Improved Algorithms for Distance Selection and Related Problems

Haitao Wang
School of Computing, University of Utah, Salt Lake City, UT, USA

Yiming Zhao[1]
Department of Computer Science, Utah State University, Logan, UT, USA

Abstract

In this paper, we propose new techniques for solving geometric optimization problems involving interpoint distances of a point set in the plane. Given a set P of n points in the plane and an integer $1 \leq k \leq \binom{n}{2}$, the distance selection problem is to find the k-th smallest interpoint distance among all pairs of points of P. The previously best deterministic algorithm solves the problem in $O(n^{4/3} \log^2 n)$ time [Katz and Sharir, 1997]. In this paper, we improve their algorithm to $O(n^{4/3} \log n)$ time. Using similar techniques, we also give improved algorithms on both the two-sided and the one-sided discrete Fréchet distance with shortcuts problem for two point sets in the plane. For the two-sided problem (resp., one-sided problem), we improve the previous work [Avraham, Filtser, Kaplan, Katz, and Sharir, 2015] by a factor of roughly $\log^2(m + n)$ (resp., $(m + n)^\epsilon$), where m and n are the sizes of the two input point sets, respectively. Other problems whose solutions can be improved by our techniques include the reverse shortest path problems for unit-disk graphs. Our techniques are quite general and we believe they will find many other applications in future.

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

Keywords and phrases Geometric optimization, distance selection, Fréchet distance, range searching

Digital Object Identifier 10.4230/LIPIcs.ESA.2023.101

Related Version *Full Version*: `https://arxiv.org/abs/2306.01073`

Funding This research was supported in part by NSF under Grants CCF-2005323 and CCF-2300356.

1 Introduction

In this paper, we propose new techniques for solving geometric optimization problems involving interpoint distances in a point set in the plane. More specifically, the optimal objective value of these problems is equal to the (Euclidean) distance of two points in the set. Our techniques usually yield improvements over the previous work by at least a logarithmic factor (and sometimes a polynomial factor).

The first problem we consider is the *distance selection* problem: Given a set P of n points in the plane and an integer $1 \leq k \leq \binom{n}{2}$, the problem asks for the k-th smallest interpoint distance among all pairs of points of P. The problem can be easily solved in $O(n^2)$ time. The first subquadratic time algorithm was given by Chazelle [10]; the algorithm runs in $O(n^{9/5} \log^{4/5} n)$ time and is based on Yao's technique [21]. Later, Agarwal, Aronov, Sharir, and Suri [1] gave a better algorithm of $O(n^{3/2} \log^{5/2} n)$ time and subsequently Goodrich [13] solved the problem in $O(n^{4/3} \log^{8/3} n)$ time. Katz and Sharir [14] finally presented an $O(n^{4/3} \log^2 n)$ time algorithm. All above are deterministic algorithms. Several randomized algorithms have also been proposed for the problem. The randomized algorithm of [1] runs in $O(n^{4/3} \log^{8/3} n)$ expected time. Matousek [17] gave another randomized algorithm of

[1] Corresponding author.

$O(n^{4/3}\log^{2/3} n)$ expected time. Very recently, Chan and Zheng proposed a randomized algorithm of $O(n^{4/3})$ expected time (see the arXiv version of [9]). Also, the time complexity can be made as a function of k. In particular, Chan's randomized techniques [7] solved the problem in $O(n\log n + n^{2/3}k^{1/3}\log^{5/3} n)$ expected time and Wang [18] recently improved the algorithm to $O(n\log n + n^{2/3}k^{1/3}\log n)$ expected time; these algorithms are particularly interesting when k is relatively small.

In this paper, we present a new deterministic algorithm that solves the distance selection problem in $O(n^{4/3}\log n)$ time. Albeit slower than the randomized algorithm of Chan and Zheng [9], our algorithm is the first progress on the deterministic solution since the work of Katz and Sharir [14] published 25 years ago (30 years if we consider their conference version in SoCG 1993). One technique we introduce is an algorithm for solving the following *partial batched range searching problem.*

▶ **Problem 1** (Partial batched range searching). *Given a set A of m points and a set B of n points in the plane and an interval $(\alpha,\beta]$, one needs to construct two collections of edge-disjoint complete bipartite graphs $\Gamma(A,B,\alpha,\beta) = \{A_t \times B_t \mid A_t \subseteq A, B_t \subseteq B\}$ and $\Pi(A,B,\alpha,\beta) = \{A'_s \times B'_s \mid A'_s \subseteq A, B'_s \subseteq B\}$ such that the following two conditions are satisfied.*

1. *For each pair $(a,b) \in A_t \times B_t \in \Gamma(A,B,\alpha,\beta)$, the (Euclidean) distance $\|ab\|$ between points $a \in A_t$ and $b \in B_t$ is in $(\alpha,\beta]$.*
2. *For any two points $a \in A$ and $b \in B$ with $\|ab\| \in (\alpha,\beta]$, either $\Gamma(A,B,\alpha,\beta)$ has a unique graph $A_t \times B_t$ that contains (a,b) or $\Pi(A,B,\alpha,\beta)$ has a unique graph $A'_s \times B'_s$ that contains (a,b).*

In other words, the two collections Γ and Π together record all pairs (a,b) of points $a \in A$ and $b \in B$ whose distances are in $(\alpha,\beta]$. While all pairs of points recorded in Γ have their distances in $(\alpha,\beta]$, this may not be true for Π. For this reason, we sometimes call the point pairs recorded in Π uncertain pairs*.*

Note that if context is clear, we sometimes use Γ and Π to refer to $\Gamma(A,B,\alpha,\beta)$ and $\Pi(A,B,\alpha,\beta)$, respectively. Also, for short, we use BRS to refer to batched range searching.

In the traditional BRS, which has been studied with many applications, e.g.,[20, 15, 4], the collection Π is $\emptyset$ (and thus Γ itself satisfies the two conditions in Problem 1); for differentiation, we refer to this case as the *complete BRS*. The advantage of the partial problem over the complete problem is that the partial problem can usually be solved faster, with a sacrifice that some uncertain pairs (i.e., those recorded in Π) are left unresolved. As will be seen later, in typical applications the number of those uncertain pairs can be made small enough so that they can be handled easily without affecting the overall runtime of the algorithm. More specifically, we derive an algorithm to compute a solution for the partial BRS, whose runtime is controlled by a parameter (roughly speaking, the runtime increases as the graph sizes of Π decreases). Previously, Katz and Sharir [14] gave an algorithm for the complete problem. Our solution, albeit for the more general partial problem, even improves their algorithm by roughly a logarithmic factor when applied to the complete case.

On the one hand, our partial BRS solution helps achieve our new result for the distance selection problem. On the other hand, combining some techniques for the latter problem, we propose a general algorithmic framework that can be used to solve any geometric optimization problem that involves interpoint distances of a set of points in the plane. Consider such a problem whose optimal objective value (denoted by δ^*) is equal to the distance of two points of a set P of n points in the plane. Assume that the decision problem (i.e., given δ, decide whether $\delta \geq \delta^*$) can be solved in T_D time. A straightforward algorithm for computing δ^* is to use the distance selection algorithm and the decision algorithm to perform binary search on interpoint distances of all pairs of points of P; the algorithm runs in $O(\log n)$ iterations and each iteration takes $O(n^{4/3}\log n + T_D)$ time (if we use our new distance selection algorithm). As such, the total runtime is $O((n^{4/3}\log n + T_D)\log n)$. Using our new framework, the runtime can be bounded by $O((n^{4/3} + T_D)\log n)$, which is faster when $T_D = o(n^{4/3}\log n)$.

