Efficient Approximations for the Online Dispersion Problem

Jing Chen Bo Li Yingkai Li
Department of Computer Science, Stony Brook University
Stony Brook, NY 11794
{jingchen, boli2, yingkli}@cs.stonybrook.edu

November 3, 2016

Abstract

The *dispersion* problem has been widely studied in computational geometry and facility location, and is closely related to the packing problem. The goal here is to locate n points (e.g., facilities or persons) in a k-dimensional polytope, so that *they are far away from each other and from the boundary of the polytope.* In many real-world scenarios however, the points to be located arrive and depart at different times, and decisions must be made without knowing future events. Therefore we study, for the first time in the literature, the *online dispersion* problem. There are two natural objectives when time is involved: the *all-time worst-case* (ATWC) problem tries to maximize the minimum distance that ever appears at any time; and the *cumulative distance* (CD) problem tries to maximize the integral of the minimum distance throughout the whole time interval. Interestingly, the online problems are highly non-trivial even on a segment. For the cumulative distance, this remains to be the case even when the problem is time-dependent but offline, with all the arriving and departure times given in advance.

For the online ATWC problem on a segment, we construct a deterministic polynomial-time algorithm which is a $2\ln 2 + \epsilon$ approximation, where $\epsilon > 0$ can be arbitrarily small and the algorithm's running time is polynomial in $\frac{1}{\epsilon}$. We show this algorithm is actually *optimal*. For the same problem in a square, we provide a 1.591-approximation and a 1.183 lower-bound. Furthermore, for arbitrary k-dimensional polytopes with $k \geq 2$, we provide a $\frac{2}{1-\epsilon}$-approximation and a $\frac{7}{6}$ lower-bound. All our lower-bounds come from the structure of the online problems and hold even when computational complexity is not a concern. Surprisingly, for the offline CD problem in arbitrary k-dimensional polytopes, we provide a polynomial-time black-box reduction to the online ATWC problem, and the resulting approximation ratio increases by a factor of at most 2. To establish our results, we show important connections between the dispersion problem and the ball-packing problem —both *uniform* packing and *non-uniform* packing. Our techniques also apply to online dispersion problems with different boundary conditions.

Keywords: dispersion, online algorithms, geometric optimization, packing, approximation algorithms

1 Introduction

The problem of assigning elements to locations in a given area comes up only too often in real life: where to sit the customers in a restaurant, where to put certain facilities in a city, where to build the nuclear power stations in a country, etc. Different problems have different features and constraints, but one common feature that appears in many of them is *not to locate the elements too close to each other*: for people's privacy, for the environment and/or for serving more users. Another feature is also common in many applications: that is, *not to locate the elements too close to the boundary of the area.* Indeed, for security reasons, important national industrial facilities in some countries are built at a safe distance away from the border. Such problems have been widely studied in computational geometry and facility location; see, e.g., [25, 3, 2, 4]. In particular, in the *dispersion* problem defined by [2], there is a k-dimensional polytope P and an integer $n > 0$, and the goal is to locate n points in P so as to maximize *the minimum distance among them and from them to the boundary of* P.

However, an important feature in all the scenarios mentioned above and many other real-world scenarios has been missing in the study of dispersion problems: the presence of elements is *time-dependent* and decisions need to be made along time, without knowing when the elements will come and go in the future. It may be hard to move an element once it is located, making it infeasible for the decision maker to relocate all the present elements according to the optimal static solution when an arrival/departure event occurs. In this paper we consider, for the first time in the literature, the *online dispersion* problem. Here, the arriving and departure times of points are chosen by an adversary who knows everything and works adaptively. An online dispersion algorithm decides where to locate a point upon its arrival, without any knowledge about future events.

1.1 Main Results

We focus on two natural objectives for the online problem: the *all-time worst-case* (ATWC) problem, which aims at maximizing the minimum distance that ever appears at any time; and the *cumulative distance* (CD) problem, which aims at maximizing the integral of the minimum distance through out the whole time interval. Although polynomial-time constant approximations have been given when time is not involved [2, 4], nothing was known about the online problem. In fact, as we will show, the solution here is quite complex even on a *segment*. Moreover, for the cumulative distance, even when the problem is *time-dependent but offline*, with all the arriving and departure times given in advance, it remains unclear how to efficiently compute the optimal solution. We formally define the problem in Section 2 and summarize our results in Table 1 below. In particular, the most technical parts are the online ATWC problem and the offline time-dependent CD problem. Surprisingly, the latter can use the former as a black-box. To establish our results, we have shown interesting connections between dispersion and ball-packing —both *uniform* packing (i.e., with identical balls) and *non-uniform* packing (i.e., with balls of different sizes). All our algorithms are deterministic and of polynomial-time; some of them take an arbitrarily small constant ϵ as a parameter and the running time is polynomial in $\frac{1}{\epsilon}$. All our lower-bounds hold even when the running time of an algorithm is not a concern. Most proofs are given in the appendix.

