Computing Maximally Separated Sets in the Plane and Independent Sets in the Intersection Graph of Unit Disks *

Pankaj K. Agarwal† Mark Overmars‡ Micha Sharir§

Abstract

Let S be a set of n points in $\mathbb{R}^2$. Given an integer $1 \leq k \leq n$, we wish to find a *maximally separated subset* $A \subseteq S$ of size k; this is a subset for which the minimum among the $\binom{k}{2}$ pairwise distances between its points is as large as possible. The decision problem associated with this problem is to determine whether there exists $I \subseteq S$, $|I| = k$, so that all $\binom{k}{2}$ pairwise distances in I are at least 2, say. This problem can also be formulated in terms of disk-intersection graphs: Let D be the set of unit disks centered at the points of S. The *disk-intersection* graph G of D connects pairs of points by an edge if the disks centered at those points intersect. I now forms an independent set in the graph G. This problem is known to be NP-Complete if k is part of the input.

In this paper we first present a linear-time approximation algorithm for constant k. Next we give $O(n^{4/3}\mathrm{polylog}(n))$ exact algorithms for the cases $k = 3$ and $k = 4$. We also present a simpler $n^{O(\sqrt{k})}$-time algorithm (as compared with the recent algorithm in [5]) for arbitrary values of k.

1 Introduction

Let S be a set of n points in the plane. We are interested in finding a small subset I of S such that the pairwise distances between points in I are large. To be more precise, let I be a subset of S of cardinality k, for $1 \leq k \leq n$. We define the *separation distance* $d_{\mathrm{sep}}(I)$ to be the minimum among the $\binom{k}{2}$ pairwise distances between its k points. We call I *δ-separated* if $d_{\mathrm{sep}}(I) \geq \delta$. We call I a *maximally separated* subset of S if $d_{\mathrm{sep}}(I) \geq d_{\mathrm{sep}}(I')$ for all subsets $I' \subseteq S$ of size k. Note that a set can have $\Omega(n^{k-1})$ maximally separated k-sets. Let $d^k_{\mathrm{sep}}(S) = \max_{I \subseteq S, |I|=k} d_{\mathrm{sep}}(I)$.

In this paper we study algorithms for computing such maximally separated subsets. We mostly consider small (constant) values of k, but we also address the general case. For the case $k = 2$ the problem is equivalent to finding a diametral pair of S and thus can be solved in $O(n \log n)$ time [6]. For larger k, the problem becomes much more complicated and is known to be NP-Complete if k is part of the input [8].

*Work by P.A. and M.S. was supported by a grant from the U.S.-Israeli Binational Science Foundation. Work by P.A. was also supported by NSF under grants CCR-00-86013 EIA-98-70724, EIA-99-72879, EIA-01-31905, and CCR-02-04118. Work by M.S. was also supported by NSF Grants CCR-97-32101 and CCR-00-98246, by a grant from the Israel Science Fund (for a Center of Excellence in Geometric Computing), and by the Hermann Minkowski–MINERVA Center for Geometry at Tel Aviv University. Part of the research was done during the 2003 Bellairs workshop on computational geometry, organized by Godfried Toussaint, and the 2003 Dagstuhl Workshop on Computational Geometry.

†Department of Computer Science, Duke University, Durham, NC 27708-0129, USA. E-mail: pankaj@cs.duke.edu

‡Department of Computer Science, Utrecht University, the Netherlands. E-mail: markov@cs.uu.nl

§School of Computer Science, Tel Aviv University, Tel Aviv 69978, Israel; and Courant Institute of Mathematical Sciences, New York University, New York, NY 10012, USA. E-mail: michas@post.tau.ac.il

Finding small well-separated subsets is important in certain pattern-matching problems, where the points in the subset form a representation of the total set of points. For example, Vleugels and Veltkamp [18] described a method for fast indexing of multimedia databases using vantage objects. These vantage objects are points in the feature space for the matching problem. It has been observed that the chosen vantage objects best be well-separated.

The decision problem associated with the problem of computing a maximally separated subset of size k asks us to determine whether a δ-separated subset I of size k exists for a given $\delta > 0$. This problem can also be formulated in terms of disk-intersection graphs: Let D be the set of disks of radius $\delta/2$ centered at the points of S. The *disk-intersection* graph G of D has the disks as nodes and two disks are connected by an edge if they intersect. Clearly, a δ-separated subset I is the set of centers of an independent set in G. So the decision problem is equivalent to the problem of finding an independent set of size k in the disk-intersection graph G. Recently, the problem of computing the maximum independent set in intersection graphs have received much attention because of its application in geographic information systems (GIS); see [2, 9, 10] and references therein.

Related work. The problem of computing an independent set in a graph is one of the earliest problems known to be NP-Complete [11]. In fact, for a general graph with n vertices, there cannot be a polynomial-time algorithm with approximation ratio better than $n^{1-\varepsilon}$, for any $\varepsilon > 0$, unless $NP = ZPP$ [13]. The best known algorithm finds an independent set of size $\Omega(\kappa \log^2(n)/n)$, where κ is the size of the maximum independent set in the graph [7]. However, better algorithms are known for intersection graphs of geometric objects. The maximum independent set in the intersection graph of intervals on a line can be computed in polynomial time, but the problem remains NP-Ccomplete for intersection graphs of orthogonal segments, unit disks, and unit squares [8]. For example, $(1+\varepsilon)$-approximation algorithms have been proposed for intersection graphs of unit disks, unit squares, and arbitrary disks [9, 15], and $O(\log n)$-approximation algorithm is known for intersection graphs of rectangles [2].

Little is known about computing maximally spearable sets. Formann and Wagner [10] developed a 2-approximation algorithm under the L_∞-metric. Alber and Fiala [5] present an algorithm that computes an independent set of cardinality k in arbitrary disk-intersection graphs in time $n^{O(\sqrt{k})}$. Their algorithm, however, is rather complicated, and they do not consider cases involving small values of k. Moreover, since they consider the entire graph, their algorithm takes $\Omega(n^2)$ time even for small values of k.

Our results. In this paper we mostly focus on small values of k and develop exact and approximation algorithms. The paper contains four main results:

(i) For constant values of k, we present a simple, linear-time algorithm that returns a subset I of size k such that $d_{\text{sep}}(I) \geq (1-\varepsilon) d^k_{\text{sep}}(S)$. Such an approximation algorithm is suitable for the pattern-matching application mentioned above (Section 2).

(ii) We present $O(n^{4/3}\text{polylog}(n))$ algorithms for computing maximally separated subsets of size 3 and 4 (Sections 3 and 4).

(iii) We also present a simpler $n^{O(\sqrt{k})}$-time algorithm (as compared with the algorithm in [5]) for arbitrary values of k (Section 5).

Our approximation algorithm relies on a standard bucketing technique but with an additional twist. Our exact algorithms for $k = 3, 4$ analyze the underlying geometric structure using results from the theory of araringements and show that one can represent this structure implicitly, which is sufficient for our purpose.

