Theory Comput Syst (2009) 45: 446–469
DOI 10.1007/s00224-008-9135-9

Covering Many or Few Points with Unit Disks

Mark de Berg · Sergio Cabello · Sariel Har-Peled

Published online: 30 July 2008

Abstract Let P be a set of n weighted points. We study approximation algorithms for the following two continuous facility-location problems.

In the first problem we want to place m unit disks, for a given constant $m \geq 1$, such that the total weight of the points from P inside the union of the disks is maximized. We present algorithms that compute, for any fixed $\varepsilon > 0$, a $(1-\varepsilon)$-approximation to the optimal solution in $O(n \log n)$ time.

In the second problem we want to place a single disk with center in a given constant-complexity region X such that the total weight of the points from P inside the disk is minimized. Here we present an algorithm that computes, for any fixed $\varepsilon > 0$, in $O(n \log^2 n)$ expected time a disk that is, with high probability, a $(1+\varepsilon)$-approximation to the optimal solution.

Keywords Facility location · Geometric optimization · Random sample · Weighted points

A preliminary version of this work has appeared in *Approximation and Online Algorithms—WAOA 2006*, LNCS, vol. 4368.

M. de Berg
Department of Computer Science, TU Eindhoven, Eindhoven, The Netherlands

S. Cabello (✉)
Department of Mathematics, FMF, University of Ljubljana, Ljubljana, Slovenia
e-mail: cabello@fmf.uni-lj.si

S. Cabello
Department of Mathematics, IMFM, Ljubljana, Slovenia

S. Har-Peled
Department of Computer Science, University of Illinois, Illinois, USA

1 Introduction

Let P be a set of n points in the plane, where each point $p \in P$ has a given weight $w_p > 0$. For any $P' \subseteq P$, let $w(P') = \sum_{p \in P'} w_p$ denote the sum of the weights over P'. We consider the following two geometric optimization problems:

- $\mathsf{max}(P, m)$. Here we are given a weighted point set P and a parameter m, where m is an integer constant with $m \geq 1$. The goal is to place m unit disks that maximize the sum of the weights of the covered points. With a slight abuse of notation, we also use $\mathsf{max}(P, m)$ to denote the value of an optimal solution, that is,

$$\mathsf{max}(P, m) = \max\{w(P \cap U) \mid U \text{ is the union of } m \text{ unit disks}\}.$$

- $\mathsf{min}(P, X)$. Here we are given a weighted point set P and a region X of constant complexity in the plane. The goal is to place a single unit disk with center in X that minimizes the sum of the weights of the covered points. Note that this problem is uninteresting if X is unbounded. We use $\mathsf{min}(P, X)$ as the value of an optimal solution, that is,

$$\mathsf{min}(P, X) = \min\{w(P \cap D) \mid D \text{ is a unit disk whose center is in } X\}.$$

The problems under consideration naturally arise in the context of locational analysis, namely when considering placement of facilities that have a fixed area of influence, such as antennas or sensors. $\mathsf{max}(P, m)$ models the problem of placing m such new facilities that maximize the number of covered clients, while $\mathsf{min}(P, X)$ models the placement of a single obnoxious facility. $\mathsf{min}(P, X)$ can also model the placement of a facility in an environment of obnoxious points.

Related Work and Other Variants Facility location has been studied extensively in many different variants and it goes far beyond the scope of our paper to review all the work in this area. We confine ourselves to discussing the work that is directly related to our variant of the problem. For a general overview of facility-location problems in the plane, we refer to the survey by Plastria [22].

The problem $\mathsf{max}(P, m)$ for $m = 1$ was introduced by Drezner [12]. Later Chazelle and Lee [9] gave an $O(n^2)$-time algorithm for this case. An approximation algorithm has also been given: Agarwal et al. [1] provided a Monte-Carlo $(1 - \varepsilon)$-approximation algorithm for $\mathsf{max}(P, 1)$ when P is an unweighted point set. If we replace each point $p \in P$ by a unit disk centered at p, then $\mathsf{max}(P, 1)$ is equivalent to finding a point of maximum depth in the arrangement of disks. This implies that the results of Aronov and Har-Peled [4] give a Monte-Carlo $(1 - \varepsilon)$-approximation algorithm for unweighted point sets that runs in $O(n\varepsilon^{-2} \log n)$ time. Both the running time and the approximation factor hold with high probability. The algorithm, although it is described for the unweighted case, can be extended to the weighted case, giving an algorithm that uses $O(n\varepsilon^{-2} \log^2 n)$ time.

Somewhat surprisingly, the problem $\mathsf{max}(P, m)$ seems to have not been studied so far for $m > 1$. For $m = 2$, however, Cabello et al. [6] have shown how to solve a variant of the problem where the two disks are required to be disjoint. (This condition changes the problem significantly, because now issues related to packing problems arise.) Their algorithm runs in $O(n^{8/3}\log^2 n)$ time.

The problem $\mathsf{min}(P, X)$ was first studied by Drezner and Wesolowsky [13], who gave an $O(n^2)$-time algorithm. Note that if as before we replace each point by a unit disk, the problem $\mathsf{min}(P, X)$ boils down to finding a point with minimum depth in an arrangement of disks restricted to X. This means that for unweighted point sets, we can again apply the result of Aronov and Har-Peled [4] to get, with high probability, a $(1+\varepsilon)$-approximation for $\mathsf{min}(P, X)$ in $O(n\varepsilon^{-2}\log n)$ expected time. For technical reasons, however, this algorithm cannot be trivially modified to handle weighted points.

The extension of $\mathsf{min}(P, X)$ to the problem of placing m unit disks, without extra requirements, would have a solution consisting of m copies of the same disk. Hence, we restrict our attention to the case $m = 1$. (Following the paper by Cabello et al. [6] mentioned above one could study this problem under the condition that the disks be disjoint, but in the current paper we are interested in possibly overlapping disks.)

When m is considered as part of the input, the $\mathsf{max}(P, m)$ problem is related to the problem of minimizing the number of unit disks that are needed to cover a given point set. This latter problem is known to be NP-hard [16], and therefore $\mathsf{max}(P, m)$ is NP-hard when m is taken as part of the input.

There are also papers studying these problems for other shapes than unit disks. The problem $\mathsf{min}(P, X)$ for unit squares—this problem was first considered by Drezner and Wesolowsky [13]—turns out to be significantly easier than for disks and one can get subquadratic exact algorithms: Katz et al. [17] gave an optimal $O(n\log n)$ algorithm that computes the exact optimum. For disks this does not seem to be possible: Aronov and Har-Peled [4] showed that for disks $\mathsf{min}(P, X)$ and also $\mathsf{max}(P, 1)$ are 3SUM-HARD [14], that is, these problems belong to a class of problems for which no subquadratic algorithm is known. (For some problems from this class, an $\Omega(n^2)$ lower bound has been proved in a restricted model of computation.) The problem $\mathsf{max}(P, 1)$ has also been studied for other shapes [1]. We will limit our discussion to disks from now on. Our algorithms can be trivially modified to handle squares, instead of disks, or other fixed shapes of constant description.

Our Results As discussed above, $\mathsf{max}(P, m)$ is already 3SUM-HARD for $m = 1$ and also $\mathsf{min}(P, X)$ is 3SUM-HARD. Since we are interested in algorithms with near-linear running time we therefore focus on approximation algorithms. For $\mathsf{max}(P, m)$ we aim to achieve $(1-\varepsilon)$-approximation algorithms; given a parameter $\varepsilon > 0$, such algorithms compute a set of m disks such that the total weight of all points in their union is at least $(1-\varepsilon)\,\mathsf{max}(P, m)$. Similarly, for $\mathsf{min}(P, X)$ we aim for $(1+\varepsilon)$-approximation algorithms, that is, an algorithm that finds a disk centered in X and covering a total weight of at most $(1+\varepsilon)\,\mathsf{min}(P, X)$. When stating our bounds we consider $m \geq 1$ to be a constant and assume a model of computation where the floor function takes constant time.

For $\mathsf{max}(P, m)$ we give deterministic $(1-\varepsilon)$-approximation algorithms that run in $O(n\log n + n\varepsilon^{-4m+4}\log^{2m-1}(1/\varepsilon))$ time if $m \geq 4$, in $O(n\log n + n\varepsilon^{-6m+6}\log(1/\varepsilon))$ time if $m = 2, 3$, and in $O(n\log n + n\varepsilon^{-3})$ time if $m = 1$. The different cases depending on m arise because of using specialized subroutines to compute so-called

$(1/r)$-approximations. However, the core of the algorithm is the same in all cases. With a slight modification in the subroutine, we also give a randomized version that, with high probability, gives a $(1-\varepsilon)$-approximation in $O(n\varepsilon^{-2}\log n)$ time if $m=1$, and $O(n\varepsilon^{-4m+4}\log^{2m-1} n)$ time if $m=2,3$. As a subroutine for our approximation algorithms, we show how to solve $\mathsf{max}(P,m)$ exactly in $O(n^{2m-1}\log n)$ time. For large values of m, we also give an exact algorithm that uses $n^{O(\sqrt{m})}$ time.

These are the first $(1-\varepsilon)$-approximation algorithms for $\mathsf{max}(P,m)$ that use near-linear time for $m>1$. For $m=1$, only randomized algorithms using near-linear time were known previously. For this case, $m=1$, our randomized algorithm improves previous results, while our deterministic algorithm is incomparable: we have larger dependency on ε but smaller on n.

For $\mathsf{min}(P,X)$ we give a randomized algorithm that runs in $O(n(\log^2 n + \varepsilon^{-2}\log n))$ expected time and gives a $(1+\varepsilon)$-approximation with high probability. This is the first near-linear time approximation algorithm for this problem that can handle weighted points.

2 Notation and Preliminaries

Grids It will be convenient to define a *unit disk* as a closed disk of radius 1. Let $s:=\sqrt{2}$, so that a square of side s can be covered by a unit disk, and let $\Delta = 3ms$. (Recall that m is the number of disks we want to place.) We assume without loss of generality that no coordinate of the points in P is a multiple of s. For a pair $(a,b)\in\{0,1,\dots,3m-1\}^2$, we use $G_{(a,b)}$ to denote the grid of spacing Δ such that (as,bs) is one of the grid vertices, and we define $G:=G_{(0,0)}$. We consider the cells of a grid to be open sets. Finally, we let $L_{(a,b)}$ denote the set of grid lines that define $G_{(a,b)}$. Thus $L_{(a,b)}$ is given by

$$\big\{(x,y)\in\mathbb{R}^2 \mid y = bs + k\cdot\Delta \text{ and } k\in\mathbb{Z}\big\} \cup \big\{(x,y)\in\mathbb{R}^2 \mid x = as + k\cdot\Delta \text{ and } k\in\mathbb{Z}\big\}.$$

The following lemma follows from an easy counting argument.

