Improved Approximation Algorithms for n-Pairs Shortest Paths

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

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

Virginia Vassilevska Williams
Department of Electrical Engineering and Computer Science and CSAIL, MIT, Cambridge, MA, USA

Abstract

Let $G = (V, E)$ be a graph with $n = |V|$ nodes and $m = |E|$ edges. The t-Pairs Shortest Paths problem, introduced by Cohen [FOCS'93; SICOMP'99], asks to approximate the distances between t prespecified pairs of vertices. Recently, this problem has received renewed attention, particularly in the case where $t = \Theta(n)$: the n-Pairs Shortest Paths problem. In this setting, new algorithms and conditional lower bounds have been developed by Dalirrooyfard, Jin, Vassilevska Williams, and Wein [FOCS'22], and Chechik, Hoch, and Lifshitz [SODA'25].

In this paper, we present the first algorithm for the n-Pairs Shortest Paths problem in *weighted* undirected graphs that achieves a $(2-\alpha)k$-approximation, for constant $\alpha > 0$, that runs in $\tilde{O}(mn^{1/k} + n^{1+2/k})$ time. Specifically, we present a $1.622k$-approximation, improving upon the $(2k-3)$-approximation of Chechik, Hoch, and Lifshitz [SODA'25] for graphs that are not super sparse, which answers in the affirmative the open question posed by them. We also develop improved approximation algorithms with better tradeoffs for unweighted graphs and dense weighted graphs that improve upon the results of Dalirrooyfard *et al.* and Chechik, Hoch, and Lifshitz.

Our main technical contribution is the new *heavy-edge* technique. Using this technique, we transform an algorithm with an approximation guarantee that depends on W_{uv}, the weight of the heaviest edge on the shortest path between u and v, into an algorithm with purely multiplicative approximation that does not depend on W_{uv}.

2012 ACM Subject Classification Theory of computation → Approximation algorithms analysis; Theory of computation → Shortest paths; Mathematics of computing → Graph algorithms

Keywords and phrases Fine-grained complexity, Graph algorithm, Graph distances, n pairs shortest paths, all pairs shortest paths, distance oracles

Digital Object Identifier 10.4230/LIPIcs.ESA.2026.66

Related Version *Full Version*: https://arxiv.org/abs/2607.02443

Funding *Liam Roditty*: Supported in part by BSF grants 2016365 and 2020356.
Virginia Vassilevska Williams: Supported in part by NSF CAREER Award 1651838, NSF Grants CCF-1909429 and CCF-2129139, BSF grants 2016365 and 2020356, a Google Research Fellowship, and a Sloan Research Fellowship.

1 Introduction

Let $G = (V, E)$ be a graph with $n = |V|$ nodes and $m = |E|$ edges, and let $d(u, v)$ be the distance between u and v, for every $u, v \in V$. Let $I \subseteq V \times V$ be a set of t prespecified pairs of vertices. In the t-Pairs Shortest Paths (t-PSP) problem, the goal is to efficiently compute (or approximate) the shortest path distance $d(u, v)$ for each pair $(u, v) \in I$. The

t-PSP problem was first studied in the 1990s by Cohen [9] and then by Aingworth, Chekuri, Indyk, and Motwani [5]. Aingworth *et al.* [5] presented an algorithm that computes a $(1, 2)$-approximation[1] for unweighted, undirected graphs in $\tilde{O}(n^{1.5}\sqrt{t} + n^2)$ time.[2]

A natural approach for approximating the t-PSP problem is to use the celebrated Approximate Distance Oracles (ADOs) of Thorup and Zwick [17]. These oracles are constructed for weighted undirected graphs in $\tilde{O}(mn^{1/k})$ time, use $\tilde{O}(n^{1+1/k})$ space, and guarantee a $(2k-1)$-approximation for distances between any pair of nodes. Once built, they can answer each distance query in $O(k)$ time, so solving the t-PSP problem using an ADO is straightforward and requires $\tilde{O}(mn^{1/k} + t)$ total time.

ADOs are primarily designed to optimize *space efficiency* while supporting distance queries for *all* pairs of vertices. In contrast, the t-PSP problem only requires distance queries between a specific set of t vertex pairs, and the main objective is to minimize *running time*, not space. This distinction suggests that the generality of ADOs may be unnecessary in the t-PSP setting, and that one can hope to design faster algorithms that achieve improved time-approximation tradeoffs for t-PSP.

Interestingly, although the t-PSP problem was first introduced in the 1990s, it remained largely unexplored for over two decades. In the meantime, closely related problems, primarily focused on space-efficient representations such as pairwise spanners, distance preservers, and reachability preservers, received significantly more attention (see, for example, [10, 1, 15]).

Recently, the t-PSP problem received renewed attention in the field of fine-grained complexity. Abboud *et al.* [3, 2] and independently Jin and Xu [12] established several conditional lower bounds for t-PSP based on the 3SUM hypothesis. In particular, Jin and Xu [12] proved that there is no $\tilde{O}(m^{1+\frac{1}{2k+1}-\varepsilon})$-time algorithm that achieves a $(k-\delta)$-approximation for the m-PSP problem in graphs where $m = \Theta(n^{1+\frac{1}{2k-2}})$.

Dalirrooyfard *et al.* [11] studied the n-PSP problem, which is a special case of t-PSP with $t = n$. They established several conditional lower bounds. Among their results, they showed that, assuming the combinatorial $4k$-Clique Hypothesis,[3] there is no $(1 + 1/k - \delta)$-approximation algorithm for n-PSP that runs in time $\tilde{O}(m^{2-\frac{2}{k+1}} n^{\frac{1}{k+1}-\varepsilon})$. This lower bound *matches* an upper bound that follows from the work of Agarwal [4] on distance oracles with non-constant query time.

[11] also considered n-PSP algorithms for unweighted undirected graphs. They presented a $(2+\varepsilon, \beta)$-approximation algorithm for unweighted undirected graphs running in $\tilde{O}(m+n^{3/2+\varepsilon})$ time, which nearly matches their lower bound for sparse graphs. For general approximations, they [11] designed a $(2k-2, 1)$-approximation algorithm running in $\tilde{O}(mn^{1/k})$ time, improving the classic $(2k-1)$-approximation of Thorup and Zwick for the special case of n-PSP.

Very recently, Chechik, Hoch, and Lifshitz [8] established new upper bounds for n-PSP by presenting several approximation algorithms with improved tradeoffs. In particular, they present an algorithm that computes a $(2k-3)$-approximation for n-PSP in *weighted* undirected graphs in $\tilde{O}(mn^{1/k})$ expected time, for any integer $k \geq 3$. This improves upon the $(2k-2, 1)$-approximation of Dalirrooyfard *et al.* [11] that worked only in unweighted graphs. In light of their $(2k-3)$-approximation for n-PSP in weighted graphs, Chechik, Hoch, and Lifshitz posed the following question:

---

[1] An estimation $\hat{d}(u, v)$ of $d(u, v)$ is an (α, β)-approximation if $d(u, v) \leq \hat{d}(u, v) \leq \alpha d(u, v) + \beta$. We denote purely multiplicative approximation ($\beta = 0$) with (α)-approximation.

[2] $\tilde{O}$ omits polylogarithmic factors.

[3] The combinatorial $4k$-Clique Hypothesis asserts that there is no $O(n^{k-\varepsilon})$-time combinatorial algorithm for detecting a k-clique, for any $\varepsilon > 0$.

▶ **Question 1** ([8]). *Is it possible to break the multiplicative* $(2k - O(1))$*-approximation barrier for* n*-PSP with an algorithm running in* $\tilde{O}(mn^{1/k})$ *time?*

The main result of this paper is an affirmative answer to this question for *weighted* graphs that are not super sparse. Specifically, we present a $1.622k$-approximation algorithm for n-PSP in weighted graphs that runs in $\tilde{O}(mn^{1/k} + n^{1+2/k})$ time.

▶ **Theorem 1.** *Let* $G = (V, E, \ell)$ *be a weighted undirected graph, where* $\ell : E \to \mathbb{R}_{\geq 0}$*, and let* $k \geq 4$ *be an integer parameter. Let* I *be a set of* n *vertex pairs. There is an algorithm that computes in* $\tilde{O}(mn^{1/k} + n^{1+2/k})$ *time, for every* $\langle u, v\rangle \in I$*, an estimate* $\hat{d}(u, v)$ *such that* $d(u, v) \leq \hat{d}(u, v) \leq 1.622k \cdot d(u, v)$*.*

This theorem improves on the $(2k - 3)$-approximation algorithm of [8] for all $k \geq 8$. We also note that our new techniques can be used to improve the dense-graph n-PSP algorithm of [8], which runs in $\tilde{O}\big(m^{1-1/k}n^{2/k}\big)$ time and achieves a $(2k + 1)$-approximation.

▶ **Corollary 2.** *Let* $G = (V, E, \ell)$ *be a weighted undirected graph, where* $\ell : E \to \mathbb{R}_{\geq 0}$*, and let* $k \geq 4$ *be an integer parameter. Let* I *be a set of* n *vertex pairs. There is an algorithm that computes in* $\tilde{O}(m^{1-1/k}n^{2/k})$ *time, for every* $\langle u, v\rangle \in I$*, an estimate* $\hat{d}(u, v)$ *such that* $d(u, v) \leq \hat{d}(u, v) \leq (1.622k + 4) \cdot d(u, v)$*.*

Unweighted n-PSP. In the *unweighted* setting, Chechik, Hoch, and Lifshitz [8] answered Question 1 affirmatively for graphs that are not super sparse. They presented a $(\lceil \frac{4k}{3} \rceil - 1, \lceil \frac{4k}{3} \rceil - 1)$-approximation algorithm for n-PSP with a running time of $\tilde{O}(mn^{1/k} + n^{1+2/k})$. We improve upon this result by reducing the multiplicative approximation error and eliminating the additive error.

▶ **Theorem 3.** *Let* $G = (V, E)$ *be an unweighted undirected graph, and let* $k \geq 4$ *be an integer parameter. Let* I *be a set of* n *vertex pairs. There is an algorithm that computes in* $\tilde{O}(mn^{1/k} + n^{1+2/k})$ *time, for every* $\langle u, v\rangle \in I$*, an estimate* $\hat{d}(u, v)$ *such that* $d(u, v) \leq \hat{d}(u, v) \leq \lceil \frac{4k}{3} - \frac{5}{3} \rceil d(u, v)$*.*

We remark that our improved approximation scheme yields a k-approximation algorithm for n-PSP with running time $\tilde{O}(mn^{1/k} + n^{1+2/k})$ for $4 \leq k \leq 5$. This naturally raises the question of whether a k-approximation algorithm with the running time $\tilde{O}(mn^{1/k} + n^{1+2/k})$ exists for *all* values of k.

As a byproduct of our improved algorithm for unweighted graphs, we extend the result of [8] to weighted graphs. Specifically, we present a $(\lceil \frac{4k}{3} \rceil - 1, (\lceil \frac{4k}{3} \rceil - 1) \cdot W_{uv})$-approximation algorithm for n-PSP, where W_{uv} denotes the weight of the heaviest edge on a shortest u-v path. The algorithm runs in $\tilde{O}(mn^{1/k} + n^{1+2/k})$ time.

▶ **Theorem 4.** *Let* $G = (V, E, \ell)$ *be a weighted undirected graph, where* $\ell : E \to \mathbb{R}_{\geq 0}$*, and let* $k \geq 4$ *be an integer parameter. Let* I *be a set of* n *vertex pairs. There is an algorithm that computes in* $\tilde{O}(mn^{1/k} + n^{1+2/k})$ *time, for every* $\langle u, v\rangle \in I$*, an estimate* $\hat{d}(u, v)$ *such that* $d(u, v) \leq \hat{d}(u, v) \leq (\lceil \frac{4k}{3} \rceil - 1)d(u, v) + (\lceil \frac{4k}{3} \rceil - 1)W_{uv}$*.*

Linear time algorithms for t-PSP. In the setting of *dense* unweighted graphs, Dalirrooyfard *et al.* [11] presented a $((2k - 1)(2k - 2))$-approximation algorithm for n-PSP, running in $\tilde{O}(m + n^{1+2/k})$ time, which is linear for graphs with $m = \Omega(n^{1+2/k})$. By combining their algorithm with the $(2k - 3)$-approximation algorithm of Chechik *et al.* [8], one can achieve a $((2k - 1)(2k - 3))$-approximation algorithm for weighted graphs in the same running time.

Table 1 n-PSP results in *weighted*, unless specified otherwise, undirected graphs with real edge weights. $\delta = d(u,v)$, and $W_{uv} = \max_{(w,z)\in P(u,v)} \ell(w,z)$.