One application of this new framework is the *two-sided discrete Fréchet distance with shortcuts* problem, or *two-sided DFD* for short. Fréchet distance is used to measure the similarity between two curves and many of its variations have been studied, e.g., [2, 3, 4, 5, 6, 12]. To reduce the impact of outliers between two (sampled) curves, discrete Fréchet distance with shortcuts was proposed [4, 12]. If outliers of only one curve need to be taken care of, it is called *one-sided DFD*; otherwise it is *two-sided DFD*. Avraham, Filtser, Kaplan, Katz, and Sharir [4] solved the two-sided DFD in $O((m^{2/3}n^{2/3} + m + n)\log^3(m + n))$, where m and n are the numbers of vertices of the two input curves, respectively. Using our new framework, we improve their algorithm to $O((m^{2/3}n^{2/3} \cdot 2^{O(\log^*(m+n))} + m\log n + n\log m)\log(m + n))$ time, an improvement of roughly $O(\log^2(m + n))$.

For the one-sided DFD, the authors of [4] gave a randomized algorithm of $O((m+n)^{6/5+\epsilon})$ expected time, for any constant $\epsilon > 0$. Using our solution to the partial BRS, we improve their algorithm to $O((m+n)^{6/5}\log^{8/5}(m+n))$ expected time. Based on the techniques of [4], Katz and Sharir [15] proposed an algorithmic framework for solving geometric optimization problems that involve interpoint distances in a point set. Consider such a problem whose optimal objective value (denoted by δ^*) is equal to the distance of two points of a set P of n points in the plane. The framework has two main procedures. The first procedure is to compute an interval that contains δ^* and with high probability at most L interpoint distances of P. Using the interval and a *bifurcation tree* technique, the second main procedure finally computes δ^*. Assuming that the decision problem can be solved in T_D time, the first main procedure takes $O(n^{4/3+\epsilon}/L^{1/3} + T_D \cdot \log n \cdot \log\log n)$ expected time and the second one runs in $O(L^{1/2} \cdot T_D \cdot \log n)$ time, resulting in an algorithm of $O(n^{4/3+\epsilon}/L^{1/3} + T_D \cdot \log n \cdot \log\log n + L^{1/2} \cdot T_D \cdot \log n)$ expected time in total [4, 15]. Using our partial BRS solution, we improve the first main procedure to $O(n^{4/3}/L^{1/3} \cdot \log^2 n + T_D \cdot \log n \cdot \log\log n)$ expected time, which eliminates the $O(n^\epsilon)$ factor. Thus, the total expected time of the framework becomes $O(n^{4/3}/L^{1/3} \cdot \log^2 n + T_D \cdot \log n \cdot \log\log n + L^{1/2} \cdot T_D \cdot \log n)$. Our result for the one-sided DFD is a direct application of this framework. More specifically, since $T_D = O(m + n)$ [4], we set $L = (m + n)^{2/5}\log^{6/5}(m + n)$ and replace n by $(m + n)$ in the above time complexity as there are two parameters m and n in the problem.

We demonstrate two more applications of the framework where our new techniques lead to improved results over the previous work: the *reverse shortest paths in unit-disk graphs* and its weighted case. Given a set P of n points in the plane and a parameter $\delta > 0$, the unit-disk graph $G_\delta(P)$ is an undirected graph whose vertex set is P such that an edge connects two points $p, q \in P$ if the (Euclidean) distance between p and q is at most δ. In the unweighted (resp., weighted) case, the weight of each edge is equal to 1 (resp., the distance between the two vertices). Given P, two points $s, t \in P$, and a parameter λ, the problem is to compute the smallest δ^* such that the shortest path length between s and t in $G_{\delta^*}(P)$ is at most λ.

Deterministic algorithms of $O(n^{5/4}\log^{7/4} n)$ and $O(n^{5/4}\log^{5/2} n)$ times are known for the unweighted and weighted problems, respectively [20]. The decision problem for the unweighted case can be solved in $O(n)$ time (after points of P are sorted) [8] while the weighted case can be solved in $O(n\log^2 n)$ time [19]. As such, using their framework, Katz and Sharir [15] solved both problems in $O(n^{6/5+\epsilon})$ expected time (by setting $L = n^{2/5}$). With our improvement to the framework, we can now solve the unweighted problem in $O(n^{6/5}\log^{8/5} n)$ expected time (by setting $L = n^{2/5}\log^{6/5} n$) and solve the weighted case in $O(n^{6/5}\log^{12/5} n)$ expected time (by setting $L = n^{2/5}/\log^{6/5} n$).

In summary, we propose two algorithmic frameworks for geometric optimization problems that involve interpoint distances in a set of points in the plane. The first one is deterministic while the second one is randomized. The first framework is mainly useful when the decision algorithm time T_D is relatively large (e.g., close to $O(n^{4/3})$) while the second one is more interesting when T_D is small (e.g., near linear). Both frameworks rely on our solution to the partial BRS problem. As optimization problems involving interpoint distances are very common in computational geometry, we believe our techniques will find more applications.

Outline. The rest of the paper is organized as follows. Section 2 presents our algorithm for the partial BRS. The distance selection algorithm is described in Section 3. The two-sided DFD problem is solved in Section 4, where we also propose our first algorithmic framework. The one-sided DFD and our second algorithmic framework are discussed in Section 5. Due to the space limit, some details and proofs are omitted but can be found in the full paper.

2 Partial batched range searching

In this section, we present our solution to the partial BRS problem, i.e., Problem 1. We follow the notation in the statement of Problem 1. In particular, $m = |A|$ and $n = |B|$.

For any set P of points and a compact region R in the plane, let $P(R)$ denote the subset of points of P in R, i.e., $P(R) = P \cap R$. For any point p in the plane, with respect to the interval $(\alpha, \beta]$ in Problem 1, let D_p denote the annulus centered at p and having radii α and β (e.g., see Fig. 1); so D_p has an inner boundary circle of radius α and an outer boundary circle of radius β. We assume that D_p includes its outer boundary circle but not its inner boundary circle. In this way, a point q is in D_p if and only if $\|pq\| \in (\alpha, \beta]$. Define $\mathcal{D}$ as the set of all annuli D_p for all points $p \in A$. Define $\mathcal{C}$ to be the set of boundary circles of all annuli of $\mathcal{D}$. Hence, $\mathcal{C}$ consists of $2m$ circles. For any compact region R in the plane, let $\mathcal{C}_R$ denote the subset of circles of $\mathcal{C}$ that intersect the relative interior of R.

An important tool we use is the cuttings [11]. For a parameter $1 \leq r \leq n$, a $(1/r)$-cutting Ξ of size $O(r^2)$ for $\mathcal{C}$ is a collection of $O(r^2)$ constant-complexity cells whose union covers the plane such that the interior of each cell $\sigma \in \Xi$ is intersected by at most m/r circles in $\mathcal{C}$, i.e., $|\mathcal{C}_\sigma| \leq m/r$. We actually use *hierarchical cuttings* [11]. We say that a cutting Ξ' *c-refines* a cutting Ξ if each cell of Ξ' is contained in a single cell of Ξ and every cell of Ξ contains at most c cells of Ξ'. Let Ξ_0 denote the cutting whose single cell is the whole plane. Then we define cuttings $\{\Xi_0, \Xi_1, ..., \Xi_k\}$, in which each Ξ_i, $1 \leq i \leq k$, is a $(1/\rho^i)$-cutting of size $O(\rho^{2i})$ that c-refines Ξ_{i-1}, for two constants ρ and c. By setting $k = \lceil \log_\rho r \rceil$, the last cutting Ξ_k is a $(1/r)$-cutting. The sequence $\{\Xi_0, \Xi_1, ..., \Xi_k\}$ of cuttings is called a hierarchical $(1/r)$-cutting of $\mathcal{C}$. For a cell σ' of Ξ_{i-1}, $1 \leq i \leq k$, that fully contains cell σ of Ξ_i, we say that σ' is the *parent* of σ and σ is a *child* of σ'. Thus the hierarchical $(1/r)$-cutting can be viewed as a tree structure with Ξ_0 as the root.