1.2 Related Work

Dispersion without time. In dispersion problems in general, the possible locations can be either a *continuous* region or a set of *discrete* candidates. Two objectives have been studied in the literature: the *max-min distance* as considered in this paper, and the *maximum total distance.* In

	Online	Offline time-dependent
ATWC	$k = 1$: a $2 \ln 2$ (≈ 1.386) lower-bound and an optimal algorithm; see Theorems 1 and 2. $k = 2$: a 1.183 lower-bound and a 1.591-approximation for squares; see Theorems 4 and 5. $k \geq 2$: a $\frac{7}{6}$ lower-bound and a $\frac{2}{1-\epsilon}$-approximation for arbitrary polytopes; see Theorems 6 and 7.	Equivalent to dispersion without time; see Claim 2.
CD	No non-trivial approximation even when $k = 1$; see Claim 1.	A black-box reduction to the online ATWC problem for arbitrary k and P, with the approximation ratio scaling up by at most 2; see Theorem 8.

Table 1: Online and offline time-dependent dispersion problems in a k-dimensional polytope P.

continuous settings, the authors of [2] consider the max-min distance with the boundary condition. Under L_∞-norm, they give a polynomial-time 1.5-approximation in rectilinear polygons[1] and show that a $\frac{14}{13}$-approximation in arbitrary polygons is NP-hard. Moreover, they show there is no PTAS under any norm unless P=NP. [4] considers a similar boundary condition under L_2-norm and provides a 1.5-approximation in polygons with obstacles. [7] considers the problem of selecting n points in n given unit-disks, one per disk, and the objective is the max-min distance.

In discrete settings, [25] shows that, if the distances among the candidates do not satisfy triangle inequality then there is no polynomial-time constant approximation for either distance unless P=NP; otherwise there is a 2-approximation for the max-min distance and a 4-approximation for the total distance. If the candidates are in a k-dimensional space and distances are measured under L_1-norm, a PTAS is given in [11] for the total distance. Finally, [3, 24, 20, 1] consider various dispersion problems in the context of obnoxious facility location.

Packing without time. In some sense, dispersion and packing are "dual" problems of each other. In the technical part of this paper we will highlight several interesting connections between them and use several important results for packing in our analysis. Thus we briefly introduce this literature. Indeed, the packing problem is one of the most extensively studied problems in geometric optimization, and a huge amount of works have been done on different variants of the problem; see [23, 16] for surveys on this topic.

One important problem is to *pack as many identical circles as possible in a bounded plane.* [14] shows this problem to be strongly NP-hard and [17] gives a PTAS for it. An APTAS for the *circle bin packing* problem is given in [22]. The *dispersal packing* problem tries to maximize the radius of a given number of circles packed in a square. A lot of effort has been made in finding the optimal radius and the corresponding packing when the number of circles is small; see [28, 27, 29, 23]. Heuristic methods have also been used in finding approximations when the number of circles gets larger [19, 31]. Finally, an important packing problem is to understand the *packing density*: that is, the maximum fraction of an infinite space covered by a packing of unit circles/spheres. The packing density is solved for dimension 2 in [10] and for dimension 3 in [15]. Very recently, [32]

[1] A rectilinear polygon is a 2-dimensional polytope whose edges are axis-parallel.

and [6] solve it for dimensions 8 and 24, respectively. Asymptotic lower bounds (as the dimension grows) for the density of the densest packing are provided in [26].

Online geometric optimization. Many important geometric optimization problems have been studied in online settings, although the objectives are quite different from ours. The seminal work of [30] provides a nearly-optimal approximation algorithm for the classic *online bin-packing* problem. Algorithms for variants of the problem have been considered ever since, such as a constant approximation for packing circles in square bins [18], and constant approximations for bin-packing in higher dimensions [8, 9].

In *online facility location* [21], it is the demands rather than the facilities that come along time. The facilities have open costs and the goal is to minimize the total open cost and the total distance between demands and facilities. As shown in [21], when the demands arrive adversarially, there is a randomized polynomial-time $O(\log n)$-approximation and no constant approximation exists. A deterministic $O(\frac{\log n}{\log\log n})$-approximation and a matching lower-bound are provided in [13] for the same problem. In *incremental facility location* [12], the facilities can be opened, closed or merged, depending on the arriving demands.

