right)$. Therefore, each invocation of **Dijkstra** in Phase 2 requires time $\tilde{O}\left(n^{2+\frac{x}{\lfloor k/3 \rfloor+2}}\right)$.

Finally, we balance terms to optimize,

$$\frac{\left(3 - \frac{2\lfloor k/3 \rfloor+2}{\lfloor k/3 \rfloor+2}x\right) + \omega\left(1 - \frac{\lfloor k/3 \rfloor}{\lfloor k/3 \rfloor+2}x, 1-x, 1 - \frac{\lfloor k/3 \rfloor-1}{\lfloor k/3 \rfloor+2}x\right)}{2} = 2 + \frac{x}{\lfloor k/3 \rfloor + 2}$$

To get the exact expression, we plug in $k = \frac{3\beta-2}{2}$ and in particular $\lfloor k/3 \rfloor = \frac{\beta}{2} - 1$.

$$\frac{\left(3 - \frac{2\beta}{\beta+2}x\right) + \omega\left(1 - \frac{\beta-2}{\beta+2}x, 1-x, 1 - \frac{\beta-4}{\beta+2}x\right)}{2} = 2 + \frac{2x}{\beta+2}$$
$$\left(3 - \frac{2\beta}{\beta+2}x\right) + \omega\left(1 - \frac{\beta-2}{\beta+2}x, 1-x, 1 - \frac{\beta-4}{\beta+2}x\right) = 4 + \frac{4x}{\beta+2}$$
$$\omega\left(1 - \frac{\beta-2}{\beta+2}x, 1-x, 1 - \frac{\beta-4}{\beta+2}x\right) = 1 + \frac{4+2\beta}{\beta+2}x$$

Table 2 exhibits the running times for a few values of β utilizing [Bra]. □

In the following lemma, we show that every distance estimate produced is feasible. That is, each distance estimate can be attained by some path in G.

Lemma 5.10. *Algorithm 6 returns $\hat{\delta}$ such that $\delta(u,v) \leq \hat{\delta}(u,v)$ for all $u, v \in V$.*

Proof. This holds simply by observing that every distance estimate is produced by some path in the original graph G.

Every estimate in Phase 1 found by concatenating two paths in G_i, a subgraph of G.

In Phase 2, the only edges added are of the form $u \times V$. However, the weights of these edges, if finite, are computed by a previous step as the length of some path from u to the relevant vertex $w \in V$. □

We present the key lemma for correctness below. In particular, we prove that we have a good additive approximation after the (k_0+3)-th iteration.

Lemma 5.11. *Let P be a shortest path between u, v. In Phase 2 of Algorithm 6, let w be a vertex of P with $\deg(w) \geq s_{k_0+3}$. Then, for every vertex $x \in P$, the (k_0+3)-th iteration of Phase 2 finds a path of length $\delta(w,x)+5$ from w^* to x where $w^* = r(w, D_{k_0+3})$.*

Proof. Let $P_{w,x}$ denote the sub-path of P between w and x.

Case 1: $\deg(P_{w,x}) \leq s_{k_0}$ Consider the path $P_{w,x}$. By Lemma A.3, since $\deg(w) \geq s_{k_0+3}$, we have that, $L(P_{w,x}) \subset \{k_0+1, k_0+2\}$ so that,

$$\hat{\delta}_{k_0+3}(w^*, x) \leq \delta(w,x) + 5$$

Case 2: $\deg(P_{w,x}) > s_{k_0}$ We now arrive at the key case depending on Phase 1. Let y denote the vertex in $P_{w,x}$ closest to x with degree $\deg(y) \geq s_{k_0+2}$ and z be a vertex of maximum degree on $P_{w,x}$. Then, let i^* be the integer $1 \leq i \leq l$ such that $\frac{n}{2^{i^*}} \leq \deg(z) < \frac{n}{2^{i^*-1}}$. Let $y^* = r(y, D_{k_0+2})$ and $z^* = r(z, C_{i^*})$.

In particular, since $P \subset F_{i^*}$ we have in the i^* iteration of Phase 1 that $X_{i^*}(z^*, w^*) \leq \delta(z,w)+2$ and $X_{i^*}(z^*, y^*) \leq \delta(z,y)+2$. Now, since B'_i is an Euler tour distance matrix, it is therefore a 1-bounded difference matrix by the triangle inequality. Then, the procedure **BDtoMonotone** creates a row-monotone matrix keeping values bounded in $O(n)$. Finally, B_i is a row-monotone matrix as the sub-matrix of a row-monotone matrix. Thus, we can use **MonotoneMinPlus**(A_{i^*}, B_{i^*}) to compute,

$$\hat{\delta}_{i^*}(w^*, y^*) \leq X_{i^*}(w^*, z^*) + X_{i^*}(z^*, y^*) \leq \delta(w,z) + \delta(z,y) + 4 = \delta(w,y) + 4$$

Now, in the (k_0+3)-th iteration in Phase 2, when we execute **Dijkstra** from w^*, we have an edge to y^* with weight at most $\delta(w,y)+4$. If $\deg(x) > s_{k_0+2}$, and therefore $y = x$, we take an extra edge $(x^*, x) \in E^*$ to find a path of length at most $\delta(w,y)+4+1 = \delta(w,x)+5$. Otherwise, after using the edge $(y^*, y) \in E^*$, the remaining edges of $P_{w,x}$ are in E_{k_0+3} and we can conclude,

$$\hat{\delta}_{k_0+3}(w^*, x) \leq \hat{\delta}_{i^*}(w^*, y^*) + 1 + \delta(y,x) \leq \delta(w,y) + \delta(y,x) + 5 = \delta(w,x) + 5$$

□

5.3 Additive Approximations for Constant Length Paths

In this section, we give a simpler algorithm that uses fast matrix multiplication to obtain fast algorithms for general additive approximations on small distances. We employ this result in the proof of Theorem 3.3.

Theorem 5.12. *Let $\beta \geq 4$ be an even integer and $C > 0$ be a constant. Let G be an undirected, unweighted graph with n vertices. Algorithm 7 computes $\hat{\delta}$ in time,*

$$\tilde{O}\left(n^{2+\frac{2x}{\beta+2}}\right)$$

x is the solution to,

$$\omega\left(1-\frac{\beta-2}{\beta+2}x, 1-x, 1-\frac{\beta-4}{\beta+2}x\right) = 2+\frac{2x}{\beta+2}$$

such that $\delta(u,v) \leq \hat{\delta}(u,v)$ for all u, v and whenever $\delta(u,v) \leq C$, $\hat{\delta}(u,v) \leq \delta(u,v) + \beta$.

The above result also implies a more efficient algorithm for approximating distances up to small polynomials.

Corollary 5.13. *Let $\varepsilon > 0$. Then, there is an algorithm computing $\hat{\delta}$ in time*

$$\tilde{O}\left(n^{2+\frac{2x}{\beta+2}}\right)$$

x is the solution to,

$$\omega\left(1-\frac{\beta-2}{\beta+2}x, 1-x, 1-\frac{\beta-4}{\beta+2}x\right) + \varepsilon = 2+\frac{2x}{\beta+2}$$

such that $\delta(u,v) \leq \hat{\delta}(u,v)$ for all u, v and whenever $\delta(u,v) \leq n^{\varepsilon}$, $\hat{\delta}(u,v) \leq \delta(u,v) + \beta$.

Proof. The only difference with the proof of Theorem 5.12 is the setting of x and the complexity analysis. For bounded C, the computation of the $(\min, +)$ product requires time,

$$Cn^{\omega\left(1-\frac{\beta-2}{\beta+2}x, 1-x, 1-\frac{\beta-4}{\beta+2}x\right)}$$

Now, since the entries are bounded instead by n^{ε}, we bound the computation of the $(\min, +)$ product by,

$$n^{\omega\left(1-\frac{\beta-2}{\beta+2}x, 1-x, 1-\frac{\beta-4}{\beta+2}x\right)+\varepsilon}$$

□

We now present a high level overview of Theorem 5.12.

High Level Overview The algorithm is essentially a simplification of the previous Algorithm 6. Indeed, we can invoke **BoundedMinPlus** in place of **MonotoneMinPlus**, which is a faster algorithm and does not require the transformation **BDtoMonotone**.

Algorithm Algorithm 7 is identical to Algorithm 6. The only modification is the invocation of **BoundedMinPlus** in place of **AdditiveAPASP**, as the entries to the matrices are now bounded by constant C.

Below, we exhibit some running times for specific choices of β, using [Bra].

Corollary 5.14. *We obtain the following running times for an additive approximation on distances $\delta(u,v) \leq C$.*

β	*time*
4	$n^{2.09314841}$
6	$n^{2.05794292}$
8	$n^{2.04220679}$
10	$n^{2.03322582}$

We now present the algorithm and proof of Theorem 5.12.

Warm-Up: $\beta = 4$

We begin by showing a proof of correctness for the case of $\beta = 4$. The proof is almost identical to Lemma 5.9.

Lemma 5.15. *Algorithm* $\textsc{BoundedAdditiveAPASP}(G, 4, C)$ *returns an estimate* $\hat{\delta}$ *satisfying:*

1. $\delta(u, v) \leq \hat{\delta}(u, v)$ *for all* u, v
2. $\hat{\delta}(u, v) \leq \delta(u, v) + 4$ *for all* $\delta(u, v) \leq C$

Proof. Let P be a shortest path between u, v. Note that $k = 5, k_0 = 2$.

Suppose $P \subset E_3 = E_{k_0+1}$. Then, $L(P, 5) \subset \{3, 4\}$ and $|L(P, 5)| \leq 2$. By Lemma A.3, since $D_5 = D_k = V$, we obtain a $+4$ approximation in Phase 2.

Thus, P has some vertex with degree at least $s_2 = n^x$. Let y denote the vertex in P with degree at least s_4 closest to v. Let z be a vertex of maximum degree on P and i^* be the integer $1 \leq i \leq l$ such that $\frac{n}{2^{i^*}} \leq \deg(z) < \frac{n}{2^{i^*-1}}$. Since $P \subset F_{i^*}$, we have in the i^*-th iteration of Phase 1 that $X_{i^*}(z^*, u) \leq \delta(z, u) + 1 \leq C + 1$ and $X_{i^*}(z^*, y^*) \leq \delta(z, y) + 2 \leq C + 2$ where $z^* = r(z, C_{i^*})$, $y^* = r(y, D_4)$. Therefore, since the entries in A_{i^*}, B_{i^*} are finite, we compute in Line 110,

$$\hat{\delta}_{i^*}(u, y^*) \leq X_{i^*}(u, z^*) + X_{i^*}(z^*, y^*) \leq \delta(u, z) + \delta(z, y) + 3 = \delta(u, y) + 3$$

Now, in the 5-th iteration in Phase 2, when we execute Dijkstra from u, we have an edge (u, y^*) with weight at most $\delta(u, y) + 3$, an edge $(y^*, y) \in E$ and the remaining edges in $E_5 = E_k$, so that,

$$\hat{\delta}(u, v) \leq \hat{\delta}_{i^*}(u, y^*) + 1 + \delta(y, v) \leq \delta(u, y) + \delta(y, v) + 4 \leq \delta(u, v) + 4$$

□

Algorithm 7 BoundedAdditiveAPASP(G, β, C)

Input : Unweighted, undirected Graph $G = (V, E)$ with n vertices; approximation parameter β; distance bound C

Output: Distance estimate $\hat{\delta} : U \times V \to \mathbb{Z}$ such that $\delta(u, v) \le \hat{\delta}(u, v)$ for all $u, v \in V$ and $\hat{\delta}(u, v) \le \delta(u, v) + \beta$ whenever $\delta(u, v) \le C$.

Phase 0: Set up and Decompose Graph

$\hat{\delta}(u, v) \leftarrow \begin{cases} 1 & (u, v) \in E \\ \infty & \text{o/w} \end{cases}$

$k \leftarrow \frac{3\beta - 2}{2}$

$x \leftarrow$ solution to $\omega\left(1 - \frac{\lfloor k/3 \rfloor}{\lfloor k/3 \rfloor + 2} x, 1 - x, 1 - \frac{\lfloor k/3 \rfloor - 1}{\lfloor k/3 \rfloor + 2} x\right) = 2 + \frac{x}{\lfloor k/3 \rfloor + 2}$

$s_{k-i} \leftarrow n^{i \cdot \frac{x}{\lfloor k/3 \rfloor + 2}}$ for all $1 \le i \le \lfloor k/3 \rfloor + 2$

$k_0 \leftarrow k - (\lfloor k/3 \rfloor + 2) = k - \lfloor k/3 \rfloor - 2$

$(D_{k_0}, D_{k_0+1}, \ldots, D_k), (E_{k_0}, E_{k_0+1}, \ldots, E_k), E^* \leftarrow$ **Decompose**$(G, (s_{k_0}, s_{k_0+1}, \ldots, s_{k-1}))$

Phase 1: Estimate Distances on High-Degree Paths

$t_i \leftarrow \frac{n}{2^i}$ for all $1 \le i \le l - 1 = \lceil (1 - x) \log n \rceil$

$(C_1, C_2, \ldots, C_l), (F_1, F_2, \ldots, F_l), F^* \leftarrow$ **Decompose**$(G, (t_1, t_2, \ldots, t_{l-1}))$

for $0 \le i \le l$ **do**

  for $w \in C_i$ **do**

    $X_i(w) \leftarrow$ **BFS**(w) on $G_i = (V, F_i)$

    $\hat{\delta}(w, v) \leftarrow \min(\hat{\delta}, X_i(w, v))$ for all $v \in V$

  Construct $|D_{k_0+2}| \times |C_i|$ matrix A_i where $A_i(v, w) = \begin{cases} X_i(w, v) & X_i(w, v) \le C + 2 \\ \infty & \text{o/w} \end{cases}$

  Construct $|C_i| \times |D_{k_0+3}|$ matrix B_i where $B_i(w, v) = \begin{cases} X_i(w, v) & X_i(w, v) \le C + 2 \\ \infty & \text{o/w} \end{cases}$

  $\hat{\delta} \leftarrow \min(\hat{\delta},$ **BoundedMinPlus**$(A_i, B_i, C + 2))$

Phase 2: Estimate Distances on Low-Degree Paths

for $k_0 + 1 \le j \le k$ **do**

  for $w \in D_j$ **do**

    $G_{j,w} = (V, E_j \cup (w \times V) \cup E^*)$

    $\hat{\delta} \leftarrow$ **Dijkstra**$(G_{j,w}, w, \hat{\delta})$

General Case: $\beta \ge 6$

We now present the proof of Theorem 5.12.

Proof. (Correctness) We will ignore cases that are too similar to Theorem 5.6. Note by the definition of $k \leftarrow \frac{3\beta-2}{2}$ and β is an even integer,

$$2\left(\lfloor k/3 \rfloor + 1\right) = 2\left(\left\lfloor \frac{\beta}{2} - \frac{1}{3} \right\rfloor + 1\right) \le \beta$$

By Lemma 5.16, it suffices to check that for $\delta(u, v) \le C$, we have $\hat{\delta}(u, v) \le \delta(u, v) + 2(\lfloor k/3 \rfloor + 1)$.

Let $u, v \in V$ be a pair of vertices such that $\delta(u, v) \le C$. Let P be a shortest path between u, v. Let $\deg(P) = \max_{v \in P} \deg(v)$ be the maximum degree of any vertex on P.

Case 1: $\deg(P) < s_{k_0}$ Since $L(P,k) \subset \{k_0+1,\ldots,k-1\}$, correctness holds as in Theorem 5.6.

Case 2: $\deg(P) \geq s_{k_0}$ By assumption, there is a vertex $w \in P$ with $\deg(w) \geq s_{k_0+3}$. Then, with Lemma 5.17, we have that for all $x \in P$,

$$\hat{\delta}_{k_0+3}(w^*, x) \leq \delta(w,x) + 5$$

The remaining proof follows as in Theorem 5.6. □

Proof. (Time Complexity)

As the only modifications are in Phase 1, we examine only the complexity of this phase. Each $|C_i| = \tilde{O}(\frac{n}{t_i}) = \tilde{O}(2^i)$ Each graph F_i has $O(nt_i)$ edges. Thus, each **BFS** requires $\tilde{O}(n^2)$-time. Over k iterations, this is again $\tilde{O}(n^2)$-time.

Now, $|D_{k_0+2}| = \tilde{O}\left(\frac{n}{s_{k_0+2}}\right) = \tilde{O}\left(\frac{n}{s_{k-\lfloor k/3 \rfloor}}\right) = \tilde{O}\left(n^{1-\frac{\lfloor k/3 \rfloor}{\lfloor k/3 \rfloor+2}x}\right)$. Similarly, we can upper bound the size of D_{k_0+3} as $|D_{k_0+3}| = \tilde{O}\left(\frac{n}{s_{k-(\lfloor k/3 \rfloor-1)}}\right) = \tilde{O}\left(n^{1-\frac{\lfloor k/3-1 \rfloor}{\lfloor k/3 \rfloor+2}x}\right)$. Finally, $|C_i| = \tilde{O}(2^l) = \tilde{O}(n^{1-x})$. Thus, each **BoundedMinPlus** requires time,

$$\omega\left(1 - \frac{\lfloor k/3 \rfloor}{\lfloor k/3 \rfloor + 2}x, 1-x, 1 - \frac{\lfloor k/3 \rfloor - 1}{\lfloor k/3 \rfloor + 2}x\right)$$

Recall that Phase 2 requires time $\tilde{O}\left(n^{2+\frac{x}{\lfloor k/3 \rfloor+2}}\right)$. Finally, we balance terms to optimize,

$$\omega\left(1 - \frac{\lfloor k/3 \rfloor}{\lfloor k/3 \rfloor + 2}x, 1-x, 1 - \frac{\lfloor k/3 \rfloor - 1}{\lfloor k/3 \rfloor + 2}x\right) = 2 + \frac{x}{\lfloor k/3 \rfloor + 2}$$

To get the exact expression, we plug in $k = \frac{3\beta-2}{2}$ and in particular $\lfloor k/3 \rfloor = \frac{\beta}{2} - 1$. □

In the following lemma, we show that every distance estimate produced is feasible. That is, each distance estimate can be attained by some path in G.

Lemma 5.16. *Algorithm 7 returns $\hat{\delta}$ such that $\delta(u,v) \leq \hat{\delta}(u,v)$ for all $u,v \in V$.*

Proof. This holds simply by observing that every distance estimate is produced by some path in the original graph G, as argued in Lemma 5.10. □

We present the key lemma for correctness below. In particular, we prove that we have a good additive approximation after the (k_0+3)-th iteration.

Lemma 5.17. *Let P be a shortest path between u,v. In Phase 2 of Algorithm 7, let w be a vertex of P in V_{k_0+3}. Then, for every vertex $x \in P$, the (k_0+3)-th iteration of Phase 2 finds a path of length $\delta(w,x)+5$ from w^* to x.*

Proof. Let $P_{w,x}$ denote the sub-path of P between w and x.

Case 1: $\deg(P_{w,x}) \leq s_{k_0}$ This is identical to Case 1 of Lemma 5.11.

Case 2: $\deg(P_{w,x}) > s_{k_0}$ We now arrive at the key case depending on Phase 1. Let y denote the vertex in $P_{w,x}$ closest to x with degree $\deg(y) \geq s_{k_0+2}$ and z be a vertex of maximum degree on $P_{w,x}$. Then, let i^* be the integer $1 \leq i \leq l$ such that $\frac{n}{2^{i^*}} \leq \deg(z) < \frac{n}{2^{i^*-1}}$. Let $y^* = r(y, D_{k_0+2})$ and $z^* = r(z, C_{i^*})$.

Since $P \subset F_{i^*}$, in the i^* iteration of Phase 1 $X_{i^*}(z^*, w^*) \leq \delta(z, w) + 2 \leq C + 2$ and $X_{i^*}(z^*, y^*) \leq \delta(z, y) + 2 \leq C + 2$. Since the entries are finite, we use $\textsc{BoundedMinPlus}(A_{i^*}, B_{i^*})$ to compute,

$$\hat{\delta}_{i^*}(w^*, y^*) \leq X_{i^*}(w^*, z^*) + X_{i^*}(z^*, y^*) \leq \delta(w, z) + \delta(z, y) + 4 = \delta(w, y) + 4$$

Now, in the $(k_0 + 3)$-th iteration in Phase 2, when we execute Dijkstra from w^*, we have an edge to y^* with weight at most $\delta(w, y) + 4$. If $\deg(x) > s_{k_0+2}$, and therefore $y = x$, we take an extra edge $(x^*, x) \in E^*$ to find a path of length at most $\delta(w, y) + 4 + 1 = \delta(w, x) + 5$. Otherwise, after using the edge $(y^*, y) \in E^*$, the remaining edges of $P_{w,x}$ are in E_{k_0+3} and we can conclude,

$$\hat{\delta}_{k_0+3}(w^*, x) \leq \hat{\delta}_{i^*}(w^*, y^*) + 1 + \delta(y, x) \leq \delta(w, y) + \delta(y, x) + 5 = \delta(w, x) + 5$$

□

6 Weighted Approximation via Approximate $(\min, +)$ Product

In this section, we generalize our results to graphs with arbitrary non-negative weights. Let $\mathsf{wt}(e)$ denote the weight of an edge $e \in E$. For any subset of edges $S \subset E$, let $\mathsf{wt}(S) = \sum_{e \in S} \mathsf{wt}(e)$ denote the total weight of edges in S. Throughout, we assume all weights are non-negative.