2 An ε-Approximation Algorithm

In this section we show that for any constant k and any $\varepsilon > 0$ we can find in linear time a subset I of S of cardinality k such that $d_{\text{sep}}(I) \geq (1-\varepsilon)d^k_{\text{sep}}(S)$.

If $k = 2$, then we can compute an ε-approximation of a diametral pair in $O(n)$ time [1], so assume that $k \geq 3$. Using induction, we assume that for all $2 \leq k' < k$ an ε-approximation of $d^{k'}_{\text{sep}}(S)$ can be computed in linear time. We compute the smallest axis-parallel bounding box B of S. Let w be the width of B and h the height of B. Without loss of generality we may assume that $w \geq h$.

We first consider the case in which $d^k_{\text{sep}}(S) \leq w/(k+1)$. We subdivide the box B into $k+1$ vertical strips $s_0, \ldots, s_k$, each of width $w/(k+1)$ and let $S_i = S \cap s_i$. Any solution will use points from at most k of these $k+1$ strips. For each strip s_i, we compute an ε-approximation of a maximally separated set in $S \setminus S_i$. The best among those $k+1$ solutions is the answer we are looking for. So let us assume the solution does not use strip s_i.

Let us first consider the case $i = 0$. Let p_l be the point on the left border of the box B. Let I' be an ε-approximate maximally separated set of size $k-1$ in $S \setminus S_0$. Then $I' \cup \{p_l\}$ is a solution because p_l lies at distance at least $w/(k+1)$ from all points in I'. A similar procedure works for $i = k$.

Now consider a value of i between 1 and $k-1$. Let $S_l = \bigcup_{j<i} S_j$ and $S_r = \bigcup_{j>i} S_j$. Since both S_l and S_r are nonempty and the distance between points of S_l and S_r is at least $w/(k+1)$, there exists a solution that uses points from both S_l and S_r. Let us assume we use t points from S_l and $k-t$ points from S_r. We compute an ε-approximate maximally separated set I_l of size t in S_l and a set I_r of size $k-t$ in S_r. $I_l \cup I_r$ form a solution to the problem. We need to repeat this for every value of t between 1 and $k-1$. So for each strip we must solve $2(k-1)$ problems with a size smaller than k. In total we need to solve $O(k^2)$ problems. Denoting by $T_k(n,\varepsilon)$ the maximum time needed to ε-approximate $d^k_{\text{sep}}(S)$ over sets S of n points, we thus obtain a total cost of $O(k^2 T_{k-1}(n,\varepsilon))$.

So we are left with the case in which the maximal separation distance $d^k_{\text{sep}}(S)$ is larger than $w/(k+1)$. Let $\delta = \frac{\varepsilon w}{2\sqrt{2}(k+1)}$. We split the bounding box B of the set S into $O(k^2/\varepsilon^2)$ grid cells of size at most $\delta \times \delta$. we choose an arbitrary point of S from each nonempty cell of the grid. Let A be the resulting set of representative points; $|A| = O(k^2/\varepsilon^2)$. We compute a maximally separable set I of size k for A.

We claim that $d_{\text{sep}}(I) \geq (1-\varepsilon)d^k_{\text{sep}}(S)$. Indeed, let $\{p_1, \ldots, p_k\} \subseteq S$ be a maximally separated set of size k. Assuming ε is small enough, these points will lie in different cells. Let $p'_i \in A$ be the representative point from the cell in which p_i lies, and let $I' = \{p'_1, \ldots, p'_k\}$. It is easily seen that

$$d_{\text{sep}}(I') \geq d^k_{\text{sep}}(S) - 2\sqrt{2}\delta = d^k_{\text{sep}}(S) - \frac{\varepsilon w}{k+1}.$$

Now, as $d^k_{\text{sep}}(S) > w/(k+1)$, it follows that $d_{\text{sep}}(I') > (1-\varepsilon)d^k_{\text{sep}}(S)$. Since we solve the problem exactly for A, $d_{\text{sep}}(I) \geq d_{\text{sep}}(I') > (1-\varepsilon)d^k_{\text{sep}}(S)$. The running time bound $T_k(n,\varepsilon)$ thus satisfies the recurrence $T_k(n,\varepsilon) = O(k^2 T_{k-1}(n,\varepsilon) + C_k(k^2/\varepsilon^2))$, where $C_k(m)$ is the time needed to compute exactly a maximally separated subset of size k in a set of m points. Clearly, the solution of this recurrence is $O(n)$, for any constant k, where the constant of proportionality depends exponentially on k. That is, we have:

Theorem 2.1 *For a set S of n points in the plane and any constants k and $\varepsilon < 1$ we can compute in $O(n)$ time a subset $I \subseteq S$ of size k such that $d_{\text{sep}}(I) \geq (1-\varepsilon)d^k_{\text{sep}}(S)$.*

3 Computing a Maximally Separated Triple

Let S be a set of n points in $\mathbb{R}^2$. We wish to compute a maximally separated triple in S. Our overall approach consists of three steps. First, we perform a binary search on the pairwise distances of S, and for each distance δ we determine whether S contains a δ-separated triple. Next, in order to compute a δ-separated triple, we draw a sufficiently small grid on the bounding box of S so that each point of a δ-separated triple of S lies in a distinct grid cell. We thus reduce the problem of computing a δ-separated triple to a multi-colored variant of this problem. Finally, we compute a trichomatic δ-separated triple in $O(n^{4/3}\log^{10/3} n)$ time. For simplicity, we describe these steps in the reverse order. That is, we first describe the algorithm for the multi-colored version, then we show how to reduce the original decision problem to the multi-colored problem, and finally we sketch the binary-search procedure.

We need a few notations. For a point $p \in \mathbb{R}^2$, let $\mathbb{D}(p)$ denote the disk of unit radius centered at p. For a set A of points in $\mathbb{R}^2$, let $K(A) = \bigcap_{p\in A} \mathbb{D}(p)$. $K(A)$ is a convex region bounded by circular arcs that lie on the boundaries of the disks $\mathbb{D}(p)$, and each disk contributes at most one such arc to $\partial K(A)$. $K(A)$ can be constructed in time $O(|A| \log |A|)$.

3.1 Computing a trichromatic 1-separated triangle

Let S_1, S_2, and S_3 be three sets of points in $\mathbb{R}^2$ that satisfy the following property:

($\triangle$) There is a constant $\delta \le 1/6$ so that, for $i = 1, 2, 3$, S_i is contained in a disk O_i of radius δ centered at a point c_i and $|c_1c_2| = |c_2c_3| = |c_3c_1| = 1$.

Without loss of generality, we assume that $c_1 = (0, 0), c_2 = (1, 0)$, and $c_3 = (1/2, \sqrt{3}/2)$. See Figure 1. Set $n_i = |S_i|$, for $i = 1, 2, 3$. We wish to compute a 1-separated triple in $S_1 \times S_2 \times S_3$. (Clearly, no other triple of points in $S_1 \cup S_2 \cup S_3$ can be 1-separated.)