Time	Estimation	Ref.	Comment
$\tilde{O}(mn^{1/k})$	$(2k-1)\delta$	[17]	
$\tilde{O}(mn^{1/k})$	$(2k-2)\delta$	[11]	
$\tilde{O}(mn^{1/k})$	$(2k-3)\delta$	[8]	
$\tilde{O}(mn^{1/k}+n^{1+2/k})$	$1.622k\delta$	Theorem 1	
$\tilde{O}(mn^{1/k}+n^{1+2/k})$	$\lceil\frac{4k}{3}-1\rceil\delta+\lceil\frac{4k}{3}-1\rceil W_{uv}$	Theorem 4	
$\tilde{O}(mn^{1/k}+n^{1+2/k})$	$\lceil\frac{4k}{3}-1\rceil\delta+\lceil\frac{4k}{3}-1\rceil$	[8]	Unweighted graphs
$\tilde{O}(mn^{1/k}+n^{1+2/k})$	$\lceil\frac{4k}{3}-\frac{5}{3}\rceil\delta$	Theorem 3	Unweighted graphs
$\tilde{O}(m+n^{1+2/k})$	$(4k^2-8k+3)\delta$	[11]+[8]	
$\tilde{O}(m+n^{1+2/k}+t)$	$(k^2+2k)\delta$	Theorem 5	For t-PSP

We improve upon this result by presenting a (k^2+2k)-approximation algorithm for the more general t-PSP problem in weighted graphs, with running time $\tilde{O}(m+n^{1+2/k}+t)$, which is linear for graphs with $m=\Omega(n^{1+2/k})$.

▶ **Theorem 5.** *Let $G=(V,E)$ be a weighted undirected graph, and let $k\geq 3$ be an integer parameter. Let I be a set of n vertex pairs. There is an algorithm that computes in $\tilde{O}(m+n^{1+2/k})$ time, for every $\langle u,v\rangle\in I$, an estimate $\hat{d}(u,v)$ such that $d(u,v)\leq\hat{d}(u,v)\leq(k^2+2k)\cdot d(u,v)$.*

We remark that, in addition to achieving a better approximation in the same running time than [11, 8], our algorithm can also answer more than n distance queries in the same time. All of our results are summarized in Table 1.

Related shortest paths problems

ADOs with non-constant query time. A natural approach to n-PSP is to construct an ADO and then query the ADO with the n input pairs. The total running time is the ADO construction time plus the cost of n queries.

Following this approach with the Thorup-Zwick ADO [17], yields a total running time of $\tilde{O}(mn^{1/k}+nk)$. Notice that the $\tilde{O}(nk)$ cost of the queries is negligible when compared to the $\tilde{O}(mn^{1/k})$ cost of the construction. This imbalance suggests that by using more time in the query, improved approximation guarantees might be achieved without increasing the overall runtime.

To keep the total running time $\tilde{O}(mn^{1/k})$, the per-query time may be as large as $\tilde{O}(\frac{m}{n}n^{1/k})$. Recently, Kadria and Roditty [14] studied ADOs with query time $\tilde{O}(\frac{m}{n}n^{1/k})$. They constructed a $(2k-5)$-ADO with query time $\tilde{O}(\frac{m}{n}n^{1/k})$ and space $\tilde{O}(m+n^{1+1/k})$. They asked whether one can obtain an approximation strictly better than $2k-O(1)$ using the same space and query time. However, ADOs typically prioritize space efficiency at the cost of increased construction time. Indeed, the construction time in [14] is $\tilde{O}(mn^{3/k})$ time, which exceeds our target of $\tilde{O}(mn^{1/k})$ time. Therefore, the two questions are distinct: an affirmative answer to the question of [14] does not answer Question 1 due to the larger construction time, and our affirmative answer to Question 1 does not answer the question of [14] because of our larger space usage.

Single-Source and Multi-Source Shortest paths. The Single-Source Shortest Paths (SSSP) and Multi-Source Shortest Paths (MSSP) problems are fundamental shortest paths problems. In SSSP, given a source vertex, the goal is to compute distances to all n vertices in the graph. In MSSP, one is given a set of n^α sources, for $0 < \alpha < 1$, and the goal is to compute distances from each source to all n vertices in the graph.

Unlike the above two problems, n-PSP allows instances in which the query set involves $\Omega(n)$ distinct sources, and these instances appear to be the hardest. Indeed, existing conditional lower bounds for n-PSP [11, 3, 2, 12] hold only when the set of queries contains $\Omega(n)$ disjoint vertex pairs.

Consequently, the hard instances of n-PSP and t-PSP have high vertex diversity. From this perspective, SSSP and MSSP represent structurally restricted cases that centralize source vertices, allowing for specialized, faster algorithms. This fundamental difference motivates the study of n-PSP as a distinct problem in its own right, necessitating novel algorithmic frameworks and techniques.

Paper organization. In the next section, we provide a technical overview of the paper. In Section 4 we present as a warm-up a simple $(\lceil \frac{4k}{3} \rceil - 1, (\lceil \frac{4k}{3} \rceil - 1)_{\text{ODD}})$-approximation n-PSP algorithm, that we will use as a base algorithm in the later sections. Section 5 is the main technical section. The section starts with adapting the results of Section 4 to weighted graphs with an approximation that depends on W_{uv}, where W_{uv} denotes the weight of the heaviest edge on a shortest u-v path. We proceed and develop the heavy-edge technique that allows us to achieve a $1.622k$-approximation n-PSP algorithm for general weighted undirected graphs, which is the main result of this paper. In Appendix A, we address the t-PSP problem in dense weighted graphs, and present our improved running time for t-PSP, which not only improves the running time for previous n-PSP algorithms, but supports more distance queries in the same time.

Finally, in Appendix B, we revisit the n-PSP algorithm from Section 4, and using tighter analysis and slight modifications to the algorithm, we improve the multiplicative approximation to $(\lceil \frac{4k}{3} - \frac{5}{3} \rceil)$, and remove the additive error entirely.

2 Preliminaries

Let $G = (V, E)$ be an undirected graph with $n = |V|$ vertices and $m = |E|$ edges. Throughout the paper, we consider both unweighted graphs and weighted graphs with non-negative real edge weights. Let $u, v \in V$. The distance $d(u, v)$ between u and v is the length of a shortest path between u and v. Let $P(u, v)$ be a shortest path between u and v; when there are several, we take one that minimizes the maximum edge weight. In weighted graphs, let W_{uv} be the weight of the longest edge in $P(u, v)$, i.e. $W_{uv} = \max_{(w,z)\in P(u,v)} \ell(w, z)$.[4] Let x_{ODD} be $x \cdot (d(u, v) \pmod 2)$. The distance $d(u, X)$ between u and X is the distance between u and the closest vertex to u from X, that is, $d(u, X) = \min_{x\in X}(d(u, x))$. Let $p(u, X) = \arg\min_{x\in X}(d(u, x))$ (ties are broken in favor of the vertex with a smaller identifier).

Following many previous algorithms ([17, 8, 11, 6] and more.) we let the bunch $B(u, X, Y)$ be $\{y \in Y \mid d(u, y) < d(u, X)\} \cup \{p(u, X)\}$, where $X, Y \subseteq V$. The bunch can be viewed as a set of vertices closer to u than its closest pivot.

[4] When there are multiple shortest paths P between u and v, W_{uv} denotes $\min_P \max_{(w,z)\in P} \ell(w, z)$. When the path $P(u, v)$ is clear from context, W_{uv} denotes the weight of the longest edge in $P(u, v)$.

In our algorithms, as well as in many distance-related algorithms (n-PSP, distance oracles, all-pairs shortest paths, multiple-source shortest paths, spanners, etc.), there is a vertex hierarchy $V = A_0 \supset A_1 \supset \ldots \supset A_k = \emptyset$, where A_{i+1} contains every vertex from A_i with probability $n^{-1/k}$. Let $B_i(u)$ be $B(u, A_{i+1}, A_i)$, $p_i(u) = p(u, A_i)$, and $h_i(u) = d(u, A_i)$. Let $B(u) = \cup_i B_i(u)$.

In their seminal work, Thorup and Zwick [17] also presented an efficient algorithm for computing $B(u)$ for every $u \in V$, as described in the following lemma.

▶ **Lemma 6** ([17]). *There is an $\tilde{O}(mn^{1/k})$ time algorithm that computes $B(u)$ and the distances from u to every $w \in B(u)$, for every $u \in V$.*

The following classic lemma from [17] bounds $h_i(u)$ in the case that $p_j(u) \notin B(v)$ and $p_j(v) \notin B(u)$ for every $0 < j < i$. This lemma plays a crucial role in the design of their distance oracle.

▶ **Lemma 7** ([17]). *Let $u, v \in V$ and let $0 < i \leq k-1$. If $p_j(u) \notin B(v)$ and $p_j(v) \notin B(u)$ for every $0 < j < i$, then*

$$h_i(u) \leq i \cdot d(u,v) \quad \text{and} \quad h_i(v) \leq i \cdot d(u,v).$$

Let $\texttt{TZQuery}(u, v)$ be the query of the distance oracle of [17]. $\texttt{TZQuery}(u, v)$ finds the minimum i such that $p_i(u) \in B(v)$ or $p_i(v) \in B(u)$, and then returns $\min(d(u, p_i(v)) + d(p_i(v), v), d(u, p_i(u)) + d(p_i(u), v))$ as the estimation. Using Lemma 7, [17] proved:

▶ **Lemma 8** ([17]). *For every $0 \leq i \leq k-1$ it holds that $\texttt{TZQuery}(u, v) \leq (2k-2i-1)d(u, v) + 2h_i(u)$.*

The following lemma is implicit in [16], and explicit in [8]. Using this lemma, one can bound $h_i(u)$ in the case that $p_{i-1}(v) \notin B(u)$. We add its proof for completeness since it is used in the correctness of Theorem 1.

▶ **Lemma 9** ([16, 8]). *Let $u, v \in V$, if $p_{i-1}(v) \notin B(u)$ then $d(v, p_i(v)) \leq d(v, p_{i-1}(v)) + 2d(u, v)$.*

Proof. From the definition of $p_i(v)$, we know that it is the closest vertex to v in A_i, and since $p_i(u) \in A_i$ we get that $d(v, p_i(v)) \leq d(v, p_i(u))$. From the triangle inequality, it follows that $d(v, p_i(u)) \leq d(v, u) + d(u, p_i(u))$. By the assumption of the lemma, we know that $p_{i-1}(v) \notin B(u)$. Therefore, $d(u, p_i(u)) \leq d(u, p_{i-1}(v))$. From the triangle inequality, it follows that $d(u, p_{i-1}(v)) \leq d(u, v) + d(v, p_{i-1}(v))$. Overall, we get that:

$$\begin{aligned} d(v, p_i(v)) &\leq d(v, p_i(u)) \leq d(v, u) + d(u, p_i(u)) \leq d(v, u) + d(u, p_{i-1}(v)) \\ &\leq d(v, u) + d(u, v) + d(v, p_{i-1}(v)) = 2d(u, v) + d(v, p_{i-1}(v)), \text{ as required.} \end{aligned}$$ ◀

Thorup and Zwick [17] proved the following bound on the size of the bunches.

▶ **Lemma 10** (Lemma 3.5 in [17]). *$|B(u)| = O(k \cdot n^{1/k} \log^{1-1/k} n) = \tilde{O}(n^{1/k})$ with high probability.*

3 Technical Overview

n-PSP in unweighted graphs. The $(\lceil \frac{4k}{3} \rceil - 1, \lceil \frac{4k}{3} \rceil - 1)$-approximation algorithm of Chechik *et al.* [8] works only in unweighted graphs because its analysis relies on the ability to select three specific vertices on the shortest path between u and v, positioned approximately

at distances $d(u,v)/4$, $d(u,v)/2$, and $3d(u,v)/4$ from u. Our first contribution is a simplification of the algorithm of [8] in which we show that it suffices to consider only a single vertex on the shortest path that is located at roughly $d(u,v)/2$ from u. Moreover, we show that when $d(u,v) \equiv 0 \pmod 2$, the additive term in the approximation can be eliminated entirely. We present this result as a warm-up in Section 4. In Appendix B, we show that by slightly modifying the algorithm of Section 4 and using tighter analysis, one can improve the multiplicative stretch to $\lceil \frac{4k}{3} - \frac{5}{3} \rceil$ and remove the additive error entirely.

n-PSP in weighted graphs. In Section 5 we turn our attention to weighted graphs. Let $P(u,v)$ be a shortest path between u and v, let (u',v') be the heaviest edge on $P(u,v)$ and let W_{uv} be the weight of (u',v'). We begin by showing that applying the algorithm from Section 4 to weighted graphs yields a $(\lceil \frac{4k}{3} \rceil - 1, (\lceil \frac{4k}{3} \rceil - 1) \cdot W_{uv})$-approximation (see Theorem 4). The rest of Section 5 is devoted to our main result which is a $1.622k$-approximation algorithm for weighted graphs (see Theorem 1).