Preliminaries We first require some preliminary results analogous to unweighted graphs for computing dominating sets, and degree decompositions. Note that this decomposition is slightly different than what is used in previous sections.

Lemma 6.1. *Let G be a weighted, undirected graph on n vertices. For a given edge e incident to vertex v, let $\mathrm{rank}_v(e)$ be the order of e among all edges incident to v when ranked in increasing order of weight. For example, if e is the lightest edge incident to v, $\mathrm{rank}_v(e) = 1$. For a neighbor $w \in N(v)$, we also denote $\mathrm{rank}_v(w) = \mathrm{rank}_v(v, w)$ to be the rank of the edge (v, w).*

Given thresholds $s_1 > s_2 > \ldots > s_{k-1}$, there is an algorithm $\textsc{WeightedDecompose}$ that outputs edge sets $\{E_i\}_{i=1}^k$, edge set E^*, and vertex sets $\{D_i\}_{i=1}^k$ satisfying,

1. *$E_i = \{(u, v) \in E \text{ s.t. } \min(\mathrm{rank}_u(u, v), \mathrm{rank}_v(u, v)) < s_{i-1}\}$.*
2. *D_i dominates $V_{s_i} = \{v \in V \text{ s.t. } \deg(v) \geq s_i\}$ and $|D_i| = \tilde{O}\left(\frac{n}{s_i}\right)$*
3. *$E^* = \bigcup_{i=1}^k E_i^*$ where each $E_i^* \subset E$ has for every $v \in V_{s_i}$ at least one edge $(v, w) \in E_i^*$ where $w \in D_i$ and $\mathrm{rank}_v(v, w) < s_i$.*

Furthermore, $\textsc{WeightedDecompose}$ runs in $\tilde{O}(kn^2)$-time.

We define the level of a vertex and edge analogously as in the unweighted case. For a given vertex $v \in V$, define the **level of** *v, denoted $\ell(v)$, as the integer i such that $s_i \leq \deg(v) < s_{i-1}$. For a given edge $e \in E$, define the* **level of** *e, denoted $\ell(e)$, as the integer i such that $e \in E_i \setminus E_{i+1}$.*

6.1 $(1 + \varepsilon, 2w)$ Approximation

For a given path P, let $h(P)$ denote the weight of the heaviest edge in P. For a pair of vertices u, v, let $w_{u,v}$ denote the minimum $h(P)$ over all shortest paths P between u, v.

For any integer $k \geq 1$, let $h(P,k)$ denote the weight of the k-th heaviest edge in P and $\overline{h}(P,k)$ denote the sum of the weights of the k heaviest edges in P. We analogously define $w_{u,v}(k) = \min_P \overline{h}(P,k)$ over shortest paths P between u, v.

Using combinatorial methods, Cohen and Zwick [CZ01] gave a $O(n^{7/3})$ algorithm for $+2w_{u,v}$ approximation. They in fact prove a stronger result and gain a $(7/3, 0)$ multiplicative approximation. With Fast Matrix Multiplication, Berman and Kasiviswanathan [BK07] gave a $(1+\varepsilon, 2w_{u,v})$ algorithm in time $O(n^{2.25})$. In this work, we use our techniques to improve upon their algorithm and design a $(1+\varepsilon, 2w_{u,v})$ algorithm with running time roughly $\tilde{O}(n^{2.152})$.

Theorem 6.2. *Let $\varepsilon > 0$. Let G be an undirected, weighted graph with n vertices and weight function $w : E \to \mathbb{R}^+$. Algorithm 8 computes $\hat{\delta}$ in time,*

$$\tilde{O}\left(\frac{n^{2.15195331}}{\varepsilon}\right)$$

such that $\delta(u,v) \leq \hat{\delta}(u,v) \leq (1+\varepsilon)\delta(u,v) + 2w_{u,v}$.

Throughout, we use the approximate $(\min, +)$ product of Bringmann et al. [BKW19].

Theorem B.12. *Let A, B be two integer $n \times n$ matrices with non-negative entries. Let $\varepsilon > 0$. There is an algorithm* **ApproximateMinPlus** *that returns C in time $\tilde{O}(n^\omega/\varepsilon)$ such that for all $1 \leq i, j \leq n$,*

$$(A * B)_{ij} \leq C_{ij} \leq (1+\varepsilon)(A * B)_{ij}$$

We provide the proof in Appendix B.2 for completeness.

High Level Overview Let x be a parameter to be set later. Let $s_1 = n^x$ and $s_2 = n^{x/2}$ and $D_1 \subset D_2 \subset D_3 = V$ and $E_3 \subset E_2 \subset E_1 = E$ be the output of **WeightedDecompose**. There are 3 cases to consider. Let P be a shortest path between u, v. If $P \subset E_3$, then we compute an exact distance from u in the graph $G \subset G_{3,u}$.

Suppose $P \subset E_2$ but $P \not\subset E_3$. Let $e = (a, b)$ be the closest edge to u such that $e \notin E_3$ and let a be the vertex closer to u so that $P_{a,u} \subset E_3$. Then, $\deg(a) \geq s_2$ so there is $a^* = r(a, D_2)$ such that $\text{rank}_a(a^*) < s_2$. Since $P \subset E_2$, $\hat{\delta}(a^*, v) \leq \mathsf{wt}(a^*, a) + \delta(a, v)$. Then, $(v, a^*, a) \circ P_{a,u} \subset G_{3,v}$ so **Dijkstra** computes an estimate,

$$\hat{\delta}(v,u) \leq \hat{\delta}(v,a^*) + \mathsf{wt}(a^*,a) + \delta(a,u) \leq \delta(v,u) + 2\mathsf{wt}(a^*,a) \leq \delta(v,u) + 2\mathsf{wt}(a,b) \leq \delta(v,u) + 2w_{u,v}$$

as $\text{rank}_a(a^*) < s_2 \leq \text{rank}_a(b)$.

Finally, suppose $P \not\subset E_2$. Let $P \subset F_i$ and $P \not\subset F_{i+1}$. Let $e = (a, b) \in F_i \setminus F_{i+1}$. Since $P \subset F_i$,

$$\hat{\delta}(a^*, u) \leq \mathsf{wt}(a^*, a) + \delta(a, u)$$
$$\hat{\delta}(a^*, v) \leq \mathsf{wt}(a^*, a) + \delta(a, v)$$

Then, from the **ApproximateMinPlus** product,

$$\hat{\delta}(u,v) \leq \left(1+\frac{\varepsilon}{3}\right)(\delta(u,v) + \mathsf{wt}(a^*,a)) \leq \left(1+\frac{\varepsilon}{3}\right)(\delta(u,v) + 2\mathsf{wt}(a,b)) \leq (1+\varepsilon)\delta(u,v) + 2w_{u,v}$$

where $\text{rank}_a(a^*) < t_j < t_{j-1} \leq \text{rank}_a(b)$.

Algorithm 8 WEIGHTED2ADDITIVEAPASP(G)

Input : Weighted, undirected Graph $G = (V, E)$ with n vertices

Output: Distance estimate $\hat{\delta} : U \times V \to \mathbb{Z}$ such that $\delta(u, v) \leq \hat{\delta}(u, v) \leq \delta(u, v) + 2w_{u,v}$ for all $u, v \in V$

Phase 0: Set up and Decompose Graph

$\hat{\delta}(u, v) \leftarrow \begin{cases} 1 & (u, v) \in E \\ \infty & \text{o/w} \end{cases}$

$x \leftarrow$ solution to $\omega(1, 1 - x, 1) = 2 + \frac{x}{2}$

$s_1, s_2 \leftarrow n^x, n^{\frac{x}{2}}$

$(D_1, D_2, D_3), (E_1, E_2, E_3), E^* \leftarrow$ WEIGHTEDDECOMPOSE$(G, (s_1, s_2))$

Phase 1: Estimate Distances on High-Degree Paths

$t_i \leftarrow \frac{n}{2^i}$ for all $1 \leq i \leq l - 1 = \lceil (1 - x) \log n \rceil$

$(C_1, C_2, \ldots, C_l), (F_1, F_2, \ldots, F_l), F^* \leftarrow$ WEIGHTEDDECOMPOSE$(G, (t_1, t_2, \ldots, t_{l-1}))$

for $0 \leq i \leq l$ **do**

  for $w \in C_i$ **do**

    $X_i(w) \leftarrow$ DIJKSTRA(w) on $G_i = (V, F_i)$

    $\hat{\delta}(w, v) \leftarrow \min(\hat{\delta}, X_i(w, v))$ for all $v \in V$

  Construct $V \times C_i$ matrix A_i where $A_i(v, w) = X_i(w, v)$

  $\hat{\delta} \leftarrow \min\left(\hat{\delta}, \text{APPROXIMATEMINPLUS}\left(A_i, A_i^T, 1 + \frac{\varepsilon}{3}\right)\right)$

Phase 2: Estimate Distances on Low-Degree Paths

for $2 \leq j \leq 3$ **do**

  for $w \in D_j$ **do**

    $G_{j,w} = (V, E_j \cup (u \times V) \cup E^*)$

    $\hat{\delta} \leftarrow$ DIJKSTRA$(G_{j,w}, w, \hat{\delta})$

Proof. (Correctness) We follow a similar proof to that of Cohen and Zwick [CZ01]. Fix a pair of vertices u, v and let P be a shortest path between u, v.

Case 1: $P \not\subset E_2$ Let $e_0 \in P$ be edge such that $e_0 \notin E_2$. Let $1 \leq j \leq l - 1$ such that $P \subset F_j$ but $P \cap (F_j \setminus F_{j+1}) = \emptyset$. Such a j must exist as $F_1 = E$ and F_l contains edges with rank at most $t_{l-1} = \frac{n}{2^{l-1}} \leq n^x$ so $e_0 \notin F_l$.

In particular, we now choose a specific $e = (a, b)$ such that $e \in F_j \setminus F_{j+1}$. By the correctness of WEIGHTEDDECOMPOSE, $\text{rank}_a(e), \text{rank}_b(e) \geq t_j$. Since both a, b have degree at least t_j, let $x \in D_j$ such that $(x, a) \in E_j^*$ so that $\text{rank}_a(x, a) < t_j \leq \text{rank}_a(e)$. Now, since $(x, a), P \subset F_j$, when we execute DIJKSTRA from x, we have

$$\hat{\delta}(x, u) \leq \delta(a, u) + \mathsf{wt}(x, a) \leq \delta(a, u) + \mathsf{wt}(a, b)$$
$$\hat{\delta}(x, v) \leq \delta(a, v) + \mathsf{wt}(x, a) \leq \delta(a, v) + \mathsf{wt}(a, b)$$

Then, from the APPROXIMATEMINPLUS product,

$$\hat{\delta}(u, v) \leq \left(1 + \frac{\varepsilon}{3}\right)\left(\hat{\delta}(x, u) + \hat{\delta}(x, v) + 2h(P)\right) \leq (1 + \varepsilon)\delta(u, v) + 2h(P)$$

where the second inequality follows as $\frac{2}{3}w_{u,v} \leq \frac{2}{3}\delta(u, v)$.

Case 2: $P \subset E_2$ **and** $P \not\subset E_3$ Let $e = (a, b) \in E_2 \setminus E_3$ be the closest such edge to v so that $\text{rank}_a(e), \text{rank}_a(b) \geq s_2$. Let b be the closest vertex to v. Then, there is $x \in D_2$ such that $(x, b) \in E_2^*$ and $\text{rank}_b(x, b) < s_2 \leq \text{rank}_b(e)$. When we execute **Dijkstra** from x, since $P \subset E_2$, we find,

$$\hat{\delta}(x, u) \leq \delta(b, u) + \mathsf{wt}(x, b) \leq \delta(b, u) + \mathsf{wt}(a, b)$$

Then, when we execute **Dijkstra** from u, we use the edge (u, x) and obtain,

$$\hat{\delta}(u, v) \leq \hat{\delta}(x, u) + \mathsf{wt}(x, b) + \delta(b, v) \leq \delta(u, v) + 2\mathsf{wt}(a, b) \leq \delta(u, v) + 2h(P)$$

Case 3: $P \subset E_3$ Note $D_3 = V$. In this case, $P \subset E_3$ so that when we execute **Dijkstra** from $u \in D_3$ we find exactly the path P and therefore $\hat{\delta}(u, v) \leq \delta(u, v)$.

In all cases, we obtain an approximation $\hat{\delta}(u, v) \leq (1 + \varepsilon)\delta(u, v) + 2h(P)$ for some shortest path P. We complete the proof by taking the minimum over all shortest paths P. □

Proof. (Time Complexity)

The analysis of time complexity is extremely similar to that of Deng et al. [DKR+22].

From Lemma 6.1, both calls to **WeightedDecompose** require time $\tilde{O}(n^2)$. In Phase 1, for a fixed $i \in [l]$, we can bound $|C_i| = \tilde{O}(t_i)$ and each graph F_i has at most $O\left(\frac{n^2}{t_i}\right)$ edges so that all invocations of **Dijkstra** require $\tilde{O}(n^2)$ time over all l. Since $|C_i| \leq \tilde{O}(t_i) = \tilde{O}(n^{1-x})$, the invocations of **ApproximateMinPlus** requires $\tilde{O}\left(\frac{n^{\omega(1,1-x,1)}}{\varepsilon}\right)$ time.

In Phase 2, $|D_2| = \tilde{O}(n^{1-x/2}), |E_2| = O(n^{1+x})$ and $|D_3| = \tilde{O}(n^{1-x}), |E_3| = O(n^{1+x/2})$ so that both invocations of **Dijkstra** require $\tilde{O}(n^{2+x/2})$ time.

Thus, we solve for x as the solution to,

$$\omega(1, 1 - x, 1) = 2 + \frac{x}{2}$$

Using [Bra], we choose $x = 0.30390661$ and obtain the running time $\tilde{O}(n^{2.15195331})$. □

6.2 $(1 + \varepsilon, \beta w)$ Approximation for $\beta \geq 4$

We now extend Algorithm 6 to handle weighted graphs, proving the following analogue of Theorem 5.6.

Theorem 6.3. *Let $\beta \geq 2$ be an integer and $\varepsilon > 0$. Let G be an undirected, unweighted graph with n vertices. Algorithm 9 computes $\hat{\delta}$ such that $\delta(u, v) \leq \hat{\delta}(u, v) \leq (1 + \varepsilon)\delta(u, v) + 2w_{u,v}(\beta)$ in time $\tilde{O}\left(\frac{n^{2+\frac{x}{\beta+1}}}{\varepsilon}\right)$ where x is the solution to, $\omega\left(1 - \frac{\beta-1}{\beta+1}x, 1 - x, 1 - \frac{\beta-2}{\beta+1}x\right) = 2 + \frac{x}{\beta+1}$. Here, $w_{u,v}(\beta)$ denotes the total weight of the β heaviest edges of a shortest path P.*

High Level Overview The weighted approximation differs from **AdditiveAPASP** in two key points. First, we use a weighted decomposition of the graph computed by **WeightedDecompose** in place of **Decompose**. Second, we compute an *approximate* $(\min, +)$ product with in place of an *exact* $(\min, +)$ product.

Whereas **Decompose** returns edge sets E_i which require that at least one endpoint is of low degree, **WeightedDecompose** returns edge sets E_i which contain the s_{i-1} edges of least weight incident to each vertex. In particular, for a given vertex v of level j, there is an edge (v, v^*) to $v^* \in D_j$ that is among the s_{j-1} lightest edges incident to v. This ensures that when we encounter

a blocking vertex, the edge that we take from its dominating vertex is lighter than the edge on the path we would have traversed, ensuring that the additive error incurred is at most twice the weight of some edge in the path. Since the blocking vertices are distinct, this allows us to bound our overall error in terms of the weights of the k heaviest edges on the path P.

However, since weights can be large and not necessarily bounded by $O(n)$, in order to compute a fast $(\min,+)$ product, we must make a sacrifice in the accuracy of the $(\min,+)$ product and settle for an approximate product. By choosing an appropriate error parameter ε, we can compute an efficient approximate $(\min,+)$ product while incurring only a $(1+\varepsilon)$-multiplicative factor in our approximation.

Analysis Following Theorem 5.6, we exhibit an $(1+\varepsilon, 2w_{u,v}(2))$ as an example, and note that the general algorithm for $\beta \geq 2$ follows similarly.

For $\beta = 2$, we set $k = 5$ and $k_0 = 2$. If $P \subset E_3$, then the **SparseAPASP** algorithm obtains a $+2w_{u,v}(2)$ approximation following Lemma 6.4. Suppose therefore $P \not\subset E_3$. As in the unweighted case, we decompose G into $\log n$ levels with degree thresholds $t_i = \frac{n}{2_i}$ and compute dominating sets C_i of size $\tilde{O}(n/t_i)$ and edge sets F_i of size $O(nt_{i-1})$ so that all invocations of **Dijkstra** require $\tilde{O}(n^2)$ time.

Let $P \subset F_j$ but $P \not\subset F_{j+1}$ and let $e = (a,b) \in F_j \setminus F_{j+1}$. Then, $a^* = r(a, C_j)$ and $\text{rank}_a(a^*) \leq t_j$. Let $e' = (c,d)$ be the edge closest to u such that $e' \notin E_5$ and c the endpoint closest to u. Let $c^* = r(c, D_4)$ so $\text{rank}_c(c^*) < s_4$. In the overview, we will assume $e' \neq e$ as otherwise we obtain a $+2w_{u,v}$ approximation.

$$\hat{\delta}(a^*, c^*) \leq \mathsf{wt}(a^*, a) + \delta(a, c) + \mathsf{wt}(c, c^*)$$
$$\hat{\delta}(a^*, v) \leq \mathsf{wt}(a^*, a) + \delta(a, v)$$

Then, the approximate $(\min,+)$ product computes,

$$\hat{\delta}(v, c^*) \leq \left(1 + \frac{\varepsilon}{3}\right)(\delta(v, c) + 2\mathsf{wt}(a^*, a) + \mathsf{wt}(c, c^*))$$

Then, $(v, c^*, c) \circ P_{c,u} \subset G_{5,u}$ so that **Dijkstra** computes,

$$\begin{aligned}
\hat{\delta}(v, u) &\leq \hat{\delta}(v, c^*) + \mathsf{wt}(c^*, c) + \delta(c, u) \\
&\leq \left(1 + \frac{\varepsilon}{3}\right)(\delta(v, c) + 2\mathsf{wt}(a^*, a) + \mathsf{wt}(c, c^*)) + \mathsf{wt}(c^*, c) + \delta(c, u) \\
&\leq \left(1 + \frac{\varepsilon}{3}\right)(\delta(v, u) + 2\mathsf{wt}(a^*, a) + 2\mathsf{wt}(c, c^*)) \\
&\leq \left(1 + \frac{\varepsilon}{3}\right)\left(\delta(v, u) + 2\mathsf{wt}(e) + 2\mathsf{wt}(e')\right) \\
&\leq (1+\varepsilon)\delta(u, v) + 2w_{u,v}(2)
\end{aligned}$$

We note that the general argument follows from a similar adaptation to Theorem 5.6.

Algorithm Algorithm 9 closely follows Algorithm 6. Observe that the only differences are the usage of **WeightedDecompose** in place of **Decompose** and the usage of **ApproximateMinPlus** in place of **MonotoneMinPlus**.

We present the algorithm pseudocode and detailed analysis below.