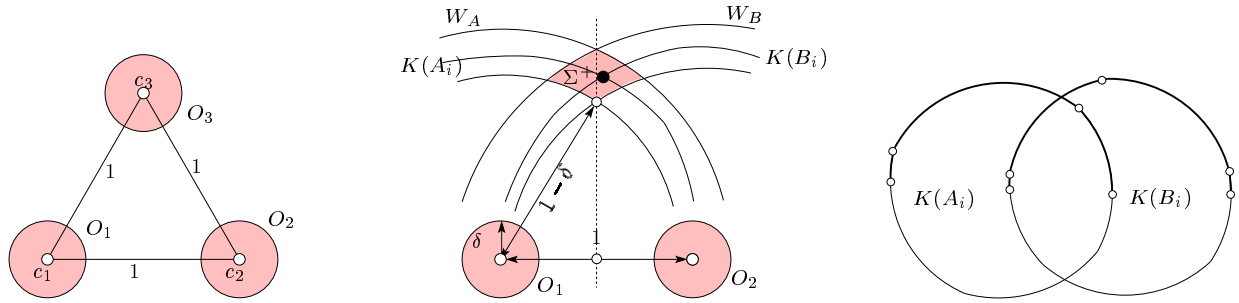

Figure 1: (i) An instance of three point sets (contained in the shaded disks) with property ($\triangle$). (ii) The annuli W_A, W_B and their top intersection Σ^+. (iii) $K(A_i)$, $K(B_i)$, and the edges of Γ_i (drawn as thick lines).

Let $G \subseteq S_1 \times S_2$ denote the bipartite graph

$$G = \{(p, q) \mid p \in S_1, q \in S_2;\ |pq| \ge 1\}.$$

Using the algorithm of Katz and Sharir [16], we compute in $O((n_1^{2/3}n_2^{2/3} + n_1 + n_2)\log n)$ time a family $\mathcal{F} = \{A_1 \times B_1, \ldots, A_u \times B_u\}$, which is a partition of G into complete bipartite graphs, satisfying

$$\sum_i (|A_i| + |B_i|) = O((n_1^{2/3}n_2^{2/3} + n_1 + n_2)\log n).$$

For each $1 \leq i \leq u$, let $R_i = K(A_i) \cup K(B_i)$. Set $\mathcal{R} = \bigcap_{i=1}^{u} R_i$. The following lemma is a straightforward reformulation of the original problem.

Lemma 3.1 *There exists a* 1*-separated triple in* $S_1 \times S_2 \times S_3$ *if and only if* $S_3 \not\subseteq \mathcal{R}$.

The following simple observation is crucial for our algorithm.

Lemma 3.2 *Let* P *be a set of points lying in a disk of radius* δ *centered at a point* c*. Then* $\partial K(P)$ *lies between two concentric circles of radius* $1 + \delta$ *and* $1 - \delta$ *centered at* c*.*

Lemma 3.3 *For each* $1 \leq i \leq u$*, the upper (resp., lower) boundaries of* $K(A_i)$ *and* $K(B_i)$ *cross at exactly one point.*

Proof: Let W_A (resp., W_B) denote the annulus bounded by the concentric circles of radii $1 + \delta$ and $1 - \delta$ centered at c_1 (resp., c_2). By Lemma 3.2, $\partial K(A_i)$ (resp., $\partial K(B_i)$) is contained in W_A (resp., W_B). Therefore $\partial K(A_i) \cap \partial K(B_i) \subseteq W_A \cap W_B$. Since $\delta < 1/6$ and $|c_1c_2| = 1$, the inner circles of W_A and W_B intersect and thus $W_A \cap W_B$ consists of two connected components Σ^+, Σ^-, where Σ^+ lies above the x-axis and Σ^- below the x-axis; see Figure 1(ii). Moreover, by the choice of δ, Σ^+ lies fully to the right of O_1, to the left of O_2, and above both these disks. This implies that within Σ^+, the boundary of each $\mathbb{D}(p)$, for $p \in A_i$, is the graph of a strictly monotone decreasing function, and thus $\partial K(A_i)$ is also the graph of a strictly decreasing function within Σ^+. By a fully symmetric argument, $\partial K(B_i)$ is the graph of a strictly monotone increasing function within Σ^+. Moreover, $\partial K(A_i) \cap \Sigma^+$ is contained in the upper boundary of $K(A_i)$, and similarly for $K(B_i)$, because Σ^+ lies above O_1 and O_2. This is easily seen to imply the assertion of the lemma. □

Lemma 3.3 implies that ∂R_i consists of a connected portion of $\partial K(A_i)$ and a connected portion of $\partial K(B_i)$. The leftmost and the rightmost points of R_i partition ∂R_i into two parts, which we refer to as its upper and lower boundaries. Let Γ_i be the set of circular arcs forming the upper boundary of R_i; we have $|\Gamma_i| \leq |A_i| + |B_i|$. Set $\Gamma = \bigcup_{i=1}^{u} \Gamma_i$; $|\Gamma| \leq \sum_{i=1}^{u}(|A_i| + |B_i|)$. Let $\mathcal{L}_\Gamma$ denote the lower envelope of Γ.

Lemma 3.4 *A point* $p \in S_3$ *lies inside* $\mathcal{R}$ *if and only if* p *lies below the lower envelope* $\mathcal{L}_\Gamma$.

Proof: If $p \in \mathcal{R}$, then it lies below the upper boundary of each R_i, thereby implying that p lies below $\mathcal{L}_\Gamma$. Conversely, suppose that p lies below $\mathcal{L}_\Gamma$. Then p lies below the upper boundary of every R_i. Let Σ^+ be the same as in the proof of Lemma 3.3. Since $|c_1c_3| = |c_2c_3| = 1$, $O_3 \subset \Sigma^+$, and thus S_3 is also contained in Σ^+. The argument in the proof of Lemma 3.3 implies that Σ^+ lies above the lower boundaries of every $K(A_i)$ and of every $K(B_i)$. Hence, p lies in each R_i and thus also in $\mathcal{R}$. □

In view of Lemma 3.4, we may proceed as follows. For each i, we compute $K(A_i)$, $K(B_i)$, R_i, and Γ_i. The total time spent in this step is

$$O\left(\sum_{i=1}^{u}(|A_i| + |B_i|)\log(n_1 + n_2)\right) = O\left((n_1^{2/3}n_2^{2/3} + n_1 + n_2)\log^2(n_1 + n_2)\right).$$

Since each arc in Γ is a portion of the upper boundary of a unit-radius disk, two arcs of Γ intersect in at most one point. Hence, we can compute the lower envelope $\mathcal{L}_\Gamma$ of Γ in $O(|\Gamma| \log n)$ time using the algorithm of Hershberger [14]. For each edge ξ of $\mathcal{L}_\Gamma$ we store the index j such that $\xi \in \Gamma_j$. Finally, for each point $p \in S_3$ we determine whether p lies below or above $\mathcal{L}_\Gamma$. If p lies above $\mathcal{L}_\Gamma$, then the test yields an arc of $\mathcal{L}_\Gamma$ that lies below p. If this arc belongs to Γ_i then we deduce that $p \notin R_i$ (by Lemma 3.4). Then, scanning the points of $A_i \cup B_i$ in additional $O(|A_i| + |B_i|)$ time, we are certain to find a 1-separated triple $(a, b, p) \in A_i \times B_i \times S_3$.