To achieve a $1.622k$-approximation, we distinguish between two cases based on the value of W_{uv}. When W_{uv} is small, we use Theorem 4 to get a good approximation. The interesting case is when W_{uv} is large. Consider for example the case that $W_{uv} = \ell(u',v') \approx d(u,v)/2$. In this case, the $(\lceil \frac{4k}{3} \rceil - 1, (\lceil \frac{4k}{3} \rceil - 1) \cdot W_{uv})$-approximation algorithm gives an approximation of $\approx \frac{4k}{3} d(u,v) + \frac{4k}{3} \cdot W_{uv} \overset{W_{uv} \approx d(u,v)/2}{\approx} 2k \cdot d(u,v)$, which does not break the $2k - O(1)$ approximation barrier. To handle the case that W_{uv} is large, we develop a new technique called the *heavy-edge* technique.

Consider again the case that $W_{uv} = \ell(u',v') \approx d(u,v)/2$. For simplicity, assume that the edge (u',v') is given to us. In this case, we can compute an estimation $\hat{d}(u,v)$ for $d(u,v)$ as follows.

$$\begin{aligned}\hat{d}(u,v) &= \texttt{TZQuery}(u,u') + \ell(u',v') + \texttt{TZQuery}(v',v) \\ &\leq (2k-1)(d(u,u') + d(v',v)) + \ell(u',v') \\ &\overset{W_{uv} \approx d(u,v)/2}{\approx} (2k-1) \cdot \tfrac{d(u,v)}{2} + \tfrac{d(u,v)}{2} = k \cdot d(u,v),\end{aligned}$$

which breaks the $2k - O(1)$ approximation barrier as wanted. The difficulty is that the heavy edge (u',v') is not known *a priori*. We therefore preprocess over all possible edges $(u',v') \in E$, treating each edge as a candidate heavy edge on a queried shortest path. Specifically, we create a hash table $H(\cdot,\cdot)$ of distance estimations, and for each edge $(u',v') \in E$, $w \in B(v')$ and $i \in [k]$, we set

$$H(p_i(u'), w) = \min(H(p_i(u'), w), h_i(u') + \ell(u',v') + d(v',w))$$

Intuitively, this stores the cost of the walk $p_i(u') \rightsquigarrow u' \to v' \rightsquigarrow w$ in $H(p_i(u'), w)$. To compute the estimation $\hat{d}(u,v)$, the algorithm considers all vertices $x \in B(u)$ and $w \in B(v)$, and updates $\hat{d}(u,v)$ as follows:

$$\hat{d}(u,v) = \min(\hat{d}(u,v), d(u,x) + H(x,w) + d(w,v))$$

For every pair $\langle u,v \rangle \in I$ the query takes $O(|B(u)| \cdot |B(v)|) = O(n^{2/k})$ time, since $|B(u)|, |B(v)| = O(n^{1/k})$. Therefore, the total running time for n queries is $O(n^{1+2/k})$, as required.

The main technical contribution is the approximation analysis provided in Lemma 14, which shows that this algorithm breaks the $2k - O(1)$ approximation barrier. Specifically, we prove that this algorithm achieves a stretch of $1.622k$, regardless of the value of W_{uv}. We

note that the heavy-edge technique might be found useful to convert other W_{uv}-dependent approximation algorithms into purely multiplicative approximation algorithms that do not depend on W_{uv}.

t-PSP in dense weighted graphs. In Appendix A, we address the t-PSP problem in dense weighted graphs. Unlike previous approaches, which construct a spanner and then run an n-PSP algorithm on top of it, our approach does not use the spanner as a black-box. We leverage a key property of the inner structure of the spanner to obtain an improved algorithm. The $(2k-1)$-spanner of [7] is constructed by sparsifying the graph G in k iterations into a subgraph H. We show that if an edge (u,v) is removed in iteration i, then the distance between u and v in H is at most $(2i-1)d_G(u,v)$ rather than $(2k-1)d_G(u,v)$.

Consider the shortest path $P_G(u,v)$, and let i be the iteration in which the last edge from $P_G(u,v)$ is removed, i.e., after the i'th iteration of the spanner construction, all the edges of $P_G(u,v)$ are already removed from the graph. If i is small, then the edges of $P_G(u,v)$ are removed at an early stage, and therefore, the stretch of the entire path in H is small, satisfying $d_H(u,v) \leq (2i-1)d_G(u,v)$. If i is large, then there exists $(x,y) \in P_G(u,v)$ that remained until a late iteration, implying that $h_i(x) \leq i \cdot \ell(x,y)$. We use this property to get that $h_i(u) \leq i \cdot d(u,v)$, and utilize the fact that $h_i(u) \leq i \cdot d(u,v)$ in our algorithm, by incorporating the parameterized distance oracle of Kadria and Roditty [13]. See Theorem 5 for the complete proof.

4 Warm-up: $(\lceil 4k/3 \rceil - 1, (\lceil 4k/3 \rceil - 1)_{\text{ODD}})$-approximation in $\tilde{O}(mn^{1/k} + n^{1+2/k})$ time

In this section, we prove the following theorem:

▶ **Theorem 11.** *Let $G = (V,E)$ be an unweighted undirected graph, and let $k \geq 4$ be an integer parameter. Let I be a set of n vertex pairs. There is an algorithm that computes in $\tilde{O}(mn^{1/k} + n^{1+2/k})$ time, for every $\langle u,v \rangle \in I$, an estimate $\hat{d}(u,v)$ such that $d(u,v) \leq \hat{d}(u,v) \leq (\lceil \frac{4k}{3} \rceil - 1)d(u,v) + (\lceil \frac{4k}{3} \rceil - 1)_{\text{ODD}}$.* [5]

The algorithm is simple and works as follows. First, the algorithm initializes two empty hash tables $H, \hat{d}$, and for every $u \in V$ computes $B(u)$ using Lemma 6. Then, for every $u \in V$, and every $v,w \in B(u)$ the algorithm sets $H(v,w)$ to $\min(H(v,w), d(u,v) + d(u,w))$. Next, for every vertex pair $\langle u,v \rangle \in I$, and for every $w \in B(u)$ and $z \in B(v)$, the algorithm sets $\hat{d}(u,v)$ to $\min(\hat{d}(u,v), d(u,w) + H(w,z) + d(z,v))$. The algorithm returns $\hat{d}(u,v)$ for every $\langle u,v \rangle \in I$. A pseudocode for the algorithm is presented in Algorithm 1.

Let $\langle u,v \rangle \in I$ be a vertex pair from the input. To prove Theorem 11 we need to prove that:

$$\hat{d}(u,v) \leq (\lceil 4k/3 \rceil - 1)d(u,v) + (\lceil 4k/3 \rceil - 1)_{\text{ODD}}$$

Let $t = d(u,v)/2 + 0.5_{\text{ODD}}$. Since the graph is unweighted, there is a vertex $\tau \in P(u,v)$ such that $d(u,\tau) = t$ and $d(v,\tau) = t - 1_{\text{ODD}} \leq t$. In Lemma 12, we show that in such a case $\hat{d}(u,v) \leq (\lceil 4k/3 \rceil - 1)2t$. Since $t = d(u,v)/2 + 0.5_{\text{ODD}}$ this implies that

$$\hat{d}(u,v) \leq (\lceil 4k/3 \rceil - 1)2t = (\lceil 4k/3 \rceil - 1)d(u,v) + (\lceil 4k/3 \rceil - 1)_{\text{ODD}},$$

[5] $x_{\text{ODD}} = x \cdot (d(u,v) \pmod 2)$.

Algorithm 1 $(\lceil 4k/3 \rceil - 1, (\lceil 4k/3 \rceil - 1)_{\text{ODD}}$-approximation for n-PSP.

```
H ← HashTable(); d̂ ← HashTable()
Compute B(u) for every u ∈ V [Lemma 6]
for u ∈ V do
    for v, w ∈ B(u) do
        H(v, w) = min(H(v, w), d(u, v) + d(u, w))
for ⟨u, v⟩ ∈ I do
    for w ∈ B(u) and z ∈ B(v) do
        d̂(u, v) = min(d̂(u, v), d(u, w) + H(w, z) + d(z, v))
return {d̂(u, v) | ⟨u, v⟩ ∈ I}
```

as required. We remark that the only point at which the assumption that G is unweighted is used is in setting $t = d(u,v)/2 + 0.5_{\text{ODD}}$. Apart from this, the proof of Lemma 12 remains valid for weighted graphs, provided the lemma's condition is satisfied.

▶ **Lemma 12.** *Let t be a value such that there exists a vertex $\tau \in P(u,v)$ satisfying $d(u,\tau), d(v,\tau) \le t$. Then, $\hat{d}(u,v) \le (\lceil 4k/3 \rceil - 1) \cdot 2t$.*

Proof. First, we show that $\hat{d}(u,v) \le \texttt{TZQuery}(u,v)$.

▷ Claim 12.1. $\hat{d}(u,v) \le \texttt{TZQuery}(u,v)$

Proof. The query of $\texttt{TZQuery}(u,v)$ finds the minimum i such that $p_i(u) \in B(v)$ or $p_i(v) \in B(u)$. Then it returns $h_i(u)+d(p_i(u),v)$ or $d(u,p_i(v))+h_i(v)$. Wlog assume that $p_i(v) \in B(u)$, then in our algorithm, in the final loop there is an iteration with $w = z = p_i(v)$, therefore after this iteration we have that $\hat{d}(u,v) \le d(u,p_i(v)) + h_i(v) = \texttt{TZQuery}(u,v)$, as required. ◁

Let i be the maximum index for which $h_i(\tau) \le i \cdot t$. Such an index must exist since $h_0(\tau) = d(\tau,\tau) = 0 = 0 \cdot t$. We divide the proof into three cases (See Figure 1 for an illustration of the 3 cases):

1. $p_i(\tau) \in B(u) \cap B(v)$.
2. $p_i(\tau) \in B(v) \setminus B(u)$ or $p_i(\tau) \in B(u) \setminus B(v)$.
3. $p_i(\tau) \notin B(u)$ and $p_i(\tau) \notin B(v)$.

We start with case 1. In this case $p_i(\tau) \in B(u) \cap B(v)$. Therefore, in the final loop of the algorithm, there is an iteration in which $w = z = p_i(\tau)$ and the algorithm sets $\hat{d}(u,v)$ to $\min(\hat{d}(u,v), d(u,p_i(\tau)) + H(p_i(\tau),p_i(\tau)) + d(p_i(\tau),v))$. Therefore:

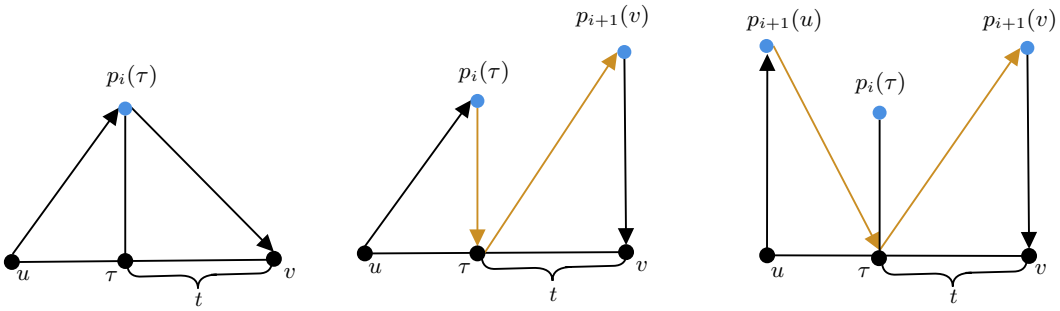
$$\hat{d}(u,v) \le d(u,p_i(\tau)) + H(p_i(\tau),p_i(\tau)) + d(p_i(\tau),v) \overset{\triangle}{\le} d(u,\tau) + h_i(\tau) + 0 + h_i(\tau) + d(\tau,v)$$
$$\overset{h_i(\tau)\le it}{\le} d(u,v) + 2it \overset{d(u,v)\le 2t}{\le} (2i+2)t \overset{i\le k-1}{\le} 2kt \le 2(\lceil 4k/3 \rceil - 1)t,$$

as required.