Algorithm 9 $\textsc{WeightedAdditiveAPASP}(G, \beta)$

Input : Unweighted, undirected Graph $G = (V, E)$ with n vertices; approximation parameter β
Output: Distance estimate $\hat{\delta} : U \times V \to \mathbb{Z}$ such that $\delta(u, v) \le \hat{\delta}(u, v) \le \delta(u, v) + \beta$ for all $u, v \in V$

Phase 0: Set up and Decompose Graph

$\hat{\delta}(u, v) \leftarrow \begin{cases} 1 & (u, v) \in E \\ \infty & \text{o/w} \end{cases}$

$k \leftarrow 3\beta - 1$

$x \leftarrow$ solution to $\omega\left(1 - \frac{\lfloor k/3 \rfloor}{\lfloor k/3 \rfloor + 2}x, 1 - x, 1 - \frac{\lfloor k/3 \rfloor - 1}{\lfloor k/3 \rfloor + 2}x\right) = 2 + \frac{x}{\lfloor k/3 \rfloor + 2}$

$s_{k-i} \leftarrow n^{i \cdot \frac{x}{\lfloor k/3 \rfloor + 2}}$ for all $1 \le i \le \lfloor k/3 \rfloor + 2$

$k_0 \leftarrow k - (\lfloor k/3 \rfloor + 2) = k - \lfloor k/3 \rfloor - 2$

$(D_{k_0}, D_{k_0+1}, \ldots, D_k), (E_{k_0}, E_{k_0+1}, \ldots, E_k), E^* \leftarrow \textsc{WeightedDecompose}(G, (s_{k_0}, s_{k_0+1}, \ldots, s_{k-1}))$

Phase 1: Estimate Distances on High-Degree Paths

$t_i \leftarrow \frac{n}{2^i}$ for all $1 \le i \le l - 1 = \lceil (1 - x) \log n \rceil$

$(C_1, C_2, \ldots, C_l), (F_1, F_2, \ldots, F_l), F^* \leftarrow \textsc{WeightedDecompose}(G, (t_1, t_2, \ldots, t_{l-1}))$

for $0 \le i \le l$ **do**

  for $w \in C_i$ **do**

    $X_i(w) \leftarrow \textsc{Dijkstra}(w)$ on $G_i = (V, F_i)$

    $\hat{\delta}(w, v) \leftarrow \min(\hat{\delta}, X_i(w, v))$ for all $v \in V$

  Construct $|D_{k_0+2}| \times |C_i|$ matrix A_i where $A_i(v, w) = X_i(w, v)$

  Construct $|C_i| \times |V|$ matrix B_i where $B_i(w, v) = X_i(w, v)$

  $\hat{\delta} \leftarrow \min\left(\hat{\delta}, \textsc{ApproximateMinPlus}\left(A_i, B_i, 1 + \frac{\varepsilon}{3}\right)\right)$

Phase 2: Estimate Distances on Low-Degree Paths

for $k_0 + 1 \le j \le k$ **do**

  for $w \in D_j$ **do**

    $G_{j,w} = (V, E_j \cup (u \times V) \cup E^*)$

    $\hat{\delta} \leftarrow \textsc{Dijkstra}(G_{j,w}, w, \hat{\delta})$

To verify the correctness of our algorithm, we will require the following analog of Lemma A.3. Note that this is equivalent to Lemma 5.2 in Cohen and Zwick [CZ01].

Lemma 6.4. *Let G be an undirected, weighted graph with n vertices and m edges and weight function $w : E \to \mathbb{R}$. Let $n = s_{k_0} > s_{k_0+1} > \ldots > s_{k-1} > s_k = 0$ be degree thresholds.*

For $k_0 + 1 \le j \le i$, let δ_j denote the estimate after the j-th iteration. Let P be a path between $u \in D_j$ and $v \in V$ such that $P \subset E_{k_0+1}$. Define the set $L(P, j) = \{k_0 + 1 \le i \le j - 1 \text{ s.t. } P \cap (E_i \setminus E_{i+1}) \ne \emptyset\}$. Then, there is a set $\mathcal{E}_j \subset P$ such that $|\mathcal{E}_j| \le |L(P, j)|$, $\mathcal{E}_j \cap E_j = \emptyset$, and $\delta_j(u, v) \le \mathsf{wt}(P) + 2\mathsf{wt}(\mathcal{E}_j)$.

Proof. We proceed by induction. In the base case, let $|L(P, j)| = 0$ so that $P \subset E_j$. In particular, since $u \in D_j$, when we execute $\textsc{Dijkstra}$ in the j-th iteration, we find P so that $\hat{\delta}_j(u, v) = \mathsf{wt}(P)$.

Now, let $|L(P, j)| > 0$. Let e_0 be the closest edge to v not in E_j. Then, $e_0 \in E_l \setminus E_{l+1}$ for some $l < j$. Let w be the endpoint of e_0 closest to v. Since $\deg(w) \ge s_l$, there is some $w^* = r(w, D_l) \in D_l$ such that $\text{rank}_w(w^*, w) < s_l$ so that $(w^*, w) \in E_{l+1} \subset E_l$. In particular, $\text{rank}_w(w^*, w) < s_l < \text{rank}_w(e_0)$. If we consider the sub-path $P_{u,w} \subset P$, we have $L(P_{u,w}, l) \subset L(P, j) \cap \{k_0 + 1, k_0 + 2, \ldots, l - 1\}$. Now, applying the inductive hypothesis to the path $P_{u,w} \cup (w, w^*)$,

$$\hat{\delta}_l(w^*, u) \le w(P_{u,w}) + \mathsf{wt}(w^*, w) + 2\mathsf{wt}(\mathcal{E}_l)$$

where $\mathcal{E}_l \subset P_{u,w} \cup (w, w^*)$, $|\mathcal{E}_l| \leq |L(P_{u,w} \cup (w, w^*), l)|$ and $\mathcal{E}_l \cap E_l = \emptyset$. Note since $(w, w^*) \in E_l$, $L(P_{u,w}, l) = L(P_{u,w} \cup (w, w^*), l)$.

Now, in the j-th iteration, we use the path $(u, w^*), (w^*, w)$ and the remaining sub-path $P_{w,v} \subset E_j$ so that,

$$\hat{\delta}_j(w^*, u) \leq \mathsf{wt}(P_{u,w}) + 2\mathsf{wt}(\mathcal{E}_l) + 2\mathsf{wt}(w^*, w) + \mathsf{wt}(P_{w,v}) \leq \mathsf{wt}(P) + 2\mathsf{wt}(\mathcal{E}_l) + 2\mathsf{wt}(e_0)$$

Note $e_0 \subset E_l$ so that $e_0 \notin \mathcal{E}_l$ and we can define $\mathcal{E}_j = \mathcal{E}_l \cup \{e_0\}$. Finally, it is easy to check $|\mathcal{E}_j| = |\mathcal{E}_l| + 1 \leq |L(P_{u,w}, l)| + 1 \leq |L(P, j)|$ as $l \in L(P, j) \setminus L(P_{u,w}, l)$. □

We are now ready to prove Theorem 6.3.

Proof. (Correctness)

Recall that $\beta \geq 2$. We prove the following for all $j \geq k_0 + 3$. Let $\hat{\delta}_j$ be the estimate $\hat{\delta}$ after the j-th iteration of Phase 2 in Algorithm 9. For $u \in D_j, v \in V$ and P a shortest path from u to v,

$$\hat{\delta}_j(u, v) \leq \delta(u, v) + 2\mathsf{wt}(\mathcal{E}_j)$$

where $\mathcal{E}_j \subset P$, $\mathcal{E}_j \cap E_j = \emptyset$ and $|\mathcal{E}_j| \leq j - (k_0 + 1)$. From Lemma 6.5, we have proved the base case for $j = k_0 + 3$. Let $j > k_0 + 3$ and $e = (a, b)$ be the closest edge in P to v not in E_j and b the endpoint closer to v.

Then, $\mathrm{rank}_a(e), \mathrm{rank}_b(e) \geq s_{j-1}$ and $j \geq k_0 + 3$. In particular, $\deg(b) \geq s_{j-1}$ so there is a vertex $b^* = r(b, D_{j-1})$ satisfying $\mathrm{rank}_b(b, b^*) < s_{j-1}$ and therefore $\mathsf{wt}(b, b^*) \leq \mathsf{wt}(e)$ and $(b^*, b) \in E_j$, By induction,

$$\hat{\delta}_{j-1}(b^*, u) \leq \delta(b^*, u) + 2\mathsf{wt}(\mathcal{E}_{j-1})$$

Then,

$$\hat{\delta}_j(u, v) \leq \hat{\delta}_{j-1}(b^*, u) + \mathsf{wt}(b^*, b) + \delta(b, v) \leq \delta(u, b) + \delta(b, v) + 2\mathsf{wt}(\mathcal{E}_{j-1}) + 2\mathsf{wt}(b, b^*)$$

Thus, define $\mathcal{E}_j = \mathcal{E}_{j-1} \cup \{(b, b^*)\}$ which satisfies the desired conditions since $(b, b^*) \in E_j \subset E_{j-1}$ and is therefore not in $\mathcal{E}_{j-1}$. Furthermore, with induction, $|\mathcal{E}_j| \leq j - (k_0 + 1)$. In particular, for $j = k$, since $D_k = V$,

$$\hat{\delta}(u, v) \leq \delta(u, v) + 2\mathsf{wt}(\mathcal{E}_k) \leq \delta(u, v) + 2\overline{h}(P, \lfloor k/3 \rfloor + 1) = \delta(u, v) + 2\overline{h}(P, \beta)$$

as desired, since $k - (k_0 + 1) = \lfloor k/3 \rfloor + 1$. □

Proof. (Time Complexity)

Phase 0 requires $O(m)$ time, as **Decompose** requires $O(k(m + n))$-time.

We now examine Phase 1. Each $|C_i| = \tilde{O}\left(\frac{n}{t_i}\right) = \tilde{O}(2^i)$ Each graph F_i has $O(nt_i)$ edges. Thus, each **Dijkstra** requires $\tilde{O}(n^2)$-time. Over k iterations, this is again $\tilde{O}(n^2)$-time.

Now, $|D_{k_0+2}| = \tilde{O}\left(\frac{n}{s_{k_0+2}}\right) = \tilde{O}\left(\frac{n}{s_{k-\lfloor k/3 \rfloor}}\right) = \tilde{O}\left(n^{1-\frac{\lfloor k/3 \rfloor}{\lfloor k/3 \rfloor + 2}x}\right)$. Similarly, we can upper bound the size of D_{k_0+3} as $|D_{k_0+3}| = \tilde{O}\left(\frac{n}{s_{k-(\lfloor k/3 \rfloor - 1)}}\right) = \tilde{O}\left(n^{1-\frac{\lfloor k/3 \rfloor - 1}{\lfloor k/3 \rfloor + 2}x}\right)$. Finally, $|C_i| = \tilde{O}(2^l) = \tilde{O}(n^{1-x})$. Thus, each **ApproximateMinPlus** requires time,

$$\frac{1}{\varepsilon} n^{\omega\left(1-\frac{\lfloor k/3 \rfloor}{\lfloor k/3 \rfloor + 2}x, 1-x, 1-\frac{\lfloor k/3 \rfloor - 1}{\lfloor k/3 \rfloor + 2}x\right)}$$

In Phase 2, each $|D_j| = \tilde{O}\left(n^{1-(k-j)\frac{x}{\lfloor k/3\rfloor+2}}\right)$ and $|E_j| \le O\left(n^{1+(k-j+1)\frac{x}{\lfloor k/3\rfloor+2}}\right)$. Therefore, each invocation of **Dijkstra** in Phase 2 requires time $\tilde{O}\left(n^{2+\frac{x}{\lfloor k/3\rfloor+2}}\right)$. Finally, we balance terms to optimize,

$$\omega\left(1-\frac{\lfloor k/3\rfloor}{\lfloor k/3\rfloor+2}x, 1-x, 1-\frac{\lfloor k/3\rfloor-1}{\lfloor k/3\rfloor+2}x\right) = 2+\frac{x}{\lfloor k/3\rfloor+2}$$

To get the exact expression, we plug in $k = 3\beta-1$ and in particular $\lfloor k/3\rfloor = \beta-1$. □

Lemma 6.5. *Let $u \in D_{k_0+3}$ and $v \in V$ and P be a shortest path between u, v. Then, after the (k_0+3)-th iteration of Phase 2 of Algorithm 9,*

$$\delta(u,v) \le \hat{\delta}(u,v) \le \delta(u,v) + 2\mathsf{wt}(\mathcal{E}_{k_0+3})$$

for some $\mathcal{E}_{k_0+3} \subset P$ of size at most 2 and $\mathcal{E}_{k_0+3} \cap E_{k_0+3} = \emptyset$.

Proof. We proceed by case analysis.

Case 1: $P \subset E_{k_0+1}$ By Lemma 6.4 and we have $\hat{\delta}(u,v) \le \delta_{k_0+3}(u,v) \le \delta(u,v) + 2\mathsf{wt}(\mathcal{E}_{k_0+3})$ where $|\mathcal{E}_{k_0+3}| \le |L(P, k_0+3)| \le 2$ and $\mathcal{E}_{k_0+3} \cap E_{k_0+3} = \emptyset$.

Case 2: $P \not\subset E_{k_0+1}$ Finally, suppose P contains some edge not in E_{k_0+1}. Let e_0 be the closest edge in P to v not in E_{k_0+3} and w_0 be the endpoint closer to v. Then, both endpoints of e_0 have degree at least s_{k_0+2} so that there is $w_0^* = r(w_0, D_{k_0+2})$ such that $\mathrm{rank}_w(w_0, w) < s_{k_0+2}$ and therefore $\mathsf{wt}(w_0^*, w) \le \mathsf{wt}(e_0)$.

Note that there is some $1 \le j \le l-1$ such that $P \subset F_j \setminus F_{j+1}$ as $t_l = \frac{n}{2^l} \le n^x = s_{k_0}$. Let $e = (a,b) \in P$ such that $e \in F_j \setminus F_{j+1}$. Then, $\mathrm{rank}_a(a,b), \mathrm{rank}_b(a,b) \ge t_j$. Since $\deg(a) > s_j$, there is vertex $a^* = r(a, D_j)$ such that $\mathrm{rank}_a(a,a^*) < s_j$ and $(a,a^*) \in F_{j+1} \subset F_j$. Since $P \subset F_j$, in the j-th iteration and $w_0^* \in D_{k_0+2}$,

$$X_j(a^*, u) \le \mathsf{wt}(a^*, a) + \delta(a,u) \le \delta(a,u) + \mathsf{wt}(a,b)$$
$$X_j(a^*, w_0^*) \le \mathsf{wt}(a^*, a) + \delta(a, w_0) + \mathsf{wt}(w_0, w_0^*) \le \delta(a, w_0) + \mathsf{wt}(a,b) + \mathsf{wt}(e_0)$$

From the **ApproximateMinPlus**, we have,

$$\hat{\delta}(u, w_0^*) \le \left(1+\frac{\varepsilon}{3}\right)(\delta(a,u) + \delta(a,w_0) + 2\mathsf{wt}(a,b) + \mathsf{wt}(e_0)) \le (1+\varepsilon)\delta(u,w_0) + 2\mathsf{wt}(a,b) + \mathsf{wt}(e_0)$$

If $e = e_0$, then we automatically have $X_j(w_0^*, u) \le \delta(w_0, u) + \mathsf{wt}(w_0^*, w_0) \le \delta(w_0, u) + \mathsf{wt}(e_0)$. Thus, we can assume $e \ne e_0$.

Then, when we execute **Dijkstra** from u in the (k_0+3)-th iteration, we use the edges (u, w_0^*), (w_0^*, w_0) and the remaining edges in E_{k_0+3}, so that,

$$\hat{\delta}(u,v) \le \hat{\delta}(u, w_0^*) + \mathsf{wt}(e_0) + \delta(w_0, v) \le (1+\varepsilon)\delta(u,v) + \le (1+\varepsilon)\delta(u,v) + 2(\mathsf{wt}(a,b) + \mathsf{wt}(e_0))$$

We claim $\{(a,b), e_0\}$ satisfies the desired conditions. Clearly $\mathcal{E}_{k_0+3}$ has size 2. By definition, $e_0 \notin E_{k_0+3}$. Recall $\mathrm{rank}_a(e), \mathrm{rank}_b(e) \ge t_j \ge t_l \ge \frac{n^x}{4} > s_{k_0+2}$ so that $(a,b) \notin E_{k_0+3}$. □

7 (α, β)-Approximate APSP

In Section 3, we showed the best known algorithm for $(2, 0)$-multiplicative approximation on general graphs, as well as observed that $(\frac{7}{3}, 0)$-approximation can be done in quadratic time, building on a quadratic $(2, 1)$-approximation due to Baswana and Kavitha [BK10] and Berman and Kasiviswanathan [BK07]. Furthermore, it is well known that any $(2 - \varepsilon, 0)$-approximation is as hard as Boolean Matrix Multiplication for any $\varepsilon > 0$, and therefore requires $\Omega(n^\omega)$ time. However, it is possible to obtain (α, β) approximations for $\alpha < 2$ if we allow sufficient additive error β. We will briefly describe the trade-off in this section.

7.1 (α, β)-Approximation for Unweighted Graphs

Since our algorithms on unweighted graphs generally provide additive guarantees, the limiting factors are the pairs of vertices with short distances. The following result shows the running time required to obtain an (α, β) approximation using our previous results for $\alpha < 2$.

Theorem 7.1. *Let $\beta \geq 2$ and $1 < \alpha < 2$. Let $T(\gamma)$ denote the running time of Algorithm 7 when given approximation parameter γ.*

Then, Algorithm 10 produces a (α, β)-approximate $\hat{\delta}$ such that for all u, v,

$$\delta(u, v) \leq \hat{\delta}(u, v) \leq \alpha\delta(u, v) + \beta$$

in time $\tilde{O}(T(\gamma))$ where γ is the largest even integer satisfying,

$$\gamma \leq 1 + \frac{\beta - 1}{2 - \alpha}$$

Before discussing the algorithm and proof, we give a high level overview and discuss our parameter choices.

High Level Overview Our goal in this section is to use our previous algorithms to obtain good bi-criteria approximations for the APSP Problem. The idea is that by increasing the tolerance for additive error, we can in fact obtain approximations where the multiplicative factor is less than 2.

The key component of our algorithm is running Algorithm 7. We will ensure small error on short paths by invoking **(2,1)-APASP** and ensure accuracy on long paths with **AdditiveAPASP** (or alternatively **DenseAPASP**) since we can choose an arbitrary constant k such that executing **AdditiveAPASP** or **DenseAPASP** is not the computational bottleneck. The constant C is chosen so that any path incurring the larger additive error k is long enough to ensure that the multiplicative error component α is small.

Why do we require $\beta \geq 2$? Let γ be the approximation parameter used in the invocation of Algorithm 7. Then for all paths of length $\gamma - 1 \leq \delta(u, v) \leq C$, we obtain a $+\gamma$ additive error. Therefore, for a fixed β, the paths of length $\gamma - 1$ have the worst multiplicative error. Let P be a path of length $\gamma - 1$. On such paths, we incur an error of γ and thus $\hat{\delta}(u, v) \leq 2\gamma - 1$. If $\beta = 1$, then we must have $\alpha \geq 2$, whereas we are interested in $\alpha < 2$. Thus, we require $\beta \geq 2$ in the input of our algorithm.

As a sub-routine, we will require the $(2, 1)$-approximation algorithm of Baswana and Kavitha [BK10] and Berman and Kasiviswanathan [BK07].

Lemma 7.2. *There is an algorithm **(2,1)-APASP** that runs in time $\tilde{O}(n^2)$ and outputs estimate $\hat{\delta}$ satisfying,*

$$\delta(u, v) \leq \hat{\delta}(u, v) \leq 2\delta(u, v) + 1$$

for all $u, v \in V$.

We will briefly remark that Algorithm **(2,1)-APASP** in fact returns a $(2, 0)$-approximation for paths of even length, only returning a $(2, 1)$-approximation for paths of odd length.

Algorithm 10 MultAddAPASP(G, α, β)

Input : Unweighted, undirected Graph $G = (V, E)$ with n vertices; approximation parameters α, β

Output: Distance estimate $\hat{\delta} : U \times V \to \mathbb{Z}$ such that $\delta(u, v) \leq \hat{\delta}(u, v) \leq \alpha\delta(u, v) + \beta$ for all $u, v \in V$

157 $\gamma \leftarrow$ maximum even γ such that $\gamma \leq 1 + \frac{\beta-1}{2-\alpha}$

158 $k \leftarrow$ minimum k such that AdditiveAPASP(G, k) requires $O(T(\gamma))$ time

159 $C \leftarrow \frac{k-\beta}{\alpha-1}$

160 $\hat{\delta}(u, v) \leftarrow \begin{cases} 1 & (u, v) \in E \\ \infty & \text{o/w} \end{cases}$

161 $\hat{\delta} \leftarrow \min(\hat{\delta},$ **(2,1)-APASP**$(G))$

162 $\hat{\delta} \leftarrow \min(\hat{\delta},$ BoundedAdditiveAPASP$(G, \gamma, C))$

163 $\hat{\delta} \leftarrow \min(\hat{\delta},$ AdditiveAPASP$(G, k))$

Proof. (Correctness) Let $u, v \in V$.