A hierarchical $(1/r)$-cutting of $\mathcal{C}$ can be computed in $O(mr)$ time, e.g., by the algorithm in [18], which adapts Chazelle's algorithm [11] for hyperplanes. The algorithm also produces the subset $\mathcal{C}_\sigma$ for all cells $\sigma \in \Xi_i$ for all $i = 0, 1, \ldots, k$, implying that the total size of these subsets is $O(mr)$. In particular, each cell of the cutting produced by the algorithm of [18] is a *pseudo-trapezoid* that is bounded by two vertical line segments from left and right, an arc of a circle of $\mathcal{C}$ from top, and an arc of a circle of $\mathcal{C}$ from bottom (e.g., see Fig. 2).

Using cuttings, we obtain the following solution to the partial BRS problem.

▶ **Lemma 1.** *For any r with $1 \leq r \leq \min\{m^{1/3}, n^{1/3}\}$, we can compute in $O(mr \log r + nr)$ time two collections $\Gamma(A, B, \alpha, \beta) = \{A_t \times B_t \mid A_t \subseteq A, B_t \subseteq B\}$ and $\Pi(A, B, \alpha, \beta) = \{A'_s \times B'_s \mid A'_s \subseteq A, B'_s \subseteq B\}$ of edge-disjoint complete bipartite graphs that satisfy the conditions*

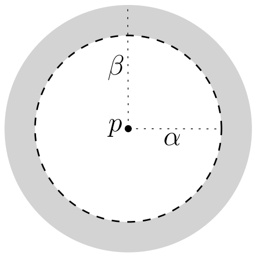

Figure 1 An annulus D_p (the grey region).

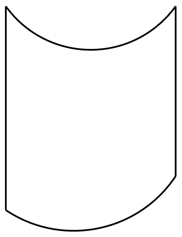

Figure 2 Illustrating a pseudo-trapezoid.

of Problem 1, with the following complexities: (1) $|\Gamma| = O(r^4)$*; (2)* $\sum_t |A_t|, \sum_t |B_t| = O(mr \log r + nr)$*; (3)* $|\Pi| = O(r^4)$*; (4)* $|A'_s| = O(m/r^3)$ *and* $|B'_s| = O(n/r^3)$ *for each* $A'_s \times B'_s \in \Pi$*; (5) the number of pairs of points recorded in* Π *is* $O(r^4 \cdot m/r^3 \cdot n/r^3) = O(mn/r^2)$.

Proof. We begin with constructing a hierarchical $(1/r)$-cutting $\{\Xi_0, \Xi_1, ..., \Xi_k\}$ for $\mathcal{C}$, which takes $O(mr)$ time as discussed above. We use Ξ to refer to the set of all cells σ in all cuttings Ξ_i, $0 \le i \le k$. Next we compute the set $B(\sigma)$ for each cell σ in the cutting (recall that $B(\sigma)$ refers to the subset of points of B inside σ; we call $B(\sigma)$ a *canonical subset*). This can be done in $O(n \log r)$ time in a top-down manner by processing each point of B individually. Specifically, for each point $p \in B$, suppose we know that p is in σ' for a cell σ' in Ξ_{i-1} (which is true initially when $i = 1$ as Ξ_0 has a single cell that is the entire plane). By examining each child of σ' we can find in $O(1)$ time the cell σ of Ξ_i that contains p and then we add p to $B(\sigma)$. Since $k = \Theta(\log r)$, each point of B is stored in $O(\log r)$ canonical subsets and the total size of all canonical subsets $B(\sigma)$ for all cells $\sigma \in \Xi$ is $O(n \log r)$.

Next, for each cell σ of Ξ, we compute another canonical subset $A_\sigma \subseteq A$. Specifically, a point $p \in A$ is in A_σ if the annulus D_p contains σ but not σ's parent. The subsets A_σ for all cells σ of Ξ can be computed in $O(mr)$ time. Indeed, recall that the cutting algorithm already computes $\mathcal{C}_\sigma$ for all cells $\sigma \in \Xi$. For each Ξ_{i-1}, $1 \le i \le k$, for each cell σ' of Ξ_{i-1}, we consider each circle $C \in \mathcal{C}_{\sigma'}$. Let p be the point of A such that C is a bounding circle of the annulus D_p. For each child σ of σ', if D_p fully contains σ, then we add p to A_σ. In this way, A_σ for all cells σ of Ξ can be computed in $O(mr)$ time since $\sum_{0 \le i \le k} \sum_{\sigma' \in \Xi_i} |\mathcal{C}_{\sigma'}| = O(mr)$ and each cell σ' has $O(1)$ children. As such, the total size of A_σ for all cells $\sigma \in \Xi$ is $O(mr)$.

By definition, for each cell $\sigma \in \Xi$, for any point $a \in A_\sigma$ and any point $b \in B(\sigma)$, we have $\|ab\| \in (\alpha, \beta]$. As such, we return $\{A_\sigma \times B(\sigma) \mid \sigma \in \Xi\}$ as a subcollection of $\Gamma(A, B, \alpha, \beta)$ to be computed for the lemma. Note that the complete bipartite graphs of $\{A_\sigma \times B(\sigma) \mid \sigma \in \Xi\}$ are edge-disjoint. The size of the subcollection is equal to the number of cells of the hierarchical cutting, which is $O(r^2)$. Also, we have shown above that $\sum_{\sigma \in \Xi} |A_\sigma| = O(mr)$ and $\sum_{\sigma \in \Xi} |B(\sigma)| = O(n \log r)$.

For each cell σ of the last cutting Ξ_k, we have $|\mathcal{C}_\sigma| \le m/r$. Let $\hat{A}_\sigma$ denote the subset of points $p \in A$ such that D_p has a bounding circle in $\mathcal{C}_\sigma$. We do not know whether distances between points of $\hat{A}_\sigma$ and points of $B(\sigma)$ are in $(\alpha, \beta]$ or not. If $|B(\sigma)| > n/r^2$, then we arbitrarily partition $B(\sigma)$ into subsets of size between $n/(2r^2)$ and n/r^2. We call these subsets *standard subsets* of $B(\sigma)$. Since $|B| = n$ and we have $O(r^2)$ cells in cutting Ξ_k, the number of standard subsets of all cells of Ξ_k is $O(r^2)$. For each standard subset $\hat{B}(\sigma) \subseteq B(\sigma)$, we form a pair $(\hat{A}_\sigma, \hat{B}(\sigma))$ as an "unsolved" *subproblem*. Then we have $O(r^2)$ subproblems. Note that $|\hat{A}_\sigma| \le m/r$ and $|\hat{B}(\sigma)| \le n/r^2$. If we apply the same algorithm recursively on each subproblem, then we have the following recurrence relation (which holds for any $1 \le r \le m$):

$$T(m, n) = O(mr + n \log r) + O(r^2) \cdot T(\frac{m}{r}, \frac{n}{r^2}). \tag{1}$$

Note that if we use $T(m,n)$ to represent the total size of A_t and B_t of all complete bipartite graphs $A_t \times B_t$ in the subcollection of $\Gamma(A,B,\alpha,\beta)$ that have been produced as above, then we have the same recurrence as above. If $N(m,n)$ denotes the number of these graphs, then we have the following recurrence:

$$N(m,n) = O(r^2) + O(r^2) \cdot N(\frac{m}{r}, \frac{n}{r^2}).$$

We now solve the problem in a "dual" setting by switching the roles of A and B, i.e., define annuli centered at points of B and compute the hierarchical cutting for their bounding circles. Then, symmetrically we have the following recurrences (which holds for any $1 \leq r \leq n$):

$$T(m,n) = O(nr + m\log r) + O(r^2) \cdot T(\frac{m}{r^2}, \frac{n}{r}), \tag{2}$$

$$N(m,n) = O(r^2) + O(r^2) \cdot N(\frac{m}{r^2}, \frac{n}{r}).$$

By applying (2) to each subproblem of (1) using the same parameter r and we can obtain the following recurrence:

$$T(m,n) = O(mr\log r + nr) + O(r^4) \cdot T(\frac{m}{r^3}, \frac{n}{r^3}).$$

Similarly, we have

$$N(m,n) = O(r^4) + O(r^4) \cdot N(\frac{m}{r^3}, \frac{n}{r^3}).$$

The above recurrences tell us that in $O(mr\log r + nr)$ time we can compute a collection of $O(r^4)$ edge-disjoint complete bipartite graphs $A_t \times B_t$ with $A_t \subseteq A$ and $B_t \subseteq B$ such that for any two points $a \in A_t$ and $b \in B_t$ their distance $\|ab\|$ lies in $(\alpha, \beta]$. Further, the size of all such A_t's and B_t's is bounded by $O(mr\log r + nr)$. We return the above collection as $\Gamma(A,B,\alpha,\beta)$ for the lemma.