The total running time of the algorithm is $O((n_1^{2/3}n_2^{2/3}+n_1+n_2)\log^2(n_1+n_2)+n_3\log(n_1+n_2))$. Hence, we obtain the following result.

Theorem 3.5 *Let S_1, S_2, and S_3 be three sets of points in $\mathbb{R}^2$ that satisfy property ($\triangle$), and put $n_i = |S_i|$, for $i = 1, 2, 3$. Then one can construct, in $O((n_1^{2/3}n_2^{2/3}+n_1+n_2)\log^2(n_1+n_2)+n_3\log(n_1+n_2))$ time, a 1-separated triple in $S_1 \times S_2 \times S_3$, if one exists, or determine that no such triple exists.*

3.2 Reduction to 3-partite graphs and finding a maximally separated triple

Let S be a set of n points in $\mathbb{R}^2$. We wish to compute a 1-separated triple in S. We fix a small constant $\varepsilon \ll 1/16$, and set $\mu = \lceil 1/\varepsilon \rceil$. We draw a square grid of size ε in the plane. For $i, j \in \mathbb{Z}$, let C_{ij} denote the grid cell $[i\varepsilon, (i+1)\varepsilon) \times [j\varepsilon, (j+1)\varepsilon)$, and let $S_{ij} = S \cap C_{ij}$. Let $\mathcal{C}$ denote the set of nonempty grid cells (i.e., those with $S_{ij} \neq \emptyset$). We construct a graph $\mathcal{G} = (\mathcal{C}, \mathcal{E})$ where $(C, C') \in \mathcal{E}$ if $\min\{|pp'| \mid p \in C, p \in C'\} < 1$.

Lemma 3.6 *If $\mathcal{G}$ is not connected or $\mathcal{C}$ spans more than $3\mu + 1$ columns or rows, then a 1-separated triple in S can be computed in $O(n)$ time.*

Proof: Omitted. Informally, if G is disconnected, then the problem reduces to computing the diameter of various pairwise-disjoint subsets of S. If $\mathcal{C}$ spans more than $3\mu + 1$ rows or columns, then a 1-separated triple can easily be computed in $O(n)$ time. □

By the above lemma, it remains to consider the case where $\mathcal{G}$ is connected and $\mathcal{C}$ spans at most $3\mu + 1$ rows and columns. Clearly, in this case $|\mathcal{C}| \leq (3\mu + 1)^2$. We consider all triples $C_1, C_2, C_3 \in \mathcal{C}$ and determine whether $S_1 \times S_2 \times S_3$ contains a 1-separated triple, where $S_i = C_i \cap S$, for $i = 1, 2, 3$. If the maximum distance between two of these three cells, say, C_1 and C_2, is less than 1, then no 1-separated triple in $S_1 \times S_2 \times S_3$ exists. Hence, we can assume that the maximum distance between every pair of C_1, C_2, C_3 is at least 1. There are four cases to consider, depending on the number k of edges of $\mathcal{G}$ between these three cells:

(i) $k = 0$**;** that is, C_1, C_2, C_3 is an independent set in $\mathcal{G}$. Then any triple in $S_1 \times S_2 \times S_3$ is 1-separated and we return any of them.

(ii) $k = 1$**;** suppose, without loss of generality, that $(C_1, C_2) \in \mathcal{E}$ and $(C_1, C_3), (C_2, C_3) \notin \mathcal{E}$. We compute the diametral pair (p, q) of $S_1 \cup S_2$. If $|pq| \geq 1$, then we return (p, q, r), where r is any point of S_3. If $|pq| < 1$, no triple in $S_1 \times S_2 \times S_3$ is 1-separated.

(iii) $k = 2$**;** suppose, without loss of generality, that $(C_1, C_2), (C_1, C_3) \in \mathcal{E}$ and $(C_2, C_3) \notin \mathcal{E}$. We compute $K(S_2)$ and $K(S_3)$. If a point $p \in S_1$ lies neither in $K(S_2)$ nor in $K(S_3)$, then there exists a pair $(q, r) \in S_2 \times S_3$ so that $p \notin \mathbb{D}(q) \cup \mathbb{D}(r)$ and thus (p, q, r) is 1-separated. If $S_1 \subseteq K(S_2) \cup K(S_3)$ then, arguing as in the proof of Lemma 3.1, no triple in $S_1 \times S_2 \times S_3$ is 1-separated.

(iv) $k = 3$**;** that is, $(C_1, C_2), (C_1, C_3), (C_2, C_3) \in \mathcal{E}$. In other words, for any pair $i \neq j \in \{1, 2, 3\}$ we have

$$\min\{|xy| \mid x \in C_i, y \in C_j\} < 1 \leq \max\{|xy| \mid x \in C_i, y \in C_j\}.$$

By the triangle inequality, this implies that any $x \in C_i$, $y \in C_j$ satisfy $1 - 2\sqrt{2}\varepsilon \leq |xy| \leq 1 + 2\sqrt{2}\varepsilon$. We claim that our choice of ε implies that there exist points $c_1, c_2, c_3 \in \mathbb{R}^2$ so that $|c_ic_j| = 1$ for each pair of distinct points c_i, c_j, and S_i is contained in the disk O_i of radius $\delta \leq 1/6$ centered at c_i, for $i = 1, 2, 3$. To see this, let $c_1 \in C_1$, $c_2 \in C_2$ be such that $|c_1c_2| = 1$. Let $c_3 \in \mathbb{R}^2$ be a point such that $\Delta c_1c_2c_3$ is equilateral and c_3 lies on the same side of the line through c_1 and c_2 as C_3 (our choice of ε is easily seen to imply that C_3 does not intersect such a line). It can be shown that C_3 is fully contained in the disk of radius

$1/6$ centered at c_3. We can therefore use Theorem 3.5 to compute a 1-separated triple in $S_1 \times S_2 \times S_3$, if one exists, or to determine that no such triple exists.

The total running time of the algorithm is dominated by the overall cost of handling case (iv), and is thus, by Theorem 3.5, $O(\mu^3 n^{4/3} \log^2 n)$. We thus obtain the following main result of this section.