Case 1: $\delta(u, v) \geq C$ Then, from AdditiveAPASP,

$$\hat{\delta}(u, v) \leq \delta(u, v) + k = (\alpha - 1)C + \beta \leq \alpha\delta(u, v) + \beta$$

Case 2: $\gamma \leq \delta(u, v) \leq C$ Then, from BoundedAdditiveAPASP(G, γ, C),

$$\hat{\delta}(u, v) \leq \delta(u, v) + \gamma$$

Note that if $\beta \geq \gamma$, we have in fact already obtained a $(1, \beta)$ approximation. Thus, in the following, assume $\beta < \gamma$.

For now, assume $\delta(u, v) = \gamma$. Then,

$$\begin{aligned} \hat{\delta}(u, v) &\leq 2\delta(u, v) \\ &= (2\delta(u, v) - \beta) + \beta \\ &= \frac{2\delta(u, v) - \beta}{\delta(u, v)}\delta(u, v) + \beta \\ &= \left(2 - \frac{\beta}{\gamma}\right)\delta(u, v) + \beta \end{aligned}$$

noting that $2 - \frac{\beta}{\gamma} \leq 2 - \frac{\beta-1}{\gamma-1} \leq \alpha$ as $\beta < \gamma$. We conclude the proof by noting that for all larger paths, an additive error of γ implies an (α, β) approximation, as desired.

Case 3: $\delta(u, v) \leq \gamma - 1$ From **(2,1)-APASP**,

$$\hat{\delta}(u, v) \leq 2\delta(u, v) + 1 = 2\delta(u, v) - (\beta - 1) + \beta \leq \left(2 - \frac{\beta - 1}{\gamma - 1}\right)\delta(u, v) + \beta$$

By our choice of γ,

$$2 - \frac{\beta - 1}{\gamma - 1} \leq \alpha$$

□

Proof. (Time Complexity) Throughout, we use the fact that γ, β are constants. Then, we chose a constant k to satisfy $T'(k) = O(T(\gamma))$ where $T'(k)$ is the running time of **AdditiveAPASP** when required to produce additive error k. Since C is a function of constants, C is a constant as well. In particular, **BoundedAdditiveAPASP** requires $O(T(\gamma))$ time, **(2,1)-APASP** requires $\tilde{O}(n^2) = O(T(\gamma))$ time, and **AdditiveAPASP** requires $O(T(\gamma))$ time, proving the desired time complexity. □

We give some examples for $\gamma = 6$.

Corollary 7.3. *In time $T(6) = n^{2.06382}$ (Corollary 5.14), we can obtain any of the following approximations.*

$$\left(\frac{9}{5}, 2\right), \left(\frac{8}{5}, 3\right), \left(\frac{7}{5}, 4\right), \left(\frac{6}{5}, 5\right)$$

7.2 Extension to Weighted Graphs

In this section, we give the simple observation that a good additive approximation can imply a multiplicative approximation even for weighted graphs, if given sufficient additional additive error.

Theorem 7.4. *Let $1 \le \alpha < 3$ and $\beta \ge 1$ be an integer. Let $T(\gamma)$ be the running time of Theorem 6.3 when asked to produce a $(1+\varepsilon, 2w_{u,v}(\gamma))$ approximation. Let $\gamma = \left\lceil \frac{\alpha(\beta+1)+(\beta-1)}{2} \right\rceil - 1$. Then, Algorithm 9 produces an $(\alpha, 2w_{u,v}(\beta))$-approximation in time $T(\gamma)$.*

Suppose for some integer $\gamma \ge 1$ we have a $(1+\varepsilon, 2w_{u,v}(\gamma))$ approximation from Theorem 6.3. Given an integer value $\beta < \gamma$, suppose we want a $(\alpha, 2w_{u,v}(\beta))$ approximation for APSP on weighted graphs. Consider the edges that are the $\beta+1, \beta+2, \ldots, \gamma$-th heaviest edges in a given shortest path P. Then,

$$\sum_{i=\beta+1}^{\gamma} h(P,i) \le (\gamma-\beta)h(P,\beta+1) \le \frac{\gamma-\beta}{\beta+1}\mathsf{wt}(P)$$

as the weight of the $\beta+1$-heaviest edge is at most a $\frac{1}{\beta+1}$ fraction of the path's total weight. In particular, if $\hat{\delta}$ is the distance estimate given by Theorem 6.3,

$$\begin{aligned}
\hat{\delta}(u,v) &\le (1+\varepsilon)\mathsf{wt}(P) + 2\overline{h}(P,\gamma) \\
&\le (1+\varepsilon)\mathsf{wt}(P) + 2\frac{\gamma-\beta}{\beta+1}\mathsf{wt}(P) + 2\overline{h}(P,\beta) \\
&\le \left(\frac{2\gamma-\beta+1}{\beta+1} + \varepsilon\right)\delta(u,v) + 2\overline{h}(P,\beta)
\end{aligned}$$

Then, let us express γ in terms of α, β. In particular, we require γ such that,

$$\begin{aligned}
\alpha &\ge \frac{2\gamma-\beta+1}{\beta+1} + \varepsilon \\
\gamma &\le \frac{(\alpha-\varepsilon)(\beta+1)+(\beta-1)}{2}
\end{aligned}$$

Since γ is an integer, we conclude that to give any $(\alpha, 2w_{u,v}(\beta))$ approximation, it suffices to find a $(1+\varepsilon, 2w_{u,v}(\gamma))$ approximation for $\gamma = \left\lfloor \frac{(\alpha-\varepsilon)(\beta+1)+(\beta-1)}{2} \right\rfloor$. In particular, since we are free to choose ε, we can define,

$$\gamma = \begin{cases} \frac{\alpha(\beta+1)+(\beta-1)}{2} - 1 & \frac{\alpha(\beta+1)+(\beta-1)}{2} \in \mathbb{Z} \\ \left\lfloor \frac{\alpha(\beta+1)+(\beta-1)}{2} \right\rfloor & \text{o/w} \end{cases}$$

or equivalently,

$$\gamma = \left\lceil \frac{\alpha(\beta+1)+(\beta-1)}{2} \right\rceil - 1$$

As an example, suppose we want a $\left(\frac{4}{3}+\varepsilon, 2w_{u,v}(2)\right)$ approximation. Then, define $\gamma = 3$ and apply Theorem 6.3.

References

[ACIM99] 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.

[AG13] Rachit Agarwal and Philip Brighten Godfrey. Brief announcement: a simple stretch 2 distance oracle. In *Proceedings of the 2013 ACM symposium on Principles of distributed computing*, pages 110–112, 2013.

[AGM97] Noga Alon, Zvi Galil, and Oded Margalit. On the exponent of the all pairs shortest path problem. *Journal of Computer and System Sciences*, 54(2):255–262, 1997.

[AR20] Maor Akav and Liam Roditty. An almost 2-approximation for all-pairs of shortest paths in subquadratic time. In *Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1–11. SIAM, 2020.

[BGS05] Surender Baswana, Vishrut Goyal, and Sandeep Sen. All-pairs nearly 2-approximate shortest-paths in o (n 2 polylog n) time. In *STACS 2005: 22nd Annual Symposium on Theoretical Aspects of Computer Science, Stuttgart, Germany, February 24-26, 2005. Proceedings 22*, pages 666–679. Springer, 2005.

[BGSW19] Karl Bringmann, Fabrizio Grandoni, Barna Saha, and Virginia Vassilevska Williams. Truly subcubic algorithms for language edit distance and rna folding via fast bounded-difference min-plus product. *SIAM Journal on Computing*, 48(2):481–512, 2019.

[BK07] Piotr Berman and Shiva Prasad Kasiviswanathan. Faster approximation of distances in graphs. In *Algorithms and Data Structures: 10th International Workshop, WADS 2007, Halifax, Canada, August 15-17, 2007. Proceedings 10*, pages 541–552. Springer, 2007.

[BK10] 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.

[BKW19] Karl Bringmann, Marvin Künnemann, and Karol Węgrzycki. 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*, pages 943–954, 2019.

[Bra] Jan van den Brand. Complexity term balancer. www.ocf.berkeley.edu/~vdbrand/complexity/. Tool to balance complexity terms depending on fast matrix multiplication.

[CDX22] Shucheng Chi, Ran Duan, and Tianle Xie. Faster algorithms for bounded-difference min-plus product. In *Proceedings of the 2022 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1435–1447. SIAM, 2022.

[CDXZ22] Shucheng Chi, Ran Duan, Tianle Xie, and Tianyi Zhang. Faster min-plus product for monotone instances. In *Proceedings of the 54th Annual ACM SIGACT Symposium on Theory of Computing*, pages 1529–1542, 2022.

[Cha21] Timothy M Chan. All-pairs shortest paths for real-weighted undirected graphs with small additive error. In *29th Annual European Symposium on Algorithms (ESA 2021)*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik, 2021.

[Che14] Shiri Chechik. Approximate distance oracles with constant query time. In *Proceedings of the forty-sixth annual ACM symposium on Theory of computing*, pages 654–663, 2014.

[Che15] Shiri Chechik. Approximate distance oracles with improved bounds. In *Proceedings of the forty-seventh annual ACM symposium on Theory of Computing*, pages 1–10, 2015.

[CZ01] Edith Cohen and Uri Zwick. All-pairs small-stretch paths. *Journal of Algorithms*, 38(2):335–353, 2001.

[CZ22] Shiri Chechik and Tianyi Zhang. Nearly 2-approximate distance oracles in subquadratic time. In *Proceedings of the 2022 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 551–580. SIAM, 2022.

[DFK+23] Michal Dory, Sebastian Forster, Yael Kirkpatrick, Yasamin Nazari, Virginia Vassilevska Williams, and Tijn de Vos. Fast 2-approximate all-pairs shortest paths. *CoRR*, abs/2307.09258, 2023.

[DHZ00] Dorit Dor, Shay Halperin, and Uri Zwick. All-pairs almost shortest paths. *SIAM Journal on Computing*, 29(5):1740–1759, 2000.

[DKR+22] 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, 2022.

[Dür23] Anita Dürr. Improved bounds for rectangular monotone min-plus product and applications. *Information Processing Letters*, page 106358, 2023.

[DWZ22] Ran Duan, Hongxun Wu, and Renfei Zhou. Faster matrix multiplication via asymmetric hashing. *arXiv preprint arXiv:2210.10173*, 2022.

[Elk05] Michael Elkin. Computing almost shortest paths. *ACM Transactions on Algorithms (TALG)*, 1(2):283–323, 2005.

[ENWN16] Michael Elkin, Ofer Neiman, and Christian Wulff-Nilsen. Space-efficient path-reporting approximate distance oracles. *Theoretical Computer Science*, 651:1–10, 2016.

[EP16] Michael Elkin and Seth Pettie. A linear-size logarithmic stretch path-reporting distance oracle for general graphs. *ACM Transactions on Algorithms (TALG)*, 12(4):1–31, 2016.

[Gal12] François Le Gall. Faster algorithms for rectangular matrix multiplication. In *2012 IEEE 53rd annual symposium on foundations of computer science*, pages 514–523. IEEE, 2012.

[GU18] François Le Gall and Florent Urrutia. Improved rectangular matrix multiplication using powers of the coppersmith-winograd tensor. In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1029–1046. SIAM, 2018.

[HP98] Xiaohan Huang and Victor Y Pan. Fast rectangular matrix multiplication and applications. *Journal of complexity*, 14(2):257–299, 1998.

[LR83] Grazia Lotti and Francesco Romani. On the asymptotic complexity of rectangular matrix multiplication. *Theoretical Computer Science*, 23(2):171–185, 1983.

[MN07] Manor Mendel and Assaf Naor. Ramsey partitions and proximity data structures. *Journal of the European Mathematical Society*, 9(2):253–275, 2007.

[PR10] Mihai Patrascu and Liam Roditty. Distance oracles beyond the thorup-zwick bound. In *2010 IEEE 51st Annual Symposium on Foundations of Computer Science*, pages 815–823. IEEE, 2010.

[Rod23] Liam Roditty. New algorithms for all pairs approximate shortest paths. In *Proceedings of the 55th Annual ACM Symposium on Theory of Computing*, pages 309–320, 2023.

[Sei95] Raimund Seidel. On the all-pairs-shortest-path problem in unweighted undirected graphs. *Journal of computer and system sciences*, 51(3):400–403, 1995.

[Som16] Christian Sommer. All-pairs approximate shortest paths and distance oracle preprocessing. In *43rd International Colloquium on Automata, Languages, and Programming (ICALP 2016)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2016.

[SZ99] A. Shoshan and U. Zwick. All pairs shortest paths in undirected graphs with integer weights. In *40th Annual Symposium on Foundations of Computer Science (Cat. No.99CB37039)*, pages 605–614, 1999.

[TZ05] Mikkel Thorup and Uri Zwick. Approximate distance oracles. *Journal of the ACM (JACM)*, 52(1):1–24, 2005.

[Wil14] Ryan Williams. Faster all-pairs shortest paths via circuit complexity. In *Proceedings of the forty-sixth annual ACM symposium on Theory of computing*, pages 664–673, 2014.

[WN12] Christian Wulff-Nilsen. Approximate distance oracles with improved preprocessing time. In *Proceedings of the twenty-third annual ACM-SIAM symposium on Discrete Algorithms*, pages 202–208. SIAM, 2012.

[WW10] Virginia Vassilevska Williams and Ryan Williams. Subcubic equivalences between path, matrix and triangle problems. In *2010 IEEE 51st Annual Symposium on Foundations of Computer Science*, pages 645–654. IEEE, 2010.

[WX20] Virginia Vassilevska Williams and Yinzhan Xu. Truly subcubic min-plus product for less structured matrices, with applications. In *Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 12–29. SIAM, 2020.

[Yus12] Raphael Yuster. Approximate shortest paths in weighted graphs. *Journal of Computer and System Sciences*, 78(2):632–637, 2012.

[Zwi98] Uri Zwick. All pairs shortest paths in weighted directed graphs-exact and almost exact algorithms. In *Proceedings 39th Annual Symposium on Foundations of Computer Science (Cat. No. 98CB36280)*, pages 310–319. IEEE, 1998.

[Zwi02] Uri Zwick. All pairs shortest paths using bridging sets and rectangular matrix multiplication. *Journal of the ACM (JACM)*, 49(3):289–317, 2002.

A Additive Approximation Algorithms of [DHZ00] with a Different Analysis

In this appendix, we state the approximation algorithms of Dor, Halperin and Zwick [DHZ00], proving some useful properties about the algorithms that we require in our algorithm design.

Lemma 3.2. *([DHZ00]) Let $\beta \geq 2$ be even. Let G be an undirected, unweighted graph with n vertices and m edges. There is an algorithm computing a $+\beta$-approximate APSP solution in time*

$$\tilde{O}(\min(n^{2-\frac{2}{\beta+2}}m^{\frac{2}{\beta+2}}, n^{2+\frac{2}{3\beta-2}}))$$

Denote **SparseAPASP** *the algorithm running in time $\tilde{O}(n^{2-\frac{2}{\beta+2}}m^{\frac{2}{\beta+2}})$ and* **DenseAPASP** *the algorithm running in time $\tilde{O}(n^{2+\frac{2}{3\beta-2}})$.*

The pseudocode for **DenseAPASP** and **SparseAPASP** are provided in Algorithms 11 and 12 respectively. Although we never explicitly call **SparseAPASP**, we implicitly run it as a sub-routine in many of our algorithms.

Algorithm 11 **DenseAPASP**(G, β)

Input : Unweighted, undirected Graph $G = (V, E)$ with n vertices and even integer β
Output: $\hat{\delta} : U \times V \to \mathbb{Z}$ such that $\delta(u, v) \leq \hat{\delta}(u, v) \leq \delta(u, v) + \beta$ for all $u, v \in V$

$\hat{\delta} \leftarrow \begin{cases} 1 & (u, v) \in E \\ \infty & \text{o/w} \end{cases}$
$k \leftarrow \frac{3\beta-2}{2}$
$s_i \leftarrow n^{1-\frac{i}{k}}$ for all $1 \leq i \leq k-1$
$(D_1, D_2, \ldots, D_k), (E_1, E_2, \ldots, E_k), E^* \leftarrow$ **Decompose**$(G, (s_1, s_2, \ldots, s_{k-1}))$
for $1 \leq i \leq k$ **do**
  for $w \in D_i$ **do**
    $G_{i,w} \leftarrow \left(V, E_i \cup (u \times V) \cup E^* \cup \left(\bigcup_{i+j_1+j_2 \leq 2k+1} D_{j_1} \times D_{j_2}\right)\right)$
    $\hat{\delta} \leftarrow$ **Dijkstra**$(G_{i,w}, w, \hat{\delta})$

Algorithm 12 **SparseAPASP**(G, β)

Input : Unweighted, undirected Graph $G = (V, E)$ with n vertices, m edges, and even integer β
Output: $\hat{\delta} : U \times V \to \mathbb{Z}$ such that $\delta(u, v) \leq \hat{\delta}(u, v) \leq \delta(u, v) + \beta$ for all $u, v \in V$

$\hat{\delta} \leftarrow \begin{cases} 1 & (u, v) \in E \\ \infty & \text{o/w} \end{cases}$
$k \leftarrow \frac{\beta+2}{2}$
$s_i \leftarrow (m/n)^{1-\frac{i}{k}}$ for all $1 \leq i \leq k-1$
$(D_1, D_2, \ldots, D_k), (E_1, E_2, \ldots, E_k), E^* \leftarrow$ **Decompose**$(G, (s_1, s_2, \ldots, s_{k-1}))$
for $1 \leq i \leq k$ **do**
  for $w \in D_i$ **do**
    $\hat{\delta} \leftarrow$ **Dijkstra**$(G_{i,w}, w, \hat{\delta})$ where $G_{i,w} \leftarrow (V, E_i \cup (u \times V) \cup E^*)$

A.1 Closer Analysis of SparseAPASP

Instead of bounding the approximation error simply by using the number of thresholds used in the degree decomposition, we perform a more fine-grained analysis to bound the approximation error based on the structure and degrees of a given path P. Recall the definition of blocking vertices and blocking levels.

Definition 3.4. *Let u, v be vertices in a graph G and P a path between u, v. Let $(D_1, \ldots, D_k)$, $(E_1, \ldots, E_k)$ be the outputs of a call to* Decompose *with degree thresholds $s_1, \ldots, s_{k-1}$. If there is an edge $(a, b) \in P$ such that $(a, b) \notin E_{\ell(u)}$, the* **blocking vertex** *of P from u, denoted $b(u, P)$, is the closest vertex to v that is an endpoint to such an edge. If no such edge exists, then $b(u, P) = v$.*

Then, the **blocking vertices of** *P is the set $B(P) = \{x_0, x_1, \ldots, x_t\}$ defined in the following manner:*

1. $x_0 = u$
2. $x_1 = v$
3. $x_i = b\left(x_{i-1}, P_{x_{i-1}, x_{i-2}}\right)$ *where $P_{x_{i-1}, x_{i-2}}$ is the sub-path of P between x_{i-1}, x_{i-2}.*

For any $j \geq \ell(v)$, **blocking levels of** *P* **at level** *j is the set $L_B(P, j) = \{\ell(x_i)$ s.t. $x_i \in B(P)$ and $\ell(x_i) < j\}$. Denote $L_B(P) = L_B(P, \ell(v))$.*

We begin with some simple properties to note about the blocking vertices and blocking levels of a given path P.

Lemma A.1. *Let P be a shortest path and $B(P) = \{x_0, x_1, \ldots, x_t\}$ be the set of blocking vertices. Then,*

1. $k \geq \ell(x_1) > \ell(x_2) > \ldots > \ell(x_t) \geq 1$.
2. *For all $2 \leq i \leq t$, $x_{i-1} \notin P_{x_i, x_{i-2}}$.*
3. $B(P_{x_i, x_{i+1}}) \subset B(P)$ *and* $L_B(P_{x_i, x_{i+1}}) \subset L_B(P)$ *for* $1 \leq i \leq t-1$.
4. $P_{x_{t-1}, x_t} \subset P_{x_{t-2}, x_{t-1}} \subset \ldots \subset P_{x_0, x_1} = P$

Proof. Property 1 follows as for any edge (a, b), $\ell(a, b) = \min(\ell(a), \ell(b))$.

Property 2 follows as for all $i \geq 2$, $x_i \in P_{x_{i-1}, x_{i-2}}$.

$B(P_{x_i, x_{i+1}}) \subset B(P)$ follows by the construction of the blocking vertices, noting that x_i, x_{i+1} determine the remaining blocking vertices. Then, $L_B(P_{x_i, x_{i+1}}) \subset L_B(P)$ follows as the blocking levels are constructed from the blocking vertices.

Property 4 follows as x_i is chosen as a vertex on the path $P_{x_{i-2}, x_{i-1}}$. □

We now give the deferred proof that the number of blocking levels directly determines the additive error accumulated on a given path when executing SparseAPASP (Algorithm 12).