For cases 2 and 3, we will show that $\hat{d}(u,v) \le (4i+6)t$ and that $\min(h_{i+1}(u), h_{i+1}(v)) \le (i+1)t$.

From Claim 12.1 we have that $\hat{d}(u,v) \le \texttt{TZQuery}(u,v)$, and we can apply Lemma 8 with parameter $i+1$ to obtain:

$$\hat{d}(u,v) \le (2k - 2(i+1) - 1)d(u,v) + 2\min(h_{i+1}(u), h_{i+1}(v))$$
$$\overset{\min(h_{i+1}(u),h_{i+1}(v))\le (i+1)t}{\le} (2k - 2i - 3)d(u,v) + 2(i+1)t \overset{d(u,v)\le 2t}{\le} (4k - 2i - 4)t$$



(1)
$p_i(\tau) \in B(u) \cap B(v)$
$\hat{d}(u,v) \leq$
$d(u,v) + 2h_i(\tau)$

(2)
$p_i(\tau) \in B(u) \setminus B(v) \rightarrow$
$p_{i+1}(v) \in B(\tau)$
$\hat{d}(u,v) \leq$
$d(u,v) + 2h_i(\tau) + 2h_{i+1}(v)$

(3)
$p_i(\tau) \notin B(u) \cup B(v) \rightarrow$
$p_{i+1}(u), p_{i+1}(v) \in B(\tau)$
$\hat{d}(u,v) \leq$
$d(u,v) + 2h_i(\tau) + 2h_{i+1}(v)$

Figure 1 The three cases of Lemma 12. $i = \max\{j \mid h_j(\tau) \leq j \cdot t\}$.

By combining this bound with the fact that $\hat{d}(u,v) \leq (4i+6)t$ we get that:

$$\hat{d}(u,v) \leq \min(4k-2i-4, 4i+6)t \leq (\lceil 4k/3 \rceil - 1)2t,$$

as required.

Therefore, to complete the proof of the lemma, it remains to show that in cases 2 and 3, we have that $\hat{d}(u,v) \leq (4i+6)t$ and $\min(h_{i+1}(u), h_{i+1}(v)) \leq (i+1)t$. To prove cases 2 and 3, we first prove the following useful claim.

▷ Claim 12.2. If $p_i(\tau) \notin B(u)$ then $p_{i+1}(u) \in B(\tau)$ and $h_{i+1}(u) \leq (i+1)t$, for every $0 \leq i \leq k-2$.

Proof. Since $p_i(\tau) \notin B(u)$, we have that

$$d(u, p_{i+1}(u)) \leq d(u, p_i(\tau)) \overset{\triangle}{\leq} d(u,\tau) + d(\tau, p_i(\tau)) \overset{h_i(\tau) \leq it}{\leq} t + it = (i+1)t.$$

Since i is the maximum index such that $h_i(\tau) \leq it$, we have that $h_{i+2}(\tau) > (i+2)t$. From the triangle inequality we have:

$$d(\tau, p_{i+1}(u)) \leq d(\tau, u) + d(u, p_{i+1}(u)) \leq t + (i+1)t = (i+2)t < h_{i+2}(\tau),$$

and therefore $p_{i+1}(u) \in B_{i+1}(\tau)$, as required. ◁

We are now ready to prove cases 2 and 3. In case 2 $p_i(\tau) \in B(v) \setminus B(u)$ or $p_i(\tau) \in B(u) \setminus B(v)$. Wlog, we assume that $p_i(\tau) \in B(v) \setminus B(u)$. Since $p_i(\tau) \notin B(u)$, from Claim 12.2 we have that $p_{i+1}(u) \in B(\tau)$ and $h_{i+1}(u) \leq (i+1)t$. Since $p_i(\tau), p_{i+1}(u) \in B(\tau)$ we have that $H(p_{i+1}(u), p_i(\tau)) \leq d(p_{i+1}(u), \tau) + h_i(\tau)$. Since $p_{i+1}(u) \in B(u)$ and $p_i(\tau) \in B(v)$, in the final loop of the algorithm, we have an iteration in which $w = p_{i+1}(u)$ and $z = p_i(\tau)$. Therefore:

$$\begin{aligned}
\hat{d}(u,v) &\leq d(u, p_{i+1}(u)) + H(p_{i+1}(u), p_i(\tau)) + d(p_i(\tau), v) \\
&\leq d(u, p_{i+1}(u)) + d(p_{i+1}(u), \tau) + d(\tau, p_i(\tau)) + d(p_i(\tau), v) \\
&\overset{\triangle}{\leq} d(u,v) + 2h_{i+1}(u) + 2h_i(\tau) \overset{h_{i+1}(u) \leq (i+1)t}{\leq} d(u,v) + 2(i+1)t + 2it \overset{d(u,v) \leq 2t}{\leq} (4i+4)t,
\end{aligned}$$

as required.

Finally, we complete the proof by proving case 3. In case 3 we have $p_i(\tau) \notin B(u)$ and $p_i(\tau) \notin B(v)$. From Claim 12.2 we have that $p_{i+1}(u), p_{i+1}(v) \in B(\tau)$ and $h_{i+1}(u), h_{i+1}(v) \leq (i+1)t$. Since $p_{i+1}(u), p_{i+1}(v) \in B(\tau)$ we have that $H(p_{i+1}(u), p_{i+1}(v)) \leq d(p_{i+1}(u), \tau) + d(\tau, p_{i+1}(v))$. Since $p_{i+1}(u) \in B(u)$ and $p_{i+1}(v) \in B(v)$, in the final loop of the algorithm we have an iteration in which $w = p_{i+1}(u)$, and $z = p_{i+1}(v)$. Therefore:

$$\begin{aligned}\hat{d}(u,v) &\leq d(u, p_{i+1}(u)) + H(p_{i+1}(u), p_{i+1}(v)) + d(p_{i+1}(v), v) \\ &\leq d(u, p_{i+1}(u)) + d(p_{i+1}(u), \tau) + d(\tau, p_{i+1}(v)) + d(p_{i+1}(v), v) \\ &\overset{\triangle}{\leq} d(u,v) + 2h_{i+1}(u) + 2h_{i+1}(v) \\ &\leq d(u,v) + 4(i+1)t \overset{d(u,v)\leq 2t}{\leq} (4i+6)t,\end{aligned}$$

as required. ◀

Next, we show that the running time of the algorithm is $\tilde{O}(mn^{1/k} + n^{1+2/k})$.

▶ **Lemma 13.** *The algorithm takes $\tilde{O}(mn^{1/k} + n^{1+2/k})$ time.*

Proof. Computing $B(u)$ for every $u \in V$ takes $\tilde{O}(mn^{1/k})$ time. Constructing H requires $O\left(\sum_{u\in V} |B(u)|^2\right) = \tilde{O}(n^{1+2/k})$. Estimating the distances takes $O\left(\sum_{(u,v)\in I} |B(u)| \cdot |B(v)|\right) = \tilde{O}(n^{1+2/k})$. Therefore, the running time is $\tilde{O}(mn^{1/k}+n^{1+2/k})$, as required. ◀

Theorem 11 follows from Lemma 12 and Lemma 13.

5 A $1.622k$-approximation algorithm for n-PSP in weighted graphs

In this section, we turn our attention to the n-PSP problem in weighted undirected graphs. We start by extending Theorem 11 to weighted graphs by bounding the additive error with $(\lceil 4k/3 \rceil - 1)W_{uv}$ instead of $(\lceil 4k/3 \rceil - 1)$ while keeping the multiplicative error the same.[6]

▶ **Reminder of Theorem 4.** *Let $G = (V, E, \ell)$ be a weighted undirected graph, where $\ell : E \to \mathbb{R}_{\geq 0}$, and let $k \geq 4$ be an integer parameter. Let I be a set of n vertex pairs. There is an algorithm that computes in $\tilde{O}(mn^{1/k} + n^{1+2/k})$ time, for every $\langle u, v\rangle \in I$, an estimate $\hat{d}(u,v)$ such that $d(u,v) \leq \hat{d}(u,v) \leq (\lceil \frac{4k}{3} \rceil - 1)d(u,v) + (\lceil \frac{4k}{3} \rceil - 1)W_{uv}$.*

Proof. To prove the theorem we analyze Algorithm 1 when the input graph is weighted and show that $d(u,v) \leq \hat{d}(u,v) \leq (\lceil \frac{4k}{3} \rceil - 1)d(u,v) + (\lceil \frac{4k}{3} \rceil - 1)W_{uv}$. Let $t = \min_{\tau \in P(u,v)}(\max(d(u,\tau), d(v,\tau)))$. Notice that in unweighted graphs $\min_{\tau\in P(u,v)}(\max(d(u,\tau), d(v,\tau))) = d(u,v)/2 + 0.5_{\text{ODD}}$. This generalized definition of t ensures that there exists $\tau \in P(u,v)$ such that $d(u,\tau), d(v,\tau) \leq t$, and therefore we can apply Lemma 12 to get that $\hat{d}(u,v) \leq (\lceil \frac{4k}{3} \rceil - 1)2t$.

We now bound t in terms of $d(u,v)$ and W_{uv}. Consider an edge (τ_u, τ_v) along the shortest path $P(u,v)$ such that $d(u,\tau_u) \leq d(u,v)/2$ and $d(u,\tau_v) > d(u,v)/2$. (See Figure 2 for an illustration.) Since $d(u,\tau_v) + d(v,\tau_u) = d(u,v) + \ell(\tau_u,\tau_v)$, it follows that:

$$t \leq \min(d(u,\tau_v), d(v,\tau_u)) \leq (d(u,\tau_v) + d(v,\tau_u))/2 \leq (d(u,v) + \ell(\tau_u,\tau_v))/2 \leq d(u,v)/2 + W_{uv}/2,$$

[6] Recall that $W_{uv} = \max_{(w,z)\in P(u,v)} \ell(w,z)$.

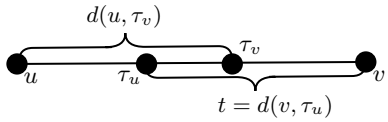


Figure 2 Illustration of $P(u,v)$, where $(\tau_u, \tau_v) \in P(u,v)$ and $d(u,\tau_u) \leq d(u,v)/2$, $d(u,\tau_v) > d(u,v)/2$.

therefore $t \leq d(u,v)/2 + W_{uv}/2$ and we get that:

$$\hat{d}(u,v) \leq (\lceil \tfrac{4k}{3} \rceil - 1)2t \leq (\lceil \tfrac{4k}{3} \rceil - 1)d(u,v) + (\lceil \tfrac{4k}{3} \rceil - 1)W_{uv},$$

as required. From Lemma 13 it follows that the running time is $\tilde{O}(mn^{1/k} + n^{1+2/k})$. ◀

Next, we present the main result of this paper: the first truly $(2-\alpha)k$, for $\alpha > 0$, approximation n-PSP algorithm for weighted undirected graphs with non-negative real edge weights. We prove:

▶ **Reminder of Theorem 1.** *Let $G = (V, E, \ell)$ be a weighted undirected graph, where $\ell : E \to \mathbb{R}_{\geq 0}$, and let $k \geq 4$ be an integer parameter. Let I be a set of n vertex pairs. There is an algorithm that computes in $\tilde{O}(mn^{1/k} + n^{1+2/k})$ time, for every $\langle u, v\rangle \in I$, an estimate $\hat{d}(u,v)$ such that $d(u,v) \leq \hat{d}(u,v) \leq 1.622k \cdot d(u,v)$.*

The algorithm works as follows. First, similar to Section 4, the algorithm initializes two empty hash tables $H, \hat{d}$, and for every $u \in V$ computes $B(u)$ using Lemma 6. For every $u \in V$, and for every $v, w \in B(u)$ the algorithm sets $H(v,w)$ to $\min(H(v,w), d(u,v) + d(u,w))$.

The new insight comes from the following step (the heavy-edge technique): For every edge $(u,v) \in E$ and for every $i \in [k]$, the algorithm sets $H(p_i(u), w)$ to $\min(H(p_i(u), w), h_i(u) + \ell(u,v) + d(v,w))$, for every $w \in B(v)$.

Next, the algorithm approximates the distances for every pair $\langle u, v\rangle \in I$, the same as Section 4: For every $w \in B(u)$ and $z \in B(v)$, the algorithm sets $\hat{d}(u,v)$ to $\min(\hat{d}(u,v), d(u,w) + H(w,z) + d(z,v))$. The algorithm returns $\hat{d}(u,v)$ for every $\langle u, v\rangle \in I$. A pseudocode for the algorithm is presented in Algorithm 2.