In addition, we have also $O(r^4)$ graphs $A'_s \times B'_s$ with $A'_s \subseteq A$ and $B'_s \subseteq B$ corresponding to the unsolved subproblems $T(m/r^3, n/r^3)$ and we do not know whether $\|ab\| \in (\alpha,\beta]$ for points $a \in A'_s$ and $b \in B'_s$. We return the collection of all such graphs as $\Pi(A,B,\alpha,\beta)$ for the lemma. Hence, $|\Pi(A,B,\alpha,\beta)| = O(r^4)$, and $|A'_s| \leq m/r^3$ and $|B'_s| \leq n/r^3$ for each graph $A'_s \times B'_s$ in the collection. The number of pairs of points recorded in $\Pi(A,B,\alpha,\beta)$ is $O(|\Pi(A,B,\alpha,\beta)| \cdot m/r^3 \cdot n/r^3)$, which is $O(mn/r^2)$. This proves the lemma. ◀

Theorem 2 solves the complete BRS by running the algorithm of Lemma 1 recursively.

▶ **Theorem 2.** *We can compute in $O(m^{2/3}n^{2/3} \cdot 2^{O(\log^*(m+n))} + m\log n + n\log m)$ time a collection $\Gamma(A,B,\alpha,\beta) = \{A_t \times B_t \mid A_t \subseteq A, B_t \subseteq B\}$ of edge-disjoint complete bipartite graphs that satisfy the conditions of Problem 1 (with $\Pi(A,B,\alpha,\beta) = \emptyset$), with the following complexities: (1) $|\Gamma| = O(m^{2/3}n^{2/3} \cdot \log^*(m+n) + m + n)$; (2) $\sum_t |A_t|, \sum_t |B_t| = O(m^{2/3}n^{2/3} \cdot 2^{O(\log^*(m+n))} + m\log n + n\log m)$.*

Proof. To solve the complete BRS problem, the main idea is to apply the recurrence (2) recursively until the size of each subproblem becomes $O(1)$. We first consider the symmetric case where $m = n$. By setting $r = n^{1/3}/\log n$ and applying (2) with $m = n$, we obtain the following

$$T(n,n) = O(n^{4/3}) + O(n^{4/3}/\log^4 n) \cdot T(\log^3 n, \log^3 n). \tag{3}$$

Similarly, we have

$$N(n,n) = O(n^{4/3}/\log^4 n) + O(n^{4/3}/\log^4 n) \cdot N(\log^3 n, \log^3 n). \tag{4}$$

The recurrences solve to $T(n,n) = n^{4/3} \cdot 2^{O(\log^* n)}$ and $N(n,n) = O(n^{4/3} \cdot \log^* n)$. This means that in $n^{4/3} \cdot 2^{O(\log^* n)}$ time we can compute a collection $\Gamma(A,B,\alpha,\beta) = \{A_t \times B_t \mid A_t \subseteq A, B_t \subseteq B\}$ of $O(n^{4/3}\log^* n)$ edge-disjoint complete bipartite graphs, with $\sum_t |A_t|, \sum_t |B_t| = n^{4/3} \cdot 2^{O(\log^* n)}$, and it satisfies the conditions of Problem 1 with $\Pi(A,B,\alpha,\beta) = \emptyset$.

For the asymmetric case, i.e., $m \neq n$, it is solved by utilizing the above symmetric case result; the details can be found in the full paper. ◀

For comparison, Katz and Sharir [14] solved the complete BRS problem in $O((m^{2/3}n^{2/3} + m + n)\log m)$ time by producing $O(m^{2/3}n^{2/3} + m + n)$ complete bipartite graphs whose total vertex set size is $O((m^{2/3}n^{2/3} + m + n)\log m))$. Our result improves their runtime and vertex set size by almost a logarithmic factor with slightly more graphs produced.

3 Distance selection

In this section, we present our algorithm for the distance selection problem. Let P be a set of n points in the plane. Define $\mathcal{E}(P)$ as the set of distances of all pairs of points of P. Given an integer $1 \leq k \leq \binom{n}{2}$, the problem is to find the k-th smallest value in $\mathcal{E}(P)$, denoted by δ^*.

Given any δ, the *decision problem* is to determine whether $\delta \geq \delta^*$. Wang [18] recently gave an $O(n^{4/3})$ time algorithm that can compute the number of values of $\mathcal{E}(P)$ at most δ, denoted by k_δ. Observe that $\delta \geq \delta^*$ if and only if $k_\delta \geq k$. Thus, using Wang's algorithm [18], the decision problem can be solved in $O(n^{4/3})$ time. We should point out that the $O(n^{4/3}\log^2 n)$ time algorithm of Katz and Sharir [14] for computing δ^* utilizes a decision algorithm of $O(n^{4/3}\log n)$ time. However, even if we replace their decision algorithm by Wang's $O(n^{4/3})$ time algorithm, the runtime of the overall algorithm for computing δ^* is still $O(n^{4/3}\log^2 n)$ because other parts of the algorithm dominate the total time. To reduce the overall time to $O(n^{4/3}\log n)$, new techniques are needed, in addition to using the faster $O(n^{4/3})$ time decision algorithm. These new techniques include, for instance, Lemma 1 for the partial BRS problem, as will be seen below.

Before presenting the details of our algorithm, we first give the following lemma, which is critical to our algorithm and is obtained by using Lemma 1.

▶ **Lemma 3.** *Given an interval $(\alpha,\beta]$, Problem 1 with $A = P$ and $B = P$ can be solved in $O(n^{4/3})$ time by computing two collections $\Gamma(P,P,\alpha,\beta) = \{A_t \times B_t \mid A_t, B_t \subseteq P\}$ and $\Pi(P,P,\alpha,\beta) = \{A'_s \times B'_s \mid A'_s, B'_s \subseteq P\}$ with the following complexities: (1) $|\Gamma| = O(n^{4/3}/\log^4 \log n)$; (2) $\sum_t |A_t|, \sum_t |B_t| = O(n^{4/3})$; (3) $|\Pi| = O(n^{4/3}/\log^4 \log n)$; (4) $|A'_s|, |B'_s| = O(\log^3 \log n)$, for each $A'_s \times B'_s \in \Pi$.*

Proof. We first apply Lemma 1 with $A = P$, $B = P$, and $r = n^{1/3}/\log n$. This constructs a collection $\Gamma_1 = \{A_t \times B_t \mid A_t, B_t \subseteq P\}$ of $O(n^{4/3}/\log^4 n)$ edge-disjoint complete bipartite graphs in $O(n^{4/3})$ time. The total size of vertex sets of these graphs is $O(n^{4/3})$, i.e., $\sum_t |A_t|, \sum_t |B_t| = O(n^{4/3})$. We also have a collection $\Pi_1 = \{A'_s \times B'_s \mid A'_s, B'_s \subseteq P\}$ of $O(n^{4/3}/\log^4 n)$ edge-disjoint complete bipartite graphs that record uncertain point pairs, with $|A'_s|, |B'_s| = O(\log^3 n)$.