Lemma A.2. *Let u, v be vertices in a graph G and P be a shortest path between u, v. Let $B(P)$ be the blocking vertices of P. Then, Algorithm 12 in the $\ell(v)$-th iteration computes distance estimate $\hat{\delta}$ such that,*

$$\hat{\delta}(v^*, u) \leq \hat{\delta}_{\ell(v)}(v^*, u) \leq \delta(v, u) + 2|L_B(P)| + 1$$

For any level $j \geq \ell(v)$, if $v \in D_j$,

$$\hat{\delta}(v, u) \leq \hat{\delta}_j(v, u) \leq \delta(v, u) + 2|L_B(P, j)|$$

Proof. We proceed by induction. Suppose $|L_B(P)| = 0$. Then, $B(P) = \{x_0, x_1\} = \{u, v\}$. In particular, $P \subset E_{\ell(v)}$ so that Algorithm 12 finds the exact distance in the $\ell(v)$-th iteration.

Therefore, assume $|L_B(P)| > 0$ and denote $B(P) = \{x_0, x_1, \ldots, x_t\}$. Since $\ell(v) \le \ell(u)$, $x_2 \neq u$. Consider the sub-path $P_{v,x_2} = P_{x_1,x_2}$. Then, the set $L_B(P_{v,x_2}) \subset L_B(P)$ by Lemma A.1 and $\ell(x_2) \in L_B(P) \setminus L_B(P_{v,x_2})$ so that $L_B(P)$ is strictly larger. By the inductive hypothesis, in the $\ell(v)$-th iteration of Algorithm 12,

$$\begin{aligned}\hat{\delta}_{\ell(v)}(v^*, u) &\le (\hat{\delta}_{\ell(x_2)}(x_2^*, v) + 1) + \delta(x_2, u) + 1\\ &\le \delta(x_2, v) + 2|L_B(P_{v,x_2})| + \delta(x_2, u) + 3\\ &\le \delta(u, v) + 2|L_B(P)| + 1\end{aligned}$$

When $v \in D_{\ell(v)}$,

$$\begin{aligned}\hat{\delta}(v, u) &\le \hat{\delta}_{\ell(x_2)}(x_2^*, v) + 1 + \delta(x_2, u)\\ &\le \delta(x_2, v) + 2|L_B(P_{x_1,x_2})| + \delta(x_2, u) + 2\\ &\le \delta(v, u) + 2|L_B(P)|\end{aligned}$$

Now, let $j \ge \ell(v)$ and $v \in D_j$. If $|L_B(P, j)| = 0$, then $P \subset E_j$ and we compute an exact distance in the j-th iteration. Otherwise, let $b = b(v, P)$ and $b^* \in D_{\ell(b)}$ where $\ell(b) < j$ so that,

$$\begin{aligned}\hat{\delta}_j(v, u) &\le \hat{\delta}_{\ell(b)}(v, b^*) + 1 + \delta(b, u)\\ &\le \delta(v, b) + 2|L_B(P_{b,v})| + 2 + \delta(b, u)\\ &\le \delta(u, v) + 2|L_B(P, j)|\end{aligned}$$

where we have used the inductive hypothesis and $L_B(P_{b,v}) \subset L_B(P, j)$ and $\ell(b) \in L_B(P, j) \setminus L_B(P_{b,v})$. □

In many cases for our algorithms, it suffices to use the following weaker lemma. Here, we bound the additive error simply by the number of levels occurring in the shortest path, whereas above the notion of blocking vertices takes into account the specific positions where each level occurs.

Lemma A.3. *Let u, v be vertices in a graph G and P be a shortest path between u, v. Suppose $\ell(v) \le \ell(u)$.*

For $j \ge \ell(v)$, let $L(P, j) = \{\ell(w)$ s.t. $w \in P$ and $\ell(w) < j\}$ and let $L(P) = L(P, \ell(v)) = \{\ell(w)$ s.t. $w \in P$ and $\ell(w) < \ell(v)\}$. Then, Algorithm 12 in the $\ell(v)$-th iteration computes distance estimate $\hat{\delta}$ such that,

$$\delta(v^*, u) \le \hat{\delta}_{\ell(v)}(v^*, u) \le \delta(v, u) + 2|L(P)| + 1$$

Furthermore, for $j \ge \ell(v)$ such that $v \in D_j$,

$$\hat{\delta}_j(v, u) \le \delta(v, u) + 2|L(P, j)|$$

Proof. This follows from Lemma A.2 as $L_B(P, j) \subset L(P, j)$ for all paths P and iterations j. □

B Dominating Sets and Degree Decomposition

In this section, we give the proofs of Lemmas 2.4, 2.5, 2.6 and 6.1.

Lemma 2.4. *Let U be a universe of n elements. Let $\mathcal{F} = \{S_1, \ldots, S_n\}$ denote a collection of subsets $S_i \subset U$ such that $|S_i| \ge s$ for all i. Then, there is a deterministic algorithm* **HittingSet** *that computes a hitting set X of size $O\left(\frac{n \log n}{s}\right)$ of $\mathcal{F}$ in time $\tilde{O}(ns)$. There is also a randomized algorithm* **rHittingSet** *that with high probability computes a hitting set X of size $O\left(\frac{n \log n}{s}\right)$ of $\mathcal{F}$ in time $O(n)$.*

Algorithm 13 HittingSet$(U, \mathcal{F}, s)$

Input : Universe U of n elements and a collection $\mathcal{F}$ of n subsets of size at least s
Output: Hitting set X of size $O(n \log n/s)$

```
X ← ∅
c(j) ← |{S_i ∈ F s.t. j ∈ S_i}| for all j ∈ U
while F ≠ ∅ do
    v ← arg max_{j∈U} c(j)
    X ← X ∪ {v}
    F ← {S_i ∈ F s.t. v ∉ S_i}
    c(j) ← |{S_i ∈ F s.t. j ∈ S_i}| for all j ∈ U
```

Proof. We begin with the deterministic algorithm HittingSet with pseudocode provided in Algorithm 13. Without loss of generality, assume all $|S_i| = s$. Otherwise, drop all but the first s elements of each subset S_i and observe that any hitting set of the resulting collection must also be a hitting set for the original collection.

Now, initially we have $\sum_{j\in U} c(j) = \sum_{S_i\in\mathcal{F}} |S_i| = ns$. When the algorithm terminates, we have an empty collection $\mathcal{F}$ so that $\sum_{j\in U} c(j) = 0$. In each iteration, the most expensive operation is to update $\mathcal{F}$ and $c(j)$. By keeping $\{S_i \in \mathcal{F} \text{ s.t. } j \in S_i\}$ in addition to the count $c(j)$ alone, we can update $\mathcal{F}$ and $c(j)$ in $\tilde{O}(1)$ time for each decrement of $c(j)$. Thus, the overall running time can be bounded by $\tilde{O}(ns)$.

We now bound the size of output set X. Let T_j denote the number of sets remaining in $\mathcal{F}$ after j elements have been added to X. Then, $T_0 = |\mathcal{F}| = n$, $|X| = \min_{j\geq 0 \text{ s.t. } T_j=0} j$, and $T_j = T_{j-1} - c(u_j)$ where u_j is the j-th element added to X. By our choice of $c(u_j)$, we have $c(u_j) \geq \frac{T_{j-1}s}{n-j+1}$ since there are T_{j-1} sets of size s and $n-(j-1) = n-j+1$ elements not in X. Then,

$$T_j \leq \left(1 - \frac{s}{n-j+1}\right) T_{j-1} \leq T_0 \prod_{l=0}^{j-1} \left(1 - \frac{s}{n-l}\right) < n\left(1-\frac{s}{n}\right)^j \leq ne^{-\frac{sj}{n}}$$

When $j = \frac{n\log n}{s}$ we have $T_j < 1$ so $T_j = 0$. In particular, $|X| \leq \frac{n\log n}{s}$. □

Proof. The randomized algorithm rHittingSet simply samples each element with probability p. Clearly this algorithm runs in time $O(n)$.

First, we bound the probability that X is not a hitting set. For each subset of size s, the probability that no element is sampled can be bounded as,

$$(1-p)^s \leq e^{-ps}$$

By the union bound, the probability some subset has no element sampled can be bounded as,

$$ne^{-ps}$$

so if $p = c\log n/s$, we upper bound this as n^{1-c} for some constant $c > 1$.

Then, to upper bound the size of X, we apply a standard Chernoff bound to see that $|X| = O(n\log n/s)$ with high probability. □

Lemma 2.5. *Let G be a graph on n vertices. There is a deterministic algorithm* Dominate *and randomized algorithm* rDominate *that computes a dominating set D of size $O\left(\frac{n\log n}{s}\right)$ of V_s in time $O(m+ns)$ and $O(n)$ respectively.*

Proof. Given the previous result, we simply apply **HittingSet** or **rHittingSet** on the subsets $N(v) \cup \{v\}$ for all $v \in V_s$. Since $|V_s| \leq n$, note that our previous result is also correct for all collections $\mathcal{F}$ of at most n subsets as we can augment $\mathcal{F}$ with $n - |\mathcal{F}|$ subsets and any hitting set of the augmented collection is also a hitting set for the original collection. □

Lemma 2.6. *Let G be a graph on n vertices. Given degree thresholds $s_1 > s_2 > \ldots > s_{k-1}$, there is a deterministic algorithm* **Decompose** *and a randomized algorithm* **rDecompose** *that outputs edge sets $\{E_i\}_{i=1}^k$, edge set E^*, and vertex sets $\{D_i\}_{i=1}^k$ satisfying,*

1. *$E_i = \{(u,v) \in E$ s.t. $\min(\deg(u), \deg(v)) < s_{i-1}\}$ is the set $E_{s_{i-1}}$.*
2. *D_i dominates $V_{s_i} = \{v \in V$ s.t. $\deg(v) \geq s_i\}$ and $|D_i| = \tilde{O}\left(\frac{n}{s_i}\right)$. For convenience, V_{s_i} may also be denoted V_i.*
3. *$D_1 \subset D_2 \subset \ldots \subset D_k = V$ and $E_k \subset E_{k-1} \subset \ldots \subset E_1 = E$.*
4. *$E^* = \bigcup_{i=1}^k E_i^*$ where each $E_i^* \subset E$ has for every $v \in V_i$, at least one edge $(v,w) \in E_i^*$ for $w \in D_i$.*

Furthermore, **Decompose** *runs in $\tilde{O}(kn^2)$ time and* **rDecompose** *runs in $\tilde{O}(kn)$ time. Note that* **rDecompose** *satisfies the above conditions with high probability.*

For a given vertex $v \in V$, define the **level of** *v, denoted $\ell(v)$, as the integer i such that $s_i \leq \deg(v) < s_{i-1}$. For a given edge $e \in E$, define the* **level of** *e, denoted $\ell(e)$, as the integer i such that $e \in E_i \setminus E_{i+1}$.*

We first discuss the deterministic algorithm **Decompose**.

Proof. By Lemma 2.5, we can compute each D_i with **Dominate** in $\tilde{O}(ns_i) = \tilde{O}(n^2)$ time, so that overall we can compute all D_i in $\tilde{O}(kn^2)$ time. Furthermore, we can ensure $D_1 \subset D_2 \subset \ldots \subset D_k = V$ by augmenting the D_i as necessary. Of course, when constructing D_i, we can easily construct E_i^* by choosing the appropriate edges when we add a vertex to v to D_i. Finally, we can construct E_i in $O(m)$ time by examining each edge for the degrees of its endpoints. □

Next, we discuss the randomized algorithm **rDecompose**.

Proof. Using the randomized construction **rDominate**, we can likewise choose $p_i = \frac{c \log n}{s_i}$ for large enough c to ensure correctness by the union bound. □

Lemma 6.1. *Let G be a weighted, undirected graph on n vertices. For a given edge e incident to vertex v, let $\text{rank}_v(e)$ be the order of e among all edges incident to v when ranked in increasing order of weight. For example, if e is the lightest edge incident to v, $\text{rank}_v(e) = 1$. For a neighbor $w \in N(v)$, we also denote $\text{rank}_v(w) = \text{rank}_v(v,w)$ to be the rank of the edge (v,w).*

Given thresholds $s_1 > s_2 > \ldots > s_{k-1}$, there is an algorithm **WeightedDecompose** *that outputs edge sets $\{E_i\}_{i=1}^k$, edge set E^*, and vertex sets $\{D_i\}_{i=1}^k$ satisfying,*

1. *$E_i = \{(u,v) \in E$ s.t. $\min(\text{rank}_u(u,v), \text{rank}_v(u,v)) < s_{i-1}\}$.*
2. *D_i dominates $V_{s_i} = \{v \in V$ s.t. $\deg(v) \geq s_i\}$ and $|D_i| = \tilde{O}\left(\frac{n}{s_i}\right)$*
3. *$E^* = \bigcup_{i=1}^k E_i^*$ where each $E_i^* \subset E$ has for every $v \in V_{s_i}$ at least one edge $(v,w) \in E_i^*$ where $w \in D_i$ and $\text{rank}_v(v,w) < s_i$.*

Furthermore, **WeightedDecompose** *runs in $\tilde{O}(kn^2)$-time.*

We define the level of a vertex and edge analogously as in the unweighted case. For a given vertex $v \in V$, define the **level of** *v, denoted $\ell(v)$, as the integer i such that $s_i \leq \deg(v) < s_{i-1}$. For a given edge $e \in E$, define the* **level of** *e, denoted $\ell(e)$, as the integer i such that $e \in E_i \setminus E_{i+1}$.*

Proof. As before, constructing E_i can easily be done in $O(m) = O(n^2)$ time. To construct D_i, for every vertex with degree at least s_i, we compute a hitting set for its neighborhood restricted to the $s_i - 1$ lightest neighbors. In particular, we compute a hitting set for the sets $\{v\} \cup \{w \in N(v) \text{ s.t. } \text{rank}_v(w) < s_i\}$ of size s_i for each $v \in V_{s_i}$. Then, this guarantees that the edge $(v, w) \in E_i^*$ satisfies the additional constraint $\text{rank}_v(v, w) < s_i$ for $v \in V_{s_i}$. By our previous arguments, D_i dominates V_{s_i} and has size $\tilde{O}\left(\frac{n}{s_i}\right)$. Note that as before computing D_i requires $\tilde{O}(n^2)$ time. □

B.1 Useful Results on Fast Matrix Multiplication

We will extensively use various algorithms for efficiently computing matrix products. We lay out some of the key results used below. The matrix multiplication exponent $\omega(a, b, c)$ denotes infimum over all constants t such that we can multiply two matrices of dimension $n^a \times n^b$ and $n^b \times n^c$ in $O(n^t)$ time. Denote $\omega(1, 1, 1)$ as the constant ω. As the results given in this section are quite involved, we omit them for brevity and instead provide references to the relevant proofs.

Theorem B.1. *([DWZ22]) The matrix multiplication exponent $\omega \leq 2.37188$.*

See Duan, Wu, and Zhou [DWZ22] for the appropriate proof. The dual matrix multiplication exponent α is the supremum over all constants k such that $\omega(1, k, 1) \leq 2$.

Theorem B.2. *([GU18]) The dual matrix multiplication exponent $\alpha \geq 0.31389$.*

Lemma B.3. *([Gal12]) For any fixed value $0 \leq k_0 < 1$ and $k_0 \leq k \leq 1$,*

$$\omega(1, k, 1) \leq \omega(1, k_0, 1) + (\omega - \omega(1, k_0, 1))\frac{k - k_0}{1 - k_0}$$

We refer the reader to Le Gall and Urrutia [GU18] and Le Gall [Gal12] for the proofs of the above two results. We will also use some useful properties of ω, in particular that it is convex, symmetric and linear.

Lemma B.4. *(Corollary 5.1 of [LR83]) Suppose $\omega(1, a, b) = \omega(1, a', b') = c$. Then, for $t \in [0, 1]$,*

$$\omega(1, t * a + (1 - t) * a', t * b + (1 - t) * b') \leq c$$

Lemma B.5. *Let $t, a, b, c \geq 0$. Then,*

$$\omega(a, b, c) = \omega(a, c, b) = \omega(b, a, c) = \omega(b, c, a) = \omega(c, a, b) = \omega(c, b, a)$$

and

$$\omega(ta, tb, tc) = t\omega(a, b, c)$$

See, for example, Lotti and Romani [LR83] for the former result and Huang and Pan [HP98] for the latter. The following theorem of Huang and Pan [HP98] gives improved bounds on general rectangular matrix multiplication for matrix products where a, b, c are distinct.

Theorem B.6. *(Theorem 8.1 of [HP98]) Suppose $r > 1 > t > 0$. Then,*

$$\omega(t, 1, r) \leq \begin{cases} r + 1 & t \leq \alpha \\ \frac{r(1-\alpha)+(1-t)+(\omega-1)(t-\alpha)}{1-\alpha} & \alpha < t \leq 1 \end{cases}$$

See Huang and Pan [HP98] for the proof of the above result. We will use the following corollary as a generalization,

Corollary B.7. *Suppose $0 < a < b < c$. Then,*

$$\omega(a,b,c) \leq \begin{cases} b+c & \frac{a}{b} \leq \alpha \\ \frac{a(\omega-2)+b(1+\alpha-\alpha\cdot\omega)+c(1-\alpha)}{1-\alpha} & \alpha < \frac{a}{b} \leq 1 \end{cases}$$

Proof. By linearity (Lemma B.5), we have,

$$\begin{aligned} \omega(a,b,c) &= b \cdot \omega\left(\frac{a}{b}, 1, \frac{c}{b}\right) \\ &\leq \begin{cases} b \cdot \left(\frac{c}{b}+1\right) & \frac{a}{b} \leq \alpha \\ b \cdot \frac{\frac{c}{b}(1-\alpha)+\left(1-\frac{a}{b}\right)+(\omega-1)\left(\frac{a}{b}-\alpha\right)}{1-\alpha} \end{cases} \\ &= \begin{cases} b+c & \frac{a}{b} \leq \alpha \\ \frac{a(\omega-2)+b(1+\alpha-\alpha\omega)+c(1-\alpha)}{1-\alpha} \end{cases} \end{aligned}$$

□

B.2 Fast $(\min,+)$ Matrix Multiplication

Recall that for two matrices A, B, their $(\min,+)$ product, denoted $A * B$, is defined as,

$$(A*B)_{ij} = \min_k A_{ik} + B_{kj}$$

While computing $(\min,+)$ product in general is difficult, there are several structured instances in which efficient algorithms are known beyond the trivial $\tilde{O}(n^3)$ algorithm. We describe a few that we will require here. First, we begin with the case that all entries are bounded.

Theorem B.8. *([AGM97]) Let A, B be two $n \times n$ integer matrices with entries bounded in absolute value by M. Then, there is an algorithm* **BoundedMinPlus** *computing the $(\min,+)$ product $C = A * B$ in time $\tilde{O}(Mn^\omega)$.*

Proof. Without loss of generality, we can assume the entries of A, B are in fact non-negative and therefore in the range $[0, M]$. Otherwise, define non-negative matrices $A'_{ik} = A_{ik} + M$ and $B'_{kj} = B_{kj} + M$ with values in the range $[0, 2M]$. Given the product $A' * B'$, we can easily recover $A * B$ as,

$$(A'*B')_{ij} - 2M = \min_k A'_{ik} + B'_{kj} - 2M = \min_k A_{ik} + B_{kj} = (A*B)_{ij}$$

Given A, we define the polynomial matrices $A^p_{ik} = x^{A_{ik}}$ if A_{ik} is finite and 0 otherwise. We define B^p analogously, and compute $C^p = A^p B^p$ using fast matrix multiplication in $\tilde{O}(Mn^\omega)$ time. Then, for each entry in C^p, if $C^p_{ij} = 0$, we set $C_{ij} = \infty$ and otherwise, we compute the lowest degree term x^c and set $C_{ij} = c$. Now, we prove that that $C = A * B$. Recall that $(A*B)_{ij} = \min_k A_{ik} + B_{kj}$. $(A*B)_{ij}$ is infinite if and only if all $A_{ik} + B_{kj}$ are infinite. If either A_{ik}, B_{kj} is infinite, then the corresponding entry in A^p, B^p is zero. Thus, if all $A_{ik} + B_{kj}$ are infinite, $C^p_{ij} = 0$ so $C_{ij} = \infty$.

Suppose $(A*B)_{ij}$ is finite. Then $C^p_{ij} = \sum_k x^{A_{ik}+B_{kj}}$ where the sum ranges over all k where $A_{ik} + B_{kj}$ is finite. Taking the minimum degree, we have $(A*B)_{ij}$, proving correctness.