We note that Algorithm 2 differs from Algorithm 1 in the loop on lines 6–8, which iterates over every $(u,v) \in E$, $i \in [k]$ and $w \in B(v)$ and sets $H(p_i(u), w)$ to $\min(H(p_i(u), w), h_i(u) + \ell(u,v) + d(v,w))$. Notice that this adaptation takes $O(\sum_{(u,v)\in E} k \cdot |B(v)|) = \tilde{O}(m \cdot n^{1/k})$, and therefore it does not increase our running time. This modification helps us bound $\hat{d}(u,v)$ in the case where W_{uv} is large (hence we call it the heavy-edge technique), and obtain an approximation that does not depend on W_{uv}, as shown in the following lemma. Let $\langle u, v\rangle \in I$ be a vertex pair from the input.

▶ **Lemma 14.** $\hat{d}(u,v) \leq 1.622kd(u,v)$

Proof. Let $t = \min_{w \in P(u,v)}(\max(d(u,w), d(w,v)))$. This implies that $d(u,v)/2 \leq t \leq d(u,v)$. Let $\tau \in P(u,v)$ be the vertex, for which $\max(d(u,\tau), d(\tau,v)) = t$. The proof is divided into two cases, according to the value of t. The case that $t \leq (1/2 + c)d(u,v)$, and the case that $t > (1/2 + c)d(u,v)$, for some constant $0 < c < 1/2$, to be determined later.

At a high level, if $t \leq (1/2 + c) \cdot d(u,v)$, then there exists a vertex on the path $P(u,v)$ that lies close to its midpoint. In this situation, we can directly apply the bound established for the unweighted case. On the other hand, if $t > (1/2 + c)d(u,v)$, then the next edge

Algorithm 2 $1.622k$-approximation for n-PSP.

$H \leftarrow HashTable(); \hat{d}(u,v) \leftarrow HashTable()$
Compute $B(u)$ for every $u \in V$ [Lemma 6]
for $u \in V$ **do**
  for $v, w \in B(u)$ **do**
    $H(v,w) = \min(H(v,w), d(u,v) + d(u,w))$
for $(u,v) \in E$ **do**
  for $i \in [k]$ *and* $w \in B(v)$ **do**
    $H(p_i(u), w) = \min(H(p_i(u), w), h_i(u) + \ell(u,v) + d(v,w))$
for $\langle u, v\rangle \in I$ **do**
  for $w \in B(u)$ *and* $z \in B(v)$ **do**
    $\hat{d}(u,v) = \min(\hat{d}(u,v), d(u,w) + H(w,z) + d(z,v))$
return $\{\hat{d}(u,v) \mid \langle u, v\rangle \in I\}$

$(\tau, \tau_v) \in P(u,v)$ must have relatively large weight, in this case we use the heavy-edge technique, which is the main technical contribution of this lemma. Since (τ, τ_v) is considered in the algorithm explicitly, the approximation for its length is 1 (exact weight), since it is a heavy edge in the path, we get an overall less than $2k$-approximation for the entire path.

In the proof, we have two bounds according to the value of c, and then we choose the optimal c that minimizes the maximum of these bounds. Formally, the analysis yields two different upper bounds on the approximation, each corresponding to one of the two cases and parameterized by c. The final step of the proof selects the value of c that minimizes the worst case of these two bounds, thereby optimizing the overall guarantee.

Consider first the case that $t \leq (1/2 + c)d(u,v)$. Since $d(u,\tau), d(v,\tau) \leq t$ it follows from Lemma 12 that:

$$\hat{d}(u,v) \leq (\lceil \tfrac{4k}{3} \rceil - 1) \cdot 2t \leq (4k/3)(1 + 2c)d(u,v)$$

Next, we consider the case where $t > (1/2 + c)d(u,v)$. Assume, wlog, that $t = d(\tau, v) > d(u,\tau)$. Since $d(u,\tau) + d(\tau,v) = d(u,v)$, and $t \geq (1/2 + c)d(u,v)$ we get that $d(u,\tau) = d(u,v) - t \leq (1/2 - c)d(u,v)$. Let $\tau_v \in P(\tau, v)$ be the first vertex from τ to v, i.e., $d(v,\tau) = d(v,\tau_v) + \ell(\tau_v, \tau)$. From the minimality of t, we know that $d(u,\tau_v) \geq t \geq (1/2 + c)d(u,v)$, and therefore $d(v,\tau_v) = d(u,v) - d(u,\tau_v) \leq d(u,v) - t \leq (1/2 - c)d(u,v)$. We conclude that

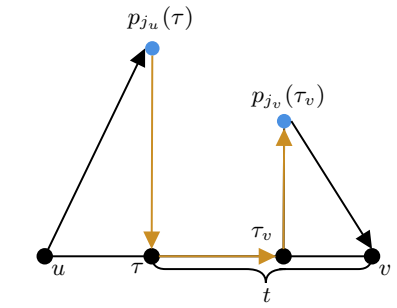
$$d(u,\tau), d(v,\tau_v) \leq (1/2 - c)d(u,v) \tag{5.1}$$

Let j_u be the first index such that $p_{j_u}(\tau) \in B(u)$. Since $p_j(\tau) \notin B(u)$ for every $j < j_u$, we can apply Lemma 9 to obtain:

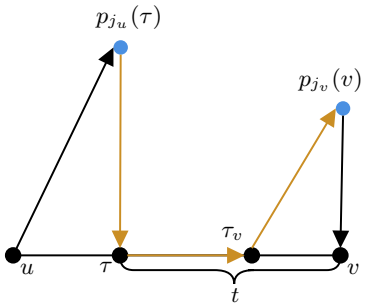
$$h_{j_u}(\tau) \leq j_u \cdot 2d(u,\tau) \overset{5.1}{\leq} (1 - 2c)j_u d(u,v) \tag{5.2}$$

Let j_v be the first index such that $p_{j_v}(\tau_v) \in B(v)$ or $p_{j_v}(v) \in B(\tau_v)$. Since $p_j(\tau_v) \notin B(v)$ and $p_j(v) \notin B(\tau_v)$ for every $j < j_v$, we can apply Lemma 7 to obtain:

$$h_{j_v}(\tau_v), h_{j_v}(v) \leq j_v d(\tau_v, v) \overset{5.1}{\leq} (1/2 - c)j_v d(u,v) \tag{5.3}$$



(a)
$p_{j_v}(\tau_v) \in B(v)$
$\hat{d}(u,v) \leq \delta + 2h_{j_u}(\tau) + 2h_{j_v}(\tau_v)$



(b)
$p_{j_v}(v) \in B(\tau_v)$
$\hat{d}(u,v) \leq \delta + 2h_{j_u}(\tau) + 2h_{j_v}(v)$

Figure 3 The two cases of Claim 14.1. $t \geq (1/2+c)\delta$. $j_u = \min(i \mid p_i(\tau) \in B(u))$, $j_v = \min(i \mid p_i(v) \in B(\tau_v) \vee p_i(\tau_v) \in B(v))$.

Next, we prove the following three bounds on $\hat{d}(u,v)$:

14.1. $\hat{d}(u,v) \leq (1 + 2(1-2c)j_u + (1-2c)j_v)d(u,v)$

14.2. $\hat{d}(u,v) \leq (2k - j_v - 1 - 2cj_v)d(u,v)$

14.3. $\hat{d}(u,v) \leq (2k - 4j_u c + 2c - 3)d(u,v)$

▷ Claim 14.1. $\hat{d}(u,v) \leq d(u,v) + 2(1-2c)j_u d(u,v) + (1-2c)j_v d(u,v)$

Proof. We divide the proof into two cases. The case that $p_{j_v}(v) \in B(\tau_v)$, and the case that $p_{j_v}(\tau_v) \in B(v)$. (See Figure 3 for an illustration.)

Consider first, the case that $p_{j_v}(v) \in B(\tau_v)$. Since $p_{j_v}(v) \in B(\tau_v)$, after the edge (τ, τ_v) is considered in the loop of lines 6–8, with $i = j_u$ and $w = p_{j_v}(v)$, it is guaranteed that:

$$H(p_{j_u}(\tau), p_{j_v}(v)) \leq h_{j_u}(\tau) + \ell(\tau, \tau_v) + d(\tau_v, p_{j_v}(v)) \tag{5.4}$$

Since $p_{j_u}(\tau) \in B(u)$ and $p_{j_v}(v) \in B(v)$, in the final loop of the algorithm there is an iteration in which $w = p_{j_u}(\tau)$ and $z = p_{j_v}(v)$. Therefore:

$$\begin{aligned}
\hat{d}(u,v) &\leq d(u, p_{j_u}(\tau)) + H(p_{j_u}(\tau), p_{j_v}(v)) + h_{j_v}(v) \\
&\overset{5.4}{\leq} d(u, p_{j_u}(\tau)) + (h_{j_u}(\tau) + \ell(\tau, \tau_v) + d(\tau_v, p_{j_v}(v))) + h_{j_v}(v) \\
&\overset{\triangle}{\leq} d(u,\tau) + h_{j_u}(\tau) + h_{j_u}(\tau) + \ell(\tau, \tau_v) + d(\tau_v, v) + h_{j_v}(v) + h_{j_v}(v) \\
&= d(u,v) + 2h_{j_u}(\tau) + 2h_{j_v}(v) \overset{5.2,5.3}{\leq} d(u,v) + 2(1-2c)j_u d(u,v) + (1-2c)j_v d(u,v),
\end{aligned}$$

as required. We consider now the case that $p_{j_v}(\tau_v) \in B(v)$. Since $p_{j_v}(\tau_v) \in B(\tau_v)$, after the edge (τ, τ_v) is considered in the loop of lines 6–8, with $i = j_u$ and $w = p_{j_v}(\tau_v)$, we get:

$$H(p_{j_u}(\tau), p_{j_v}(\tau_v)) \leq h_{j_u}(\tau) + \ell(\tau, \tau_v) + h_{j_v}(\tau_v) \tag{5.5}$$

Since $p_{j_u}(\tau) \in B(u)$ and $p_{j_v}(\tau_v) \in B(v)$, in the final loop of the algorithm there is an iteration in which $w = p_{j_u}(\tau)$ and $z = p_{j_v}(\tau_v)$. Therefore:

$$\begin{aligned}\hat{d}(u,v) &\leq d(u,p_{j_u}(\tau)) + H(p_{j_u}(\tau), p_{j_v}(\tau_v)) + d(p_{j_v}(\tau_v), v)\\ &\overset{5.5}{\leq} d(u,p_{j_u}(\tau)) + (h_{j_u}(\tau) + \ell(\tau,\tau_v) + h_{j_v}(\tau_v)) + d(p_{j_v}(\tau_v), v)\\ &\overset{\triangle}{\leq} d(u,\tau) + h_{j_u}(\tau) + h_{j_u}(\tau) + \ell(\tau,\tau_v) + h_{j_v}(\tau_v) + h_{j_v}(\tau_v) + d(\tau_v, v)\\ &= d(u,v) + 2h_{j_u}(\tau) + 2h_{j_v}(\tau_v) \overset{5.2,5.3}{\leq} d(u,v) + 2(1-2c)j_u d(u,v) + (1-2c)j_v d(u,v),\end{aligned}$$

as required. ◁

▷ Claim 14.2. $\hat{d}(u,v) \leq (2k - j_v - 1 - 2cj_v)d(u,v)$

Proof. From Claim 12.1 we have that $\hat{d}(u,v) \leq \texttt{TZQuery}(u,v)$. Therefore, applying Lemma 8 with j_v, we get that:

$$\begin{aligned}\hat{d}(u,v) &\leq (2k - 2j_v - 1)d(u,v) + 2h_{j_v}(v) \overset{5.3}{\leq} (2k - 2j_v - 1)d(u,v) + (1-2c)j_v d(u,v)\\ &= (2k - j_v - 1 - 2cj_v)d(u,v),\end{aligned}$$

as required. ◁

▷ Claim 14.3. $\hat{d}(u,v) \leq (2k - 4j_u c + 2c - 3)d(u,v)$

Proof. From Lemma 9 we have that

$$h_{j_u-1}(\tau) \leq (j_u - 1) \cdot 2d(u,\tau) \tag{5.6}$$

From the minimality of j_u it follows that $p_{j_u-1}(\tau) \notin B(u)$. Thus:

$$h_{j_u}(u) \leq d(u, p_{j_u-1}(\tau)) \overset{\triangle}{\leq} d(u,\tau) + h_{j_u-1}(\tau) \overset{5.6}{\leq} d(u,\tau) + (2j_u - 1)d(u,\tau) \overset{5.1}{\leq} (1/2 - c)(2j_u - 1)d(u,v) \tag{5.7}$$