Hence, the number of uncertain pairs of points of P (i.e., we do not know whether their distances are in $(\alpha,\beta]$) is $\sum_s |A'_s| \cdot |B'_s| = O(n^{4/3}\log^2 n)$. To further reduce this number, we apply Lemma 1 on every pair (A'_s, B'_s) of Π_1. More specifically, for each pair (A'_s, B'_s)

of Π_1, we apply Lemma 1 with $A = A'_s$, $B = B'_s$, and $r = \log n/\log\log n$. This computes a collection Γ_s of $O(\log^4 n/\log^4 \log n)$ edge-disjoint complete bipartite graphs in $O(\log^4 n)$ time; the total size of vertex sets of all graphs in Γ_s is $O(\log^4 n)$. We also have a collection Π_s of $O(\log^4 n/\log^4 \log n)$ edge-disjoint complete bipartite graphs. The size of each vertex set of each graph of Π_s is bounded by $O(\log^3 \log n)$. The total time for Lemma 1 on all pairs (A'_s, B'_s) of Π_1 as above is $O(n^{4/3})$. We return $\Gamma_1 \cup \bigcup_s \Gamma_s$ as collection Γ, and $\bigcup_s \Pi_s$ as collection Π in the lemma statement. As such, the complexities in the lemma hold. ◀

In what follows, we describe our algorithm for computing δ^*. Like Katz and Sharir's algorithm [14], our algorithm proceeds in stages. Initially, we have $I_0 = (0, +\infty]$. In each j-th stage, an interval $I_j = (\alpha_i, \beta_j]$ is computed from I_{j-1} such that I_j must contain δ^* and the number of values of $\mathcal{E}(P)$ in I_j is a constant fraction of that in I_{j-1}. Specifically, we will prove that $|\mathcal{E}(P) \cap I_j| = O(n^2\rho^j)$ holds for each j, for some constant $\rho < 1$. Once $|\mathcal{E}(P) \cap I_j|$ is no more than a threshold (to be given later; as will be seen later, this threshold is not constant, which is a main difference between our algorithm and Katz and Sharir's algorithm [14]), we will compute δ^* directly. In the following we discuss the j-th stage of the algorithm. We assume that we have an interval $I_{j-1} = (\alpha_{j-1}, \beta_{j-1}]$ containing δ^*.

We first apply Lemma 3 with $(\alpha, \beta] = (\alpha_{j-1}, \beta_{j-1}]$. This is another major difference between our algorithm and Katz and Sharir's algorithm [14], where they solved the complete BRS problem, while we only solve a partial problem (this saves time by a logarithmic factor). Applying Lemma 3 produces a collection $\Gamma_{j-1} = \{A_t \times B_t \mid A_t, B_t \subseteq P\}$ of $O(n^{4/3}/\log^4 \log n)$ edge-disjoint complete bipartite graphs, with $\sum_t |A_t|, \sum_t |B_t| = O(n^{4/3})$, as well as another collection Π_{j-1} of $O(n^{4/3}/\log^4 \log n)$ graphs. By Lemma 3 (3) and (4), the number of pairs of points of P in Π_{j-1} is $O(n^{4/3} \log^2 \log n)$.

If $\sum_t |A_t| \cdot |B_t| \leq n^{4/3} \log n$, which is our threshold, then this is the last stage of the algorithm and we compute δ^* directly by Lemma 4. Each edge of the graph in $\Gamma_{j-1} \cup \Pi_{j-1}$ connects two points of P; we say that the distance of the two points is *induced* by the edge.

▶ **Lemma 4.** *If $\sum_t |A_t| \cdot |B_t| \leq n^{4/3} \log n$, then δ^* can be computed in $O(n^{4/3} \log n)$ time.*

Proof. We first explicitly compute the set S of distances induced from edges of all graphs of Γ_{j-1} and Π_{j-1}. Since $\sum_t |A_t| \cdot |B_t| \leq n^{4/3} \log n$ and the number of edges of all graphs of Π_{j-1} is $O(n^{4/3} \log^2 \log n)$, we have $|S| = O(n^{4/3} \log n)$ and S can be computed in $O(n^{4/3} \log n)$ time by brute force. Then, we compute the number $k_{\alpha_{j-1}}$ of values of $\mathcal{E}(P)$ that are at most α_{j-1}, which can be done in $O(n^{4/3})$ time [18]. Observe that δ^* is the $(k - k_{\alpha_{j-1}})$-th smallest value in S. Hence, using the linear time selection algorithm, we can find δ^* in $O(|S|)$ time, which is $O(n^{4/3} \log n)$. ◀

We now assume $\sum_t |A_t| \cdot |B_t| > n^{4/3} \log n$. The rest of the algorithm for the j-th iteration takes $O(n^{4/3})$ time. For each graph $A_t \times B_t \in \Gamma_{j-1}$, if $|A_t| < |B_t|$, then we switch the name of A_t and B_t, i.e., A_t now refers to B_t and B_t refers to the original A_t. Note that this does not change the solution of the partial BRS produced by Lemma 3 and it does not change the complexities of Lemma 3 either. This name change is only for ease of the exposition. Now we have $|A_t| \geq |B_t|$ for each graph $A_t \times B_t \in \Gamma_{j-1}$. Let $m_t = |A_t|$ and $n_t = |B_t|$.

We partition each A_t into $g = \lfloor m_t/n_t \rfloor$ subsets $A_{t1}, A_{t2}, \ldots, A_{tg}$ so that each subset contains n_t elements except that the last subset A_{tg} contains at least n_t but at most $2n_t - 1$ elements. Each pair (A_{ti}, B_t), $1 \leq i \leq g$, can be viewed as a complete bipartite graph. As in [14], we construct a *d-regular LPS-expander graph G_{ti}* on the vertex set $A_{ti} \cup B_t$, for a constant d to be fixed later.[2] The expander G_{ti} has $O(|A_{ti}|+|B_t|)$ edges and can be computed in $O(|A_{ti}|+|B_t|)$ time [14, 16]. Let G_t be the union of all these expander graphs G_{ti} over all $i = 1, 2, \ldots, g$. The construction of G_t takes $\sum_{i=1}^{g} O(|A_{ti}|+|B_t|) = O(|A_t| + \lfloor \frac{m_t}{n_t} \rfloor \cdot |B_t|) = O(|A_t|)$ time. Hence, computing all graphs $\{G_t\}_t$ for all $O(n^{4/3}/\log^4 \log n)$ pairs $A_t \times B_t$ in Γ_{j-1} takes $\sum_t O(|A_t|) = O(n^{4/3})$ time. The number of edges in G_t is $O(|A_t|+|B_t|)$, and thus the number of edges in all graphs $\{G_t\}_t$ is $\sum_t O(|A_t|+|B_t|) = O(n^{4/3})$.

For each edge (a, b) in graph G_t that connects a point $a \in A_t$ and a point $b \in B_t$, we associate it with the interpoint distance $\|ab\|$. We compute all these distances for all graphs $\{G_t\}_t$ to form a set S. The size of S is bounded by the number of edges in all graphs $\{G_t\}_t$, which is $O(n^{4/3})$. Note that all values of S are in the interval I_{j-1}.

One way we could proceed from here is to find the largest value δ_1 of S with $\delta_1 < \delta^*$ and the smallest value δ_2 with $\delta^* \le \delta_2$, and then return $(\delta_1, \delta_2]$ as the interval I_j and finish the j-th stage of the algorithm. Finding δ_1 and δ_2 could be done by binary search on S using the linear time selection algorithm and the $O(n^{4/3})$ time decision algorithm. Then the runtime of this step would be $O(n^{4/3}\log n)$, resulting in a total of $O(n^{4/3}\log^2 n)$ time for the overall algorithm for computing δ^* since there are $O(\log n)$ stages. To improve the time, as in [14], we use the "Cole-like" technique to reduce the number of calls to the decision algorithm to $O(1)$ in each stage, as follows.