Let us now analyze the performance of the above algorithm. Constructing A^p from A, B^p from B, and C from C^p requires time at most $\tilde{O}(Mn^2)$. Computing $C^p = A^p B^p$ requires $\tilde{O}(Mn^\omega)$ as each multiplication of at most M degree polynomials requires $\tilde{O}(M)$ time. □

More generally, there is also a sub-cubic algorithm when the entries have bounded differences [BGSW19].

Definition B.9. *An $n \times n$ matrix is called* b**-bounded difference** *if for all i, j, $|A_{i,j} - A_{i,j+1}| \leq b$ and $|A_{i,j} - A_{i+1}, j| \leq b$. If only the former holds, the matrix is* **row bounded difference**. *If only the latter holds, the matrix is* **column bounded difference**.

A more general requirement may be that the entries are monotone. That is, the values of each row (or column) are monotonically increasing.

Definition B.10. *An $n \times n$ matrix is called* **row-monotone** *if all entries are non-negative integers bounded by $O(n)$ and each row of this matrix is non-decreasing. A* **column-monotone** *matrix is defined similarly.*

Theorem B.11. *([CDXZ22]) Let A, B be two $n \times n$ integer matrices with entries $A_{ij}, B_{ij} = O(n)$. Suppose furthermore that B is either row-monotone or column-monotone. Then, there is an algorithm* **MonotoneMinPlus** *computing the $(\min, +)$ product $C = A * B$ in time $\tilde{O}\left(n^{\frac{3+\omega}{2}}\right)$*

See Chi, Duan, Xie, and Zhang [CDXZ22] for a detailed proof. We also provide a proof overview in the proof of Theorem 5.1. Given a b-bounded difference matrix, it is easy to construct a row-monotone matrix, by adding the $b \cdot j$ to the j-th column. To recover the $(\min, +)$ product, simply subtract the same constant from each column in the result. We describe this procedure as **BDtoMonotone**. By a similar argument, we can convert row-bounded difference matrices to row-monotone matrices, and column-bounded difference matrices to column-monotone matrices. We will also use the simple observation that the sub-matrix of a monotone matrix (chosen by selecting a subset of rows or columns) is a monotone matrix.

We also require the approximate $(\min, +)$ product of Zwick [Zwi02].

Theorem B.12. *Let A, B be two integer $n \times n$ matrices with non-negative entries. Let $\varepsilon > 0$. There is an algorithm* **ApproximateMinPlus** *that returns C in time $\tilde{O}\left(n^{\omega}/\varepsilon\right)$ such that for all $1 \leq i, j \leq n$,*

$$(A * B)_{ij} \leq C_{ij} \leq (1 + \varepsilon)(A * B)_{ij}$$

Proof. Suppose the entries of A, B are in the range $[0, M]$. Let the resolution R be a parameter to be set later. Initialize $C_{ij} \leftarrow \infty$ for all i, j. For every $\lfloor \log R \rfloor \leq r \leq \lceil \log M \rceil$, define,

$$A^{(r)}_{ik} \leftarrow \begin{cases} \lceil RA_{ik}/2^r \rceil & 0 \leq A_{ik} \leq 2^r \\ \infty & \text{o/w} \end{cases}$$

$$B^{(r)}_{kj} \leftarrow \begin{cases} \lceil RB_{kj}/2^r \rceil & 0 \leq B_{kj} \leq 2^r \\ \infty & \text{o/w} \end{cases}$$

and compute $C^{(r)} \leftarrow$ **BoundedMinPlus**$(A^{(r)}, B^{(r)}, R)$. Update $C_{ij} \leftarrow \min\left(C_{ij}, \frac{2^r}{R} C^{(r)}\right)$. The inequality $(A * B)_{ij} \leq C_{ij}$ follows as $A^{(r)}, B^{(r)}$ always round its entries up. Suppose $(A * B)_{ij} = A_{iq} + B_{qj}$ and without loss of generality $A_{iq} \leq B_{qj}$. Let $1 \leq s \leq \lceil \log M \rceil$ where $2^{s-1} \leq B_{kj} \leq 2^s$. If $s \leq \log R$, then the $C^{\log R}$ computes $(A * B)_{ij}$ exactly. Thus, assume $s \geq \log R$. Then,

$$\frac{2^r}{R} C^{(r)}_{ij} \leq \frac{2^r}{R}\left(A^{(r)}_{iq} + B^{(r)}_{qj}\right) \leq A_{iq} + B_{qj} + \frac{2^{r+1}}{R} \leq \left(1 + \frac{4}{R}\right)(A * B)_{ij}$$

Then, if we set $R = O\left(1/\varepsilon\right)$, we obtain the approximation error $(1+\varepsilon)$ and note that each invocation of **BoundedMinPlus** requires $\tilde{O}(n^{\omega}/\varepsilon)$ time. □

C Missing Proofs

Lemma C.1. *Let G be an unweighted, undirected graph on n vertices. Let $\hat{\delta}$ be the estimate output by Algorithm 2. Then, $\delta(u,v) \leq \hat{\delta}(u,v)$ for all $u, v \in V$.*

Proof. Consider some $t = 2^j$. Then,

$$\hat{\delta}(u,v) = \min_{w \in W} \hat{\delta}(u,w) + \hat{\delta}(w,v)\delta(u,v)$$

so that Algorithm 2 outputs $\delta(u,v) \leq \hat{\delta}(u,v)$ for all u, v. □

Lemma C.2. *Let G be an unweighted, undirected graph on n vertices. Let $\hat{\delta}$ be the estimate output by Algorithm 3. Then, $\delta(u,v) \leq \hat{\delta}(u,v)$ for all $u, v \in V$.*

Proof. We will show for all vertex pairs (u,v) that $\delta(u,v) \leq \hat{\delta}(u,v)$. To do so, it suffices to observe that every estimated distance is witnessed by some path in the graph G.

Initially, the only finite entries of $\hat{\delta}$ are edges (u,v) in the graph G. Phase 1 calls $\textsc{DominatingSetAPASP}(G_t, V, V, C_t, C+1)$ so that $\hat{\delta}(u,v) \geq \delta(G_t,u,v) \geq \delta(u,v)$ for all u, v since $G_t \subset G$ is a sub-graph and Lemma C.1.

Similarly, Phase 2.1 calls $\textsc{DominatingSetAPASP}(G_{s_1}, V, D_{k-1}, D_1, C+2)$ so that after Phase 2.1, $\hat{\delta}(u,v) \geq \delta(G_{s_1},u,v) \geq \delta(u,v)$ as $G_{s_1} \subset G$.

In Phase 2.2, any path found by **BFS** in subgraphs $G_x, G_2 \subset G$ is naturally also a path in G.

In Phase 2.3, since for any $u \in q(w, D_i)$, $(u,w) \in E$, by the triangle inequality,

$$1 + \hat{\delta}(u,v) \geq 1 + \delta(u,v) \geq \delta(w,v)$$

Thus, any distance in $G_{i,w}$ is witnessed by some path in G. Therefore, executing $\textsc{Dijkstra}$ on $G_{i,w}$ with weight function $\hat{\delta}$ will not return any distance $\hat{\delta}$ smaller than the shortest distance between u, v.

Finally, we conclude by noting the correctness of $\textsc{BoundedAdditiveAPASP}$ and $\textsc{DenseAPASP}$ implies $\delta(u,v) \leq \hat{\delta}(u,v)$. □

Proposition C.3. *For any $x \in [0,1]$ and any integer $k \geq 1$,*

$$\omega\left(1, 1 - \frac{(k-1)}{k}x, 1 - \frac{x}{k}\right) \leq \omega(1, 1-x, 1)$$

Proof. Recall that by symmetry of ω, $\omega(1,1-x,1) = \omega(1,1,1-x)$. The claim follows from applying Lemma B.4 with $t = \frac{1}{k}$. □

Proposition C.4. *We obtain the running times in Table 1 for $(2,0)$ approximation on paths of length $\delta(u,v) \geq k$ from Theorem 4.2.*

Proof. Recall from Theorem 4.2 we wish to minimize the following,

$$\max\left(\omega\left(1 - \frac{k-2}{2(l-k+2)}x, 1-x, 1 - \frac{k-4}{2(l-k+2)}x\right), 2 + \frac{x}{l-k+2}, 1.5 + x\right)$$

In all cases, we can set M to be some arbitrary constant without affecting the asymptotic running time.

$k = 6$ We minimize,

$$\max\left(\omega\left(1-\frac{2x}{l-4}, 1-x, 1-\frac{x}{l-4}\right), 2+\frac{x}{l-4}, 1.5+x\right)$$

Using [Bra], we set $l = 51, x = 0.50980335$ and obtain the running time $\tilde{O}(n^{2.01084688})$.

$k = 8$ We minimize,

$$\max\left(\omega\left(1-\frac{3x}{l-6}, 1-x, 1-\frac{2x}{l-6}\right), 2+\frac{x}{l-6}, 1.5+x\right)$$

Using [Bra], we set $l = 74, x = 0.50716126$ and obtain the running time $\tilde{O}(n^{2.00745825})$.

$k = 10$ We minimize,

$$\max\left(\omega\left(1-\frac{4x}{l-8}, 1-x, 1-\frac{3x}{l-8}\right), 2+\frac{x}{l-8}, 1.5+x\right)$$

Using [Bra], we set $l = 96, x = 0.50496404$ and obtain the running time $\tilde{O}(n^{2.00573823})$.

$k = 12$ We minimize,

$$\max\left(\omega\left(1-\frac{5x}{l-10}, 1-x, 1-\frac{4x}{l-10}\right), 2+\frac{x}{l-10}, 1.5+x\right)$$

Using [Bra], we set $l = 119, x = 0.50432041$ and obtain the running time $\tilde{O}(n^{2.00462679})$. □

D A Quadratic $(\frac{7}{3}, 0)$-Approximate APSP on Unweighted Graphs

In the section, we briefly observe that a $\frac{7}{3}$-approximate APSP solution can be obtained in $\tilde{O}(n^2)$-time. This follows simply from a $\tilde{O}(n^2)$-time $(2,1)$-approximate APSP algorithm due to Baswana and Kavitha [BK10] and Berman and Kasiviswanathan [BK07] along with the result of [DHZ00]

Lemma D.1. *Given an undirected, unweighted graph, Algorithm 14 outputs a $(2,0)$ approximation for paths of length $\delta(u,v) \leq 2$. in $\tilde{O}(n^2)$-time.*

Algorithm 14 DISTANCE2APASP(G)

Input : Unweighted, undirected Graph $G = (V, E)$ with n vertices

Output: Distance estimate $\hat{\delta} : U \times V \to \mathbb{Z}$ such that $\delta(u,v) \leq \hat{\delta}(u,v)$ for all u, v and $\hat{\delta}(u,v) \leq 4$ for all $\delta(u,v) = 2$

$\hat{d}(u,v) \leftarrow \begin{cases} 1 & (u,v) \in E \\ \infty & \text{o/w} \end{cases}$

$s_i \leftarrow \frac{n}{2^i}$ for all $1 \leq i \leq k-1 = \lfloor \log n \rfloor$

$(D_1, D_2, \ldots, D_k), (E_1, E_2, \ldots, E_k), E^* \leftarrow$ DECOMPOSE$(G, (s_1, s_2, \ldots, s_{k-1}))$

for $1 \leq i \leq k = \lfloor \log n \rfloor + 1$ **do**

  for $w \in D_i$ **do**

    $X_i(w) \leftarrow$ DIJKSTRA(w) on graph $\left(V, E_i \cup (w \times V) \cup E^*, \hat{\delta}\right)$

    $\hat{\delta}(w,v) \leftarrow \min(\hat{\delta}(w,v), X_i(w,v))$

$\hat{\delta}(u,v) \leftarrow \min(\hat{\delta}(u,v), \hat{\delta}(v,u))$

Proof. First we analyze the running time. Note $k = \tilde{O}(1)$ so that **Decompose** requires $\tilde{O}(n^2)$ time. Fix an iteration i. Then $|D_i| = \tilde{O}\left(\frac{n}{s_i}\right)$, E_i has ns_{i-1} edges, while $E^* \cup (w \times V)$ have size $\tilde{O}(n)$. Thus, each iteration i requires $\tilde{O}(n^2)$ time. Over $\log n$ iterations, Algorithm 14 requires $\tilde{O}(n^2)$ time.

Consider now $\delta(u,v) = 2$ and a path $P = (u,w,v)$. Without loss of generality, assume $\deg(u) \geq \deg(v)$ so that $P \subset E_{\ell(u)}$. Let $u^* \in D_{\ell(u)}$ be its representative $r(u, D_{\ell(u)})$. Then, the execution of **Dijkstra** from u^* returns $\hat{\delta}(u^*, v) \leq 3$ via the path (u^*, u, w, v). Now, in the final iteration, we have $\hat{\delta}(v,u) \leq 4$ via the path (v, u^*, u) when executing **Dijkstra** from vertex v.

Finally, it is easy to observe that $\hat{\delta}(u,v) \geq \delta(u,v)$ for all u, v (without any assumption on the distance) since all traversed paths exist in the original graph G. □

Combining Lemma D.1 with the algorithm of Baswana et al. [BK10] or Berman et al. [BK07] gives the desired result.

Corollary D.2. *There is an $\tilde{O}(n^2)$-time algorithm that computes a $\left(\frac{7}{3}, 0\right)$-approximate APSP solution in undirected, unweighted graphs.*

Proof. Of course length 1 paths can be directly deduced from the adjacency matrix in time $O(n^2)$. From Lemma D.1, all distance 2 paths can be approximated within $(2,0)$-approximation in $\tilde{O}\left(n^2\right)$ time. For paths of length 3 or more, a $(2,1)$-approximation is at least as good as a $(\frac{7}{3}, 0)$ approximation. □

E Faster Combinatorial $(2,0)$-approximation for path lengths ≥ 4.

In this section, we give an improved combinatorial algorithm for finding $(2,0)$ approximations for paths with length $k \geq 4$. We obtain a $+\beta$-additive approximations for paths of length at most $\beta+1$ for every even β on dense graphs. In particular, we apply the algorithm with $\beta = 4$ to compute a $+4$ approximation on paths with length at most 5. This implies a $(2,0)$ approximation for paths of length at least 4.

Theorem E.1. *Let G be an undirected, unweighted graph with n vertices. Let $\beta \geq 4$ be an even odd integer. Algorithm 15 computes in expected time $\tilde{O}\left(n^{2+\frac{2}{3\beta+2}}\right)$ a distance estimate $\hat{\delta}$ such that $\delta(u,v) \leq \hat{\delta}(u,v)$ for all $u, v \in V$ and $\hat{\delta}(u,v) \leq \delta(u,v) + \beta$ for all $\delta(u,v) \leq \beta + 1$.*

Notation and Definitions We define some notation that will be useful in this section. Let G be an undirected, unweighted graph. Let β be an even integer, $k = \frac{3\beta+2}{2}$ and $k_0 = \beta - 2$. Let $s_i, s_2, \ldots, s_{k-1}$ be degree thresholds. For $k_0 + 1 \leq k \leq k_0 + \frac{\beta}{2} - 1$ define $r_i = \frac{\beta}{2} - (i - (k_0+1))$ and let $N(u, r_i) = \{v \in V \text{ s.t. } \delta(u,v) \leq r_i\}$ be the r_i-neighborhood of u.

Definition E.2. *For a vertex $u \in V$ and set $S \subset V$, let $p(u,S) = \arg\min_{v \in S} \delta(u,v)$ be the closest vertex in S to u.*

We note that $p(u,S)$ can be efficiently computed by executing **Dijkstra** on the graph G augmented with a dummy vertex w with 0 weight edges to every vertex $v \in S$. For a given i, $X(u,i)$ is an arbitrary subset of $N(u,r_i)$ if $|N(u,r_i)| \geq s_i$ and $C(u_i) = N(u,r_i) \cap D_{i+1}$ if $|N(u,r_i) \cap D_{i+1}| \leq 12 s_{k-1} \log n$. Roughly speaking, $X(u,i)$ is the first s_i vertices encountered when conducting a **BFS** from u. For details on how $X(u,i), C(u,i)$ are computed see Lemma E.3.

High Level Overview To illustrate our algorithm, we walk through an overview of how to compute a $+4$ approximation for paths of length at most 5. For a $+4$ approximation, the **DenseAPASP** algorithm decomposes graph G into 5 levels. Instead, we will show that we can decompose the graph G into 7 levels. Thus, define thresholds $s_1 = n^{6/7}, s_2 = n^{5/7}, \ldots, s_6 = n^{1/7}$. As length 5 paths are the most challenging to handle, let u, v be vertices and P a shortest path of length at exactly 5. If P has a vertex of degree at least s_2, since $2 + 6 + 7 = 15 = 2 * 7 + 1$, we may include the edge sets $\bigcup_{i=1}^{2} D_i \times D_6 = \tilde{O}(|E_9|)$. we can include these edges and obtain a $+4$ approximation following similar arguments to **DenseAPASP** (Figure 1).

Thus, let us assume $P \subset E_3$. If $P \subset E_5$, then the blocking levels $L(P, 7) \subset \{5, 6\}$ (Definition 3.4) so that we can obtain a $+4$ approximation via Lemma A.3. Then, assume $P \not\subset E_5$, so P has some edge in $E_j \setminus E_{j+1}$ for $j \in \{3, 4\}$. Denote the length 5 path $P = (u, u_2, u_3, v_3, v_2, v)$.

We will argue that if $\ell(u_2) \leq 4$, then we obtain a good additive approximation. Suppose $|N(u, 2)| \geq s_3$, so that from $p(u, D_3) \in N(u, 2)$ (Definition E.2) we compute an exact distance $\hat{\delta}(p(u, D_3), v) = \delta(p(u, D_3), v) \leq \delta(u, v) + 2$. When we add 2, we obtain a $+4$ approximation. Figure 7 gives an illustration

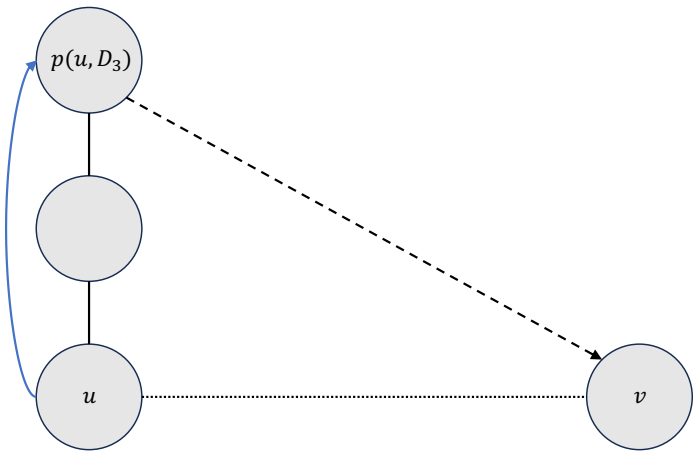


Figure 7: Estimate when $|N(u, 2)| \geq s_3$. Solid lines denote edges and dotted lines denote paths in G. Black dashed arrows denote computed distances estimates. Blue solid arrows denote a constant additive term.

ShortAdditiveAPASP (Algorithm 15) samples each dominating set D_i randomly with the algorithm **rDominate**, rather than following the greedy deterministic construction of **Dominate**. With high probability, if $|N(u, 2)| \leq s_3$, then $|C(u, 3)| = |N(u, 2) \cap D_4| = \tilde{O}(s_6)$ and if $\ell(u_2) \leq 4$, then Lemma A.3 guarantees that $\hat{\delta}(u_2^*, v) \leq \delta(u_2, v) + 3 = \delta(u, v) + 2$. We efficiently iterate over $C(u, 3)$ and update $\hat{\delta}(u, v) \leq 2 + \min_{w \in C(u,3)} \hat{\delta}(w, v)$ to obtain a $+4$ approximation. Figure 8 gives an illustration.

We can compute the sets $X(u, i), C(u, i)$ efficiently as we will terminate our **BFS** whenever s_i distinct vertices are encountered, which requires time at most $s_i^2 \leq s_3^2 = \tilde{O}(n^{8/7})$.

Then, we can assume $\deg(u_2), \deg(v_2) < s_4$. In particular, only one edge (u_3, v_3) is not in E_5. We now execute **Dijkstra** from every vertex $w \in D_j$ on graph $G_{i,w}$ containing (at least) the edges $E_j \cup E^* \cup (w \times V)$.