From Claim 12.1 we have that $\hat{d}(u,v) \leq \texttt{TZQuery}(u,v)$. Therefore, from Lemma 8 with parameter j_u, we get that

$$\begin{aligned}\hat{d}(u,v) &\leq (2k - 2j_u - 1)d(u,v) + 2h_{j_u}(u) \overset{5.7}{\leq} (2k - 2j_u - 1)d(u,v) + 2(1/2 - c)(2j_u - 1)d(u,v)\\ &= (2k - 4j_u c + 2c - 3)d(u,v), \text{ as required.}\end{aligned}$$

◁

Using the three bounds on $\hat{d}(u,v)$ we get that:

$$\hat{d}(u,v) \leq \min\left(2k - j_v - 1 - 2cj_v, 2k - 4j_u c + 2c - 3, 1 + 2(1-2c)j_u + (1-2c)j_v\right) \cdot d(u,v)$$

From [7] we have that:

$$\hat{d}(u,v) \leq (2k - 4c\frac{k + 2ck - 3}{1 + 4c - 4c^2} + 2c - 3) \cdot d(u,v)$$

[7] `https://www.wolframalpha.com/input?i=2k-y-1-2cy+%3D+2k-4xc%2B2c-3%2C++2k-4xc%2B2c-3+%3D+1%2B2%281-2c%29x%2B%281-2c%29y`, where $j_u = x$ and $j_v = y$

Therefore, we obtained that if $t \geq (1/2+c)d(u,v)$ then $\hat{d}(u,v) \leq (2k - 4c\frac{k+2ck-3}{1+4c-4c^2} + 2c - 3) \cdot d(u,v)$, and if $t \leq (1/2+c)d(u,v)$ then $\hat{d}(u,v) \leq (4k/3)(1+2c)d(u,v)$. We want to choose the optimal value of c such that the maximum of these two bounds is minimal. Since when c increases, one term increases and the other decreases, we want to have that

$$\begin{aligned}(2k - 4c\frac{k+2ck-3}{1+4c-4c^2} + 2c - 3) &= (4k/3)(1+2c)\\ 2 - \frac{4c(2c+1)}{1+4c-4c^2} &= (4/3)(1+2c)\\ \frac{16c^3 - 32c^2 - 6c + 1}{4c^2 - 4c - 1} &= 0\end{aligned}$$

From [8] we have that $c \approx 0.107912$ is the solution for this example, and therefore we have that:

$$\hat{d}(u,v) \leq \frac{4k}{3}d(u,v)(1+2c) \leq (1 + 2 \cdot 0.108)\frac{4k}{3}d(u,v) < 1.622kd(u,v),$$

as required.[9] ◀

Next, we bound the running time of the algorithm.

▶ **Lemma 15.** *The running time of the algorithm is* $\tilde{O}(mn^{1/k} + n^{1+2/k})$

Proof. Computing $B(u)$ for every $u \in V$ takes $\tilde{O}(mn^{1/k})$ time. Constructing H requires $O\left(\sum_{u \in V} |B(u)|^2 + \sum_{(u,v)\in E} |\{p_i(u) \mid i \in [k]\} \times B(v)|\right) = \tilde{O}(n^{1+2/k} + mn^{1/k})$. Estimating the distances takes $O\left(\sum_{(u,v)\in I} |B(u)| \cdot |B(v)|\right) = \tilde{O}(n^{1+2/k})$. Therefore, the running time is $\tilde{O}(mn^{1/k} + n^{1+2/k})$, as required. ◀

Theorem 1 follows from Lemmas 14 and 15.

References

1 Amir Abboud and Greg Bodwin. Reachability preservers: New extremal bounds and approximation algorithms. In Artur Czumaj, editor, *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 1865–1883, New Orleans, LA, USA, january 7–10 2018. SIAM. doi:10.1137/1.9781611975031.122.

2 Amir Abboud, Karl Bringmann, and Nick Fischer. Stronger 3-sum lower bounds for approximate distance oracles via additive combinatorics. In Barna Saha and Rocco A. Servedio, editors, *Proceedings of the 55th Annual ACM Symposium on Theory of Computing, STOC 2023, Orlando, FL, USA, June 20-23, 2023*, pages 391–404. ACM, 2023. doi:10.1145/3564246.3585240.

3 Amir Abboud, Karl Bringmann, Seri Khoury, and Or Zamir. Hardness of approximation in p via short cycle removal: cycle detection, distance oracles, and beyond. In Stefano Leonardi and Anupam Gupta, editors, *STOC '22: 54th Annual ACM SIGACT Symposium on Theory of Computing, Rome, Italy, June 20 - 24, 2022*, pages 1487–1500. ACM, 2022. doi:10.1145/3519935.3520066.

[8] https://www.wolframalpha.com/input?i=%281%2B2x%29%5Cfrac%7B4k%7D%7B3%7D+%3D+2k-4x%5Cfrac%7Bk%2B2xk%7D%7B1%2B4x-4x%5E2%7D, where $x = c$

[9] Notice that for $c \approx 0.107912$ we have that $2c - 3$ is negative, and therefore removing this term from the first line does not harm the upper bound.

4 Rachit Agarwal. The space-stretch-time tradeoff in distance oracles. In Andreas S. Schulz and Dorothea Wagner, editors, *Algorithms - ESA 2014 - 22th Annual European Symposium, Wroclaw, Poland, September 8-10, 2014. Proceedings*, volume 8737 of *Lecture Notes in Computer Science*, pages 49–60. Springer, 2014. doi:10.1007/978-3-662-44777-2_5.

5 D. Aingworth, C. Chekuri, P. Indyk, and R. Motwani. Fast estimation of diameter and shortest paths (without matrix multiplication). *SIAM Journal on Computing*, 28(4):1167–1181, 1999. doi:10.1137/S0097539796303421.

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

7 Surender Baswana and Sandeep Sen. A simple and linear time randomized algorithm for computing sparse spanners in weighted graphs. *Random Struct. Algorithms*, 30(4):532–563, 2007. doi:10.1002/RSA.20130.

8 Shiri Chechik, Itay Hoch, and Gur Lifshitz. New approximation algorithms and reductions for n-pairs shortest paths and all-nodes shortest cycles. In Yossi Azar and Debmalya Panigrahi, editors, *Proceedings of the 2025 Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2025, New Orleans, LA, USA, January 12-15, 2025*, pages 5207–5238. SIAM, 2025. doi:10.1137/1.9781611978322.177.

9 Edith Cohen. Fast algorithms for constructing t-spanners and paths with stretch t. *SIAM J. Comput.*, 28(1):210–236, 1998. doi:10.1137/S0097539794261295.

10 Don Coppersmith and Michael Elkin. Sparse sourcewise and pairwise distance preservers. *SIAM Journal on Discrete Mathematics*, 20(2):463–501, 2006. doi:10.1137/050630696.

11 Mina Dalirrooyfard, Ce Jin, Virginia Vassilevska Williams, and Nicole Wein. Approximation algorithms and hardness for n-pairs shortest paths and all-nodes shortest cycles. In *63rd IEEE Annual Symposium on Foundations of Computer Science, FOCS 2022, Denver, CO, USA, October 31 - November 3, 2022*, pages 290–300. IEEE, 2022. doi:10.1109/FOCS54457.2022.00034.

12 Ce Jin and Yinzhan Xu. Removing additive structure in 3sum-based reductions. In *Proceedings of the 55th Annual ACM Symposium on Theory of Computing*, pages 405–418, 2023. doi:10.1145/3564246.3585157.

13 Avi Kadria and Liam Roditty. Faster algorithms for $(2k-1)$-stretch distance oracles, 2025. doi:10.48550/arXiv.2507.06721.

14 Avi Kadria and Liam Roditty. New Approximate Distance Oracles and Their Applications. In *36th International Symposium on Algorithms and Computation (ISAAC 2025)*, volume 359 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 43:1–43:17. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2025. doi:10.4230/LIPIcs.ISAAC.2025.43.

15 Shimon Kogan and Merav Parter. Having hope in hops: New spanners, preservers and lower bounds for hopsets. In *63rd IEEE Annual Symposium on Foundations of Computer Science, FOCS 2022, Denver, CO, USA, October 31 - November 3, 2022*, pages 766–777. IEEE, 2022. doi:10.1109/FOCS54457.2022.00078.

16 Mikkel Thorup and Uri Zwick. Compact routing schemes. In Arnold L. Rosenberg, editor, *Proceedings of the Thirteenth Annual ACM Symposium on Parallel Algorithms and Architectures, SPAA 2001, Heraklion, Crete Island, Greece, July 4-6, 2001*, pages 1–10. ACM, 2001. doi:10.1145/378580.378581.

17 Mikkel Thorup and Uri Zwick. Approximate distance oracles. *J. ACM*, 52(1):1–24, 2005. doi:10.1145/1044731.1044732.

A (k^2+2k)-approximation for t-PSP in $O(m+n^{1+2/k}+t)$ time

In this section, we consider the t-PSP problem in dense weighted graphs, and present a near linear time algorithm for graphs with $m=\Omega(n^{1+2/k})$ edges that has a (k^2+2k)-approximation. We prove:

▶ **Reminder of Theorem 5.** *Let $G = (V, E)$ be a weighted undirected graph, and let $k \geq 3$ be an integer parameter. Let I be a set of n vertex pairs. There is an algorithm that computes in $\tilde{O}(m + n^{1+2/k})$ time, for every $\langle u, v\rangle \in I$, an estimate $\hat{d}(u, v)$ such that $d(u, v) \leq \hat{d}(u, v) \leq (k^2 + 2k) \cdot d(u, v)$.*

Our algorithm relies on the following fast $(2k-1)$-emulator construction, which is a simplified variant of the $(2k-1)$-spanner construction of Baswana and Sen [7]. A graph H is called an α-emulator of G if $d_G(u, v) \leq d_H(u, v) \leq \alpha \cdot d_G(u, v)$, for every pair of vertices $u, v \in V$.

▶ **Lemma 16.** *Let G be a weighted graph. For any integer $k \geq 1$, a $(2k-1)$-emulator with $\tilde{O}(n^{1+1/k})$ edges can be constructed in $\tilde{O}(m)$ time.*

Proof. First, the algorithm computes a vertex hierarchy $V = A_0, A_1, \ldots, A_k = \emptyset$, such that A_i contains every vertex from A_{i-1} with probability $n^{-1/k}$, for every $0 < i < k$. Then, the algorithm computes $d(A_i, u)$, adds $(u, p_i(u))$ with weight $d(u, p_i(u))$ to H, for every $0 < i < k$, and $u \in V$ using k calls to `Dijkstra`'s algorithm.

Then, for every edge $(u, v) \in E$, let $i_{(u,v)}$ be the first index such that $d(u, p_{i_{(u,v)}+1}(u)) > (i_{(u,v)} + 1) \cdot \ell(u, v)$, and add the edge $(p_{i_{(u,v)}}(u), v)$ with weight $d(u, p_{i_{(u,v)}}(u)) + \ell(u, v)$.

It is straightforward to see that the running time is $\tilde{O}(m)$, and the proof that $|H| = O(n^{1+1/k})$ follows from [7].

Next, we move to bound the approximation of the emulator.

▷ Claim 16.1. Let $(u, v) \in E$, it holds that $d_H(u, v) \leq (2i_{(u,v)} + 1)\ell(u, v)$.

Proof. By the definition of $i_{(u,v)}$ we have that $d_G(u, p_{i_{(u,v)}}(u)) \leq i\ell(u, v)$. Since the edge $(p_i(u), v)$ with weight $d_G(u, p_{i_{(u,v)}}(u)) + \ell(u, v)$ is added to H, together with the edge $(u, p_i(u))$ of weight $d(u, p_i(u))$ we get that:

$$d_H(u, v) \overset{\triangle}{\leq} d_G(u, p_{i_{(u,v)}}(u)) + d_G(u, p_{i_{(u,v)}}(u)) + \ell(u, v) \leq (2i + 1)\ell(u, v),$$

as required. ◁

Since $i_{(u,v)} \leq k - 1$ for every edge $(u, v) \in E$, we get from Claim 16.1 that $d_H(u, v) \leq (2k-1)\ell(u, v)$, for every edge $(u, v) \in E$, as required. ◀

We let $\texttt{Emulator}(G, k)$ be the emulator of Lemma 16 on the graph G with parameter k. In addition, our algorithm uses the parameterized distance oracle of Kadria and Roditty [13], which we denote with $\texttt{ADO}_P(G, k, S)$.