We assign a *weight* to each value of S. Note that since each graph $G_{ti} \in G_t$ is a d-regular LPS-expander, the degree of G_{ti} is d [14]. Hence, G_{ti} has at most $(|A_{ti}|+|B_t|)\cdot d/2$ edges and thus it contributes at most $(|A_{ti}|+|B_t|)\cdot d/2$ values to S. We assign each distance induced from G_{ti} a weight equal to $|A_{ti}|\cdot|B_t|/(|A_{ti}|+|B_t|)$. As such, the total weight of the values of S is at most

$$\sum_{t,i}(|A_{ti}|+|B_t|)\cdot\frac{d}{2}\cdot\frac{|A_{ti}|\cdot|B_t|}{|A_{ti}|+|B_t|} = \frac{d}{2}\cdot\sum_{t,i}|A_{ti}|\cdot|B_t| = \frac{d}{2}\cdot m_{j-1},$$

where $m_{j-1} = \sum_t |A_t|\cdot|B_t|$. Recall that $m_{j-1} > n^{4/3}\log n$ and $|B_t| \le |A_{ti}|$ in each G_{ti}. We can assume $n \ge 16$ so that $m_{j-1} \ge 16$. As such, we have the following bound for the weight of each value in S: $|A_{ti}|\cdot|B_t|/(|A_{ti}|+|B_t|) \le |B_t| \le \sqrt{|B_t|\cdot|A_{ti}|} \le \sqrt{m_{j-1}} \le m_{j-1}/4$.

We partition the values of S into at most $2d$ intervals $\{I'_1, I'_2, ..., I'_h\}$, $1 \le h \le 2d$, such that the total weight of values in every interval is at least $m_{j-1}/4$ and but at most $m_{j-1}/2$. The partition can be done in $O(|S|)$ time, which is $O(n^{4/3})$, using the linear time selection algorithm. Then, we invoke the decision algorithm $\log(2d) = O(1)$ times to find the interval I'_l that contains δ^*, for some $1 \le l \le h$. We set $I_j = I'_l$. Since the decision algorithm is called $O(1)$ times, this step takes $O(n^{4/3})$ time. This finishes the j-th stage of the algorithm.

The following Lemma 5 shows that the number of values of $\mathcal{E}(P)$ in I_j is a constant portion of that in I_{j-1}. This guarantees that the algorithm will finish in $O(\log n)$ stages since $|\mathcal{E}(P)| = O(n^2)$. As each stage runs in $O(n^{4/3})$ time (except that the last stage takes $O(n^{4/3}\log n)$ time), the total time of the algorithm is $O(n^{4/3}\log n)$.

▶ **Lemma 5.** *There exists a constant ρ with $0 < \rho < 1$ such that the number of values of $\mathcal{E}(P)$ in I_j is at most ρ times the number of values of $\mathcal{E}(P)$ in I_{j-1}.*

[2] A good summary of definitions and properties of expanders can be found in Section 2 of [14]. Here it suffices for the reader to know the following property (which is needed in the proof of Lemma 5): If X and Y are two vertex subsets of a d-regular expander graph of M vertices and there are fewer than $3M$ edges connecting points of X and points of Y, then $|X|\cdot|Y| \le 9M^2/d$.

Proof. Define n_j (resp., n_{j-1}) as the number of values of $\mathcal{E}(P)$ in I_j (resp., I_{j-1}). Our goal is to find a constant $\rho \in (0,1)$ so that $n_j \le \rho \cdot n_{j-1}$ holds.

Recall that m_{j-1} is the number of distances induced from the graphs of Γ_{j-1}. Define m'_{j-1} as the number of distances induced from the graphs of Π_{j-1}. Define q_j (resp., q'_j) as the number of interpoint distances of $\mathcal{E}(P) \cap I_j$ whose point pairs are recorded in Γ_{j-1} (resp., Π_{j-1}). Note that all interpoint distances induced from graphs of Γ_{j-1} are in I_{j-1}. Hence, $m_{j-1} \le n_{j-1}$. By definition, $n_j = q_j + q'_j$ and $q'_j \le m'_{j-1}$. By Lemma 3 (3) and (4), we have $m'_{j-1} = O(n^{4/3} \log^2 \log n)$.

We first make the following **claim:** there exists a constant $\gamma \in (0, 1/3)$ such that $q_j \le \gamma \cdot m_{j-1}$. The proof of this claim is similar to the analysis in [14] and can be found in the full paper. Next, we prove the lemma by using this claim.

As this is not the last stage of the algorithm (since otherwise δ^* would have already been computed without producing interval I_j), it holds that $m_{j-1} > n^{4/3} \log n$. Since $m'_{j-1} = O(n^{4/3} \log^2 \log n)$, there exists a constant $c' \in (0, 1/3)$ such that $\frac{m'_{j-1}}{m_{j-1}} \le c'$ when n is sufficiently large. As $n_j = q_j + q'_j$, $q'_j \le m'_{j-1}$, and $m_{j-1} \le n_{j-1}$, we can obtain the following using the above claim:

$$n_j = q_j + q'_j \le q_j + m'_{j-1} \le \gamma \cdot m_{j-1} + c' \cdot m_{j-1} \le (\gamma + c') \cdot m_{j-1} \le (\gamma + c') \cdot n_{j-1}.$$

Set $\rho = \gamma + c'$. Since both γ and c' are in $(0, 1/3)$, we have $\rho \in (0, 2/3)$ and $n_j \le \rho \cdot n_{j-1}$. This proves the lemma. ◀

We conclude with the following result. Note that once δ^* is computed, one can find a pair of points of P whose distance is equal to δ^* in additional $O(n^{4/3})$ time [18].

▶ **Theorem 6.** *Given a set P of n points in the plane and an integer $1 \le k \le \binom{n}{2}$, the k-th smallest interpoint distance of P can be computed in $O(n^{4/3} \log n)$ time.*

Remark. Our algorithm can be easily extended to the following *bipartite version* of the distance selection problem: Given a set A of m points and a set B of n points in the plane, and an integer $1 \le k \le mn$, compute the k-th smallest interpoint distance δ^* in the set $\{\|ab\| \mid a \in A, b \in B\}$. This problem can be solved in $O((m^{2/3}n^{2/3} + m \log n + n \log m) \log(m + n))$ time by extending our algorithm. More detailed discussions can be found in the full paper.

4 Two-sided discrete Fréchet distance with shortcuts

In this section, we show that our techniques in Section 3 can be used to solve the two-sided DFD problem. Let $A = \{a_1, a_2, ..., a_m\}$ and $B = \{b_1, b_2, ..., b_n\}$ be two sequences of points in the plane. Consider two frogs connected by an inelastic leash, initially placed at a_1 and b_1, respectively. Each frog is allowed to jump forward at most one step in one move, i.e., if the first frog is currently at a_i, then in the next move it can either jump to a_{i+1} or stay at a_i. Note that frogs are not allowed to go backwards. The *discrete Fréchet distance* (or DFD for short) is defined as the minimum length of the inelastic leash that allows two frogs to reach their destinations, i.e., a_m and b_n, respectively.

Because the Fréchet distance is very sensitive to outliers, to reduce the sensitivity, DFD with outliers have been proposed [4]. Specifically, if we allow the A-frog to jump from its current point to any of its succeeding points in each move but B-frog has to traverse all points in B in order plus one restriction that only one frog is allowed to jump in each move (i.e., in each move one of the frogs must stay still), then this problem is called *one-sided discrete Fréchet distance with shortcuts* (or *one-sided DFD* for short), where the goal is to compute the minimum length of the inelastic leash that allows two frogs to reach their destinations. If we allow both frogs to skip points in their sequences (but again with the restriction that only one frog is allowed to jump in each move), then problem is called *two-sided DFD*.

We focus on the two-sided DFD in this section while the one-sided version will be treated in the next section. Let δ^* denote the optimal objective value, i.e., the minimum length of the leash. Avraham, Filtser, Kaplan, Katz, and Sharir [4] presented an algorithm that can compute δ^* in $O((m^{2/3}n^{2/3} + m + n) \log^3(m + n))$ time. In what follows, we show that our techniques in Section 3 can improve their algorithm to $O((m^{2/3}n^{2/3} \cdot 2^{O(\log^*(m+n))} + m \log n + n \log m) \log(m + n))$ time, roughly a factor of $O(\log^2(m + n))$ faster.