Theorem 3.7 *Let S be a set of n points in $\mathbb{R}^2$. We can compute, in $O(n^{4/3} \log^2 n)$ time, a 1-separated triple in S, if one exists, or determine that no such triple exists.*

Finally, we run a binary search on the $\binom{n}{2}$ pairwise distances in S. The kth smallest pairwise distance δ_k in S, for any $1 \leq k \leq \binom{n}{2}$, can be computed in time $O(n^{4/3} \log^2 n)$ [16], and by Theorem 3.7, we can determine whether a δ_k-separated triple exists in S within the same time bound. Hence, we obtain the following.

Theorem 3.8 *Let S be a set of n points in $\mathbb{R}^2$. We can compute, in $O(n^{4/3} \log^3 n)$ time, a maximally separated triple in S.*

4 Computing a Maximally Separated Quadruple

Our overall approach is the same as in Section 3. We first consider a multi-colored version of this problem, in which we are given four sets S_1, S_2, S_3, and S_4 of points, placed "reasonably far" from each other, and we wish to compute a 1-separated quadruple in $S_1 \times S_2 \times S_3 \times S_4$. We present an algorithm for a special configuration of these sets, which we refer to as a *diamond* configuration. We then sketch the overall algorithm, which, as above, reduces the general problem to a constant number of multi-colored instances.

4.1 The diamond configuration

Let S_1, S_2, S_3, and S_4 be four sets of points in $\mathbb{R}^2$ that satisfy the following property:

($\diamond$) There is a constant $\delta \leq 1/8$ so that each S_i is contained in a disk O_i of radius δ centered at c_i so that $|c_1c_3|, |c_2c_3|, |c_1c_4|, |c_2c_4| = 1$, $1 \leq |c_1c_2| \leq |c_3c_4|$, and $|c_3c_4| > 1 + 2\delta$.

Without loss of generality assume that $c_3 = (0, 0)$, c_4 lies on the x-axis to the right of c_3, and c_1 (resp., c_2) lies below (resp., above) the x-axis (in symmetric positions). The conditions on the c_i's imply that $|c_1c_2|, |c_3c_4| \leq \sqrt{3}$. See Figure 2 (i). Set $n_i = |S_i|$, for $i = 1, \ldots, 4$.

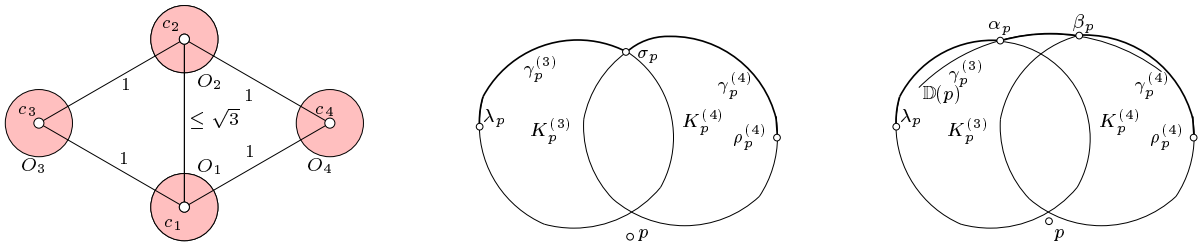

Figure 2: (i) A diamond configuration. (ii) Implicit representation of ∂R_p; $\mathbb{D}(p)$ does not appear on ∂R_p. (iii) Implicit representation of ∂R_p; $\mathbb{D}(p)$ appears on ∂R_p.

For a point $p \in S_1$, let $S_p^{(3)} = \{q \in S_3 \mid |pq| \geq 1\}$ and $S_p^{(4)} = \{q \in S_4 \mid |pq| \geq 1\}$. (We ignore for the time being the issue of efficient construction of these sets; this will be addressed later on.) We remove

from S_1 any point p for which one of these sets is empty, because such a p cannot be part of a 1-separated quadruple. Set

$$K_p^{(3)} = \bigcap_{q \in S_p^{(3)}} \mathbb{D}(q), \qquad K_p^{(4)} = \bigcap_{q \in S_p^{(4)}} \mathbb{D}(q), \qquad R_p = K_p^{(3)} \cup K_p^{(4)} \cup \mathbb{D}(p), \qquad \mathcal{R} = \bigcap_{p \in S_1} R_p.$$

The following lemma is fairly straightforward.

Lemma 4.1 *There exists a* 1*-separated quadruple in* $S_1 \times S_2 \times S_3 \times S_4$ *if and only if* $S_2 \not\subseteq \mathcal{R}$.

The following is a variant of Lemma 3.3, proved in a similar manner.

Lemma 4.2 *(i) For any* $p \in S_1$, $\partial K_p^{(3)}$ *and* $\partial K_p^{(4)}$ *intersect above the* x*-axis at exactly one point* σ_p*, which lies on the upper boundaries of both regions.*

(ii) For any $p \in S_1$, $\partial K_p^{(3)}$ *and* $\partial \mathbb{D}(p)$ *intersect above the* x*-axis exactly once, and similarly for* $\partial K_p^{(4)}$ *and* $\partial \mathbb{D}(p)$.

Definition 4.3 For a point $p \in S_1$, let σ_p denote, as in Lemma 4.2, the unique intersection point of the upper boundaries of $K_p^{(3)}$ and $K_p^{(4)}$, and let α_p (resp., β_p) denote the intersection point of the upper boundary of $K_p^{(3)}$ (resp., $K_p^{(4)}$) with $\mathbb{D}(p)$, if such a point exists.

Consider the upper envelope of the upper boundaries of $K_p^{(3)}$, $K_p^{(4)}$, and $\mathbb{D}(p)$. The preceding analysis implies that the envelope has one of the following two structures: (i) Either $\mathbb{D}(p)$ does not appear on the envelope, and then it consists of a connected portion $\gamma_p^{(3)}$ of the upper boundary of $K_p^{(3)}$ and a connected portion $\gamma_p^{(4)}$ of the upper boundary of $K_p^{(4)}$, meeting at the point σ_p (see Figure 2 (ii)); or (ii) $\mathbb{D}(p)$ appears on the envelope, and then it consists of a connected portion $\gamma_p^{(3)}$ of the upper boundary of $K_p^{(3)}$, a connected portion δ_p of the upper boundary of $\mathbb{D}(p)$, and a connected portion $\gamma_p^{(4)}$ of the upper boundary of $K_p^{(4)}$, so that the first and second portions meet at α_p and the second and third portions meet at β_p (see Figure 2 (iii)).

Let $\Gamma^{(3)} = \{\gamma_p^{(3)} \mid p \in S_1\}$, $\Gamma^{(4)} = \{\gamma_p^{(4)} \mid p \in S_1\}$, and $\Delta = \{\delta_p \mid p \in S_1\}$. Let $\mathcal{L}^{(3)}$ (resp., $\mathcal{L}^{(4)}$, $\mathcal{L}^{(1)}$) denote the lower envelope of $\Gamma^{(3)}$ (resp., $\Gamma^{(4)}$, Δ).