Lemma 1 *Let $U := D_1\cup\cdots\cup D_m$ be the union of m unit disks. There is some $(a,b)\in\{0,1,\dots,3m-1\}^2$ such that $L_{(a,b)}$ does not intersect U, which means that each disk D_i is fully contained in a cell of $G_{(a,b)}$.*

Proof For $a\in\{0,1,\dots,3m-1\}$, let L_a be the vertical lines of $L_{(a,\cdot)}$. Each D_i is intersected by L_a for at most 2 values of $a\in\{0,1,\dots,3m-1\}$. It follows that at most $2m$ of the $3m$ possible values for a intersect $U=D_1\cup\cdots\cup D_m$, and therefore there is some $a_0\in\{0,1,\dots,3m-1\}$ such that L_{a_0} does not intersect U. The same applies to the horizontal lines: there is some $b_0\in\{0,1,\dots,3m-1\}$ such that the horizontal lines of $L_{(\cdot,b_0)}$ do not intersect U. Then, no line of $L_{(a_0,b_0)}$ intersects U. □

Samples and High Probability Throughout the paper we use the expression *with high probability*, or *whp* for short, to indicate that, for any given constant $c>0$, the failure probability can be bounded by n^{-c}. (In our algorithms, the value c affects the constant factor in the O-notation expressing the running time.)

An *integer-weighted* point set Q is a weighted point set with integer weights. We can see Q as a multiset where each point is repeated as many times as its weight. We use P for arbitrary weighted point sets and Q for integer-weighted point sets. A *ρ-sample* R of Q, for some $0 \le \rho \le 1$ is obtained by adding each point of the multiset Q to R with probability ρ, independently. A ρ-sample of an integer-weighted point set is also an integer-weighted point set. If R is a ρ-sample of Q and $\rho \cdot w(Q) \ge c \log n$, for an appropriate constant c, then it follows from Chernoff bounds that R has $\Theta(\rho \cdot w(Q))$ points whp.

3 Approximation Algorithms for $\max(P, m)$

Our algorithm uses $(1/r)$-approximations [7, 8, 21]. In our application they can be defined as follows. Let $\mathcal{U}$ be the collection of sets $U \subset \mathbb{R}^2$ that are the union of m unit disks, and let P be a weighted point set. Let A be a subset of P, possibly with different weights in the points. We say that A is a $(1/r)$-*approximation* for P if for any $U \in \mathcal{U}$ we have: $|w(U \cap A) - w(U \cap P)| \le w(P)/r$. There are different constructions of $(1/r)$-approximations. For our application, the following lemma will give the best results for $m \ge 4$. Other constructions that are more suitable when $m \le 3$ will be considered in the Appendix.

Lemma 2 *Let P be a weighted point set with n points and let $1 \le r \le n$ be a real parameter. We can construct in $O(n r^{12} \log^6 r)$ time a $(1/r)$-approximation A for P consisting of $O(r^2 \log r)$ points.*

Proof We assume that the reader is familiar with $(1/r)$-approximations for general range spaces [7, 8, 21]. Let $\mathcal{V}$ be the infinite set of cells that can arise in a vertical decomposition [15] of any collection of unit circles in the plane. The shatter function of the range space $(P, \mathcal{V})$ has exponent 6, that is, the set $\{P \cap c \mid c \in \mathcal{V}\}$ consists of at most $O(n^6)$ subsets of P. To see this, consider a set $P \cap c$, where c is a cell arising in the vertical decomposition of some collection of unit disks. Then $P \cap c$ is characterized by at most six points from P, namely the leftmost point in $P \cap c$, the rightmost point in $P \cap c$, one or two points for the top boundary, and one or two points for the bottom boundary. (The points for the top boundary, for example, are the points that can be reached when the disk bounding c from above is translated and or rotated down.)

Let $r' = v \cdot r$, where $v = v(m)$ is the maximum number of cells that the vertical decomposition of m unit circles can have. Since m is a fixed constant, $r' = O(r)$. Consider a $(1/r')$-approximation A for P with respect to the ranges $\mathcal{V}$. Then A is a $(1/r)$-approximation for P: for any $U \in \mathcal{U}$, if $\mathcal{V}(U)$ denotes the vertical decomposition given by the unit disks that define U, we have

$$\begin{aligned} |w(U \cap P) - w(U \cap A)| &= \left| \sum_{c \in \mathcal{V}(U),\, c \subseteq U} w(c \cap P) - w(c \cap A) \right| \\ &\le \sum_{c \in \mathcal{V}(U),\, c \subseteq U} |w(c \cap P) - w(c \cap A)| \\ &\le v \cdot \frac{w(P)}{r'} = \frac{w(P)}{r}. \end{aligned}$$

When P is an unweighted point set, we can use known algorithms [7, 8] to find a $(1/r')$-approximation A for P with respect to $\mathcal{V}$ consisting of $O((r')^2 \log r') = O(r^2 \log r)$ points in $O(n((r')^2 \log r')^6) = O(n r^{12} \log^6 r)$ time. Moreover, Matoušek [21] shows how an algorithm for finding $(1/r)$-approximations for unweighted point sets can be used as a subroutine to find $(1/r)$-approximation for weighted point sets, without affecting the asymptotic running time or the number of points in the approximation. $\square$

At first sight it may seem that this solves our problem: compute a $(1/r)$-approximation for $r = 1/\varepsilon$, and solve the problem for the resulting set of $O(\varepsilon^{-2} \log(1/\varepsilon))$ points. Unfortunately, this is not true: the error in the approximation is $w(P)/r$, not $w(U \cap P)/r$. Hence, when $w(P)$ is significantly larger than $w(U \cap P)$ we do not get a good approximation. Indeed, to obtain a good approximation we need to choose $r = w(P)/(\varepsilon \cdot \mathsf{max}(P, m))$. But now r may become quite large—in fact $\Theta(n)$ in the worst case—and it seems we do not gain anything. Nevertheless, this is the route we take. The crucial fact is that, even though the size of the approximation may be $\Theta(n)$, we can still gain something: we can ensure that any cell of $G = G_{(0,0)}$ contains only a few points. This will allow us to compute the optimal solution within a cell quickly. By combining this with a dynamic-programming approach and using several shifted grids, we can then obtain our result. We start with a lemma guaranteeing the existence of an approximation with few points per grid cell.

Lemma 3 *Let $0 < \varepsilon < 1$ be a parameter and let P be a set with n weighted points. Let $r := w(P)/(\varepsilon \mathsf{max}(P, m))$; note that the value of r is not known. We can find in $O(n \log n + n \varepsilon^{-12} \log^6(1/\varepsilon))$ time a $(1/2r)$-approximation A for P such that any cell of G contains $O(\varepsilon^{-2} \log(1/\varepsilon))$ points from A.*

Proof Let $\mathcal{C}$ be the collection of cells from G that contain some point of P. For a cell $C \in \mathcal{C}$, define $P_C := P \cap C$. Set $r' := 72m^2/\varepsilon$. For each cell $C \in \mathcal{C}$, compute a $(1/r')$-approximation A_C for P_C. We next show that the set $A := \bigcup_{C \in \mathcal{C}} A_C$ is a $(1/2r)$-approximation for P with the desired properties.

For any cell C we have $w(P_C) \le 9m \cdot \mathsf{max}(P, m)$ because C can be decomposed into $9m$ rectangles of size $s \times ms$, and for each of these rectangles R we have $w(R \cap P) \le \mathsf{max}(P, m)$. Since A_C is a $(1/r')$-approximation for P_C, we therefore have for any $U \in \mathcal{U}$,

$$\left| w(U \cap A_C) - w(U \cap P_C) \right| \le \frac{w(P_C)}{r'} \le \frac{9m \cdot \mathsf{max}(P, m)}{72m^2/\varepsilon} = \frac{\varepsilon}{8m} \cdot \mathsf{max}(P, m).$$

A unit disk of $U \in \mathcal{U}$ can intersect at most 4 cells of G, and therefore the total number of cells intersected by any $U \in \mathcal{U}$ is at most $4m$. If $\mathcal{C}_U$ denotes the cells of G intersected by U, we have $|\mathcal{C}_U| \le 4m$, so

$$\begin{aligned} |w(U \cap A) - w(U \cap P)| &= \left| \sum_{C \in \mathcal{C}_U} (w(U \cap A_C) - w(U \cap P_C)) \right| \\ &\le \sum_{C \in \mathcal{C}_U} |w(U \cap A_C) - w(U \cap P_C)| \end{aligned}$$

$$\leq \sum_{C \in \mathcal{C}_U} \frac{\varepsilon}{8m} \cdot \mathsf{max}(P,m) \leq (\varepsilon/2) \cdot \mathsf{max}(P,m)$$
$$= w(P)/2r.$$

We conclude that A is indeed a $(1/2r)$-approximation for P.

For constructing the set A, we can classify the points P by cells of G in $O(n \log n)$ time, and then for each non-empty cell C apply Lemma 2 to get a $(1/r')$-approximation A_C for P_C. Since m is a fixed constant, we have $r' = O(1/\varepsilon)$, and according to Lemma 2, A_C will contain $O((r')^2 \log(r')) = O(\varepsilon^{-2} \log(1/\varepsilon))$ points. Also, computing A_C takes $O(|P_C| \cdot (r')^{12} \log^6 r') = O(|P_C| \cdot \varepsilon^{-12} \log^6(1/\varepsilon))$ time, and adding the time over all cells $C \in \mathcal{C}$, we obtain the claimed running time. □

Choosing the value of r as in Lemma 3 indeed leads to a $(1-\varepsilon)$-approximation, which we show in Lemma 4 next.

Lemma 4 *Let* $0 < \varepsilon < 1$ *be a parameter and let* P *be a set with* n *weighted points. Let* A *be a* $(1/2r)$*-approximation for* P*, where* $r = w(P)/(\varepsilon \mathsf{max}(P,m))$*. If* U_A^* *is an optimal solution for* $\mathsf{max}(A,m)$*, then* $w(P \cap U_A^*) \geq (1-\varepsilon) \cdot \mathsf{max}(P,m)$.