To solve the problem, the authors of [4] first proposed an algorithm to solve the decision problem, i.e., given any δ, decide whether $\delta^* \leq \delta$; the algorithm runs in $O((m^{2/3}n^{2/3} + m + n) \log^2(m + n))$ time. Then, to compute δ^*, the authors of [4] used the bipartite version of the distance selection algorithm from Katz and Sharir [14] for point sets A and B together with their decision algorithm to do binary search on the interpoint distances between points in A and those in B, i.e., in each iteration, using the distance selection algorithm to find the k-th smallest distance δ_k for an appropriate k and then call the decision algorithm on δ_k to decide which way to search. As both the distance selection algorithm [14] and the decision algorithm run in $O((m^{2/3}n^{2/3} + m + n) \log^2(m + n))$ time, computing δ^* takes $O((m^{2/3}n^{2/3} + m + n) \log^3(m + n))$ time.

The following lemma (whose proof is in the full paper) shows that the runtime of their decision algorithm [4] can be improved by a factor of roughly $O(\log^2(m + n))$, by using our result in Theorem 2 for the complete BRS problem.

▶ **Lemma 7.** *Given any δ, we can decide whether the two-sided DFD $\delta^* \leq \delta$ in $O(m^{2/3}n^{2/3} \cdot 2^{O(\log^*(m+n))} + m \log n + n \log m)$ time.*

Improving the optimization algorithm for computing δ^*. With our new $O((m^{2/3}n^{2/3} + m \log n + n \log m) \log(m + n))$ time bipartite distance selection algorithm in Section 3 and the above faster decision algorithm, following the same binary search scheme as discussed above, δ^* can be computed in $O((m^{2/3}n^{2/3} + m \log n + n \log m) \log^2(m + n))$ time, a logarithmic factor improvement over the result of [4]. Notice that the time is dominated by the calls to the bipartite distance selection algorithm.

To further improve the algorithm, an observation is that we do not have to call the distance selection algorithm as an oracle and instead we can use that algorithm as a framework and replace the decision algorithm of the distance selection problem by the decision algorithm of the two-sided DFD problem. This will roughly reduce another logarithmic factor. The proof of the following theorem provides the details about this idea.

▶ **Theorem 8.** *Given two sequences of points $A = (a_1, a_2, ..., a_m)$ and $B = (b_1, b_2, ..., b_n)$ in the plane, the two-sided DFD problem can be solved in $O((m^{2/3}n^{2/3} \cdot 2^{O(\log^*(m+n))} + m \log n + n \log m) \log(m + n))$ time.*

Proof. Following our distance selection algorithm, we run in stages and each j-th stage will compute an interval I_j that contains δ^*. In the j-th stage, we first perform the partial BRS on point sets A and B with respect to interval I_{j-1}, in the same way as before. This produces a collection Γ of $(m^{2/3}n^{2/3}/\log^4 \log(m^2/n) + m^{2/3}n^{2/3}/\log^4 \log(n^2/m) + m + n)$ edge-disjoint complete bipartite graphs that record some pairs of $A \times B$ whose interpoint distances are in I_{j-1}. The total size of vertex sets of all graphs in Γ is $O(m^{2/3}n^{2/3} + m \log n + n \log m)$. In addition, we also have a collection Π of complete bipartite graphs representing $O(m^{2/3}n^{2/3} \log^2 \log(m + n))$ uncertain pairs of $A \times B$. The total runtime is $O(m^{2/3}n^{2/3} + m \log n + n \log m)$.

We next compute the number n_Γ of distances induced from the graphs of Γ. If n_Γ is larger than the threshold $\tau = (m^{2/3}n^{2/3} + m \log n + n \log m) \log(m+n)$, then we use the "Cole-like" technique to perform a binary search on the interpoint distances induced from the expander graphs that are built on the vertex sets of the graphs in Γ, which calls the decision algorithm $O(1)$ times. The runtime for this stage is $O(m^{2/3}n^{2/3} \cdot 2^{O(\log^*(m+n))} + m \log n + n \log m)$. If $n_\Gamma \leq \tau$, then we reach the last stage of the algorithm and we can compute δ^* as follows. We compute the interpoint distances induced from the graphs in Γ and Π. The total number of such distances is $O((m^{2/3}n^{2/3} + m \log n + n \log m) \log(m+n))$. Using the decision algorithm and the linear time selection algorithm, a binary search on these interpoint distances is performed to compute δ^*, which takes $O((m^{2/3}n^{2/3} \cdot 2^{O(\log^*(m+n))} + m \log n + n \log m) \log(m+n))$ time as the decision algorithm is called $O(\log(m+n))$ times. The algorithm finishes within $O(\log(m+n))$ stages by an analysis similar to Lemma 5 (indeed, the proof of Lemma 5 does not rely on which decision algorithm is used).

In summary, the total runtime for computing δ^* is bounded by $O((m^{2/3}n^{2/3} \cdot 2^{O(\log^*(m+n))} + m \log n + n \log m) \log(m+n))$. ◀

A general (deterministic) algorithmic framework. The algorithm of Theorem 8 can be made into a general algorithmic framework for solving geometric optimization problems involving interpoint distances in the plane. Specifically, suppose we have an optimization problem $\mathcal{P}$ whose optimal objective value δ^* is equal to $\|ab\|$ for a point $a \in A$ and a point $b \in B$, with A as a set of m points and B as a set of n points in the plane. The goal is to compute δ^*. Suppose that we have a decision algorithm that can determine whether $\delta \geq \delta^*$ in T_D time for any δ. Then, we can compute δ^* by applying exactly the same algorithm of Theorem 8 except that we use the decision algorithm for $\mathcal{P}$ instead. The total time of the algorithm is $O((m^{2/3}n^{2/3} + m \log n + n \log m + T_D) \cdot \log(m+n))$. Note that in the case $T_D = o((m^{2/3}n^{2/3} + m \log n + n \log m) \log(m+n))$ this is faster than the traditional binary search approach by repeatedly invoking the distance selection algorithm.

▶ **Theorem 9.** *Given two sets A and B of m and n points respectively in the plane, any geometric optimization problem whose optimal objective value is equal to the distance between a point of $a \in A$ and a point of $b \in B$ can be solved in $O((m^{2/3}n^{2/3} + m \log n + n \log m + T_D) \cdot \log(m+n))$ time, where T_D is the time for solving the decision version of the problem.*

5 One-sided discrete Fréchet distance with shortcuts

We consider the one-sided DFD problem defined in Section 4. Let δ^* denote the optimal objective value. Avraham, Filtser, Kaplan, Katz, and Sharir [4] proposed an a randomized algorithm of $O((m+n)^{6/5+\epsilon})$ expected time. We show that using our result in Lemma 1 for the partial BRS the runtime of their algorithm can be reduced to $O((m+n)^{6/5} \log^{8/5}(m+n))$.

Define $\mathcal{E}(A, B) = \{\|ab\| \mid a \in A, b \in B\}$. It is known that $\delta^* \in \mathcal{E}(A, B)$ [4]. The decision problem is to decide whether $\delta \geq \delta^*$ for any δ. The authors [4] solved the decision problem in $O(m+n)$ (deterministic) time. To compute δ^*, their algorithm has two main procedures.

The first main procedure computes an interval $(\alpha, \beta]$ that is guaranteed to contain δ^*, and in addition, with high probability the interval contains at most L values of $\mathcal{E}(A, B)$, given any $1 \leq L \leq mn$; the algorithm runs in $O((m+n)^{4/3+\epsilon}/L^{1/3} + (m+n) \log(m+n) \log\log(m+n))$ time, for any $\epsilon > 0$. More specifically, during the course of the algorithm, an interval $(\alpha, \beta]$ containing δ^* is maintained; initially $\alpha = 0$ and $\beta = \infty$. In each iteration, the algorithm first determines, through random sampling, whether the number of values of $\mathcal{E}(A, B)$ in $(\alpha, \beta]$ is at most L with high probability. If so, the algorithm stops by returning the current interval

$(\alpha, \beta]$. Otherwise, a subset R of $O(\log(m+n))$ values of $\mathcal{E}(A, B)$ is sampled which contains with high probability an approximate median (in the middle three quarters) among the values of $\mathcal{E}(A, B)$ in $(\alpha, \beta]$. A binary search guided by the decision algorithm is performed to narrow down the interval $(\alpha, \beta]$; the algorithm then proceeds with the next iteration. As such, after $O(\log(m+n))$ iterations, the algorithm eventually returns an interval $(\alpha, \beta]$ with the property discussed above.