The following lemma is a corollary of Lemma 3.4 and Lemma 4.2. Its proof uses the obvious observation that the lower envelope of $\mathcal{L}^{(3)}$, $\mathcal{L}^{(4)}$, and $\mathcal{L}^{(1)}$ is the same as the lower envelope of the upper boundaries of the regions R_p, for $p \in S_1$. (We follow here the convention that if a lower envelope is undefined at some x, it is assumed to be $+\infty$ there.)

Corollary 4.4 *A point* $q \in S_2$ *lies in* $\mathcal{R}$ *if and only if* q *lies below each of* $\mathcal{L}^{(3)}$, $\mathcal{L}^{(4)}$, *and* $\mathcal{L}^{(1)}$.

We thus compute $\mathcal{L}^{(3)}$, $\mathcal{L}^{(4)}$, and $\mathcal{L}^{(1)}$ separately, and determine whether any point of S_2 lies above any of them. If the answer is yes, we can conclude that a 1-separated quadruple in $S_1 \times S_2 \times S_3 \times S_4$ exists, and we can compute it in additional linear time. Otherwise no such quadruple exists. Computing these envelopes explicitly is however expensive, so we represent them implicitly. We first describe the implicit representation of the envelopes and of its arcs, following a similar representation used by Agarwal et al. [4], and then present the algorithm for computing the envelopes.

Implicit representation of $K_p^{(3)}$, $K_p^{(4)}$, and of the lower envelopes. For a subset $Q \subseteq S_1$, let $\mathcal{L}_Q^{(3)}$ denote the lower envelope of arcs in the set $\{\gamma_p^{(3)} \mid p \in Q\}$. We represent $\mathcal{L}_Q^{(3)}$ by the sequence of its *breakpoints* in increasing order of their x-coordinates. The breakpoints are defined so that each portion ξ of $\mathcal{L}_R^{(3)}$ between

two consecutive breakpoints is contained in a single $\gamma_p^{(3)}$ (such a ξ may overlap with many $\gamma_p^{(3)}$'s, but there is (at least) one point $p \in S_1$ such that ξ is fully contained in $\gamma_p^{(3)}$). We maintain ξ implicitly, by recording a point $p \in R$ that satisfies $\xi \subseteq \gamma_p^{(3)}$.

Let $\mathcal{D} = \{\mathbb{D}(q) \mid q \in S_3\}$. To represent each $\gamma_p^{(3)}$ implicitly, we choose a parameter $n_3 \le s \le n_3^2$, compute a family $\{\mathcal{D}^{(1)}, \ldots, \mathcal{D}^{(u)}\}$ of *canonical* subsets of $\mathcal{D}$ and associate with each $\mathcal{D}^{(i)}$ its intersection region $\bar{K}^{(i)} = \bigcap \mathcal{D}^{(i)}$, such that $\sum_{i=1}^{u} |\mathcal{D}^{(i)}| = O(s \log n_3)$, and such that for any $p \in S_1$, $K_p^{(3)}$ can be represented as the intersection of $O((n_3/\sqrt{s}) \log n_3)$ of these canonical regions $\bar{K}^{(i)}$. Let J_p denote the set of indices of these canonical subsets (i.e., $K_p^{(3)} = \bigcap_{i \in J_p} \bar{K}^{(i)}$). Katz and Sharir [16] have shown that the construction of such a family of canonical sets, and of the corresponding sets of indices $\{J_p\}_{p \in S_1}$, can be accomplished in time $O(s \log n_3)$. For each canonical subset $\mathcal{D}^{(j)}$, the construction of $\bar{K}^{(j)}$ can be performed in $O(|\mathcal{D}^{(j)}| \log |\mathcal{D}^{(j)}|)$ time. We store the vertices of the upper boundary of $\bar{K}^{(j)}$ in a list, sorted in increasing order of their x-coordinates. For each such boundary vertex, we also store the disk whose boundary appears on $\partial \bar{K}^{(j)}$ immediately to its right. The total time spent in computing this implicit representation of the $K_p^{(3)}$'s is $\sum_{j=1}^{u} O(|\mathcal{D}^{(j)}| \log n_3) = O(s \log^2 n_3)$.

As shown by Agarwal *et al.* [4], each of the following three operations on the arcs in $\Gamma^{(3)}$ and $\{\partial K_p^{(3)}\}_{p \in S_1}$ can be performed in $O((n_3/\sqrt{s}) \log^3 n_3)$ time.

(S1) *Leftmost and rightmost points:* Given a point $p \in S_1$, compute the leftmost and the rightmost points of $K_p^{(3)}$.

(S2) *Intersection point(s) with a vertical line:* Given a vertical line ℓ and a point $p \in S_1$, determine the intersection point(s) of ℓ with $\gamma_p^{(3)}$ or with $\partial K_p^{(3)}$.

(S3) *Intersection points with a unit disk:* Given a unit disk $\mathbb{D}$ and a point $p \in S_1$, determine the intersection point(s) of $\mathbb{D}$ with $\gamma_p^{(3)}$ or with $\partial K_p^{(3)}$.

(S4) *Crossing point of two arcs:* Given two points $p, q \in S_1$ and an x-interval $[a, b]$ contained in the x-span of both $\gamma_p^{(3)}$ and $\gamma_q^{(3)}$, determine whether the top boundaries of $K_p^{(3)}$ and $K_q^{(3)}$ cross in $[a, b]$. If so, return their crossing point. If they *weakly cross* in $[a, b]$, i.e., overlap over some subinterval J of $[a, b]$ and their vertical order to the right of J is the reverse of their vertical order to the left of J, then return the leftmost endpoint of their common overlap in $[a, b]$.

In a fully analogous fashion, we choose a parameter $n_4 \le s' \le n_4^2$ and process S_4 in $O(s' \log^2 n_4)$ time to compute an implicit representation of all the arcs $\gamma_p^{(4)}$. Each of the operations (S1)–(S4) on $\Gamma^{(4)}$ can be performed in $O((n_4/\sqrt{s'}) \log^3 n_4)$ time.

Computing $\mathcal{L}^{(3)}$, $\mathcal{L}^{(4)}$, and $\mathcal{L}^{(1)}$. Using the subroutine (S1), we first compute the lefmost point λ_p of $K_p^{(3)}$ and the rightmost point ρ_p of $K_p^{(4)}$, for each $p \in S_1$. Next, using (S3) and (S4), we compute the intersection point σ_p of the upper boundaries of $K_p^{(3)}$ and $K_p^{(4)}$, the intersection point α_p of the upper boundaries of $K_p^{(3)}$ and $\mathbb{D}(p)$, and the intersection point β_p of the upper boundaries of $K_p^{(4)}$ and $\mathbb{D}(p)$. By comparing the x-coordinates of these three points, we can determine whether the upper boundary of R_p is of the first kind (without $\mathbb{D}(p)$) or of the second kind (with $\mathbb{D}(p)$). In the first case, we represent the arc $\gamma_p^{(3)}$ by the interval $[x(\lambda_p), x(\sigma_p)]$ and by the set J_p of indices as described above, and the arc $\gamma_p^{(4)}$ by the interval

$[x(\sigma_p), x(\rho_p)]$ and by the corresponding set J'_p of indices. In the second case, we represent $\gamma_p^{(3)}$ by the interval $[x(\lambda_p), x(\alpha_p)]$, δ_p by the interval $[x(\alpha_p), x(\beta_p)]$, and $\gamma_p^{(4)}$ by the interval $[x(\sigma_p), x(\rho_p)]$; in addition, each arc is associated with its corresponding set J_p of indices.