Proof Let U^* be an optimal solution for P, that is, $w(U^* \cap P) = \mathsf{max}(P,m)$. Since U_A^* is optimal for A, we have $w(U_A^* \cap A) \geq w(U^* \cap A)$. On the other hand, since A is a $(1/2r)$-approximation for P, we have

$$|w(U_A^* \cap A) - w(U_A^* \cap P)| \leq (1/2r) \cdot w(P) = (\varepsilon/2) \cdot \mathsf{max}(P,m)$$

and

$$|w(U^* \cap A) - w(U^* \cap P)| \leq (\varepsilon/2) \cdot \mathsf{max}(P,m).$$

Therefore

$$\begin{aligned}
w(U_A^* \cap P) &= w(U_A^* \cap A) - w(U_A^* \cap A) + w(U_A^* \cap P) \\
&\geq w(U_A^* \cap A) - (\varepsilon/2) \cdot \mathsf{max}(P,m) \\
&\geq w(U^* \cap A) - (\varepsilon/2) \cdot \mathsf{max}(P,m) \\
&= w(U^* \cap P) - w(U^* \cap P) + w(U^* \cap A) - (\varepsilon/2) \cdot \mathsf{max}(P,m) \\
&\geq w(U^* \cap P) - (\varepsilon/2) \cdot \mathsf{max}(P,m) - (\varepsilon/2) \cdot \mathsf{max}(P,m) \\
&= (1-\varepsilon) \cdot \mathsf{max}(P,m).
\end{aligned}$$
□

It remains to find an optimal solution U_A^* for A. For a point set B, an integer m, and a cell C, define $\mathsf{max}(B,m,C)$ to be the maximum sum of weights of B that can be covered by placing m unit disks contained in C. Assume that we have an algorithm *Exact*(B,m,C)—later we will provide such an algorithm—that finds the exact value $\mathsf{max}(B,m,C)$ in $T(k,m)$ time for point sets B with k points. For technical reasons, we also assume that $T(k,m)$ has the following two properties: $T(k,j) \leq T(k,m)$ for $j \leq m$ and $T(k,m)$ is superlinear but polynomially bounded for any fixed m. The next lemma shows that we can then compute the optimal solution for A quickly, using a dynamic-programming approach.

Lemma 5 *Let A be a point set with at most n points such that each cell of G contains at most k points. We can find* $\mathsf{max}(A, m)$ *in* $O(n \log n + (n/k) \cdot T(k, m))$ *time.*

Proof For each $(a, b) \in \{0, 1, \dots, 3m-1\}^2$, let $\mathsf{max}_{(a,b)}(A, m)$ be the optimal weight we can cover with m unit disks that are disjoint from $L_{(a,b)}$. Lemma 1 implies that

$$\mathsf{max}(A, m) = \max_{(a,b)\in\{0,1,\dots,3m-1\}^2} \mathsf{max}_{(a,b)}(A, m).$$

We will show how to compute each $\mathsf{max}_{(a,b)}(A, m)$ in $O(n \log n + (n/k) \cdot T(k, m))$ time, which proves our statement because $m^2 = O(1)$. First we give the algorithm, and then discuss its time bound.

Consider a fixed $(a, b) \in \{0, 1, \dots, 3m-1\}^2$. Let $\mathcal{C} = \{C_1, \dots, C_t\}$ be the cells of $G_{(a,b)}$ that contain some point from P; we have $|\mathcal{C}| = t \le n$. For any cell $C_i \in \mathcal{C}$, define $A_i = A \cap C_i$.

For each cell $C_i \in \mathcal{C}$ and each $j \in \{1, \dots, m\}$, compute $\mathsf{max}(A_i, j, C_i)$ by calling the procedure *Exact*(A_i, j, C_i). From the values $\mathsf{max}(A_i, j, C_i)$ we can compute $\mathsf{max}_{(a,b)}(A, m)$ using dynamic programming across the cells of $\mathcal{C}$, as follows. Define $B_i = A_1 \cup \dots \cup A_i$. We want to compute $\mathsf{max}_{(a,b)}(B_i, j)$ for all i, j. To this end we note that an optimal solution $\mathsf{max}_{(a,b)}(B_i, j)$ will have ℓ disks inside A_i, for some $0 \le \ell \le j$, and the remaining $j - \ell$ disks spread among the cells $C_1, \dots, C_{i-1}$. This leads to the following recursive formula:

$$\mathsf{max}_{(a,b)}(B_i, j) = \begin{cases} \mathsf{max}(A_1, j, C_1) & \text{if } i = 1, \\ \max_{0\le\ell\le j}\{\mathsf{max}(A_i, \ell, C_i) + \mathsf{max}_{(a,b)}(B_{i-1}, j-\ell)\}, \\ \quad \text{otherwise.} \end{cases}$$

Since $\mathsf{max}_{(a,b)}(B_t, m) = \mathsf{max}_{(a,b)}(A, m)$, we end up computing the value $\mathsf{max}_{(a,b)}(A, m)$. This finishes the description of the algorithm.

The time used to compute $\mathsf{max}_{(a,b)}(A, m)$ can be bounded as follows. Firstly, observe that constructing A_i for all $C_i \in \mathcal{C}$ takes $O(n \log n)$ time. For computing the values $\mathsf{max}(A_i, j, C_i)$ for all i, j we need time

$$\sum_{C_i\in\mathcal{C}} \sum_{j=1}^{m} T(|A_i|, j) \le \sum_{C_i\in\mathcal{C}} m \cdot T(|A_i|, m) = O\Bigg(\sum_{C_i\in\mathcal{C}} T(|A_i|, m)\Bigg),$$

where the first inequality follows because for $j \le m$ we have $T(k, j) \le T(k, m)$ by assumption, and the second one follows since m is a constant. We have $|A_i| \le 4k$ for any $C_i \in \mathcal{C}$ because C_i intersects at most 4 cells of G. Moreover, because $T(k, m)$ is superlinear in k for fixed m, the sum is maximized when the points concentrate in as few sets A_i as possible. Therefore, the needed time can be bounded by

$$\begin{aligned} O\Bigg(\sum_{C_i\in\mathcal{C}} T(|A_i|, m)\Bigg) &\le O\Bigg(\sum_{i=1}^{\lceil n/4k\rceil} T(4k, m)\Bigg) \\ &= O((n/4k) \cdot T(4k, m)) = O((n/k) \cdot T(k, m)), \end{aligned}$$

where we have used that $T(4k,m) = O(T(k,m))$ because T is polynomially bounded by assumption. Once we have the values $\mathsf{max}(A_i, j, C_i)$ for all i, j, the dynamic programming requires computing $O(tm) = O(n)$ values $\mathsf{max}_{(a,b)}(B_i, j)$, and each element requires $O(m) = O(1)$ time. Therefore, the dynamic programming takes $O(n)$ time. We conclude that finding $\mathsf{max}_{(a,b)}(A, m)$ takes $O(n \log n + (n/k) \cdot T(k,m))$ time for any $(a,b) \in \{0, 1, \ldots, 3m-1\}^2$. □

Putting everything together, we obtain the following result.

Lemma 6 *For any weighted point set P with n points, we can find a set of m disks that cover a weight of at least $(1-\varepsilon)\mathsf{max}(P,m)$ in $O(n \log n + n\,\varepsilon^{-12} \log^6(1/\varepsilon) + (n/k) \cdot T(k,m))$ time, where $k = O(\varepsilon^{-2} \log(1/\varepsilon))$.*

Proof Given P and a parameter ε, consider the (unknown) value $r = \frac{w(P)}{\varepsilon \cdot \mathsf{max}(P,m)}$. We use Lemma 3 to compute a point set A with at most n points and such that A is a $(1/2r)$-approximation for P and any cell of G contains $O(\varepsilon^{-2} \log(1/\varepsilon))$ points.

We then use Lemma 5 to find an optimal solution U_A^* for $\mathsf{max}(A,m)$ in $O(n \log n + (n/k) \cdot T(k,m))$, where $k = O(\varepsilon^{-2} \log(1/\varepsilon))$. From Lemma 4, we know that $w(U_A^* \cap P) \geq (1-\varepsilon)\,\mathsf{max}(P,m)$, and the result follows. □

Theorem 9 below states there is an algorithm for the exact problem with $T(k,m) = O(k^{2m-1} \log k)$ for $m > 1$. For $m = 1$, we have $T(k,1) = O(k^2)$ because of the results of Chazelle and Lee [9] mentioned in the Related Work of Sect. 1. This bound $T(k,m)$ satisfies the two technical conditions mentioned above, so we can apply Lemma 6 to it. A simple calculation shows that we obtain a $(1-\varepsilon)$-approximation algorithm using $O(n \log n + n\,\varepsilon^{-12} \log^6(1/\varepsilon) + n\,\varepsilon^{-4m+4} \log^{2m-1}(1/\varepsilon))$ time. When $m \geq 4$, the second term is absorbed by the third term, and this is the best algorithm we will obtain for this case. This proves Theorem 7(i) below.

Considering different variants of Lemma 2, while leaving the rest of the algorithm basically unchanged, we can obtain results that have a better dependency on ε for $m = 1, 2, 3$. These results require no significant new ideas, and we have included them in the Appendix. Here, we only summarize the final results.

Theorem 7 *Given a parameter $0 < \varepsilon < 1$ and a weighted point set P with n points, we can find a set of m disks that cover a weight of at least $(1-\varepsilon)\mathsf{max}(P,m)$:*

(i) *in $O(n \log n + n\,\varepsilon^{-4m+4} \log^{2m-1}(1/\varepsilon))$ time if $m \geq 4$;*
(ii) *in $O(n \log n + n\varepsilon^{-6m+6} \log(1/\varepsilon))$ time if $m = 2, 3$;*
(iii) *in $O(n \log n + n\varepsilon^{-3})$ time if $m = 1$.*

Also, there is a randomized algorithm that, with high probability, returns a set of m disks that cover a weight of at least $(1-\varepsilon)\,\mathsf{max}(P,m)$

(iv) *in $O(n\varepsilon^{-4m+4} \log^{2m-1} n)$ time if $m = 2, 3$;*
(v) *in $O(n\varepsilon^{-2} \log n)$ time if $m = 1$.*

The term $O(n \log n)$ in the running times of cases (i)–(iii) comes from the classification of points according to cells. Using hashing, this can be done with $O(n)$

expected time [19, Chap. 13]. Therefore, we can remove the term $O(n \log n)$ in the running times, and obtain a bound on the expected running time.