2 The Online Dispersion Problem

Given a k-dimensional polytope P, the *dispersion* problem [2] takes as input a positive integer n and outputs n locations, $X_1, \ldots, X_n \in P$, specifying how to locate n points. For each point i, let $dis(X_i, \partial P)$ be the distance from X_i to ∂P, the boundary of P, measured by L_2-norm. Also, let $dis(X_i, X_j)$ be the distance between X_i and X_j for any $i \neq j$. The objective is

$$Disp(n; P) \triangleq \max_{X_1,\ldots,X_n \in P} \min_{i,j\in[n]} \{dis(X_i, \partial P), dis(X_i, X_j)\}.$$

In Appendix F we also consider the dispersion problem where the distances to the boundary are not taken into consideration. Most of our techniques can be applied there.

We now define the *online dispersion* problem, where each point i arrives at time s_i and departs at time d_i, with $d_i > s_i$. Without loss of generality, $0 = s_1 \leq s_2 \leq \cdots \leq s_n$. An online algorithm is notified upon the occurrence of an arrival/departure event. It must decide the location X_i for a point i upon its arrival, knowing neither the future events nor the total number of points n. An adversary knows how the algorithm works and chooses future events after seeing the output of the algorithm so far. In the *time-dependent offline* version of the problem, the times of all events, denoted by a vector $S = ((s_1, d_1), \ldots, (s_n, d_n))$, is given to the algorithm in advance.

Given such a vector S, let $T = \max_{i\in[n]} d_i$ be the last departure time. Moreover, given locations $X = (X_1, \ldots, X_n)$, for any $t \leq T$, let

$$d_{min}(t; X) = \min_{i,j\in[n]: s_i \leq t \leq d_i, s_j \leq t \leq d_i} \{dis(X_i, \partial P), dis(X_i, X_j)\}$$

be the minimum distance corresponding to the points that are present at time t. When X is clear from the context, we may write $d_{min}(t)$ for short. We consider two natural objectives: the *all-time worst-case* (ATWC) problem, where the objective is

$$OPT_A(S; P) \triangleq \max_{X_1,\ldots,X_n} \min_{t\leq T} d_{min}(t);$$

and the *cumulative distance* (CD) problem, where the objective is

$$OPT_C(S; P) \triangleq \max_{X_1,\ldots,X_n} \int_0^T d_{min}(t)dt.$$

Note that both objectives are the optimum of the corresponding offline problems, as well as the *ex-post* optimum for the online problems. Below we provide two simple observations about them, proved in Appendix A.

Claim 1. *For the CD problem, even when $k = 1$ and P is the unit segment, no (randomized) online algorithm provides an approximation to OPT_C better than $\Omega(n)$.*

Note that an $O(n)$ approximation is trivial, whether online or offline. Next, given any ex-post instance $S = ((s_1, d_1), \ldots, (s_n, d_n))$, let m be the maximum number of points simultaneously present at any time t: that is, $m = \max_{t \le T} |\{i : s_i \le t \le d_i\}|$.

Claim 2. *$\forall\, S = ((s_1, d_1), \ldots, (s_n, d_n))$, letting $m = \max_{t \le T} |\{i : s_i \le t \le d_i\}|$, we have $OPT_A(S; P) = Disp(m; P)$.*

In light of the claims above, we will focus on the online ATWC problem and the offline CD problem, in particular the former. We now highlight some interesting connections between dispersion and ball-packing. The *(uniform) ball-packing* problem [17] in a polytope P takes as input a non-negative value r and outputs an integer n, the maximum number of balls of radius r that can be packed non-overlappingly in P, together with a corresponding packing. Denote the solution by $Pack(r; P)$. The *dispersal packing* problem [2] is a "mixture" of dispersion and packing: it takes as input an integer n and outputs the maximum radius for n identical balls that can be packed in P. That is, $DP(n; P) \triangleq \max\{r : Pack(r; P) \ge n\}$.

Recall that a k-dimensional convex polytope P has an *insphere* if the largest ball contained wholly in P is tangent to *all* the facets (i.e., $(k-1)$-faces) of P. Such a ball, if exists, is unique. It is referred to as the insphere of P. The center of the insphere maximizes the minimum distance for any point in P to its facets, and has the same distance to all facets —the radius of the insphere. We have the following two claims.

Claim 3. *For any $k \ge 1$ and any k-dimensional convex polytope P with an insphere, letting x be the radius of the insphere, we have $Disp(n; P) = \frac{2xDP(n;P)}{x+DP(n;P)}$.*

Claim 4. *For any $k \ge 1$ and any k-dimensional convex polytope P with an insphere, given the radius of the insphere,*

(1) any polynomial-time algorithm for $Disp(n; P)$ implies such an algorithm for $Pack(r; P)$, and

(2) any polynomial-time algorithm for $Pack(r; P)$ implies an FPTAS for $Disp(n; P)$.