The x-coordinate of any point $p \in S_1$ is at most $\sqrt{3}/2 + \delta$ and the x-coordinate of the rightmost point of R_p is at least $1 - \delta$. This easily implies that p lies below $K_p^{(3)}$ (i.e., the vertical ray emanating upward from p intersects $K_p^{(3)}$). Since this holds for any point $p \in S_1$, Theorem 2.8 of Agarwal *et al.* [4] (concerning the "pseudo-diskness" of the regions $K_p^{(3)}$) implies that, for any $p, q \in S_1$, $\gamma_p^{(3)}$ and $\gamma_q^{(3)}$ cross in at most one point. A similar argument proves the corresponding claim for $\Gamma^{(4)}$. Hence, each of $\Gamma^{(3)}, \Gamma^{(4)}$ is a collection of pseudo-segments. We can therefore compute the lower envelopes $\mathcal{L}^{(3)}, \mathcal{L}^{(4)}$ using the divide-and-conquer algorithm of Hershberger [14], mentioned above. In the main step of this algorithm, we have available the envelopes $\mathcal{L}_A^{(3)}, \mathcal{L}_B^{(3)}$ of two subsets $A, B \subseteq \Gamma^{(3)}$, and we need to merge these envelopes to compute $\mathcal{L}_{A\cup B}^{(3)}$. The only nontrivial part in the merge step is computing the crossing point of two arcs $\gamma_p^{(3)}$ and $\gamma_q^{(3)}$ in a given x-interval $[a, b]$. Using the subroutine (S4), we can perform this step in $O((n_3/\sqrt{s}) \log^3 n_3)$ time. Plugging this bound into Hershberger's algorithm, we can compute an implicit representation of $\mathcal{L}^{(3)}$ in overall time $O((n_1 n_3/\sqrt{s}) \log^3 n_3 \log n_1 + s \log^2 n_3)$. Similarly, we can compute an implicit representation of $\mathcal{L}^{(4)}$ in time $O((n_1 n_4/\sqrt{s'}) \log^3 n_4 \log n_1 + s' \log^2 n_4)$. Computing $\mathcal{L}^{(1)}$ is easier, since no implicit representation is needed here: We simply have to compute the lower envelope of n_1 upper unit circular arcs, which behave as pseudo-segments, so their envelope can be computed in $O(n_1 \log n_1)$ time (as in [14]). Finally, for each point $p \in S_2$, we determine in $O((n_3/\sqrt{s}) \log^3 n_3 + (n_4/\sqrt{s'}) \log^3 n_4)$ time, using the subroutine (S2), whether p lies above $\mathcal{L}^{(3)}$ or above $\mathcal{L}^{(4)}$. Testing whether p lies above $\mathcal{L}^{(1)}$ is easy to accomplish in $O(\log n)$ time. The total time spent is thus

$$O\left((n_1 + n_2)\left(\frac{n_3}{\sqrt{s}} + \frac{n_4}{\sqrt{s'}}\right) \log^3(n_3 + n_4) \log(n_1 + n_2) + (s + s') \log^2(n_3 + n_4)\right). \tag{4.1}$$

In summary, we have shown:

Lemma 4.5 *Let S_1, S_2, S_3, and S_4 be four sets of points in $\mathbb{R}^2$ that satisfy property ($\diamond$). Let $n_i = |S_i|$, for $i = 1, \ldots, 4$, and let $n_3 \le s \le n_3^2$, $n_4 \le s' \le n_4^2$, be two parameters. One can determine whether $S_1 \times S_2 \times S_3 \times S_4$ contains a 1-separated quadruple, and, if so, compute such a quadruple, within the time bound in (4.1).*

In particular, bounding n_1, n_2, n_3, n_4 by n and choosing $s = s' = n^{4/3} \log^{4/3} n$, we obtain:

Theorem 4.6 *For a set S of n points in the plane of the form $S_1 \cup S_2 \cup S_3 \cup S_4$, where the S_i's satisfy property ($\diamond$), one can determine whether S contains a 1-separated quadruple, and, if so, compute such a quadruple, in $O(n^{4/3} \log^{10/3} n)$ time.*

4.2 Reducing to 4-partite graphs and finding a maximally separated quadruple

As in Section 3, we construct a square grid of size ε, for a sufficiently small constant parameter $\varepsilon > 0$. Let $C_{ij}, S_{ij}, \mathcal{C}, \mathcal{G}$ be as before. Following the same argument used earlier, we can assume that $\mathcal{G}$ is connected and $\mathcal{C}$ spans at most 4μ rows and columns. We now try all quadruples $C_1, C_2, C_3, C_4 \in \mathcal{C}$ and determine whether the corresponding quadruple $S_1 \times S_2 \times S_3 \times S_4$ contains a 1-separated quadruple. We can assume that the maximum distance between any pair of these cells is at least one and that the subgraph induced by these four cells is connected, because if the former assumption is violated then no 1-separated quadruple exists, and if the latter is violated then the problem can be reduced to finding a 1-separated triple (or two

pairs of diametral points). Following a case analysis similar to the previous section, but somewhat more involved, we obtain:

Theorem 4.7 *Let S be a set of n points in $\mathbb{R}^2$. A 1-separated quadruple in S can be computed in time $O(n^{4/3}\log^{10/3} n)$.*

Finally, by performing a binary search on the pairwise distances in S, as above, we obtain the following.

Theorem 4.8 *Let S be a set of n points in $\mathbb{R}^2$. A maximally separated quadruple in S can be computed in $O(n^{4/3}\log^{13/3} n)$ time.*

5 An Exact Algorithm for an Arbitrary Value of k

Let S be a set of n points in $\mathbb{R}^2$, and let $k > 0$ be an integer. We describe an $n^{O(\sqrt{k})}$-time algorithm for computing a maximally separated subset of S of size k. Because of lack of space we only provide a sketch of the algorithm and omit the details. As in the previous sections, we focus on the decision problem: Given a set $\mathcal{D}$ of n unit disks and an integer $1 \leq k \leq n$, is there a subset $I \subset \mathcal{D}$ of k pairwise-disjoint disks? Suppose $\mathcal{D}$ lies inside a strip W of width w. By a sweep-line algorithm we can compute the largest subset of pairwise disjoint disks in $O(n^w)$ time as follows. Our algorithm is similar to the one by Gonzalez [12] for computing a k-center.