3.1 Exact Algorithms for $\mathsf{max}(P, m, C)$

We want an algorithm that solves exactly the problem of placing m unit disks contained in a grid cell C such the sum of the weights of the covered points is maximized. We denote by $\mathsf{max}(P, m, C)$ the optimal value. This problem was used as a subroutine in Lemma 5. The only property of C that will be used is that C has bounded description complexity. Therefore, the same discussion provides exact algorithms for the problem $\mathsf{max}(P, m)$. Let X be the set of possible centers for a unit disk contained in C—the domain X is simply a square with the same center as C and of side length $\Delta - 2$ instead of Δ.

For a point $p \in P$, let D_p be the unit disk centered at p. The weight of D_p is w_p, the weight of p. Let $\mathcal{D}_P := \{D_p : p \in P\}$ be the set of all disks defined by P. For a point $q \in \mathbb{R}^2$ and a set $\mathcal{D}$ of weighted disks, we define depth$(q, \mathcal{D})$ to be the sum of the weights of the disks from $\mathcal{D}$ that contain q. Let $\mathcal{A}$ denote the arrangement induced by the disks from $\mathcal{D}_P$. For any point q inside a fixed cell c of $\mathcal{A}$, the function depth$(q, \mathcal{D}_P)$ is constant; we denote its value by depth$(c, \mathcal{D}_P)$. Because each disk D_p has the same size, the arrangement $\mathcal{A}$ can be constructed in $O(n^2)$ time [9]. Moreover, a simple traversal of $\mathcal{A}$ allows us to compute depth$(c, \mathcal{D}_P)$ for all cells $c \in \mathcal{A}$ in $O(n^2)$ time.

Let $V_{\mathcal{A}}$ be the set of vertices of $\mathcal{A}$, let V_X be the intersection points of the boundary of X with the boundary of some disk D_p, $p \in P$, and let V_{left} be the set of leftmost points from each disk D_p, $p \in P$. Finally, let $V = (V_{\mathcal{A}} \cup V_X \cup V_{\text{left}}) \cap X$. See Fig. 1, left. If $V = \emptyset$, then X is contained in some cell of $\mathcal{A}$ and the problem can trivially be solved. Otherwise we have

$$\mathsf{max}(P, m, C) = \max \{w(P \cap U) \mid U \text{ union of } m \text{ unit disks with centers in } V\},$$

that is, we only need to consider disks whose centers are in V. Based on this observation, we can solve $\mathsf{max}(P, m, C)$ for $m > 1$. We first consider the case $m = 2$.

Lemma 8 *We can compute* $\mathsf{max}(P, 2, C)$ *in* $O(n^3 \log n)$ *time.*

Proof Our approach is similar to the one used by Katz and Sharir [18]. Let $\mathcal{A}^*$ the arrangement induced by the set $\mathcal{D}_P$ of disks and the sets X and V. Let G be the plane graph obtained by considering the restriction of $\mathcal{A}^*$ to X: the vertices of G are the vertices of $\mathcal{A}^*$ contained in X and the edges of G are the edges of $\mathcal{A}^*$ fully contained in X—see Fig. 1, right. For simplicity, let us assume that each vertex in G has degree 4, meaning that no three points of P are on a unit circle and no intersection of the boundary of two disks lies on the boundary of X. This condition can be enforced considering the max problem for disks of radius $1 + \delta$, where $\delta > 0$ is a formal infinitesimal: since we consider closed unit disks in the problem, the quality of the solution is not affected by making the disks infinitesimally larger. Consider a spanning tree T of G and double each edge to obtain an Euler path π of T. The path π has $O(n^2)$ edges and it visits each vertex of V at least once and at most four times.

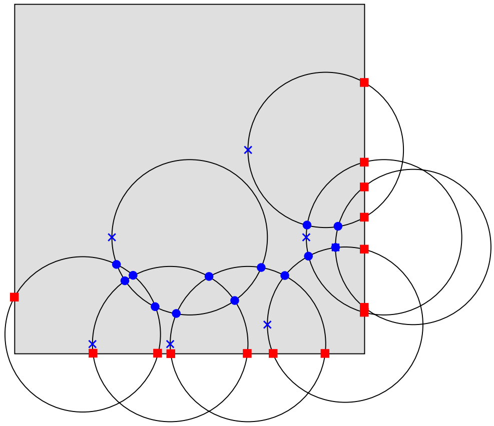

Fig. 1 *Left*: Example showing the points V. The *dots* indicate $V_{\mathcal{A}} \cap X$, the *squares* indicate V_X, and the *crosses* indicate $V_{\text{left}} \cap X$. *Right*: planar graph G with V as vertices and connected using portions of $\mathcal{A}$ or the boundary of X as edges

The algorithm is as follows. We want to find two vertices $q, v \in V$, such that $P \cap (D_q \cup D_v)$ has maximum weight. If we fix q and let $\overline{\mathcal{D}_P}(q) \subset \mathcal{D}_P$ denote the disks in $\mathcal{D}_P$ *not* containing q, then the best pair q, v (for this choice of q) covers a weight of $\text{depth}(q, \mathcal{D}_P) + \max_{v \in V} \text{depth}(v, \overline{\mathcal{D}_P}(q))$. So our approach is to walk along the tour π to visit all possible vertices $q \in V$, and maintain the set $\mathcal{D} := \overline{\mathcal{D}_P}(q)$—we call this the set of *active disks*—such that we can efficiently perform the following operations: (i) report a vertex $v \in V$ maximizing $\text{depth}(v, \mathcal{D})$, and (ii) insert or delete a disk into $\mathcal{D}$. Then we can proceed as follows. Consider two vertices q', q'' that are connected by an edge of π. The symmetric difference between the sets $\overline{\mathcal{D}_P}(q')$ and $\overline{\mathcal{D}_P}(q'')$ has at most two disks. So while we traverse π, stepping from a vertex q' to an adjacent one q'' along an edge of π, we can update $\mathcal{D}$ with at most two insertions/deletions, and then report a vertex $v \in V$ maximizing $\text{depth}(v, \mathcal{D})$. Next we show how to maintain $\mathcal{D}$ such that both operations—reporting and updating—can be performed in $O(n \log n)$ time. Since π has $O(n^2)$ vertices, the total time will then be $O(n^3 \log n)$, as claimed.

The main problem in maintaining the set of active disks $\mathcal{D}$ is that the insertion or deletion of a disk can change $\text{depth}(v, \mathcal{D})$ for $\Theta(n^2)$ vertices $v \in V$. Hence, to obtain $O(n \log n)$ update time, we cannot maintain all the depths explicitly. Instead we do this implicitly, as follows.

Let $\mathcal{T}$ be a balanced binary tree on the path π, where the leftmost leaf stores the first vertex of π, the next leaf the second vertex of π, and so on. Thus the tree $\mathcal{T}$ has $O(n^2)$ nodes. For an internal node ν we denote by $\mathcal{T}_\nu$ the subtree of $\mathcal{T}$ rooted at ν. Furthermore, we define $\pi(\nu)$ to be the subpath of π from the leftmost vertex in $\mathcal{T}_\nu$ to the rightmost vertex in $\mathcal{T}_\nu$. Note that if μ_1 and μ_2 are the children of ν, then $\pi(\nu)$ is the concatenation of $\pi(\mu_1)$ and $\pi(\mu_2)$. Also note that $\pi(\text{root}(\mathcal{T})) = \pi$. Finally, note that any subpath from π can be expressed as the concatenation of the subpaths $\pi(\nu_1), \pi(\nu_2), \ldots$ of $O(\log n)$ nodes—this is similar to the way a segment tree [11] works.

Now consider some disk $D_p \in \mathcal{D}_P$. Since D_p has $O(n)$ vertices from V on its boundary, the part of π inside D_p consists of $O(n)$ subpaths. Hence, there is a collection $N(D_p)$ of $O(n \log n)$ nodes in $\mathcal{T}$—we call this set the *canonical representation* of D_p—such that $\pi \cap D_p$ is the disjoint union of the set of paths $\{\pi(\nu) : \nu \in N(D_p)\}$. We store at each node ν of T the following two values:

- Cover(ν): the total weight of all disks $D_p \in \mathcal{D}$ (that is, all active disks) such that $\nu \in N(D_p)$.
- MaxDepth(ν): the value $\max\{\text{depth}(v, \mathcal{D}(\nu)) : v \in \pi(\nu)\}$, where $\mathcal{D}(\nu) \subset \mathcal{D}$ is the set of all active disks whose canonical representation contains a node μ in $\mathcal{T}_\nu$.

Notice that MaxDepth(root($\mathcal{T}$)) $= \max_{v \in V} \text{depth}(v, \mathcal{D})$, so MaxDepth(root($\mathcal{T}$)) is exactly the value we want to report. Hence, it remains to describe how to maintain the values Cover(ν) and MaxDepth(ν) when $\mathcal{D}$ is updated. Consider the insertion of a disk D_p into $\mathcal{D}$; deletions are handled similarly. First we find in $O(n \log n)$ time the set $N(D_p)$ of nodes in $\mathcal{T}$ that forms the canonical representation of D_p. The values Cover(ν) and MaxDepth(ν) are only influenced for nodes ν that are in $N(D_p)$, or that are an ancestor of such a node. More precisely, for $\nu \in N(D_p)$, we need to add the weight of D_p to Cover(ν) and to MaxDepth(ν). To update the values at the ancestors we use that, if μ_1 and μ_2 are the children of a node ν, then we have

$$\text{MaxDepth}(\nu) = \text{Cover}(\nu) + \max\bigl(\text{MaxDepth}(\mu_1), \text{MaxDepth}(\mu_2)\bigr).$$

This means we can update the values in a bottom-up fashion in $O(1)$ time per ancestor, so in $O(n \log n)$ time in total. This finishes the description of the data structure—see [17] or [5] for similar ideas, or how to reduce the space requirements. □

Theorem 9 *For any fixed $m > 1$, we can compute* $\mathsf{max}(P, m, C)$ *in $O(n^{2m-1} \log n)$ time.*