▶ **Lemma 17** ([13])**.** *Let $k \geq 1$ and let $S \subseteq V$. There is an $O(n|S|^{\frac{1}{k}})$-space distance oracle that given two vertices $u, v \in V$, returns in $O(k)$ time an estimation $\texttt{ADO}_P(G, k, S).\texttt{Query}(u, v)$ such that*

$$d(u, v) \leq \texttt{ADO}_P(G, k, S).\texttt{Query}(u, v) \leq 2\min(d(u, S), d(v, S)) + (2k-1)d(u, v)$$

The distance oracle is constructed in $O(m|S|^{\frac{1}{k}})$ time.

The t-PSP algorithm works as follows. Let H be a $(2k-1)$-emulator of G constructed with Lemma 16. For every $1 \leq i \leq k-1$, let A_i be the set used in the construction of H. For every i, we construct $\texttt{ADO}_P(H, k-i, A_i)$. Then for every $\langle u, v\rangle \in I$ the algorithm returns $\min_i(\texttt{ADO}_P(H, k'_i, A_i).\texttt{Query}(u, v))$ as the distance estimation. A pseudo-code for our algorithm is presented in Algorithm 3.

In the next lemma, we bound the value $\hat{d}(u, v)$ computed by t-PSP (G, k, I), for any pair $\langle u, v\rangle \in I$.

Algorithm 3 t-PSP (G, k, I).

```
1 H ← Emulator(G, k) [Lemma 16]
2 for i ← 1 to k − 1 do
3 |  Construct ADO_P(H, k − i, A_i) [Lemma 17]
4 return {min_{i∈[k]}(ADO_P(H, k − i, A_i).Query(u, v)) | ⟨u, v⟩ ∈ I}
```

▶ **Lemma 18.** $\hat{d}(u,v) \leq (k^2+2k)\cdot d(u,v)$

Proof. For every edge $(x,y) \in E$, we denote with $i_{(x,y)}$ the first index such that $d(x, p_{i_{(x,y)}+1}(x)) > (i_{(x,y)}+1)\cdot \ell(x,y)$. Let i be $\max_{(x,y)\in P(u,v)}(i_{(x,y)})$. Let $(w,z) \in P(u,v)$ be an edge such that $i_{(w,z)} = i$. Therefore, we have that $d(w, p_i(w)) \leq i \cdot \ell(w,z)$.

Wlog, assume that the edge (w,z) is the first edge in $P(w,v)$, therefore we have that $\ell(w,z) \leq d(w,v)$. Since $d(w,p_i(w)) \leq i\cdot\ell(w,z)$, we get that

$$\begin{aligned} d(u,p_i(u)) \leq d(u,p_i(w)) &\overset{\triangle}{\leq} d(u,w) + d(w,p_i(w)) \leq d(u,w) + i\ell(w,z) \\ &\overset{\ell(w,z)\leq d(w,v)}{\leq} d(u,w) + d(w,v) + (i-1)d(w,v) \leq i\cdot d(u,v), \end{aligned} \tag{A.1}$$

where the last inequality follows from the fact that $d(u,w) + d(w,v) = d(u,v)$.

Since $i \geq i_{(x,y)}$ for every $(x,y) \in P(u,v)$, from Claim 16.1 we have that $d_H(x,y) \leq (2i+1)\ell(x,y)$, for every $(x,y) \in P(u,v)$. Therefore, for the entire path $P(u,v)$ we have that

$$d_H(u,v) \leq (2i+1)d(u,v) \tag{A.2}$$

In the algorithm, we have $\hat{d}(u,v) \leq \texttt{ADO}_P(H, k-i, A_i).\texttt{Query}(u,v)$. From Lemma 17, we have that $\texttt{ADO}_P(H,k-i,A_i).\texttt{Query}(u,v) \leq 2\min(d(u,A_i), d(v,A_i)) + 2(k-i)d_H(u,v)$. Thus:

$$\begin{aligned} \hat{d}(u,v) &\leq 2\min(d(u,p_i(u)), d(v,p_i(v))) + 2(k-i)d_H(u,v) \\ &\overset{A.1,A.2}{\leq} 2\cdot i d(u,v) + 2(k-i)\cdot(2i+1)d(u,v) \\ &= (2(k-i)(2i+1)+2i)d(u,v) = (4ik+2k-4i^2)d(u,v) \leq (k^2+2k)d(u,v), \end{aligned}$$

where the last inequality follows from the following discussion. Let $f(i) = 4ik + 2k - 4i^2$. We have that $f'(i) = 4k - 8i$, and since $f''(i) = -8$ is negative, we have that this function has a maximum when $f'(i) = 0$, i.e., the maximum of the function is when $i = k/2$, therefore $f(i) \leq f(k/2) = k^2 + 2k$, as required. ◀

Next, in the following lemma, we bound the running time of Algorithm 3.

▶ **Lemma 19.** *The running time of t-PSP is* $\tilde{O}(m + n^{1+2/k})$

Proof. From Lemma 16 constructing $\texttt{Emulator}(G,k)$ takes $\tilde{O}(m)$ time. From Lemma 17 constructing $\texttt{ADO}_P(H, k_i', A_i)$ takes

$$\tilde{O}(|H||A_i|^{k_i'}) = \tilde{O}(n^{1+1/k}\cdot(n^{\frac{k-i}{k}})^{\frac{1}{k-i}}) = \tilde{O}(n^{1+1/k+\frac{k-i}{k\cdot(k-i)}}) = \tilde{O}(n^{1+2/k})$$

Overall, we get that the running time is $\tilde{O}(m + k\cdot n^{1+2/k}) = \tilde{O}(m + n^{1+2/k})$, as required. ◀

Theorem 5 follows from Lemmas 18 and 19.

B $\lceil \frac{4k}{3} - \frac{5}{3} \rceil d(u,v)$ approximation in $\tilde{O}(mn^{1/k} + n^{1+2/k})$ time in unweighted graphs

In this section, we present a tighter analysis of the algorithm from Section 4 (Algorithm 1), which yields an improved approximation bound. We show:

▶ **Reminder of Theorem 3.** *Let $G = (V, E)$ be an unweighted undirected graph, and let $k \geq 4$ be an integer parameter. Let I be a set of n vertex pairs. There is an algorithm that computes in $\tilde{O}(mn^{1/k} + n^{1+2/k})$ time, for every $\langle u, v \rangle \in I$, an estimate $\hat{d}(u, v)$ such that $d(u, v) \leq \hat{d}(u, v) \leq \lceil \frac{4k}{3} - \frac{5}{3} \rceil d(u, v)$.*

In the following lemma, we prove that the algorithm from Section 4 has a tighter guarantee on almost all the cases. To solve the remaining case (Claim 20.4.3) we set $\hat{d}(u, v)$ to $\min(\hat{d}(u, v), \min(d(u, w) + d(w, v) \mid w \in (\cup_{u_1 \in B_0(u) \cup C(u,A_1)} B(u_1) \cup C(u_1, A_1)) \cap (B_0(v) \cup C_0(v))))$ in $\tilde{O}(n^{1+2/k})$ time.

▶ **Lemma 20.** $\hat{d}(u, v) \leq \lceil \frac{4k}{3} - \frac{5}{3} \rceil d(u, v)$.

Proof. Let

$$t = d(u, v)/2 + 0.5_{\text{ODD}}, \tag{B.1}$$

and let $\tau \in P(u, v)$ be such that $d(u, \tau) = t$ and $d(v, \tau) = t - 1_{\text{ODD}}$. Let i_u be the first index such that $p_{i_u}(u) \in B(\tau)$ or $p_{i_u}(\tau) \in B(u)$, similarly, let i_v be the first index such that $p_{i_v}(v) \in B(\tau)$ or $p_{i_v}(\tau) \in B(v)$.

From the minimality of i_u, we have that for every $0 \leq i < i_u$ it holds that $p_i(u) \notin B(\tau)$ and $p_i(\tau) \notin B(u)$. (The same holds also for v.) Therefore, we can apply Lemma 7 to get:

$$h_{i_u}(u), h_{i_u}(\tau) \leq i_u \cdot d(u, \tau) = i_u \cdot t \tag{B.2}$$

Similarly, we get that:

$$h_{i_v}(v), h_{i_v}(\tau) \leq i_v \cdot d(v, \tau) = i_v \cdot (t - 1_{\text{ODD}}) \tag{B.3}$$

Next we prove the first bound on $\hat{d}(u, v)$ and show that $\hat{d}(u, v) \leq d(u, v) + 2i_u t + 2i_v(t - 1_{\text{ODD}})$.

▷ Claim 20.1. $\hat{d}(u, v) \leq d(u, v) + 2i_u t + 2i_v(t - 1_{\text{ODD}})$

Proof. Assume, wlog, that $p_{i_u}(u) \in B(\tau)$ and $p_{i_v}(\tau) \in B(v)$ (the other cases follow using similar methods). Since $p_{i_u}(u), p_{i_v}(\tau) \in B(\tau)$ we have that:

$$H(p_{i_u}(u), p_{i_v}(\tau)) \leq d(p_{i_u}(u), \tau) + h_{i_v}(\tau) \tag{B.4}$$

Since $p_{i_u}(u) \in B(u)$ and $p_{i_v}(\tau) \in B(v)$ in the final loop of the algorithm there is an iteration in which $w = p_{i_u}(u)$ and $z = p_{i_v}(\tau)$. Thus, after this iteration, we have that:

$$\begin{aligned}
\hat{d}(u, v) &\leq d(u, p_{i_u}(u)) + H(p_{i_u}(u), p_{i_v}(\tau)) + d(p_{i_v}(\tau), v) \\
&\overset{B.4}{\leq} d(u, v) + (d(p_{i_u}(u), \tau) + h_{i_v}(\tau)) + d(p_{i_v}(\tau), v) \overset{\triangle}{\leq} d(u, v) + 2h_{i_u}(u) + 2h_{i_v}(\tau) \\
&\overset{B.2,B.3}{\leq} d(u, v) + 2i_u t + 2i_v(t - 1_{\text{ODD}})
\end{aligned}$$

◁

Notice that since $\hat{d}(u,v) \leq d(u,v) + 2i_u t + 2i_v(t - 1_{\text{ODD}})$, the case that $i_u \geq i_v$ is harder than the case that $i_v < i_u$. Therefore, for the rest of the proof, we assume that $i_u \geq i_v$. Let $c \geq 0$ be the value such that $i_v = i_u - c$. From Claim 20.1 we get that:

$$\begin{aligned}\hat{d}(u,v) &\overset{20.1}{\leq} d(u,v) + 2i_u t + 2i_v(t - 1_{\text{ODD}}) \overset{i_v = i_u - c}{=} d(u,v) + 2i_u t + 2i_u(t - 1_{\text{ODD}}) - c(t - 1_{\text{ODD}}) \\ &\overset{B.1}{=} d(u,v)(1 + 2i_u - c/2) + (c/2)_{\text{ODD}} \end{aligned} \tag{B.5}$$

Next, we prove the following claim that either bounds $\min(h_{i_u+1}(u), h_{i_u+1}(v))$ or bounds $\hat{d}(u,v)$.