Since only one edge is not in E_5, the blocking levels of P (Definition 3.4), $L_B(P)$ cannot contain both $\{3, 4\}$. If the blocking levels $L_B(P)$ do not contain every level from $\{5, 6\}$, then we obtain a

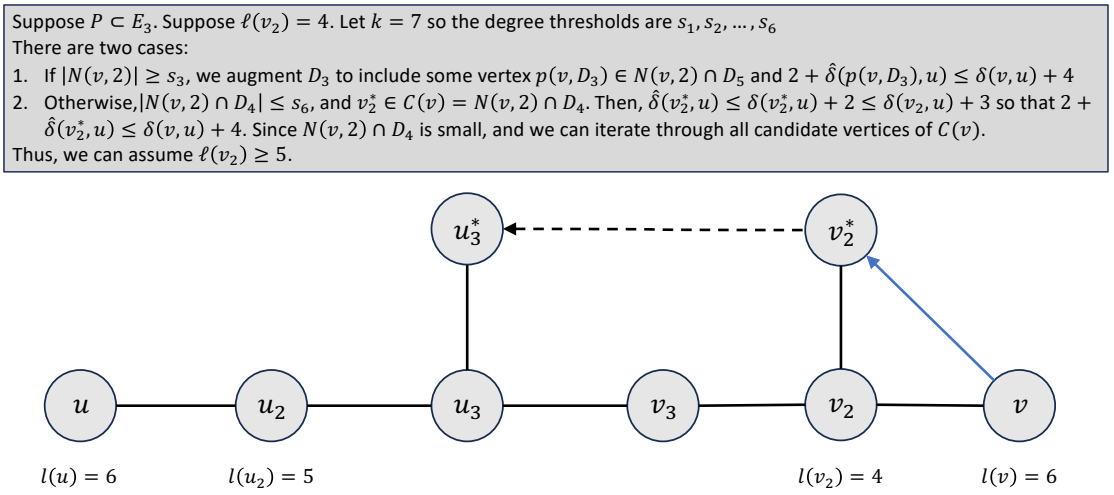


Figure 8: Imposing the degree restriction $\ell(v_2) \geq 5$

good additive approximation from Lemma A.2 as there are at most 2 blocking levels. On the other hand, suppose $\{5, 6\} \subset L_B(P)$. Note $\ell(u_3), \ell(v_3) < 5$, otherwise we obtain a $+4$ approximation as $L_B(P) = \{5, 6\}$. Assume without loss of generality that $\ell(v) \leq \ell(u)$ as in Definition 3.4. If $\ell(v) = 5$, then since $P \subset E_{\ell(u_3)}$, $\hat{\delta}(u_3^*, v) \leq \delta(u_3, v)$. Since the remaining edges are in E_5, G_{5,v^*}

Otherwise, $\ell(v) \geq 6$. If $\ell(v) = 7$, then $\ell(u) = \ell(v) = 7$ and we obtain a $+4$ approximation following the same argument as Theorem 3.3. Therefore, assume $\ell(v) = 6$. If $\ell(u_2) \geq 6$, then the blocking vertex $b(v, P) = u_3$ and we obtain a $+4$ approximation as $B(P) = \{u, v, u_3\}$.

We can therefore assume $\ell(u_2) = 5$. However, since this vertex has degree at most s_4, we use a depth 2 **BFS** in Line 209 to bypass the middle edge entirely, therefore obtaining a $+4$ approximation, as illustrated in Figure 9. The depth 2 **BFS** requires time $s_2 s_4 = \tilde{O}(n^{8/7})$.

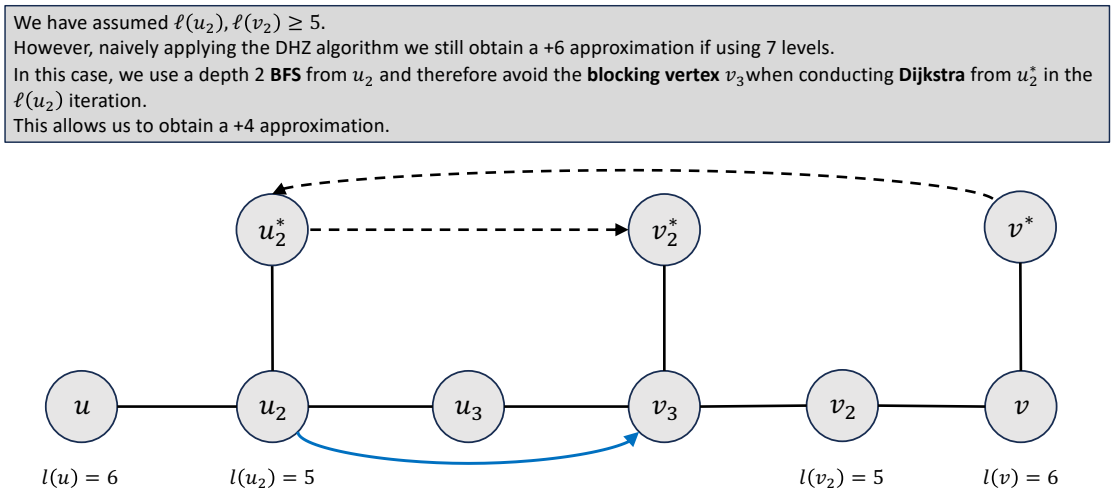


Figure 9: Using depth 2 **BFS** to bypass the central edge.

Algorithm In Phase 0, we initialize the distance estimates to the adjacency matrix, set parameters k, k_0 and decompose the graph G into k degree thresholds. To compute the dominating sets, we use

a random construction **rDecompose** which samples all vertices with identical probability. Finally, we check that the dominating sets are computed correctly and are not too large. If there is an error in the sample dominating sets, we return an exact **APSP** solution.

In Phase 1, we compute the sets $X(u,i)$, $C(u,i)$. For a given i, if $|N(u,r_i)| \geq s_i$ is large, we let $X(u,i)$ be an arbitrary subset of size s_i within this neighborhood. On the other hand, if $|N(u,r_i)| < s_i$ is small, we let $C(u,i) = N(u,r_i) \cap D_{i+1}$ given $|N(u,r_i) \cap D_{i+1}|$ is small. If we are unable to compute either $X(u,i)$ or $C(u,i)$, we return an exact **APSP** computation.

Then, for vertices u with degree at most s_{k_0+2}, we compute a depth 2 **BFS** in the graph G_{k_0+1}. Finally, we augment D_i with a hitting set for $X(u,i)$ for all u with large $N(u,r_i)$. Note that when we augment D_i, this does not affect the sets $r(u,D_i)$ for $\deg(u) \geq s_i$ or $q(w,D_i)$ for $w \in D_i$. These relationships are fixed at the computation of **rDecompose**.

In Phase 2, we begin by iteratively executing **Dijkstra** from each vertex $w \in D_j$ on the graph $G_{j,w}$ consisting of edges E_j, E^* as well as $w \times V$ and $D_{j_1} \times D_{j_2}$ for all indices $j + j_1 + j_2 \leq 2k+1$. Before the execution of **Dijkstra**, each vertex $w \in D_j$ examines its constituency (Definition 2.2) and updates its distance estimates where an improvement can be found.

Finally, each vertex u examines $p(u,D_i)$ and $C(u,i)$ where appropriate to update its distance estimates for all other vertices $v \in V$.

Preliminaries for Proving Theorem E.1

A key ingredient of our algorithm is computing small neighborhoods of each vertex. In the following lemma, we describe how to compute $X(u,i), C(u,i)$ efficiently.

Lemma E.3. *Suppose $\beta \geq 4$. Let $u \in V$ and $k_0 + 1 \leq i \leq k_0 + \frac{\beta}{2} - 1$. Algorithm 15 computes $X(u,i), C(u,i)$ in time $O\left(n^{2+\frac{1}{k}}\right)$ such that,*

1. *$X(u,i) \subset N(u,r_i)$ and $|X(u,i)| \geq s_i$ if $|N(u,r_i)| \geq s_i$.*
2. *$C(u,i) = N(u,r_i) \cap D_{i+1}$ if $|N(u,r_i) \cap D_{i+1}| < 12 s_{k-1} \log n$.*

Proof. Fix some iteration i and vertex u. We conduct a **BFS** on the graph G from u until we either encounter a vertex at depth $r_i + 1$ or s_i distinct vertices. If we encounter s_i distinct vertices first, then $|N(u,r_i)| \geq s_i$, so define $X(u,i)$ to be the distinct vertices encountered thus far. Otherwise, if we encounter a vertex at depth $r_i + 1$ first, then $|N(u,r_i)| < s_i$ and we can efficiently compute $C(u,i) = N(u,r_i) \cap D_{i+1}$ by iterating over the distinct vertices found and querying their membership in D_{i+1}.

To bound the time required to compute these sets, note that finding s_i distinct vertices requires time at most s_i^2. In particular, we can bound the time consumed by computing the **BFS** as,

$$s_i^2 \leq s_{k_0+1}^2 = n^{2-\frac{2(k_0+1)}{k}}$$

Note $k_0 = \beta - 2 = \frac{2}{3}(k-1) - 2 = \frac{2}{3}k - \frac{8}{3}$. Summing over all vertices u and iterations i, the total time consumed is,

$$O\left(n^{3-\frac{2(k_0+1)}{k}}\right) = O\left(n^{3-\frac{4k/3-10/3}{k}}\right) = O\left(n^{\frac{5}{3}+\frac{10}{3k}}\right) = O\left(n^{2+\frac{1}{k}}\right)$$

The last inequality holds whenever $k \geq 7$ or alternatively $\beta = \frac{2}{3}(k-1) \geq 4$. Computing $C(u,i)$ requires only s_i time as we only need to iterate over the vertices found in $N(u,r_i)$. □

Algorithm 15 $\textsc{ShortAdditiveAPASP}(G, \beta)$

Input : Unweighted, undirected graph $G = (V, E)$ with n vertices; approximation parameter β

Output: Distance estimate $\hat{\delta} : U \times V \to \mathbb{Z}$ such that $\delta(u, v) \le \hat{\delta}(u, v)$ for all $u, v \in V$ and $\hat{\delta}(u, v) \le \delta(u, v) + \beta$ whenever $\delta(u, v) \le \beta + 1$

Phase 0: Set up and Decompose Graph

$\hat{\delta}(u, v) \leftarrow \begin{cases} 1 & (u, v) \in E \\ \infty & \text{o/w} \end{cases}$

$k \leftarrow \frac{3\beta+2}{2}$ and $k_0 \leftarrow \beta - 2$.

$s_i \leftarrow n^{1-\frac{i}{k}}$ for all $1 \le i \le k-1$

$(D_1, D_2, \ldots, D_k), (E_1, E_2, \ldots, E_k), E^* \leftarrow \textsc{rDecompose}(G, (s_1, s_2, \ldots, s_{k-1}))$

for $1 \le i \le k$ **do**

  if *D_i does not dominate all neighborhoods $|N(v)| \ge s_i$ or $|D_i| \ge \frac{12n \log n}{s_i}$* **then**

    return $\textsc{APSP}(G)$

Phase 1: Compute $X(u), C(u)$ for all $u \in V$

for $u \in V$ **do**

  for $k_0 + 1 \le i \le k_0 + \frac{\beta}{2} - 1$ **do**

    $r_i \leftarrow \frac{\beta}{2} - (i - (k_0 + 1))$

    if $|N(u, r_i)| \ge s_i$ **then** compute $X(u, i)$ (see Lemma E.3);

    else if $|N(u, r_i) \cap D_{i+1}| \le 12 s_{k-1} \log n$ **then** compute $C(u, i) \leftarrow N(u, r_i) \cap D_{i+1}$;

    else return $\textsc{APSP}(G)$;

  if $\deg(u) \le s_{k_0+2}$ **then** $\hat{\delta}(u, v) \leftarrow \min(\hat{\delta}(u, v), \textsc{BFS}(G_{k_0+1}, u, 2))$ where $G_{k_0+1} = (V, E_{k_0+1})$;

for $k_0 + 1 \le i \le k_0 + \frac{\beta}{2} - 1$ **do** $D_i \leftarrow D_i \cup \textsc{HittingSet}(\{X(u, i)\})$ for all u such that $|X(u, i)| \ge s_i$;

Phase 2: Compute Distance Estimates

for $1 \le j \le k$ **do**

  for $w \in D_j$ **do**

    $G_{j,w} \leftarrow \left(V, E_j \cup \left(\bigcup_{j+j_1+j_2 \le 2k+1} D_{j_1} \times D_{j_2}\right) \cup E^* \cup (w \times V)\right)$

    $\hat{\delta}(w, v) \leftarrow \min(\hat{\delta}(w, v), \min_{u \in q(w, D_j)} 1 + \hat{\delta}(u, v))$ for all $v \in V$

    $\hat{\delta} \leftarrow \textsc{Dijkstra}(G_{j,w}, w, \hat{\delta})$

for $u, v \in V$ **do**

  for $k_0 + 1 \le i \le k_0 + \frac{\beta}{2} - 1$ **do**

    if $\delta(u, p(u, D_i)) \le r_i$ **then** $\hat{\delta}(u, v) \leftarrow \min(\hat{\delta}(u, v), r_i + \hat{\delta}(p(u, D_i), v))$;

    if $|C(u, i)| \le 12 s_{k-1} \log n$ **then** $\hat{\delta}(u, v) \leftarrow \min(\hat{\delta}(u, v), \min_{w \in C(u,i)} r_i + \hat{\delta}(w, v))$;

The following lemma bounds the probability that we must compute an exact **APSP** solution.

Lemma E.4. *Let $u \in V$ and $k_0 + 1 \le i \le k_0 + \frac{\beta}{2} - 1$ Define the following events for each u, i.*

$$A(u, i) = \{|N(u, r_i)| \ge s_i\}$$
$$B(u, i) = \{|N(u, r_i) \cap D_{i+1}| \ge 12 s_{k-1} \log n\}$$

For all u, let $E(u, i)$ denote the error event $E(u, i) = \neg A(u, i) \wedge B(u, i)$.

For each $1 \le i \le k$, let $E(i)$ denote the event that either D_i fails to dominate some neighborhood of size at least s_i or $|D_i| \ge \frac{12n \log n}{s_i}$.

Let $E = (\bigcup_{i=1}^{k} E(i)) \cup (\bigcup_{i,u} E(u, i))$ denote the event that any error event occurs.

Then, $\Pr(E) = O\left(\frac{1}{n^2}\right)$

Proof. Fix $u \in V$ and $k_0 + 1 \le qi \le k_0 + \frac{\beta}{2} - 1$, Consider the event $E(u,i)$.

$$\Pr(E(u,i)) = \Pr(\neg A(u,i) \wedge B(u,i) \le \Pr(B(u,i) \mid \neg A(u,i))$$

Conditioned on $\neg A(u,i)$, define the random variable $X = |N(u,r_i) \cap D_{i+1}|$. Since D_{i+1} is constructed by sampling each vertex probability $p = \frac{3\log n}{s_{i+1}}$, X is the sum of at most s_i Bernoulli random variables. Let Y be the sum of s_i Bernoulli random variables with the same probability p so that Y stochastically dominates X. Note that $\mathbb{E}[X] \le \frac{3\log n s_i}{s_{i+1}} = 3s_{k-1}\log n$. By a standard Chernoff bound,

$$\Pr(X > 12 s_{k-1}\log n) \le \Pr(Y > 12 s_{k-1}\log n) < \exp(-3 s_{k-1}\log n)$$

Thus, we can easily union bound over $O(n)$ events $E(u,i)$ as each occurs with inverse exponential probability. Note that $C(u,i)$ is defined from the randomly chosen D_i, and these sets are constructed before D_i is augmented by a deterministic hitting set of $X(u,i)$ in Line 210.

From Lemma 2.6, **rDominate** with high probability satifies the necessary requirements. To see this, consider some threshold s_i and vertex $\deg(v) \ge s_i$. The probability D_i fails to dominate s_i with P chosen above is at most,

$$(1-p)^{s_i} \le \exp(-3\log n) \le n^3$$

We union bound over all u, i to obtain the probability $\tilde{O}(1/n^2)$.

Second, the probability that $|D_i| \ge \frac{12n\log n}{s_i}$ is small by an identical argument to the error bound for E_i. Since we bound over $O(1)$ events, these also occur with negligible probability. We conclude the proof by union bounding over all the error events. □

Lemma E.5. *Suppose that the error event E does not occur. Then, Algorithm 15 outputs $\hat{\delta}$ such that,*

$$\delta(u,v) \le \hat{\delta}(u,v) \le \delta(u,v) + \beta$$

for all $\delta(u,v) \le \beta + 1$.

We delay the proof of Lemma E.5, instead beginning with a few useful lemmas that will be crucial to our proof. In the following lemma, we prove that a reasonable additive approximation is obtained whenever $A(u,i)$ is true. That is, we assume $|N(u,r_i)| \ge s_i$ is large.

Lemma E.6. *Let $k_0+1 \le i \le k_0 + \frac{\beta}{2} - 1$. Let u, v be a pair of vertices and $P \subset E_{k_0+1}$ be a shortest path. Suppose $|N(u,r_i)| \ge s_i$ where $r_i = \frac{\beta}{2} - (i - (k_0+1))$. Then, after Line 219 we have,*

$$\hat{\delta}(u,v) \le \delta(u,v) + \beta$$

Proof. Suppose $|N(u,r_i)| \ge s_i$, then there is some vertex $w \in X(u,i) \cap D_i \subset N(u,r_i) \cap D_i$ so that $\delta(u, p(u,D_i)) \le r_i$. In particular,

$$\hat{\delta}(u,v) \le r_i + \hat{\delta}(p(u,D_i), v)$$

Let us examine the quantity $\hat{\delta}(p(u,D_i),v)$. Since after Line 210 the shortest path in G between $u, p(u,D_i)$ is at most r_i, we begin by showing that the path between $u, p(u,D_i)$ exists in the graph $G_{i,u}$. If there is an edge missing from E_i, let w be the vertex closest to u such that $\deg(w) > s_{i-1}$. Therefore, $N(w) \cap D_{i-1} \neq \emptyset$ and $N(w) \cap D_i \neq \emptyset$. If w is not the vertex immediately preceding $p(u,D_i)$, then this violates the property that $p(u,D_i)$ is the nearest vertex in D_i to u. Otherwise, if w is the vertex immediately preceding $p(u,D_i)$, then the edge $(w, p(u,D_i))$ is accessible in the

graph $G_{i,p(u,D_i)}$. Thus, the shortest path of length r_i between $p(u, D_i), u$ is available in the i-th iteration.

Consider now the a path P' from $p(u, D_i)$ to v of length at most $|P| + r_i$. By our arguments above and $P \subset E_{k_0+1}$, the path P' is also in E_{k_0+1}. From Lemma A.3, $L(P') \subset \{k_0 + 1, \ldots, i - 1\}$ has size at most $i - (k_0 + 1)$, so we can conclude the total error is at most,

$$\hat{\delta}_i(p(u, D_i), v) \leq \delta(u, v) + r_i + 2(i - (k_0 + 1))$$

Then,

$$\begin{aligned}\hat{\delta}(u, v) &\leq r_i + \hat{\delta}(p(u, D_i), v) \\ &\leq \delta(u, v) + 2r_i + 2(i - (k_0 + 1)) \\ &= \delta(u, v) + 2\left(r_i + (i - (k_0 + 1))\right) \\ &\leq \delta(u, v) + \beta\end{aligned}$$

□

Next, we prove correctness under the assumption that $B(u)$ is false. In particular, assume the set $|C(u, i)| = |N(u, r_i) \cap D_{i+1}| \leq 12 s_{k-1} \log n$ is small.

Lemma E.7. *Suppose that the error event E does not occur.*

Let $k_0 + 1 \leq i \leq k_0 + \frac{\beta}{2} - 1$. Let u, v be a pair of vertices with a shortest path $P = (u, u_2, u_3, \ldots, v_3, v_2, v)$ with all edges in E_{k_0+1}. Suppose $|N(u, r_i) \cap D_{i+1}| \leq 12 s_{k-1} \log n$. If $\ell(u_{r_i}) \leq i + 1$ then, after Line 220 we have,

$$\hat{\delta}(u, v) \leq \hat{\delta}(u, v) + \beta$$

Proof. Since $\ell(u_{r_i}) \leq i + 1$, there is some vertex $z \in N(u, r_i) \cap D_{i+1}$ such that $z = r(u_{r_i}, D_{i+1})$ as $u_{r_i} \in N(u, r_i - 1)$. Since $P \subset E_{k_0+1}$, then after the $(i + 1)$-th iteration,

$$\hat{\delta}(z, v) \leq \delta(u_{r_i}, v) + 2(i - k_0) + 1$$

where we apply Lemma A.3 with $L(P) \subset \{k_0 + 1, \ldots, i\}$ has size at most $i - k_0$.