Proof For $m > 2$, fix any $m - 2$ vertices $v_1, \ldots, v_{m-2} \in V$, compute the point set $P' = P \setminus (D_{v_1} \cup \cdots \cup D_{v_{m-2}})$, and compute $\mathsf{max}(P', 2, C)$ in $O(n^3 \log n)$ time using Lemma 8. We obtain a placement of disks covering a weight of $w(P \setminus P') + \mathsf{max}(P', 2, C)$. This solution is optimal under the assumption that the first $m - 2$ disks are placed at $v_1, \ldots, v_{m-2}$. Iterating this procedure over the $O(|V|^{m-2}) = O(n^{2m-4})$ possible tuples of vertices $v_1, \ldots, v_{m-2}$, it is clear that we obtain the optimal solution for placing m disks with centers in X. The time we spend can be bounded as $O(n^{2m-4} n^3 \log n) = O(n^{2m-1} \log n)$. □

3.2 A Faster Algorithm for Large m

If m is large, we can adapt an algorithm of Agarwal and Procopiuc [2] to get a faster solution to our problem. The approach is based on the following observation: for any optimal solution, there exists a line of the integer grid (either horizontal or vertical) that stabs $O(\sqrt{m})$ of its disks. Indeed, consider the m unit disks used by an optimal solution U. All the unit disks that are stabbed by a single point are contained in a disk of radius 2, which could be covered by 7 unit disks. It follows that no point can stab more than seven disks of the optimal solution U. Then, two cases are distinguished

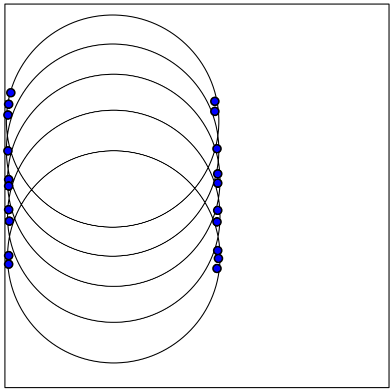

Fig. 2 Example showing that there are cases where some points pierce $\Theta(m)$ disks of any optimal solution for $\mathsf{max}(P, m, C)$

depending on whether U is contained in a vertical strip of width $\sqrt{m}$ or not. In the first case, one can argue that there is a horizontal line of the integer grid with the desired property, while in the second, one can argue for a vertical line. See [2] if more detail is needed.

We use dynamic programming to find the optimal solution, as follows. Consider all the lines of the integer grid that are at distance at most 4 from one of the points of P. There are $O(n)$ such lines. We start with a bounding rectangle that contains all the points of P. At each stage, we guess a cutting line ℓ that intersects at most $O(\sqrt{m})$ unit disks of the optimal solution, and also the $O(\sqrt{m})$ unit disks that ℓ cuts (there are $O(n^2)$ candidate unit disks overall). We also guess the number of disks in each side of ℓ in the optimal solution, and solve recursively in each side. In this recursive algorithm, a subproblem is composed of a rectangle (there are $O(n^4)$ such choices), a guess of $O(\sqrt{m})$ unit disks of the optimal solution intersecting the boundary of the rectangle (there are $n^{O(\sqrt{m})}$ such choices), and the number of disks fully contained in the rectangle (there are m such choices). Hence, the dynamic programming to compute the maximum number of covered points can be done in $n^{O(\sqrt{m})}$ time. We summarize.

Theorem 10 *For any fixed $m \geq 1$, we can compute* $\mathsf{max}(P, m)$ *in $n^{O(\sqrt{m})}$ time.*

This algorithm cannot be directly adapted to solve the problem $\mathsf{max}(P, m, C)$: it is no longer true that any point stabs at most a constant number of disks in an optimal solution; see Fig. 2. Reusing the ideas involved in Lemmas 1–6, one can give another algorithm that employs Theorem 10 as a subroutine. However, the constants involved in the exponent $O(\sqrt{m})$ of Theorem 10 make this approach uninteresting, unless m is large.

4 Approximation Algorithms for $\mathsf{min}(P, X)$

We now turn our attention to the problem $\mathsf{min}(P, X)$ where we wish to place a single disk D in X as to minimize the sum of the weights of the points in $P \cap D$. Our approach to obtain a $(1+\varepsilon)$-approximation consists of two stages. First, we make a binary search to find a value T that is a constant factor approximation for $\mathsf{min}(P, X)$. For this to work, we give a decision procedure that drives the search for the value T. Second, we use the constant factor approximation T to find a $(1+\varepsilon)$-approximation to $\mathsf{min}(P, X)$. Both stages of our algorithm are based on the same idea: convert the point set P to an integer-weighted point set Q by scaling and rounding appropriately, and then solve the problem for a random sample of Q. We first provide two technical lemmas to be used later on. Lemmas 11, 13 and 16 are simple adaptations of results by Aronov and Har-Peled [4]. The proofs are given here for the sake of clarity and self-containment.

Like with the $\mathsf{max}(P, m)$ problem, we also use a subroutine to exactly solve a slight modification of the $\mathsf{min}(P, X)$ problem. The following lemma describes the subroutine we will use. Here, like before, D_a denotes the unit disk centered at a point a.

Lemma 11 *Let Q be an integer-weighted point set with at most n points, let X be a domain of constant complexity, let A be a set of at most n points, and let $\kappa \geq 0$ be a real number. We can decide in $O(n\kappa + n\log n)$ expected time if $\mathsf{min}(Q, X \setminus (\bigcup_{a\in A} D_a)) \leq \kappa$ or $\mathsf{min}(Q, X \setminus (\bigcup_{a\in A} D_a)) > \kappa$. In the former case we can also find a unit disk D that is optimal for $\mathsf{min}(Q, X \setminus (\bigcup_{a\in A} D_a))$. The running time is randomized, but the result is always correct.*

Proof Let $\mathcal{A}$ be the arrangement induced by the $O(n)$ disks D_a, $a \in A$, and D_q, $q \in Q$, and let $\mathcal{A}_\kappa$ be the portion of $\mathcal{A}$ that has depth at most κ. The portion $\mathcal{A}_\kappa$ has complexity $O(n\kappa)$ [23] and it can be constructed using a randomized incremental construction, in $O(n\kappa + n\log n)$ expected time [10]. Then, we just discard all the cells of $\mathcal{A}_\kappa$ that are covered by any disk D_a with $a \in A$, and for the remaining cells we check if any has depth over κ and intersects X. Since X has constant complexity, in each cell we spend time proportional to its complexity, and the result follows. □

The following lemma bounds the error when scaling and rounding the weights.

Lemma 12 *Let P be a weighted point set with n points, let $S > 0$ be a real number, and let Q denote the integer-weighted point set obtained by picking each point from P and assigning to it weight $\lfloor w_p/S \rfloor$. Then, for any disk D we have*

$$w(D \cap P) - n \cdot S \leq S \cdot w(D \cap Q) \leq w(D \cap P).$$

Proof For any disk D we have

$$w(D \cap P) - n \cdot S \leq \sum_{p\in D\cap P} \left(w_p - S\right) \leq \sum_{p\in D\cap P} \left\lfloor w_p/S \right\rfloor \cdot S = w(D \cap Q) \cdot S$$

and

$$w(D \cap Q) \cdot S = \sum_{p\in D\cap P} \left\lfloor w_p/S \right\rfloor \cdot S \leq \sum_{p\in D\cap P} (w_p/S) \cdot S = w(D \cap P).$$

□

4.1 Finding a Constant Factor Approximation

Our algorithm uses the following combinatorial result for random sampling.

Lemma 13 *Let Q be an integer-weighted point set with at most n points, let Y be any domain, and let $\Delta_Q = \mathsf{min}(Q, Y)$. Given a value k, set $\rho = \min\{1, ck^{-1}\log n\}$, where $c > 0$ is an appropriate constant. If R is a ρ-sample of Q and $\Delta_R = \mathsf{min}(R, Y)$, then whp it holds*:

(i) *if $\Delta_Q \geq k/2$, then $\Delta_R \geq k\rho/4$;*
(ii) *if $\Delta_Q \leq 2k$, then $\Delta_R \leq 3k\rho$;*
(iii) *if $\Delta_Q \notin [k/8, 6k]$, then $\Delta_R \notin [k\rho/4, 3k\rho]$.*

Proof If $\rho = 1$, then the result is clearly true. The case $\rho < 1$ is handled considering each claim separately and using Chernoff bounds.

(i) Assume that $\Delta_Q \geq k/2$. Consider a fixed unit disk D centered in Y and the random variable $W = w(D \cap R)$. Since $w(D \cap Q) \geq \Delta_Q \geq k/2$, we have $\mu := \mathbb{E}[W] \geq k\rho/2$, and therefore $\mu - k\rho/4 \geq \mu/2$. Using Chernoff bounds we obtain

$$\begin{aligned}\Pr\left[W \leq \frac{k\rho}{4}\right] &= \Pr\left[-W \geq -\frac{k\rho}{4}\right] = \Pr\left[\mu - W \geq \mu - \frac{k\rho}{4}\right]\\ &\leq \Pr\left[|W-\mu| \geq \frac{\mu}{2}\right] \leq e^{-\Omega(\mu(\frac{1}{2})^2)}\\ &= e^{-\Omega(\mu)} = e^{-\Omega(k\rho)} \leq e^{-\Omega(c\log n)} \leq n^{-\Omega(c)}.\end{aligned}$$

We conclude that whp $w(D \cap R) \geq k\rho/4$. This only holds for the fixed disk D. However, since there are at most $O(n^2)$ combinatorially different unit disks, that is, $\{Q \cap D \mid D \text{ unit disk}\}$ has at most $O(n^2)$ elements, it follows from the union bound that whp $w(D \cap R) \geq k\rho/4$ for any disk D centered in Y. Therefore, whp $\Delta_R \geq k\rho/4$.

(ii) Assume that $\Delta_Q \leq 2k$. Let D^* be a unit disk centered in Y such that $w(D^* \cap Q) = \Delta_Q \leq 2k$. Consider the random variable $W = w(D^* \cap R)$. We have $\mu := \mathbb{E}[W] \leq 2k\rho$, and therefore $k\rho/\mu \geq 1/2$. Using Chernoff bounds, we obtain

$$\begin{aligned}\Pr[W \geq 3k\rho] &= \Pr[W - 2k\rho \geq k\rho] \leq \Pr[W - \mu \geq k\rho]\\ &\leq \Pr\left[W - \mu \geq \frac{k\rho}{\mu}\cdot\mu\right] \leq e^{-\Omega(\mu(\frac{k\rho}{\mu})^2)}\\ &\leq e^{-\Omega(\frac{(k\rho)^2}{\mu})} \leq e^{-\Omega(k\rho)} \leq e^{-\Omega(c\log n)} \leq n^{-\Omega(c)}.\end{aligned}$$

We conclude that whp $w(D^* \cap R) \leq 3k\rho$, and therefore $\Delta_R \leq 3k\rho$.