▷ Claim 20.2. One of the following conditions holds:

- $\min(h_{i_u+1}(u), h_{i_u+1}(v)) \leq (i_u + 1)d(u,v)/2 + (c/2 - i_v + 0.5) \cdot 1_{\text{ODD}}$
- $\hat{d}(u,v) \leq kd(u,v) + (c - 2i_v)_{\text{ODD}}$

Proof. From the definition of i_u it follows that for every $i_v < i < i_u$ it holds that $p_i(\tau) \notin B(u)$ and $p_i(u) \notin B(\tau)$. Therefore we can apply Lemma 7 and get that:

$$\begin{aligned} h_{i_u}(u), h_{i_u}(\tau) &\leq (i_u - i_v)d(u,\tau) + h_{i_v}(\tau) \overset{B.3}{\leq} (i_u - i_v)t + i_v \cdot (t - 1_{\text{ODD}}) \\ &\overset{i_v = i_u - c}{=} ct + (i_u - c)(t - 1_{\text{ODD}}) \\ &\overset{B.1}{=} i_u \cdot d(u,v)/2 + (c/2 - i_v) \cdot 1_{\text{ODD}} \end{aligned} \tag{B.6}$$

If $p_{i_u}(\tau) \in B(u) \cap B(v)$, then in the final loop in the algorithm there is an iteration in which $w = z = p_{i_u}(\tau)$, and after this iteration we have that:

$$\hat{d}(u,v) \leq d(u,v) + 2h_{i_u}(\tau) \overset{B.6}{\leq} d(u,v) + 2(i_u \cdot d(u,v)/2 + (c/2 - i_v) \cdot 1_{\text{ODD}}) \overset{i_u \leq k-1}{\leq} kd(u,v) + (c - 2i_v)_{\text{ODD}},$$

as required. Otherwise, we have that $p_{i_u}(\tau) \notin B(u)$ or $p_{i_u}(\tau) \notin B(v)$, and therefore:

$$\begin{aligned} \min(h_{i_u+1}(u), h_{i_u+1}(v)) &\leq h_{i_u}(\tau) + t \overset{B.6}{\leq} i_u \cdot d(u,v)/2 + (c/2 - i_v) \cdot 1_{\text{ODD}} + t \\ &\overset{B.1}{=} (i_u + 1)d(u,v)/2 + (c/2 - i_v + 0.5) \cdot 1_{\text{ODD}}, \end{aligned}$$

as required. ◁

Armed with bounds Claim 20.1 and Claim 20.2 we are ready to prove our lemma. We divide the proof of the lemma into two cases: the case where $c/2 - i_v \leq -1$ and the case where $c/2 - i_v \geq 0$. We start by proving the case where $c/2 - i_v \leq -1$ in the following claim.

▷ Claim 20.3. If $c/2 - i_v \leq -1$ then $\hat{d}(u,v) \leq \lceil \frac{4k}{3} - \frac{5}{3} \rceil d(u,v)$

Proof. Notice, that if $i_u \leq \lceil \frac{2k}{3} - \frac{4}{3} \rceil$, then we get that

$$\begin{aligned} \hat{d}(u,v) &\overset{B.5}{\leq} (1 + 2i_u)d(u,v) - c(t - 1_{\text{ODD}}) \overset{c \geq 0}{\leq} (1 + 2i_u)d(u,v) \\ &\overset{i_u \leq \lceil \frac{2k}{3} - \frac{4}{3} \rceil}{\leq} (1 + 2\lceil \tfrac{2k}{3} - \tfrac{4}{3} \rceil)d(u,v) \leq \lceil \tfrac{4k}{3} - \tfrac{5}{3} \rceil d(u,v), \end{aligned}$$

as required. Therefore, for the rest of the proof of this case, we assume that $i_u > \lceil \frac{2k}{3} - \frac{4}{3} \rceil$, since i_u and $\lceil \frac{2k}{3} - \frac{4}{3} \rceil$ are integers, we get that:

$$i_u \geq \lceil \tfrac{2k}{3} - \tfrac{4}{3} \rceil + 1 = \lceil \tfrac{2k}{3} - \tfrac{1}{3} \rceil \tag{B.7}$$

From Claim 12.1 we have that $\hat{d}(u,v) \leq \texttt{TZQuery}(u,v)$, and therefore from Lemma 8 with parameter (i_u+1) we get that

$$\begin{aligned}\hat{d}(u,v) &\leq d(u,v) + 2(k-1-i_u-1)d(u,v) + 2\min(h_{i_u+1}(u), h_{i_u+1}(v)) \\ &\overset{C\ 20.2}{\leq} (1+2k-2-2i_u-2)d(u,v) + 2((i_u+1)d(u,v)/2 + (c/2-i_v+0.5)\cdot 1_{\text{ODD}}) \\ &= (2k-2-i_u)d(u,v) + (c-2i_v+1)\cdot 1_{\text{ODD}} \overset{c/2-i_v\leq -1}{\leq} (2k-2-i_u)d(u,v) \\ &\overset{B.7}{\leq} (2k-2-\lceil \tfrac{2k}{3} - \tfrac{1}{3}\rceil)d(u,v) \leq \lceil \tfrac{4k}{3} - \tfrac{5}{3}\rceil d(u,v),\end{aligned}$$

as required. Notice that from Claim 20.2 we might have that $\hat{d}(u,v) \leq kd(u,v) + (c-2i_v)_{\text{ODD}}$, but since $c/2-i_v < 0$ we get that in this case $\hat{d}(u,v) \leq kd(u,v) \leq \lceil \frac{4k}{3} - \frac{5}{3}\rceil d(u,v)$, as required. ◁

Next, we consider the case that $c/2 - i_v \geq 0$.

▷ Claim 20.4. If $c/2 - i_v \geq 0$ then $\hat{d}(u,v) \leq \lceil \frac{4k}{3} - \frac{5}{3}\rceil d(u,v)$

Proof. Notice, that if $i_u \leq 2k/3 - 4/3 + c/4 + (c/4)\cdot 1_{\text{ODD}}/d(u,v)$, then:

$$\begin{aligned}\hat{d}(u,v) &\overset{B.5}{\leq} (1+2i_u)d(u,v) - c(t - 1_{\text{ODD}}) = (1+2i_u - c/2)d(u,v) - (c/2)\cdot 1_{\text{ODD}} \\ &\leq (1 + 2(\tfrac{2k}{3} - \frac{4}{3} + c/4 + (c/4)\cdot 1_{\text{ODD}}/d(u,v)) - c/2)d(u,v) - (c/2)\cdot 1_{\text{ODD}} \\ &= (\tfrac{4k}{3} - \tfrac{5}{3})d(u,v),\end{aligned}$$

as required. Therefore, for the rest of the proof, we can assume that:

$$i_u > \tfrac{2k}{3} - \tfrac{4}{3} + c/4 + c/4 \cdot 1_{\text{ODD}}/d(u,v) \tag{B.8}$$

From Claim 12.1 we have that $\hat{d}(u,v) \leq \texttt{TZQuery}(u,v)$, and therefore from Lemma 8 with parameter (i_u+1) we get that:

$$\begin{aligned}\hat{d}(u,v) &\leq d(u,v) + 2(k-1-i_u-1)d(u,v) + 2\min(h_{i_u+1}(u), h_{i_u+1}(v)) \\ &\overset{C\ 20.2}{\leq} (1+2k-2-2i_u-2)d(u,v) + 2((i_u+1)d(u,v)/2 + (c/2-i_v+0.5)\cdot 1_{\text{ODD}}) \\ &\overset{i_v=i_u-c}{=} (2k-2-i_u)d(u,v) + (3c-2i_u+1)\cdot 1_{\text{ODD}}\end{aligned} \tag{B.9}$$

Notice that in Claim 20.2 we might have that $\hat{d}(u,v) \leq kd(u,v) + (c-2i_v)_{\text{ODD}}$, to have that $kd(u,v) + (c-2i_v)_{\text{ODD}} \leq \lceil \frac{4k}{3} - \frac{5}{3}\rceil d(u,v)$, we need to have that either $d(u,v) \geq 4$ or $c < i_u$. Therefore, we divide the proof into three cases. The case that $d(u,v) \geq 4$, the case that $d(u,v) \leq 3$ and $c < i_u$, and the case that $d(u,v) \leq 3$ and $c = i_u$.

▷ Claim 20.4.1. If $d(u,v) \geq 4$ then $\hat{d}(u,v) \leq \lceil \frac{4k}{3} - \frac{5}{3}\rceil d(u,v)$

Proof. Since $d(u,v) \geq 4$, we have that $1_{\text{ODD}} \leq d(u,v)/5$. Therefore:

$$\begin{aligned}\hat{d}(u,v) &\leq (2k-2-i_u)d(u,v) + (3c-2i_u+1)\cdot 1_{\text{ODD}} \\ &\overset{1_{\text{ODD}}\leq d(u,v)/5}{\leq} (2k-2-i_u+11c/20+1/5-2i_u/5)d(u,v) + (c/4)\cdot 1_{\text{ODD}} \\ &= (2k-9/5-7i_u/5+11c/20)d(u,v) + (c/4)\cdot 1_{\text{ODD}} \\ &\overset{c\leq i_u}{\leq} (2k-9/5-i_u+3c/20)d(u,v) \\ &\overset{B.8}{\leq} (2k-9/5-(\tfrac{2k}{3} - \tfrac{4}{3} + c/4 + c/4\cdot 1_{\text{ODD}}/d(u,v)) + 3c/20)d(u,v) + (c/4)\cdot 1_{\text{ODD}} \\ &\leq (\tfrac{4k}{3} - \tfrac{7}{15} - c/10) \leq \tfrac{4k}{3} - \tfrac{7}{15}\end{aligned}$$

Since $\hat{d}(u,v)$ is an integer, we get that $\hat{d}(u,v) \leq \lfloor \frac{4k}{3} - \frac{7}{15} \rfloor$, since k is an integer we get that $\lfloor \frac{4k}{3} - \frac{7}{15} \rfloor \leq \frac{4k}{3} - \frac{5}{3}$, and overall we get that $\hat{d}(u,v) \leq \frac{4k}{3} - \frac{5}{3}$, as required. Notice that in Claim 20.2 we might have that:

$$\hat{d}(u,v) \leq kd(u,v) + (c-2i_v)_{\text{ODD}} \overset{c \leq k-1}{\leq} kd(u,v) + (k-1)_{\text{ODD}} \overset{1_{\text{ODD}} \leq d(u,v)/5}{\leq} 6/5kd(u,v),$$

as required. ◁

Next, we consider the case where $d(u,v) \leq 3$ and $c < i_u$.

▷ Claim 20.4.2. If $d(u,v) \leq 3$ and $c < i_u$ then $\hat{d}(u,v) \leq \lceil \frac{4k}{3} - \frac{5}{3} \rceil d(u,v)$

Proof. We can solve the case that $d(u,v) = 1$ in $O(n)$ time by checking whether $(u,v) \in E$. If $d(u,v) = 2$ then we have that $1_{\text{ODD}} = 0$, and the claim holds using the same methods. If $d(u,v) = 3$ we get that:

$$\begin{aligned}\hat{d}(u,v) &\overset{B.9}{\leq} (2k-2-i_u)d(u,v) + (3c-2i_u+1) \cdot 1_{\text{ODD}} \\ &\overset{d(u,v)=3}{=} 6k-6-3i_u+3c-2i_u+1 = 6k-5i_u+3c-5 \end{aligned} \tag{B.10}$$

In this case, if $i_u \leq (4k-5)/6 + c/3$, then:

$$\hat{d}(u,v) \overset{B.5}{\leq} (1+2i_u-c/2)d(u,v) - (c/2) \cdot 1_{\text{ODD}} \overset{d(u,v)=3}{=} 3+6i_u-2c \overset{i_u \leq (4k-5)/6+c/3}{\leq} (4k-5)$$
$$\overset{d(u,v)=3}{=} (\tfrac{4k}{3} - \tfrac{5}{3}) \cdot d(u,v), \text{ as required.}$$

Therefore, for the rest of the proof, we assume that $i_u > (4k-5)/6 + c/3$. Thus:

$$\begin{aligned}\hat{d}(u,v) &\leq 6k - 5i_u + 3c - 5 < 6k - 3((4k-5)/6 + c/3) - 2i_u + 3c - 5 \leq 4k - 2i_u + 2c - 2.5 \\ &\overset{c \leq i_u - 1}{\leq} 4k - 4.5\end{aligned}$$

Since k and $\hat{d}(u,v)$ are integers, having $\hat{d}(u,v) < 4k - 4.5$ guarantees that $\hat{d}(u,v) \leq 4k-5$, as required since $(\frac{4k}{3} - \frac{5}{3})d(u,v) = 4k-5$. ◁

Next, we complete the proof by considering the special case where $d(u,v) = 3$ and $c = i_u$.

▷ Claim 20.4.3. If $d(u,v) = 3$ and $c = i_u$ then $\hat{d}(u,v) = d(u,v) \leq \lceil \frac{4k}{3} - \frac{5}{3} \rceil d(u,v)$

Proof. From the definition of c, in this case, we have that $i_v = 0$. Let $P(u,v) = (u, u_1, v_1, v)$, since $i_v = 0$, we have that $v_1 \in B_0(v) \cup C(v, A_1)$, and from symmetry we can also get that $u_1 \in B_0(u) \cup C(u, A_1)$, and using similar arguments we get that $v_1 \in B(u_1) \cup C(u_1, A_1)$.

Therefore, we get that $\min(d(u,w) + d(w,v) \mid w \in (\cup_{u_1 \in B_0(u) \cup C(u,A_1)} B(u_1) \cup C(u_1, A_1)) \cap (B_0(v) \cup C_0(v))) = d(u,v)$, and therefore $\hat{d}(u,v) = d(u,v)$, as required. ◁

◁

◀