To our best knowledge, it is still unknown whether ball-packing in regular polytopes (which is a special case of convex polytopes with an insphere) is NP-hard or not. Therefore the complexity of dispersion in regular polytopes remains open. Note that ball-packing in arbitrary polytopes is NP-hard [14], so is a $\frac{14}{13}$-approximation for dispersion in rectilinear polygons [2]. Moreover, a claim similar to Claim 4 applies to $DP(n; P)$ and $Pack(r; P)$ in arbitrary polytopes. The relation between dispersion and packing in arbitrary polytopes is worth further investigation.

3 The 1-Dimensional Online All-Time Worst-Case Problem

Note that a 1-dimensional polytope is simply a segment. Without loss of generality, we consider the unit segment $P = [0, 1]$. Below we first provide a lower bound for the approximation ratio of any algorithm, even computationally unbounded ones.

3.1 The Lower Bound

Theorem 1. *No online algorithm achieves an approximation ratio better than* $2\ln 2$ *(*≈ 1.386*) for the 1-dimensional ATWC problem.*

Proof sketch. Letting $\sigma'_r = \sum_{i=r+1}^{2r} \frac{1}{i}$ for any positive integer r, we show that no algorithm achieves an approximation ratio better than $2\sigma'_r$. Roughly speaking, we construct an instance (i.e., an adversary) for the online ATWC problem with three stages. In the first stage, $r-1$ points arrive simultaneously; in the second stage, r new points arrive one by one; and finally, all $2r-1$ points depart simultaneously. If an algorithm $\mathcal{A}$ is an α-approximation to OPT_A with $\alpha < 2\sigma'_r$, it must be such an approximation after the arrival of each point, as it does not know the total number of points arriving. We show that in order for $\mathcal{A}$ to do so, the segment must be longer than P itself, a contradiction. The complete proof is provided in the Appendix B.1. □

3.2 A Polynomial-Time Online Algorithm

Next, we provide a deterministic polynomial-time online algorithm whose approximation ratio to OPT_A can be arbitrarily close to $2\ln 2$. Intuitively, a good algorithm should disperse the points as evenly as possible. However, if at some point of time with m points present, the resulting $m+1$ intervals on the segment have almost the same length, then the next arriving point will force the minimum distance to drop by a factor of 2, while the optimum only changes from $\frac{1}{m+1}$ to $\frac{1}{m+2}$, causing the approximation ratio to drop by almost 2. To overcome this problem, the algorithm must find a balance between two consecutive points, choosing a sub-optimal solution for the former so as to leave enough space for the latter. The difficulty, as for online algorithms in general, is that this balance needs to be kept for arbitrarily many pairs of consecutive point, as the sequence of points is chosen by an adversary who observes the algorithm's output.

Inspired by our lower bound, roughly speaking, our algorithm uses a parameter r to pre-fix the locations of the first r points and the resulting $r+1$ intervals, and then inserts the next $r+1$ points in the middle of these intervals. The idea is that, when done properly, after these $2r+1$ points, the resulting configuration is almost the same as if the algorithm has used parameter $2r+1$ to pre-fix the first $2r+2$ intervals: then the procedure can repeat for arbitrary sequences.

More specifically, given a positive integer r, let $Q = \{q_1, \ldots, q_r\}$ be a set of positions on the segment, such that the length ratios of the $r+1$ intervals sliced by them are $\frac{1}{r+1} : \frac{1}{r+2} : \cdots : \frac{1}{2r+1}$. That is, letting $\sigma_r = \sum_{i=r+1}^{2r+1} \frac{1}{i}$, the lengths of the intervals are $\frac{1}{\sigma_r(r+1)}$, $\frac{1}{\sigma_r(r+2)}$, $\cdots$, $\frac{1}{\sigma_r(2r+1)}$, and $q_i = \frac{1}{\sigma_r} \sum_{j=r+1}^{r+i} \frac{1}{j}$ for each $i \in [r]$, as illustrated by Figure 1, with $q_0 = 0$ and $q_{r+1} = 1$. Note that σ_r differs from σ'_r in Theorem 1 by $\frac{1}{2r+1}$. Also, σ_r is strictly decreasing in r and $\lim_{r\to\infty} \sigma_r = \ln 2$. Moreover, for any two intervals (q_{j-1}, q_j) and $(q_{j'-1}, q_{j'})$ with $j < j' \le r+1$, we have $|(q_{j'-1}, q_{j'})| < |(q_{j-1}, q_j)| < 2|(q_{j'-1}, q_{j'})|$.

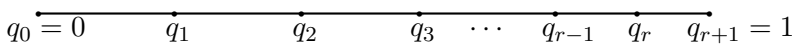

Figure 1: The pre-fixed positions in Q for dimension 1.