(iii) This is done exactly in the same way as (i) and (ii). □

The idea used in the algorithm for the decision problem is to consider heavy and light points separately. The heavy points have to be avoided, while the light ones can be approximated by a set of n *integer-weighted* points with similar weights. Then we can take a random sample of appropriate size and use Lemma 13 to solve the decision problem. This decision procedure is then used as a subroutine in Lemma 15 to find a constant-factor approximation for $\mathsf{min}(P, X)$.

Lemma 14 *Let X be a domain with constant complexity. Given a weighted point set P with n points and a value T, we can return in $O(n \log n)$ expected time whether* (i) $\mathsf{min}(P, X) < T$, *or* (ii) $\mathsf{min}(P, X) > 2T$, *or* (iii) $\mathsf{min}(P, X) \in (T/10, 10T)$, *where the returned answer is correct whp.*

Proof First we describe the algorithm then show its correctness and finally discuss its running time.

Algorithm We compute the sets $A = \{p \in P \mid w_p > 2T\}$ and $P' = P \setminus A$, as well as the domain $Y = X \setminus \bigcup_{a\in A} D_a$. If $Y = \emptyset$, then we can report $\mathsf{min}(P, X) > 2T$, since any disk with center in X covers some point with weight at least $2T$. If $Y \neq \emptyset$, we construct the integer-weighted point set Q obtained by picking each point from P' and assigning to it weight $\lfloor 2nw_p/T \rfloor$. Set $k = 2n$, and $\rho = \min\{1, ck^{-1} \log n\}$, where c is an appropriate constant. We construct a ρ-sample R of Q, and decide as follows: If $\mathsf{min}(R, Y) < k\rho/4$ then return $\mathsf{min}(P, X) < T$; if $\mathsf{min}(R, Y) > 3k\rho$ then return $\mathsf{min}(P, X) > 2T$; otherwise return $\mathsf{min}(P, X) \in (T/10, 10T)$.

Correctness We have to show that, whp, the algorithm gives a correct answer. Note that $\mathsf{min}(P, X) \le 2T$ if and only if $\mathsf{min}(P, Y) \le 2T$, and in that case we have $\mathsf{min}(P, X) = \mathsf{min}(P, Y)$. Therefore, we only need concentrate our attention to $\mathsf{min}(P, Y)$. Define $\Delta_Q = \mathsf{min}(Q, Y)$ and $\Delta_P = \mathsf{min}(P, Y)$. For any unit disk D centered in Y we have $w(D \cap P) = w(D \cap P')$, and therefore

$$w(D \cap P) - T/2 \le (T/2n) \cdot w(D \cap Q) \le w(D \cap P)$$

because of Lemma 12. This implies that

$$\Delta_P - T/2 \le \frac{T}{2n} \cdot \Delta_Q \le \Delta_P. \tag{1}$$

The value $\Delta_R = \mathsf{min}(R, Y)$ provides us information as follows:

- If $\Delta_R < k\rho/4$, then $\Delta_Q < k/2 = n$ whp because of Lemma 13(i), and using (1) we obtain that
$$\Delta_P \le \frac{T}{2n} \cdot \Delta_Q + \frac{T}{2} < \frac{T}{2n} \cdot n + \frac{T}{2} = T.$$
- If $\Delta_R > 3k\rho$, then $\Delta_Q > 2k = 4n$ whp because of Lemma 13(ii), and using (1) we obtain that
$$\Delta_P \ge \frac{T}{2n} \cdot \Delta_Q > \frac{T}{2n} \cdot 4n = 2T.$$
- If $\Delta_R \in [k\rho/4, 3k\rho]$, then $\Delta_Q \in [k/8, 6k] = [n/4, 12n]$ whp by Lemma 13(iii). Using (1) we obtain that whp
$$\Delta_P \le \frac{T}{2n} \cdot \Delta_Q + T/2 < \frac{T}{2n} \cdot 12n + T/2 < 10T$$

and

$$\Delta_P \geq \frac{T}{2n} \cdot \Delta_Q \geq \frac{T}{2n} \cdot \frac{n}{4} > \frac{T}{10}.$$

It follows that the algorithm gives the correct answer whp.

Running time We can compute A, P', Q, R in linear time, and check if $Y = \emptyset$ in $O(n \log n)$ expected time by constructing $\bigcup_{a\in A} D_a$ explicitly using a randomized incremental construction. Note that $k\rho = O(\log n)$ and R consists of at most n points. Because $Y = X \setminus \bigcup_{a\in A} D_a$, we can use Lemma 11 to find if $\Delta_R = \mathsf{min}(R, Y) > 3k\rho$ or otherwise compute Δ_R exactly, in $O(|R|\log|R| + |R|k\rho) = O(n \log n)$ expected time. □

Lemma 15 *Let X be a domain with constant complexity. Given a weighted point set P with n points, we can find in $O(n \log^2 n)$ expected time a value T that, whp, satisfies $T/10 < \mathsf{min}(P, X) < 10T$.*

Proof The idea is to make a binary search. For this, we will use Lemma 14 for certain values T. Note that, if at any stage, Lemma 14 returns that $\mathsf{min}(P, X) \in (T/10, 10T)$, then we have found our desired value T, and we can finish the search. In total, we will make $O(\log n)$ calls to the procedure of Lemma 14, and therefore we obtain the claimed expected running time. Also, the result is correct whp because we make $O(\log n)$ calls to procedures that are correct whp.

Define the interval $I_p = [w_p, (n + 1) \cdot w_p)$ for any point $p \in P$, and let $I = \bigcup_{p\in P} I_p$. It is clear that $\mathsf{min}(P, X) \in I$, since the weight of the heaviest point covered by an optimal solution can appear at most n times in the solution. Consider the values $B = \{w_p, (n + 1) \cdot w_p \mid p \in P\}$, and assume that $B = \{b_1, \dots, b_{2n}\}$ is sorted increasingly. Note that for the (unique) index i such that $\mathsf{min}(P, X) \in [b_i, b_{i+1})$, it must hold that $b_{i+1} \leq (n + 1)b_i$

We first perform a binary search to find two consecutive elements b_i, b_{i+1} such that $\mathsf{min}(P, X) \in [b_i, b_{i+1})$. Start with $\ell = 1$ and $r = 2n$. While $r \neq \ell + 1$, set $m = \lfloor(\ell + r)/2\rfloor$ and use Lemma 14 with $T = b_m$:

- If $\mathsf{min}(P, X) < T$, then set $r = m$.
- If $\mathsf{min}(P, X) > 2T$, then set $\ell = m$.
- If $T/10 < \mathsf{min}(P, X) < 10T$, then we just return T as the desired value.

Note that during the search we maintain the invariant $\mathsf{min}(P, X) \in [b_\ell, b_r)$. Since we end up with two consecutive indices $\ell = i, r = i + 1$, it must hold that $\mathsf{min}(P, X) \in [b_i, b_{i+1})$.

Next, we perform another binary search in the interval $[b_i, b_{i+1})$ as follows. Start with $\ell = b_i$ and $r = b_{i+1}$. While $r/\ell > 10$, set $m = (\ell + r)/2$ and call the procedure of Lemma 14 with $T = m$:

- If $\mathsf{min}(P, X) < T$, then set $r = m$.
- If $\mathsf{min}(P, X) > 2T$, then set $\ell = m$.
- If $T/10 < \mathsf{min}(P, X) < 10T$, then we just return T as the desired value.

Since $b_{i+1} \leq (n+1)b_i$, it takes $O(\log n)$ iterations to ensure that $r/\ell \leq 10$. During the search we maintain the invariant that $\min(P, X) \in [\ell, r)$, and therefore we can return the last value ℓ as satisfying $\min(P, X) \in (\ell/10, 10\ell)$. □

4.2 Finer Sampling and $(1+\varepsilon)$-Approximation

Assuming that we have a constant factor approximation to the value $\min(P, X)$, we will provide an algorithm that gives a $(1+\varepsilon)$-approximation. First, we provide a combinatorial lemma that resembles Lemma 13 but takes the parameter ε into account.

Lemma 16 *Let Q be an integer-weighted point set with at most n points, let Y be any domain, and let $0 < \varepsilon < 1$ be a parameter. Suppose we are given a value k such that $\min(Q, Y) = \Omega(k)$, and set $\rho = \min\{1, ck^{-1}\varepsilon^{-2}\log n\}$, where $c > 0$ is an appropriate constant. If R is a ρ-sample of Q and D_R is an optimal unit disk for $\min(R, Y)$, then $w(D_R \cap Q) \leq (1+\varepsilon/2)\min(Q, Y)$ whp.*

Proof If $\rho = 1$, then $R = Q$ and the claim is evident. Otherwise, let $\Delta_Q = \min(Q, Y)$ and $\Delta_R = \min(R, Y)$. Consider the value $Z = (1+\varepsilon/4)\rho\,\Delta_Q$. We have the following two properties:

- Whp, $\Delta_R < Z$. Indeed, if we consider a disk D^* centered at Y such that $w(D^* \cap Q) = \Delta_Q$, we can apply Chernoff bounds to the random variable $W = w(D^* \cap R)$ to obtain

$$\begin{aligned}\Pr\big[w(D^* \cap R) \geq Z\big] &= \Pr\left[W \geq \left(1+\frac{\varepsilon}{4}\right)\rho\,\Delta_Q\right] = \Pr\left[W \geq \left(1+\frac{\varepsilon}{4}\right)\mathbb{E}[W]\right]\\ &\leq e^{-\Omega(\mathbb{E}[W](\frac{\varepsilon}{4})^2)} \leq e^{-\Omega(\rho\,\Delta_Q\,\varepsilon^2)}\\ &\leq e^{-\Omega(ck^{-1}\Delta_Q\,\log n)} \leq n^{-\Omega(c)},\end{aligned}$$

  where we have used that $\Delta_Q/k = \Omega(1)$.
- Whp, for all unit disks D with $w(D \cap Q) > (1+\varepsilon/2)\Delta_Q$ we have $w(D \cap R) \geq Z$. Indeed, consider any such disk D and the related random variable $W = w(D \cap R)$. Note that $\mathbb{E}[W] \geq (1+\varepsilon/2)\rho\,\Delta_Q$. Using that $(1+\varepsilon/4) \leq (1-\varepsilon/6)(1+\varepsilon/2)$ for any $\varepsilon \in (0, 1)$, we have

$$\begin{aligned}\Pr\big[w(D \cap R) < Z\big] &= \Pr\left[W < \left(1+\frac{\varepsilon}{4}\right)\rho\,\Delta_Q\right]\\ &\leq \Pr\left[W < \left(1-\frac{\varepsilon}{6}\right)\left(1+\frac{\varepsilon}{2}\right)\rho\,\Delta_Q\right]\\ &\leq \Pr\left[W < \left(1-\frac{\varepsilon}{6}\right)\mathbb{E}[W]\right] \leq e^{-\Omega(\mathbb{E}[W](\frac{\varepsilon}{6})^2)}\\ &\leq e^{-\Omega(\rho\Delta_Q\varepsilon^2)} \leq e^{-\Omega(ck^{-1}\Delta_Q\log n)} \leq n^{-\Omega(c)},\end{aligned}$$

  where we have used that $\Delta_Q/k = \Omega(1)$. Since there are at most $O(n^2)$ combinatorially different unit disks, the claim follows from the union bound.