The second main procedure is to find δ^* from $\mathcal{E}(A, B) \cap (\alpha, \beta]$. This is done by using a *bifurcation tree* technique (Lemma 4.4 [4]), whose runtime relies on L', the true number of values of $\mathcal{E}(A, B)$ in $(\alpha, \beta]$. As it is possible that $L' > L$, if the algorithm detects that case happens, then the first main procedure will run one more round from scratch. As $L' < L$ holds with high probability, the expected number of rounds is $O(1)$. If $L' \leq L$, the runtime of the second main procedure is bounded by $O((m+n)L^{1/2}\log(m+n))$.

As such, the expected time of the algorithm is $O((m+n)^{4/3+\epsilon}/L^{1/3} + (m+n)\log(m+n)\log\log(m+n) + (m+n)L^{1/2}\log(m+n))$. Setting L to $O((m+n)^{2/5+\epsilon})$ for another small $\epsilon > 0$, the time can be bounded by $O((m+n)^{6/5+\epsilon})$.

Our improvement. We can improve the runtime of the first main procedure by a factor of $O((m+n)^{\epsilon})$, which leads to the improvement of overall algorithm by a similar factor. To this end, by applying Lemma 1 with $r = (\frac{m+n}{L})^{1/3}$, we first have the following corollary, which improves Lemma 4.1 in [4] (which is needed in the first main procedure).

▶ **Corollary 10.** *Given a set A of m points and a set B of n points in the plane, an interval $(\alpha, \beta]$, and a parameter $1 \leq L \leq mn$, we can compute in $O((m+n)^{4/3}/L^{1/3} \cdot \log(\frac{m+n}{L}))$ time two collections $\Gamma(A, B, \alpha, \beta) = \{A_t \times B_t \mid A_t \subseteq A, B_t \subseteq B\}$ and $\Pi(A, B, \alpha, \beta) = \{A'_s \times B'_s \mid A'_s \subseteq A, B'_s \subseteq B\}$ of edge-disjoint complete bipartite graphs that satisfy the conditions of Problem 1, with the following complexities: (1) $|\Gamma| = O((\frac{m+n}{L})^{4/3})$; (2) $\sum_t |A_t|, \sum_t |B_t| = O((m+n)^{4/3}/L^{1/3} \cdot \log(\frac{m+n}{L}))$; (3) $|\Pi| = O((\frac{m+n}{L})^{4/3})$; (4) $|A'_s| = O(\frac{mL}{m+n})$ and $|B'_s| = O(\frac{nL}{m+n})$ for each $A'_s \times B'_s \in \Pi$; (5) the number of pairs of points recorded in Π is $O((m+n)^{4/3}L^{2/3})$.*

Replacing Lemma 4.1 in [4] by our results in Corollary 10 and following the rest of the algorithm in [4] leads to an algorithm to compute δ^* in $O((m+n)^{6/5}\log^2(m+n))$ time. More details can be found in the full paper, which makes the discussion in the context of a more general algorithmic framework (indeed, a recent result of Katz and Sharir [15] already gave such a framework; here we improve their result by a factor of $O((m+n)^{\epsilon})$ due to Corollary 10). As discussed in Section 1, another immediate application of the framework is the reverse shortest path problem in unit-disk graphs [20].

References

1 Pankaj K. Agarwal, Boris Aronov, Micha Sharir, and Subhash Suri. Selecting distances in the plane. *Algorithmica*, 9(5):495–514, 1993.

2 Pankaj K. Agarwal, Rinat B. Avraham, Haim Kaplan, and Micha Sharir. Computing the discrete Fréchet distance in subquadratic time. *SIAM Journal on Computing*, 43:429–449, 2014.

3 Helmut Alt and Michael Godau. Computing the Fréchet distance between two polygonal curves. *International Journal of Computational Geometry and Applications*, 5:75–91, 1995.

4 Rinat B. Avraham, Omrit Filtser, Haim Kaplan, Matthew J. Katz, and Micha Sharir. The discrete and semicontinuous Fréchet distance with shortcuts via approximate distance counting and selection. *ACM Transactions on Algorithms*, 11(4):Article No. 29, 2015.

5 Kevin Buchin, Maike Buchin, and Yusu Wang. Exact algorithms for partial curve matching via the Fréchet distance. In *Proceedings of the 20th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 645–654, 2009.

6 Maike Buchin, Anne Driemel, and Bettina Speckmann. Computing the Fréchet distance with shortcuts is NP-hard. In *Proceedings of the 30th Annual Symposium on Computational Geometry (SoCG)*, pages 367–376, 2014.

7 Timothy M. Chan. On enumerating and selecting distances. *International Journal of Computational Geometry and Application*, 11:291–304, 2001.

8 Timothy M. Chan and Dimitrios Skrepetos. All-pairs shortest paths in unit-disk graphs in slightly subquadratic time. In *Proceedings of the 27th International Symposium on Algorithms and Computation (ISAAC)*, pages 24:1–24:13, 2016.

9 Timothy M. Chan and Da Wei Zheng. Hopcroft's problem, log-star shaving, 2D fractional cascading, and decision trees. In *Proceedings of the 33rd Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 190–210, 2022. Full version with new results available at `arXiv:2111.03744`.

10 Bernard Chazelle. New techniques for computing order statistics in Euclidean space. In *Proceedings of the 1st Annual Symposium on Computational Geometry (SoCG)*, pages 125–134, 1985.

11 Bernard Chazelle. Cutting hyperplanes for divide-and-conquer. *Discrete and Computational Geometry*, 9(2):145–158, 1993.

12 Anne Driemel and Sariel Har-Peled. Jaywalking your dog: computing the Fréchet distance with shortcuts. *SIAM Journal on Computing*, 42(5):1830–1866, 2013.

13 Michael T. Goodrich. Geometric partitioning made easier, even in parallel. In *Proceedings of the 9th Annual Symposium on Computational Geometry (SoCG)*, pages 73–82, 1993.

14 Matthew J. Katz and Micha Sharir. An expander-based approach to geometric optimization. *SIAM Journal on Computing*, 26(5):1384–1408, 1997.

15 Matthew J. Katz and Micha Sharir. Efficient algorithms for optimization problems involving semi-algebraic range searching. *arXiv*, 2021. `arXiv:2111.02052`.

16 Alexander Lubotzky, Ralph Phillips, and Peter Sarnak. Explicit expanders and the Ramanujan conjectures. In *Proceedings of the 18th Annual ACM Symposium on Theory of Computing (STOC)*, pages 240–246, 1986.

17 Jiří Matoušek. Randomized optimal algorithm for slope selection. *Information Processing Letters*, 39:183–187, 1991.

18 Haitao Wang. Unit-disk range searching and applications. In *Proceedings of the 18th Scandinavian Symposium and Workshops on Algorithm Theory (SWAT)*, pages 32:1–32:17, 2022.

19 Haitao Wang and Jie Xue. Near-optimal algorithms for shortest paths in weighted unit-disk graphs. *Discrete and Computational Geometry*, 64:1141–1166, 2020.

20 Haitao Wang and Yiming Zhao. Reverse shortest path problem for unit-disk graphs. *Journal of Computational Geometry*, 14(1):14–47, 2023.

21 Andrew Chi-Chih Yao. On constructing minimum spanning trees in k-dimensional spaces and related problems. *SIAM Journal on Computing*, 11(4):721–736, 1982.