Our algorithm also takes as parameter an ordering for the positions in Q, denoted by $\tau = (\tau_1, \tau_2, \ldots, \tau_r)$. It is defined in Algorithm 1 and we have the following two lemmas, proved in Appendix B.2 and B.3, respectively. Lemma 2 is the harder part and we sketch the main ideas below. Recall that, given $S = ((s_1, d_1), \ldots, (s_n, d_n))$, m is the maximum number of points simultaneously present at any time t.

Algorithm 1. A polynomial-time algorithm for the 1-dimensional online ATWC problem.

Parameter: A positive integer r, the corresponding set $Q = \{q_1, \ldots, q_r\}$, and an ordering τ for Q.
Input: A sequence of points arriving and departing along time.

Denote by $\hat{Q}$ the set of positions ever occupied by a point. At any point of time, a position in $\hat{Q}$ is labeled *occupied* if currently there is a point there and *vacant* otherwise. Initially $\hat{Q} = \emptyset$.
When a point i leaves, change the label of its position in $\hat{Q}$ from *occupied* to *vacant*.
When a point i arrives:
if $\hat{Q} = \emptyset$ or all positions in $\hat{Q}$ are labelled *occupied* **then**
  if $Q \not\subseteq \hat{Q}$ **then**
    Choose the first position q according to τ with $q \in Q \setminus \hat{Q}$, add it to $\hat{Q}$ and label it *occupied*.
    Put i at position q.
  else
    Find position q which is the middle of the largest interval created by the positions in $\hat{Q}$.
    Put i at position q, add q to $\hat{Q}$ and label it *occupied*.
  end if
else
  Arbitrarily choose a vacant position q from $\hat{Q}$ and label it *occupied*.
  Put i at position q.
end if

Lemma 1. *Given any r and τ, Algorithm 1 is a $2\sigma_r$-approximation to OPT_A for any S with $m > r$.*

Lemma 2. *For any positive integer l and $r = 2^l - 1$, there exists an ordering τ for the corresponding set Q, such that Algorithm 1 is a $2\sigma_r$-approximation to OPT_A for any S with $m \leq r$.*

Proof sketch. Interestingly, we only need to consider the instance $S = ((1, r+1), (2, r+1), \ldots, (r, r+1))$. We construct an ordering τ for Q such that the approximation ratio at any time $d \in [r]$ is smaller than $2\sigma_r$. To do so, we fill in a complete binary tree with r nodes as in Figure 2, and τ is obtained by traversing the tree in a breadth-first manner starting from the root. □

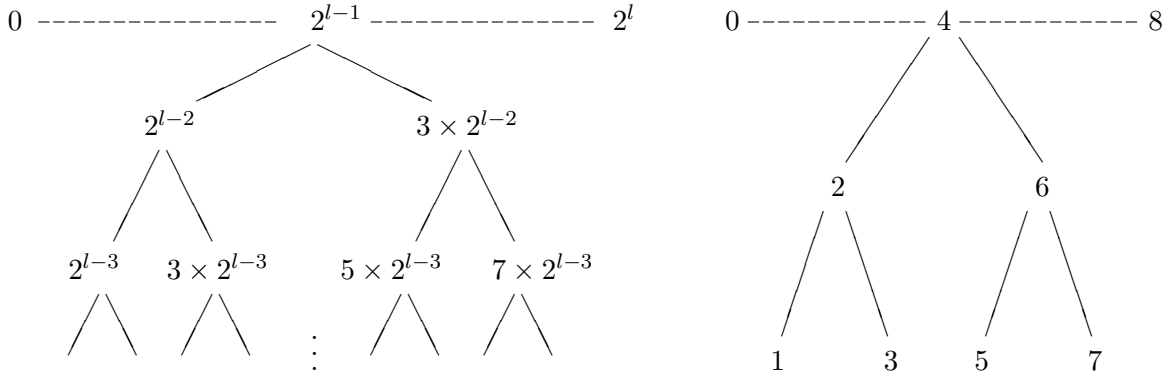

Figure 2: The left-hand side shows the top three levels of the binary tree for a general l, with $\tau = (q_{2^{l-1}}, q_{2^{l-2}}, q_{3\times 2^{l-2}}, q_{2^{l-3}}, q_{3\times 2^{l-3}}, q_{5\times 2^{l-3}}, q_{7\times 2^{l-3}}, \ldots)$. The right-hand side shows the complete binary tree for $l = 3$, with $\tau = (q_4, q_2, q_6, q_1, q_3, q_5, q_7)$.

The theorem below holds almost immediately from the two lemmas; see Appendix B.4.

Theorem 2. *There exists a deterministic polynomial-time online algorithm for the ATWC problem, whose approximation ratio can be arbitrarily close to* $2\ln 2$. *Moreover, the running time is polynomial in* $\frac{1}{\epsilon}$ *for approximation ratio* $2\ln 2+\epsilon$.