The first item implies that, whp, an optimal unit disk D_R satisfies $w(D_R \cap R) = \Delta_R < Z$. But $w(D_R \cap R) = \Delta_R < Z$ implies that, whp, $w(D_R \cap Q) < (1+\varepsilon/2)\Delta_Q$ because of the second item. $\square$

Lemma 17 *Let $0 < \varepsilon < 1$ be a parameter, let P be a weighted point set with n points, and let T be a given value such that $T/10 < \mathsf{min}(P, X) < 10T$. We can find in $O(n\varepsilon^{-2}\log n)$ expected time a unit disk D that, whp, satisfies $w(D \cap P) \le (1+\varepsilon)\mathsf{min}(P, X)$.*

Proof First we describe the algorithm, then show its correctness, and finally discuss its running time. The ideas are similar to the ones used in Lemma 14. However, now we also need to take into account the parameter ε.

Algorithm We compute the sets $A = \{p \in P \mid w_p > 10T\}$ and $P' = P \setminus A$, as well as the domain $Y = X \setminus \bigcup_{a\in A} D_a$. We construct an integer-weighted point set Q by picking each point from P' and assigning to it weight $\lfloor 20nw_p/\varepsilon T\rfloor$. Define $k = \lfloor 20n/\varepsilon\rfloor$, and let $\rho = \min\{1, ck^{-1}\varepsilon^{-2}\log n\}$, where c is the constant used in Lemma 16. Finally, compute a ρ-sample R of Q, find a best disk D_R for $\mathsf{min}(R, Y)$, and report the disk D_R as a solution.

Correctness Since $\mathsf{min}(P, X) < 10T$ by hypothesis, we know that $\mathsf{min}(P, X) = \mathsf{min}(P, Y) = \mathsf{min}(P', Y)$, because any disk with center in $\bigcup_{a\in A} D_a$ covers some point of A. We therefore concentrate on the value $\mathsf{min}(P, Y)$. Let $\Delta_Q = \mathsf{min}(Q, Y)$ and $\Delta_P = \mathsf{min}(P, Y)$. We have to show that whp the disk D_R returned by the algorithm satisfies $w(D_R \cap P) \le (1+\varepsilon)\mathsf{min}(P, X) = (1+\varepsilon)\Delta_P$. For any unit disk D centered in Y, we have $w(D \cap P) = w(D \cap P')$, and Lemma 12 implies

$$w(D\cap P) - \frac{\varepsilon T}{20} \le \frac{\varepsilon T}{20n}\cdot w(D\cap Q) \le w(D\cap P). \tag{2}$$

Therefore, if D ranges among all unit disks centered in Y, we have

$$\min_D\left\{w(D\cap P) - \frac{\varepsilon T}{20}\right\} \le \min_D\left\{\frac{\varepsilon T}{20n}\cdot w(D\cap Q)\right\} \le \min_D\left\{w(D\cap P)\right\}$$

and thus

$$\Delta_P - \frac{\varepsilon T}{20} \le \frac{\varepsilon T}{20n}\cdot \Delta_Q \le \Delta_P. \tag{3}$$

Using this last relation and the bound $\Delta_P \ge T/10$ we obtain that

$$\Delta_Q \ge \frac{20n}{\varepsilon T}\cdot\Delta_P - n \ge \frac{20n}{\varepsilon T}\cdot\frac{T}{10} - n = \frac{2n}{\varepsilon} - n \ge \frac{n}{\varepsilon} \ge \frac{k}{20}.$$

This means that $\Delta_Q = \Omega(k)$, and by Lemma 16 we conclude that $w(D_R \cap Q) \le (1+\varepsilon/2)\Delta_Q$. We can use (2), (3) and the bound $T \le 10\,\Delta_P$ to obtain

$$\begin{aligned} w(D_R\cap P) &\le \frac{\varepsilon T}{20} + \frac{\varepsilon T}{20n}\cdot w(D_R\cap Q) \le \frac{\varepsilon\cdot 10\,\Delta_P}{20} + \frac{\varepsilon T}{20n}\cdot\left(1+\frac{\varepsilon}{2}\right)\Delta_Q \\ &\le \frac{\varepsilon}{2}\cdot\Delta_P + \left(1+\frac{\varepsilon}{2}\right)\Delta_P = (1+\varepsilon)\Delta_P. \end{aligned}$$

This finishes the proof of correctness. Note that to show correctness, we only used the assumption $T < 10\,\Delta_P$. The other assumption $\Delta_P < 10T$ is used only to bound the running time.

Running time Observe that we can compute A, P', Q, R, Y in $O(n \log n)$ expected time, like in Lemma 14. The first item in the proof of Lemma 16 shows that whp $\Delta_R \le (1+\varepsilon/4)\rho\,\Delta_Q < 2p\,\Delta_Q$. Substituting ρ, k, using the relation (3), and with the assumption $\Delta_P < 10T$, we conclude that, whp,

$$\Delta_R = O\left(k^{-1}\varepsilon^{-2}\log n \cdot \frac{20n}{\varepsilon T}\cdot \Delta_P\right) = O\big((n/\varepsilon)^{-1}\,\varepsilon^{-3} n \log n\big) = O\big(\varepsilon^{-2}\log n\big).$$

Since R consists of at most n points and $Y = X \setminus \bigcup_{a\in A} D_a$, we can use Lemma 11 to find a best disk for $\mathsf{min}(R, Y)$ in $O(|R|\log|R| + |R|\Delta_R) = O(n\log n + n\,\Delta_R)$ expected time. Since whp we have $\Delta_R = O(\varepsilon^{-2}\log n)$, then whp we spend $O(n\varepsilon^{-2}\log n)$ expected time. The probability that Δ_R is not bounded by $O(\varepsilon^{-2}\log n)$ can be bounded by $O(n^{-3})$, and in this case we can solve the problem using a quadratic-time solution. It follows that the expected time is bounded by $O(n\varepsilon^{-2}\log n)$. □

By combining Lemmas 15 and 17 we get our final result:

Theorem 18 *Given a domain X of constant complexity, a parameter $0 < \varepsilon < 1$, and a weighted point set P with n points, we can find in $O(n(\log^2 n + \varepsilon^{-2}\log n))$ expected time a unit disk that, with high probability, covers a weight of at most $(1+\varepsilon)\mathsf{min}(P, X)$.*

Proof For the given point set P, we first apply Lemma 15 to obtain a value T that, whp, satisfies $T/10 < \mathsf{min}(P, X) < 10T$. This takes $O(n\log^2 n)$ expected time. Then, we apply Lemma 17 to obtain a unit disk that, whp, covers a weight of at most $(1+\varepsilon)\,\mathsf{min}(P, X)$. This step takes $O(n\,\varepsilon^{-2}\log n)$ expected time, and the theorem follows. □

5 Conclusions

We presented efficient approximation algorithms for the following two problems: (i) given a set of n weighted points in the plane and a constant m, place m disks so as to maximize the total weight of the points in the disks, (ii) given a set of n weighted points in the plane and a constant-complexity region X, place a single disk with center inside X so as to minimize the total weight of the points in the disk. Although our algorithms are near-linear in n and give $(1-\varepsilon)$ (resp. $(1+\varepsilon)$) approximations, the running times are most likely not optimal. Another direction for further research is to consider problem (i) for the case where m is not considered a constant and problem (ii) for the case where the region X has non-constant complexity. Finally, our solution to problem (ii) takes near-linear expected time and has some probability of error. It would be interesting to find algorithms for this problem that are always correct and take deterministic or randomized near-linear time.

Acknowledgements M. de Berg was supported by the Netherland's Organisation for Scientific Research (NWO) under project No. 639.023.301. S. Cabello was partially supported by the European Community Sixth Framework Programme under a Marie Curie Intra-European Fellowship, and by the Slovenian Research Agency, project J1-7218.

Appendix: Approximation Algorithms for Small *m*

We provide here the necessary material to prove Theorem 7(ii)–(v), which concern the cases $m = 1, 2, 3$. This is done considering different variants of Lemma 2, while leaving the rest of the algorithm basically unchanged. We start with the following variant of Lemma 2, which allows us to obtain a $(1/r)$-approximation faster, but of larger cardinality.

Lemma 19 *Let P be a weighted point set with n points and let $1 \le r \le n$ be a real parameter. We can construct a $(1/r)$-approximation A for P consisting of $O(r^3)$ points in $O(n \log r)$ time if $r^4 \le n$, or in $O(n^{5/4})$ time otherwise.*

Proof Like in the proof of Lemma 2, consider the set $\mathcal{V}$ of all possible cells that may arise in vertical decompositions of unit circles in the plane. As seen in the proof of Lemma 2, it is enough to show how to construct $(1/r)$-approximations for an *unweighted* point set P with respect to ranges $\mathcal{V}$. This only affects the constants hidden in the O-notation.