Then, after Line 220, since $z \in N(u, r_i) \cap D_{i+1} = C(u, i)$,

$$\hat{\delta}(u, v) \leq r_i + \hat{\delta}(z, v) \leq \delta(u_{r_i}, v) + r_i + 2(i - k_0) + 1$$

Now, $\delta(u_{r_i}, v) = \delta(u, v) - r_i + 1$ so that,

$$\hat{\delta}(u, v) \leq \delta(u, v) + 2(i - k_0 + 1) \leq \delta(u, v) + \beta$$

as $i + 1 \leq k_0 + \frac{\beta}{2}$. □

From Lemmas E.6 and E.7 we can place an upper bound on the levels of many vertices of the shortest path. The following lemma shows that we obtain a good additive approximation when these bounds hold.

Lemma E.8. *Let $P = (u, u_2, \ldots, v_2, v)$ be a shortest path of length at most $\beta + 1$ such that $P \subset E_{k_0+1}$. Suppose the degrees $\ell(u_{r_i}), \ell(v_{r_i}) > i + 1$ for all $k_0 + 1 \leq i \leq k_0 + \frac{\beta}{2} - 1$.*

Then, Algorithm 15 obtains an approximation such that $\hat{\delta}(u, v) \leq \delta(u, v) + \beta$.

Proof. As before, let $\deg(P) = \max_{v \in P} \deg(v)$ be the maximum degree of any vertex in P and $\ell(P) = \min_{v \in P} \ell(v)$ the minimum level of any vertex in P. In the following, we will frequently use (some variant of) the following equality,

$$k - k_0 = \left(\frac{3}{2}\beta + 1\right) - (\beta - 2) = \frac{\beta}{2} + 3 \tag{1}$$

Let us begin by assuming P has length exactly $\beta + 1$ so that we can label the vertices $P = (u, u_2, \dots, u_{\beta/2+1}, v_{\beta/2+1}, \dots, v_2, v)$.

If $P \subset E_{k_0+3}$, then $L(P,k) \subset \{k_0+3, \dots, k-1\}$ is a set of size at most $k - k_0 - 3 = \frac{\beta}{2}$, so we may apply Lemma A.3 and obtain a $+\beta$-approximation. Thus, in the following, assume P has an edge in $E_{k_0+1} \setminus E_{k_0+3}$. In particular, there are at least two vertices $x \in P$ with $\deg(x) > s_{k_0+2}$.

Suppose $|L_B(P,k)| \le \frac{\beta}{2}$. In the k-th iteration, as $D_k = V$, we have by A.2, $\hat{\delta}_k(u,v) \le \delta(u,v) + \beta$. Thus, we can assume $|L_B(P,k)| > \frac{\beta}{2}$.

By assumption, the levels of vertices u_{r_i}, v_{r_i} are restricted. Recall that $r_i = \frac{\beta}{2} - (i - (k_0+1))$ so that,

$$r_{k_0+1} = \frac{\beta}{2} - ((k_0+1) - (k_0+1)) = \frac{\beta}{2}$$
$$r_{k_0+\frac{\beta}{2}-1} = \frac{\beta}{2} - ((k_0 + \frac{\beta}{2} - 1) - (k_0+1)) = 2$$

In particular, other than the two middle vertices, $u_{\frac{\beta}{2}+1}, v_{\frac{\beta}{2}+1}$, the remaining vertices have level at least $i + 2 \ge k_0 + 3$, so that we can assume both central vertices have degree at least s_{k_0+2}. Furthermore, at most one of these vertices can be a blocking vertex. Suppose for contradiction that both are, so that one, for example $u_{\frac{\beta}{2}+1}$, has level k_0+1 and the other level k_0+2. However, in this case $P \subset E_{k_0+2}$ so that $u_{\frac{\beta}{2}+1}$ cannot be a blocking vertex. In particular, if $L(P,k)$ does not contain all levels $\{k_0+3, \dots, k-1\}$ then we can already obtain a β approximation by Lemma A.3.

Therefore, let $L_B(P)$ denote the blocking vertices of P. Since $k_0 + 3 \in L_B(P)$, we have either $u_{\beta/2}, v_{\beta/2} \in B(P)$. Without loss of generality, suppose $u_{\beta/2} \in B(P)$.

In Line 209, since $\deg(u_{\beta/2}) < s_{k_0+2}$ and $P \subset E_{k_0+1}$, we compute $\hat{\delta}(u_{\beta/2}, v_{\beta/2+1}) = 2$. Then, if $u^*_{\beta/2} = r(u_{\beta/2}, D_{k_0+3})$, in Line 215, we have $\hat{\delta}(u^*_{\beta/2}, v_{\beta/2+1}) \le 3$. We claim that,

$$\hat{\delta}_{k_0+3}(u^*_{\beta/2}, x) \le \delta(u_{\beta/2}, x) + 1$$

for all $x \in P$. We have shown this already for $x \in \{u_{\beta/2+1}, v_{\beta/2+1}\}$. For $x \in \{u, u_2, \dots, u_{\beta/2}\}$, this follows simply from the assumption that all these vertices have level at least k_0+3 so that $P_{w,x} \subset E_{k_0+3}$. For $x \in \{v_{\beta/2}, \dots, v_2, v\}$, when executing **Dijkstra** we take the edge $(u^*_{\beta/2}, v_{\beta/2+1})$ and the remaining edges of P in E_{k_0+3}.

This allows us to make the following claim stronger than Lemma A.2. For any $w, x \in P$ where $\ell(w) \ge k_0 + 3$,

$$\hat{\delta}_{\ell(w)}(w^*, x) \le \delta(w,x) + 2|L_B(P_{w,x})| - 1 \tag{2}$$

where we have shown the base case for $\ell(u_{\beta/2}) = k_0 + 3$. The inductive step will follow similarly. Let $j \ge k_0 + 3$ and $\ell(w) = j$. Let $y = b(w, P_{w,x})$ be the blocking vertex so that $\ell(y) < \ell(w)$. Then,

$$\begin{aligned}
\hat{\delta}_{\ell(w)}(w^*, x) &\le \hat{\delta}_{\ell(y)}(w^*, y^*) + 1 + \delta(y,x) \\
&\le \delta(w,y) + 2|L_B(P_{y,w})| + 1 + \delta(y,x) \\
&\le \delta(w,x) + 2|L_B(P_{w,x})| - 1
\end{aligned}$$

where we have used $|L_B(P_{w,x})| > |L_B(P_{y,w})|$ as $\ell(y) \in L_B(P_{w,x}) \setminus L_B(P_{y,w})$. Then, in the k-th iteration, let w be the closest vertex to v such that $\deg(w) \geq s_{k-1}$ so that,

$$\hat{\delta}(u,v) \leq \hat{\delta}_{k-1}(u,w^*) + 1 + \delta(w,v) \leq \delta(u,v) + 2|L_B(P_{w,u})| \leq \delta(u,v) + \beta$$

as $L_B(P_{w,u}) = \{k_0+3,\ldots,k-2\} \cup \{\min(\ell(u_{\beta/2+1}), \ell(v_{\beta/2+1})\}$ has size $\frac{\beta}{2}$.

To conclude, if P has length less than β, then $P \subset E_{k_0+3}$, so we obtain a β approximation by Lemma A.3. □

Finally, we claim that all the estimates produced by Algorithm 15 are feasible.

Lemma E.9. *Suppose error event E does not occur. Then, Algorithm 15 obtains an approximation such that $\delta(u,v) \leq \hat{\delta}(u,v)$ for all $u,v \in V$.*

Proof. We analyze each instance where the distance estimates are updated. In Phase 0, we initialize the distance estimates to the adjacency matrix. In Phase 1, we compute length 2 paths in a sub-graph $G_{k_0+1} \subset G$. Next, the distance estimates are updated in Phase 2 in computing $\textsc{Dijkstra}(G_{j,w}, w, \hat{\delta})$ for all $w \in D_j$. Note that this is identical to Algorithm **DenseAPASP**, and produces feasible distance estimates.

Line 36 does not violate feasibility as if $u \in q(w, D_j)$ then $(u,w) \in E$ so that,

$$\hat{\delta}(w,v) \geq 1 + \hat{\delta}(u,v) \geq 1 + \delta(u,v) \geq \delta(w,v)$$

by the triangle inequality, where we have also used that the previous estimates were feasible.

Finally, since $p(u, D_i) \in N(u, r_i)$ and $C(u,i) \subset N(u,r_i)$, the final distance estimates are feasible as for any $z \in N(u,r_i)$, the updated distance estimate,

$$\hat{\delta}(u,v) = r_i + \hat{\delta}(z,v) \geq r_i + \delta(z,v) \geq \delta(u,v)$$

where in the first inequality we used that the previous estimates are feasible and in the second we used the triangle inequality. □

Proof of Theorem E.1

To prove the correctness of Algorithm 15, it suffices to prove Lemma E.5. Indeed, whenever E occurs, we compute an exact **APSP** solution, which is obviously correct. If E does not occur, Lemma E.9 guarantees that each estimate is feasible, and is produced by some path in G.

Proof. Let u,v be a pair of vertices with shortest path P length at most $\beta+1$. We proceed by case analysis.

Case 1: $P \subset E_{k_0+1}$ Suppose the assumptions of Lemma E.8 are not met, otherwise we obtain an accurate approximation. That is, for some i, $\ell(u_{r_i}) \leq i+1$. In particular, $u^* = r(u, D_{i+1})$ must be in $N(u,r_i) \cap D_{i+1}$. Then, if $|N(u,r_i) \cap D_{i+1}| \leq 12 s_{k-1} \log n$, we obtain a correct estimate by Lemma E.7. Otherwise, since $E(u,i)$ does not occur, $|N(u,i)| \geq s_i$, and we obtain a correct estimate by Lemma E.6.

Case 2: $P \not\subset E_{k_0+1}$ We begin with the special case $\beta = 4$, generalizing to $\beta \geq 6$ later.

Warm up: $\beta = 4$ Note $k = 7, k_0 = 2$. Let z be the vertex of maximum degree. Let $w = b(u, P)$ be the vertex closest to v such that $\deg(w) \geq s_{k_0+4} = s_6$. Since $P = P_{u,w} \cdot P_{w,v}$, the sub-path $P_{u,w}$ must contain the vertex z. Let $z^* = r(z, D_{\ell(z)}), w^* = r(w, D_{k_0+4})$. Since $P \subset E_{\ell(z)}$, we have $\hat{\delta}_{\ell(z)}(z^*, u) \leq \delta(z, u)+1$ and $\hat{\delta}_{\ell(z)}(z^*, w^*) \leq \delta(z, w)+2$. Furthermore, since $\ell(z)+(k_0+4)+(k_0+5) \leq 3k_0 + 9 = 2k + 1$, in the $k_0 + 5$-th iteration,

$$\hat{\delta}(u, v) \leq \hat{\delta}_{\ell(z)}(u, z^*) + \hat{\delta}_{\ell(z)}(z^*, w^*) + 1 + \delta(w, v) \leq \delta(u, v) + 4$$

as desired.

Generalization to $\beta \geq 6$ Suppose $\ell(v) \leq \ell(u)$. Recall from Lemma A.2 the blocking vertices $B(P) = \{x_0, x_1, \ldots, x_t\}$ and levels $L_B(P)$ of path P.

Let $a = \min_{\ell(x_i) \leq k_0+5} i$ be the minimum index of an element in the blocking set such that $\ell(x_i) \leq k_0 + 5$. Since $k \geq \ell(x_1) > \ell(x_2) > \ldots > \ell(x_t) \geq 1$, we can upper bound,

$$a \leq k - (k_0 + 5) = \frac{\beta}{2} - 2$$

Since P is not contained in E_{k_0+1}, we can assume a exists and $a \geq 1$.

Let $x_a \in B(P)$ be the corresponding vertex in $B(P)$. Since P has an edge not in E_{k_0+1}, the last blocking vertex of minimum level must have $\ell(x_t) \leq k_0$. Let $v^* = r\left(v, D_{\ell(v)}\right)$ for any vertex v. Since $P \subset E_{\ell(x_t)}$, we again have $\hat{\delta}_{\ell(x_t)}(x_t^*, x_a^*) \leq \delta(x_t, x_a) + 2$ and $\hat{\delta}_{\ell(x_t)}(x_t^*, x_{a+1}^*) \leq \delta(x_t, x_{a+1}) + 2$.

Consider the $\ell(x_a)$-th iteration. The edges $D_{\ell(x_t)} \times D_{\ell(x_{a+1})}$ are in $G_{\ell(x_a),x_a}$ as,

$$\ell(z) + \ell(x_{a+1}) + \ell(x_a) \leq k_0 + (k_0 + 4) + (k_0 + 5) = 3k_0 + 9 = 3\left(k - \frac{\beta}{2} - 3\right) + 9 = 2k + 1$$

Then, since $x_t \in P_{x_a,x_{a+1}}$,

$$\begin{aligned}
\hat{\delta}_{\ell(x_a)}(x_a^*, x_{a-1}) &\leq \hat{\delta}_{\ell(x_t)}(x_a^*, x_t^*) + \hat{\delta}_{\ell(x_t)}(x_t^*, x_{a+1}^*) + 1 + \delta(x_{a+1}, x_{a-1}) \\
&\leq \delta(x_a, x_t) + \delta(x_t, x_{a+1}) + \delta(x_{a+1}, x_{a-1}) + 5 \\
&\leq \delta(x_a, x_{a-1}) + 5
\end{aligned}$$

Then, following a similar argument to the inductive step of Lemma A.2, we claim the following for all $1 \leq j \leq a$.

$$\hat{\delta}_{\ell(x_j)}(x_j^*, x_{j-1}) \leq \delta(x_j, x_{j-1}) + 2(2 + (a - j)) + 1$$

where we have established the base case $j = a$ above. We now proceed by induction for $j < a$. Consider an execution of **Dijkstra** from x_j^* in $G_{\ell(x_j),x_j^*}$. Let x_{j+1} be the blocking vertex from the previous iteration. We take the edges (x_j^*, x_{j+1}^*), $(x_{j+1}^*, x_{j+1}) \in E^*$, and the remaining edges in $E_{\ell(x_j)}$. By induction, we have,

$$\begin{aligned}
\hat{\delta}_{\ell(x_j)}(x_j^*, x_{j-1}) &\leq \hat{\delta}_{\ell(x_{j+1})}(x_j^*, x_{j+1}^*) + 1 + \delta(x_{j+1}, x_{j-1}) \\
&\leq \delta(x_j, x_{j+1}) + 2(2 + (a - (j + 1))) + 3 + \delta(x_{j+1}, x_{j-1}) \\
&\leq \delta(x_j, x_{j-1}) + 2(2 + (a - j)) + 1
\end{aligned}$$

Thus, we have,

$$\hat{\delta}_{\ell(v)}(v^*, u) = \hat{\delta}_{\ell(x_1)}(x_1^*, x_0) \leq \delta(v, u) + 2(a + 1) + 1$$

From u, we take the edge (u, v^*), followed by $(v^*, v) \in E^*$ so that,

$$\begin{aligned}\hat{\delta}(u,v) &\leq \hat{\delta}_{\ell(v)}(u,v^*) + 1 \\ &\leq \delta(u,v) + 2(a+1) + 2 \\ &\leq \delta(u,v) + \beta\end{aligned}$$

□

Thus, to prove Theorem E.1, it only remains to analyze the performance of Algorithm 15.

Proof. (Time Complexity)

We separately analyze the complexity of each phase. By Lemma E.4, E occurs with probability $O\left(\frac{1}{n^2}\right)$. Since in this case Algorithm 15 requires $\tilde{O}(n^3)$ time, this contributes at most $\tilde{O}(n)$ to the expected run-time. Thus, in the following assume that E does not occur.

Phase 0 rDecompose requires $O(m+n) = O(n^2)$ time to sample each vertex set randomly and construct the edge sets E_i. Verifying that D_i is of the appropriate size and dominates $|N(v)| \geq s_i$ requires time $\tilde{O}(n^2)$.

Phase 1 By Lemma E.3, computing the sets $X(u,i), C(u,i)$ require time at most $\tilde{O}\left(n^{2+\frac{1}{k}}\right)$. By Lemma 2.4, augmenting D_i in Line 210 requires time $\tilde{O}(n^2)$.

Line 209 requires time,

$$\tilde{O}\left(ns_{k_0+2}s_{k_0}\right) = \tilde{O}\left(n^{3-\frac{2k_0+2}{k}}\right) = \tilde{O}\left(n^{2+\frac{1}{k}}\right)$$

whenever $\beta \geq 4$ as shown in Lemma E.3.

Phase 2 First, we bound the complexity of each call to Dijkstra as in the proof of Lemma 3.2 for DenseAPASP. Fix some j. Then, $|E_j| \leq ns_{j-1} = n^{2-\frac{j-1}{k}}$ and $|D_{j_1} \times D_{j_2}| = \tilde{O}\left(\frac{n}{s_{j_1}} \times \frac{n}{s_{j_2}}\right) = \tilde{O}\left(n^{\frac{j_1+j_2}{k}}\right) = \tilde{O}\left(n^{2-\frac{j-1}{k}}\right)$ as $j_1 + j_2 \leq 2k+1-j$. Note for any fixed w that $(w \times V) \cup E^*$ has size $\tilde{O}(n)$. Summing over $|D_j| = \tilde{O}\left(\frac{n}{s_j}\right) = \tilde{O}\left(n^{\frac{j}{k}}\right)$, all invocations of Dijkstra require time $\tilde{O}\left(n^{2+\frac{1}{k}}\right)$.

Line 215 requires time $\tilde{O}(n^2)$ as every vertex u is in at most one $q(w, D_j)$ for any fixed j.

Finally, Line 219 requires time $\tilde{O}(n^2)$ over all u, v, i. By the bound on $|C(u,i)| = \tilde{O}(n^{1/k})$, Line 220 requires time $\tilde{O}\left(n^{2+\frac{1}{k}}\right)$. In particular, this is the overall time bound when E does not occur, giving the desired result by substituting $k = \frac{3\beta}{2} + 1$. □

We have the following Corollary.

Recall the following result of Roditty [Rod23].

Lemma E.10 (Theorem 5.1 of Roditty [Rod23]). *Let $k' \geq 6$ be an even integer. There is an algorithm that computes an additive $k'-2$ approximation in $\tilde{O}(n^{2-\frac{2}{k'+2}}m^{\frac{2}{k'+2}})$ expected time, for every $u, v \in V$ for which $\delta(u,v) \leq k'$.*

Corollary E.11. *Let $k \geq 4$ be an even integer. Then, we can compute a $(2,0)$-approximation for distances $\delta(u,v) \geq k$ combinatorially in expected time*

$$\tilde{O}\left(\min\left(n^{2-\frac{2}{k+4}}m^{\frac{2}{k+4}}, n^{2+\frac{1}{2(k-1)}}, n^{2+\frac{2}{3k+2}}\right)\right)$$

In particular, we output $\hat{\delta}$ such that $\delta(u,v) \leq \hat{\delta}(u,v)$ for all u,v and $\hat{\delta}(u,v) \leq 2\delta(u,v)$ whenever $\delta(u,v) \geq k$.

Proof. We combine Theorem 4.1, Theorem E.1, Lemma 3.2 and Lemma E.10. If $n^{2-\frac{2}{k+4}}m^{\frac{2}{k+4}}$ is the minimum term, we invoke the algorithm of Lemma E.10 with $k' = k+2$ and the algorithm **SparseAPASP** of *Lemma* 3.2 with $\beta = k+2$. For paths of length $k \leq \delta(u,v) \leq k+2$, we obtain a $+k$ approximation from Lemma E.10. For paths of length $\delta(u,v) \geq k+2$, we obtain a $+(k+2)$ approximation from **SparseAPASP**. In either case, we obtain a $(2,0)$ approximation for all $\delta(u,v) \geq k$. The overall running time is,

$$\tilde{O}(n^{2-\frac{2}{k+4}}m^{\frac{2}{k+4}})$$

On the other hand, if $n^{2+\frac{1}{2(k-1)}}$ is the minimum term, we invoke Algorithm 4 and obtain a $(2,0)$ approximation for paths of length at least k following Theorem 4.1.

Finally, if $n^{2+\frac{2}{3k+2}}$ is the minimum term, we call Theorem E.1 with $k' = k+1$ and **DenseAPASP** with $\beta = k+2$. For any path of length at least $k+2$, the estimate from **DenseAPASP** is a $(2,0)$-approximation. For paths of length $k \leq \delta(u,v) \leq k+1$, the estimate from **ShortAdditiveAPASP** is a $(2,0)$-approximation. The overall computation time can be bounded by,

$$\tilde{O}\left(n^{2+\frac{2}{3k+2}} + n^{2+\frac{2}{3(k+2)-2}}\right) = \tilde{O}\left(n^{2+\frac{2}{3k+2}}\right)$$

□

Note that for $k=6$, $n^{2+\frac{1}{2(k-1)}} = n^{21/10} = n^{2+\frac{2}{3k-2}}$. For larger k, the former term is smaller and we apply Theorem 4.1. However, for $k = 4, 5$, Algorithm 15 gives the best bound (see Table 1).