For a subset $A \subseteq \mathcal{D}$ and a vertical line ℓ, let $\chi(A, \ell) \subseteq A$ be the set of disks that intersect ℓ. We sweep a vertical line ℓ from left to right, stopping at the leftmost and the rightmost point of each disk in $\mathcal{D}$. At any time, the algorithm maintains a family $\mathcal{F} = \{I_1, \ldots, I_u\}$ of subsets of pairwise-disjoint disks that satisfies the following invariants:

(I.1) For every $1 \leq j \leq u$, no disk in $I_j \subseteq \mathcal{D}$ is contained in the (closed) halfplane lying to the right of the sweep line.

(I.2) For $x \neq y$, $\chi(I_x, \ell) \neq \chi(I_y, \ell)$.

(I.3) If there is a subset $A \subseteq \mathcal{D}$ of pairwise-disjoint disks so that no disk in A lies completely to the right of ℓ, then there is a subset $I_j \in \mathcal{F}$ so that $\chi(I_j, \ell) = \chi(A, \ell)$ ($\chi(I_j, \ell)$ may be empty) and $|A| \leq |I_j|$.

Since at most $O(w)$ pairwise-disjoint disks of $\mathcal{D}$ can intersect ℓ, it can be argued that $|\mathcal{F}| = n^{O(w)}$. Whenever the sweep-line passes through the leftmost or the rightmost point of a disk in $\mathcal{D}$, the family $\mathcal{F}$ can be updated in $n^{O(w)}$ time. We leave it to the reader to verify that the invariants (I.1)–(I.3) ensure that the algorithm computes the largest subset of pairwise-disjoint disks in $\mathcal{D}$.

Hence, if $w \leq \sqrt{k}$, we are done. So assume that $w > \sqrt{k}$. By invoking the above algorithm at most n times, we can determine in $n^{O(\sqrt{k})}$ time whether there is a subset $I \subseteq \mathcal{D}$ of k pairwise-dsjoint disks that lie in a horizontal strip of width at most $\sqrt{k}$. Hence, we can assume that no subset of k pairwise-disjoint disks lies in a horizontal strip of width $\sqrt{k}$. Using a straightforward packing lemma, we can prove the following.

Lemma 5.1 *Let $\mathcal{D}$ be a set of n disks in $\mathbb{R}^2$, and let $I \subseteq \mathcal{D}$ be a subset of k pairwise-disjoint disks that lie in a horizontal strip of width larger than $\sqrt{k}$. Then there exists a horizontal line tangent to one of the disks in $\mathcal{D}$ that intersects $O(\sqrt{k})$ disks of I.*

In view of this lemma, we guess the horizontal line h and a subset $\mathcal{D}_h$ of $t = O(\sqrt{k})$ pairwise-disjoint disks that intersect ℓ. Let $\mathcal{D}^+$ (resp., $\mathcal{D}^-$) be the set of disks that lie completely above (resp., below) h and do not intersect any disk in $\mathcal{D}_h$. We guess an integer $0 \leq l \leq k - t$. We recursively determine whether

$\mathcal{D}^-$ (resp., $\mathcal{D}^+$) contains a subset of l (resp., $k-t-l$) pairwise-disjoint disks. Each subproblem is defined by two horizontal lines, each tangent to a disk in $\mathcal{D}$, and $O(\sqrt{k})$ disks of $\mathcal{D}$, so we need to solve $n^{O(\sqrt{k})}$ subproblems. Using a dynamic-programming approach, we can solve each of these subproblems in $n^{O(\sqrt{k})}$ time. Omitting all the details, which are similar to those in [3], we conclude the following.

Theorem 5.2 *Let S be a set of n points in $\mathbb{R}^2$, and let $1 \leq k \leq n$ be an integer. A maximally separated subset in S of size k can be computed in $n^{O(\sqrt{k})}$ time.*

References

[1] P. K. Agarwal, S. Har-Peled, and K. R. Varadarajan, Approximating extent measures of points, submitted for publication, 2003.

[2] P. K. Agarwal, M. van Kreveld, and S. Suri. Label placement by maximum independent set in rectangles. *Computational Geometry: Theory and Applications*, **11** (1998), 209–218.

[3] P. K. Agarwal and Cecilia M. Procopiuc, Exact and approximation algorithms for clustering, *Algorithmica* **33** (2002), 201–226.

[4] P. K. Agarwal, M. Sharir, and E. Welzl, The discrete 2-center problem, *Discrete Comput. Geom.* **20** (1998), 287–305.

[5] J. Alber and J. Fiala, Geometric separation and exact solutions for the parameterized independent set problem on disk graphs, *Proc. 2nd IFIP Intern. Conf. on Theoretical Computer Science*, 2002, pp. 26–37.

[6] M. de Berg, M. van Kreveld, M. H. Overmars, and O. Schwarzkopf, *Computational Geometry: Algorithms and Applications (2nd ed.)*, Springer-Verlag, Heidelberg, 2000.

[7] R. Boppana and M. M. Halldórsson. Approximating maximum independent sets by excluding subgraphs. *Proc. 2nd Scandinavian Workshop on Algorithm Theory*, 1990, 13–25.

[8] B. N. Clark, C. J. Colbourn and D. S. Johnson, Unit disk graphs, *Discrete Mathematics* **86** (1990), 165–177.

[9] T. Erlebach, K. Jansen, and E. Seidel. Polynomial-time approximation schemes for geometric graphs. In *Proceedings of the Twelth Annual ACM-SIAM Symposium on Discrete Algorithms*, 2001, 671–679.

[10] M. Formann and F. Wangner, A packing problem with applications to letter of maps, *Proc. 7th Annu. Sympos. Comp. Geom.*, 1991, 281–288.

[11] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, New York, NY, 1979.

[12] T. Gonzalez, Clustering to minimize the maximum intercluster distance, *Theoret. Comput. Sci.* **38** (1985), 293–306.

[13] J. Hastad. Clique is hard to approximate within $n^{1-\epsilon}$. *Acta Math.*, **182** (1999), 105–142.

[14] J. Hershberger, Finding the upper envelope of n line segments in $O(n \log n)$ time, *Inform. Process. Lett.* **33** (1989), 169–174.

[15] H. B. Hunt III, M. V. Marathe, V. Radhakrishnan, S. S. Ravi, D. J. Rosenkrantz, and R. E. Stearns. NC-approximation schemes for NP- and PSPACE-hard problems for geometric graphs. *J. Algorithms*, **26** (1998), 238–274.

[16] M. Katz and M. Sharir, An expander-based approach to geometric optimization, *SIAM J. Comput.* **26** (1997), 1384–1408.

[17] M. Sharir and P. K. Agarwal, *Davenport-Schinzel Sequences and Their Geometric Applications*, Cambridge University Press, New York, NY, 1995.

[18] J. Vleugels and R. C. Veltkamp, Efficient image retrieval through vantage objects, *Patt. Recog.* **35** (2002), 69–80.