Consider the following lift $\ell : \mathbb{R}^2 \to \mathbb{R}^3$ of the plane to the paraboloid: a point $(x, y) \in \mathbb{R}^2$ is mapped to the point $\ell(p) = (x, y, x^2 + y^2) \in \mathbb{R}^3$. It is easy to check that the lift of the points of a circle are coplanar, and therefore the points of P in a given disk correspond to the points of $\ell(P)$ contained in a halfspace of $\mathbb{R}^3$. A cell $c \in \mathcal{V}$ is bounded by at most four rectilinear or circular segments. and therefore, the points of P contained in c correspond to the points of $\ell(P)$ contained in the intersection of at most four halfspaces of $\mathbb{R}^3$, that is, a (possibly unbounded) simplex of $\mathbb{R}^3$. It follows that constructing a $(1/r)$-approximation for P can be reduced to constructing a $(1/r)$-approximation for $\ell(P)$ with respect to simplices in $\mathbb{R}^3$.

Matoušek [20] provides tools to construct $(1/r)$-approximations with respect to simplices in $\mathbb{R}^3$ of size $O(r^3)$ in $O(n \log r)$ time if $r^3 \le n^{1-\delta}$, and in $O(n^{1+\delta})$ time otherwise, where $\delta > 0$ is an arbitrary fixed constant.[1] In particular, fixing $\delta = 1/4$, we conclude that a $(1/r)$-approximation with respect to simplices in $\mathbb{R}^3$ of size $O(r^3)$ can be found in $O(n \log r)$ time if $r^4 \le n$ and in $O(n^{5/4})$ time otherwise. The result follows. □

Equipped with this result we can revise the construction in Lemma 3, and prove the remaining items of Theorem 7 that concern deterministic algorithms.

Lemma 20 *For any weighted point set P with n points, we can find a set of m disks that cover a weight of at least $(1 - \varepsilon)\mathsf{max}(P, m)$ in $O(n \log n + n\varepsilon^{-2} + (n/k) \cdot T(k, m))$ time, where $k = O(\varepsilon^{-3})$.*

[1] The first case is given in the abstract of [20] and discussed after Theorem 4.7 in [20], while the second case follows from combining Theorem 4.7(iii) and Observation 4.1 in [20].

Proof Consider the construction in Lemma 3: after classifying P by cells of the grid G in $O(n\log n)$ time, for each cell C in G we construct a $(1/r')$ approximation A_C for $P_C = P\cap C$, where $r' = O(1/\varepsilon)$. For each cell C we do this using Lemma 19 if $|P_C| \geq (r')^3$, and taking $A_C = P_C$ if $|P_C| < (r')^3$. It is clear from the construction that each grid cell contains $O((r')^3) = O(\varepsilon^{-3})$ points. The time we use is bounded by

$$\begin{aligned}
&\sum_{C\in\mathcal{C},\,|P_C|\geq(r')^4} O(|P_C|\log r') + \sum_{C\in\mathcal{C},\,(r')^3\leq|P_C|<(r')^4} O(|P_C|^{5/4}) \\
&\quad + \sum_{C\in\mathcal{C},\,|P_C|\leq(r')^3} O(|P_C|) \\
&\leq O(n\log r') + \frac{n}{(r')^3}\cdot O\big(\big((r')^4\big)^{5/4}\big) + O(n) \\
&\leq O(n\log(1/\varepsilon)) + O(n\varepsilon^{-2}) = O(n\,\varepsilon^{-2}).
\end{aligned}$$

We conclude that we can rephrase Lemma 3 with a running time of $O(n\log n + n\,\varepsilon^{-2})$ time and with the property that each cell of G contains $O(\varepsilon^{-3})$ points. Leaving the rest of the discussion up to Lemma 6 unaltered, we obtain the result. □

Proof of Theorem 7(ii), (iii) As discussed in Sect. 3, we have the bounds $T(k,m) = O(k^{2m-1}\log k)$ for $m>1$, and $T(k,m) = O(k^2)$ for $m=1$. For $m>1$ we have $O((n/k)\cdot T(k,m)) = O(nk^{2m-2}\log k) = O(n\,\varepsilon^{-6m+6}\log(1/\varepsilon))$, and Lemma 6 implies the bound claimed in case (ii). (For $m\geq 4$, the bound we obtain by this method is not an improvement.) For $m=1$ we have $O((n/k)\cdot T(k,1)) = O(nk) = O(n\,\varepsilon^{-3})$, and we obtain the bound claimed in case (iii). □

An alternative approach based on random sampling gives the following counterpart of Lemma 2.

Lemma 21 *Let P be a weighted point set with n points and let $1\leq r\leq n$ be a real parameter. Whp, we can construct in $O(n)$ time a $(1/r)$-approximation A for P consisting of $O(r^2\log n)$ points.*

Proof Scaling the weights appropriately, we may assume that $w(P) = 2nr$. We construct an integer-weighted point set Q by placing each point $p\in P$ with weight $\lfloor w_p\rfloor$. For any $U\in\mathcal{U}$, we have $w(P\cap U) - n \leq w(Q\cap U)\leq w(P\cap U)$, and therefore

$$|w(P\cap U) - w(Q\cap U)| \leq \frac{w(P)}{2r}.$$

If R is a $(1/2r)$-approximation for Q, then for any $U\in\mathcal{U}$ we have

$$\begin{aligned}
&\big|w(P\cap U) - w(R\cap U)\big| \\
&\quad = \big|w(P\cap U) - w(Q\cap U) + w(Q\cap U) - w(R\cap U)\big|
\end{aligned}$$

$$\leq \left|w(P \cap U) - w(Q \cap U)\right| + \left|w(Q \cap U) - w(R \cap U)\right|$$
$$\leq \frac{w(P)}{2r} + \frac{w(Q)}{2r} \leq \frac{w(P)}{r},$$

and thus R is also a $(1/r)$-approximation for P.

Algorithmically, constructing Q requires scaling and rounding weights, which can be done in $O(n)$ time. We then take a ρ-sample R of Q, where $\rho = c \cdot w(Q)^{-1} r^2 \log nr$ for an appropriate constant c. It is well-known that, if c large enough, R is indeed a $(1/2r)$-approximation for Q whp [3, Chap. 13], and therefore, R is also a $(1/r)$-approximation for P. Moreover, since $\rho \cdot w(Q) = c \cdot r^2 \log nr$ and $r \leq n$, the set R consists of $O(r^2 \log nr) = O(r^2 \log n)$ points whp. □

Equipped with this result and revising the discussion, we obtain the following randomized counterpart of Lemma 6, which directly implies Theorem 7(iv), (v). The final result that we obtain is relevant only for $m \leq 3$.

Lemma 22 *For any weighted point set P with n points, we can find a set of m disks that cover a weight of at least $(1-\varepsilon)\mathsf{max}(P, m)$ in $O(n \log n + (n/k) \cdot T(k, m))$ time, where $k = O(\varepsilon^{-2} \log n)$. The result and the time bound are correct whp.*

Proof Using Lemma 19 instead of Lemma 2, we can rephrase Lemma 3 with a running time of $O(n \log n)$ time and with the property that each cell of G contains $O(\varepsilon^{-2} \log n)$ points. Both events happen whp. Leaving the rest of the discussion unaltered, we obtain the result. □

References

1. Agarwal, P.K., Hagerup, T., Ray, R., Sharir, M., Smid, M., Welzl, E.: Translating a planar object to maximize point containment. In: ESA 2002. LNCS, vol. 2461. Springer, Berlin (2002)
2. Agarwal, P.K., Procopiuc, C.M.: Exact and approximation algorithms for clustering. Algorithmica **33**(2), 201–226 (2002)
3. Alon, N., Spencer, J.: The Probabilistic Method, 2nd edn. Wiley, New York (2000)
4. Aronov, B., Har-Peled, S.: On approximating the depth and related problems. SIAM J. Comput. **38**, 899–921 (2008)
5. Bose, P., van Kreveld, M., Maheshwari, A., Morin, P., Morrison, J.: Translating a regular grid over a point set. Comput. Geom. Theory Appl. **25**, 21–34 (2003)
6. Cabello, S., Díaz Báñez, J.M., Seara, C., Sellarès, J.A., Urrutia, J., Ventura, I.: Covering point sets with two disjoint disks or squares. Comput. Geom. Theory Appl. **40**, 195–206 (2008)
7. Chazelle, B.: The Discrepancy Method: Randomness and Complexity. Cambridge University Press, New York (2001)
8. Chazelle, B.: The discrepancy method in computational geometry. In: Handbook of Discrete and Computational Geometry, pp. 983–996. CRC, Boca Raton (2004)
9. Chazelle, B., Lee, D.T.: On a circle placement problem. Computing **36**, 1–16 (1986)
10. Clarkson, K.L., Shor, P.W.: Applications of random sampling in computational geometry II. Discrete Comput. Geom. **4**, 387–421 (1989)
11. de Berg, M., van Kreveld, M., Overmars, M., Schwarzkopf, O.: Computational Geometry: Algorithms and Applications, 2nd edn. Springer, Berlin (2000)
12. Drezner, Z.: On a modified one-center model. Manag. Sci. **27**, 848–851 (1981)
13. Drezner, Z., Wesolowsky, G.O.: Finding the circle or rectangle containing the minimum weight of points. Location Sci. **2**, 83–90 (1994)

14. Gajentaan, A., Overmars, M.H.: On a class of $O(n^2)$ problems in computational geometry. Comput. Geom. Theory Appl. **5**, 165–185 (1995)
15. Halperin, D.: Arrangements. In: Goodman, J.E., O'Rourke, J. (eds.) Handbook of Discrete and Computational Geometry, pp. 529–562. CRC, Boca Raton (2004)
16. Hochbaum, D.S., Maass, W.: Approximation schemes for covering and packing problems in image processing and vlsi. J. ACM **32**(1), 130–136 (1985)
17. Katz, M.J., Kedem, K., Segal, M.: Improved algorithms for placing undesirable facilities. Comput. Oper. Res. **29**, 1859–1872 (2002)
18. Katz, M.J., Sharir, M.: An expander-based approach to geometric optimization. SIAM J. Comput. **26**, 1384–1408 (1997)
19. Kleinberg, J., Tardos, E.: Algorithm Design. Addison-Wesley, Reading (2005)
20. Matoušek, J.: Efficient partition trees. Discrete Comput. Geom. **8**, 315–334 (1992)
21. Matoušek, J.: Approximations and optimal geometric divide-an-conquer. J. Comput. Syst. Sci. **50**, 203–208 (1995)
22. Plastria, F.: Continuous covering location problems. In: Hamacher, H., Drezner, Z. (eds.) Location Analysis: Theory and Applications, pp. 39–83. Springer, Berlin (2001), Chap. 2
23. Sharir, M.: On k-sets in arrangements of curves and surfaces. Discrete Comput. Geom. **6**, 593–613 (1991)