Remark. When the number of arrived points is large but the maximum number m of simultaneously present points is small, the running time of the algorithm for each arriving point is polynomial in m and can be much faster than being polynomial in the size of the input.

Following Theorem 1, Algorithm 1 is essentially optimal. Inspired by our constructions of Q and τ, we actually characterize the optimal solution for the online ATWC problem, whose approximation ratio is exactly $2\ln 2$: see Theorem 3 below, proved in Appendix B.5. However, this solution involves irrational numbers and cannot be exactly computed in polynomial time.

Theorem 3. *For any integer* $d=2^i+s$ *with* $i\geq 0$ *and* $0\leq s\leq 2^i-1$, *let* $\tau_d=\log_2(1+\frac{2s+1}{2^{i+1}})$. *If Algorithm 1 creates the d-th new position in* $\hat{Q}$ *to be* τ_d, *the approximation ratio is exactly* $2\ln 2$.

4 The 2-Dimensional Online All-Time Worst-Case Problem

We now consider the 2-dimensional online ATWC problem in a square—without loss of generality, $P=[0,1]^2$. One difficulty is that, different from the 1-dimensional problem where it is trivial to have $Disp(n;P)=\frac{1}{n+1}$ for any $n\geq 1$, here neither $Disp(n;P)$ nor $Pack(r;P)$ has a known closed-form optimal solution (whether polynomial-time computable or not). Accordingly, our lower-bound and our approximation algorithm must rely on some proper upper- and lower-bounds for $Disp(n;P)$, which is why the resulting bounds are not tight. In particular, we have the following lemma, proved in Appendix C.1.

Lemma 3. *For any* $n\geq 1$, $\frac{2}{5+\sqrt{2\sqrt{3}n}}\leq Disp(n;P)\leq\frac{2}{2+\sqrt{2\sqrt{3}n}}$.

4.1 The Lower Bound

Interestingly, not only the dispersion problem is closely related to *uniform* packing (i.e., the disks all have the same radius) as we have seen in Section 2, but we also obtain a lower bound for the online ATWC problem by carefully fitting a *non-uniform* packing into the square. The idea is to imagine each position created in an online algorithm as a disk centered at that position. The radius of each disk is a function of the algorithm's approximation ratio and the optimal solutions to specific dispersion problems without time. Note that the area covered by the disks is upper-bounded by the area of the square containing them. Combining these relations together gives us the following theorem, proved in Appendix C.2.

Theorem 4. *No online algorithm achieves an approximation ratio better than 1.183 for the 2-dimensional ATWC problem in a square.*

4.2 A Polynomial-Time Online Algorithm

Now we provide a deterministic polynomial-time online algorithm which is a 1.591-approximation to OPT_A. Similar to Algorithm 1, we first construct a set Q of pre-fixed positions. However, it is unclear how to define Q of arbitrary size in the square, and we construct a set of 36 positions, denoted by $Q=\{q_1,\ldots,q_{36}\}$. It depends on a parameter $1<c<\sqrt{2}$ and $x=\frac{1}{3+4c}$, as illustrated in Figure 3. The indices of the q_i's specify the order according to which they should be occupied, thus we do not need an extra ordering τ. Note that these positions create a grid in P and split it into multiple rectangles. The choice of c (and x, Q) will become clear in the analysis.

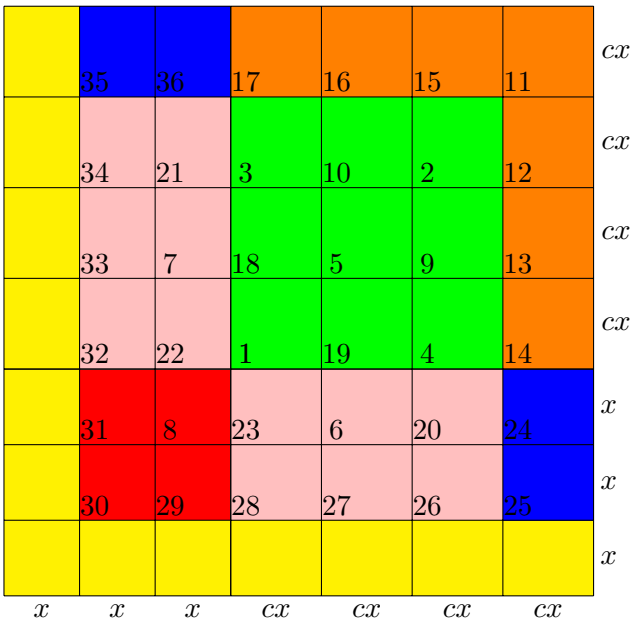

Figure 3: The set of pre-fixed positions, $Q = \{q_1, \ldots, q_{36}\}$, and the grid created by Q. A grid point labelled by i indicates the position q_i. The colored areas are used in the algorithm's description. More specifically, denoting a rectangle by the position in Q at its lower-left corner, the green area is $(3, 10, 2; 18, 5, 9; 1, 19, 4)$; the two pink areas are $(23, 6, 20; 28, 27, 26)$ and $(34, 21; 33, 7; 32, 22)$; the red area is $(31, 8; 30, 29)$; the two blue areas are $(35, 36)$ and $(24, 25)$; the orange area is $(17, 16, 15, 11, 12, 13, 14)$; and finally the yellow area contains all the remaining rectangles: that is, rectangles adjacent to the left boundary and the bottom boundary.

Algorithm 2. A polynomial-time online algorithm for the ATWC problem in a square.

Parameter: c such that $1 < c < \sqrt{2}$, the corresponding $x = \frac{1}{3+4c}$, and Q.
Input: A sequence of points arriving and departing along time.

Denote by $\hat{Q}$ the set of positions ever occupied by a point. At any point of time, a position in $\hat{Q}$ is labeled *occupied* if currently there is a point there and *vacant* otherwise. Initially $\hat{Q} = \emptyset$.
When a point w leaves, change the label of its position in $\hat{Q}$ from *occupied* to *vacant*.
When a point w arrives:
if $\hat{Q} = \emptyset$ or all positions in $\hat{Q}$ are labelled *occupied* **then**
  if $|\hat{Q}| < 36$ **then**
    Put w at position $q_{|\hat{Q}|+1}$, add this position to $\hat{Q}$ and label it *occupied*.
  else
    Compute a position q according to the Position Creation Phase defined in Algorithm 4.
    Put w at position q, add q to $\hat{Q}$ and label it *occupied*.
  end if
else
  Arbitrarily choose a vacant position q from $\hat{Q}$ and label it *occupied*.
  Put w at position q.
end if

Whenever a new position needs to be created, we pick the first position in Q that has never been occupied yet. When all positions in Q are occupied, we may (1) create a new position in the center of a current rectangle with the largest area, split this rectangle into four smaller ones accordingly, and add the vertices of the new rectangles into the grid; or (2) create a new position at a grid point that has never been occupied yet. The main algorithm is similar to Algorithm 1 and defined in Algorithm 2. It uses in Step 8 a sub-routine, the *Position Creation Phase,* as defined in Algorithm 4 in Appendix C.3. In Appendix C.4 we provide some intuition on the choices of Q, x, and c. By setting $c = 1.271$, we have the following theorem, proved in Appendix C.5.

Theorem 5. *Algorithm 2 is of polynomial time and is a* 1.591*-approximation for the 2-dimensional online ATWC problem in a square.*

Note that the upper-bound for $Disp(n; P)$ in Lemma 3 is not tight when n is small. With better upper-bounds for $Disp(n; P)$, better approximation ratios for our algorithm can be directly obtained via a similar analysis. Moreover, we believe the approximation ratio can be improved by using a larger set Q and the best ordering for positions in Q. Such a Q and a rigorous analysis based on it are left for future studies.

5 The General k-Dimensional Online ATWC Problem

Although the literature gives us little understanding about the optimal dispersion/packing problem in an arbitrary k-dimensional polytope P with $k \geq 2$, we are still able to provide a simple lower-bound and a simple algorithm for the online ATWC problem. Below we only state the theorems and briefly discuss the ideas; see Appendix D for the proofs.

Theorem 6. *For any $k \geq 2$, no online algorithm achieves an approximation ratio better than $\frac{7}{6}$ for the ATWC problem for arbitrary polytopes.*

Proof sketch. We show that even when the "polytope" is a k-dimensional sphere and there are only two points, the approximation ratio of any online algorithm is already no better than $\frac{7}{6}$. □

For any polytope P, letting the *covering rate* be the ratio between the edge-lengths of the maximum inscribed cube and the minimum bounding cube, we have the following theorem. The geometric problems of finding the minimum bounding cube, deciding whether a position is in P, and finding the distance between a point in P and the boundary of P are given as oracles.

Theorem 7. *For any constants $\gamma, \epsilon > 0$, for any integer $k \geq 2$ and any k-dimensional polytope P with covering rate at least γ, there exists a deterministic polynomial-time online algorithm for the ATWC problem, with approximation ratio $\frac{2}{1-\epsilon}$ and running time polynomial in $\frac{1}{(\gamma\epsilon)^k}$.*

Proof sketch. The main difficulty in dealing with arbitrary high-dimensional polytopes is that the shape of the densest packing can be highly irregular and the literature knows little about it: the optimal solution is known only for $k = 2, 3, 8, 24$, with the last two proved very recently [32, 6]. Instead, we show that the straightforward greedy online algorithm is a 2-approximation, although it is not necessarily efficiently computable. Furthermore, we show the greedy algorithm can be efficiently approximated arbitrarily closely. □

6 The General k-Dimensional Offline CD Problem

By Claim 1, no online algorithm provides a good approximation for the CD problem, thus we focus on the offline problem. Interestingly, the offline CD problem is reducible to the online ATWC problem for any dimension k and polytope P. Intuitively, given an input sequence $S =$

$((s_1, d_1), \ldots, (s_n, d_n))$, we first slice the whole time interval $[0, T]$ into smaller ones by the arriving time s_i and the departure time d_i of each point $i \in [n]$. Thus the set of present points only changes at the end-points of the intervals and stays the same within an interval. Then we design an algorithm to locate the points such that, *in each time interval,* the minimum distance is a constant approximation to the optimal dispersion problem without time, for the points present in this interval.

To carry out this idea, we provide a polynomial-time algorithm $\mathcal{A}_{\mathcal{I}}$, described in Algorithm 3, to iteratively choose a subset $\mathcal{I}$ of points and eliminate the chosen points from S. In each iteration, the chosen set $\mathcal{I}$ satisfies the following properties.

Φ.1 $\mathcal{I}$ can be partitioned into two groups $\mathcal{I}_1$ and $\mathcal{I}_2$ such that, for any two points in the same group, they do not have any overlap along time.

Φ.2 For any time $0 \le t \le T$, if there is still some point in S present at time t, then at least one of them is selected to $\mathcal{I}$.

Recall that m is the maximum number of points simultaneously present at any time. By Φ.2, this procedure ends in m iterations. Iteratively finding satisfiable subsets and allocating them into the positions given by the online algorithm provides a constant approximation. More specifically, we have the following theorem, proved in Appendix E.

Algorithm 3. Algorithm $\mathcal{A}_{\mathcal{I}}$ for computing $\mathcal{I}$ satisfying properties Φ.1 and Φ.2.

Input: A sequence $S = ((s_1, d_1), \ldots, (s_n, d_n))$.

Let $\mathcal{I}_1 = \mathcal{I}_2 = \emptyset$, $s = -1$, $d = 0$ and $T = \max_{i \in [n]} d_i$.
(s and d are the end-points of a "sliding window" for the arriving times under consideration.)
Let $index = 1$.
while d $\neq T$ **do**
  Let $\hat{S} = \{i | i \in S, s_i > s, s_i \le d, d_i > d\}$.
  if $\hat{S} \neq \emptyset$ **then**
    Arbitrarily choose $j \in \arg\max_{i \in \hat{S}} d_i$ and add j to $\mathcal{I}_{index}$.
    $s = d$, then $d = d_j$.
  else
    $s = d$, then $d = \min_{i \in S, s_i > d} s_i$.
  end if
  $index = 3 - index$.
end while
Output $\{\mathcal{I}_1, \mathcal{I}_2\}$.

Theorem 8. *For any $k \ge 1$ and k-dimensional polytope P, given any polynomial-time online algorithm $\mathcal{A}_{ATWC}$ for the ATWC problem with approximation ratio σ, there is a polynomial-time offline algorithm for the CD problem with approximation ratio $\sigma \max_{i \ge 1} \frac{Disp(i;P)}{Disp(2i;P)} \le 2\sigma$, using $\mathcal{A}_{ATWC}$ as a black-box.*

Proof sketch. We first iteratively use Algorithm 3 to find the desired subsets of points. For each iteration, let $\{\mathcal{I}_1, \mathcal{I}_2\}$ be the output of Algorithm 3. By locating all the points in $\mathcal{I}_1$ and $\mathcal{I}_2$ to the next two positions output by $\mathcal{A}_{ATWC}$, one for each group, the points will not overlap by property Φ.1. We show that the approximation ratio *in each time interval* is bounded by $\sigma \max_{i \ge 1} \frac{Disp(i;P)}{Disp(2i;P)}$, and it is easy to see that $\max_{i \ge 1} \frac{Disp(i;P)}{Disp(2i;P)} \le 2$ for arbitrary polytopes. □

Acknowledgement

The authors thank Joseph Mitchell for motivating us to study this problem. We thank Esther Arkin, Michael Bender, Rezaul A. Chowdhury, Jie Gao, Joseph Mitchell, Jelani Nelson, and the participants of the Algorithm Reading Group for helpful discussions, and several anonymous reviewers for their comments. This work is partially supported by NSF CAREER Award No. 1553385.

A Proofs for Section 2

Claim 1. (restated) *For the CD problem, even when $k = 1$ and P is the unit segment, no (randomized) online algorithm provides an approximation to OPT_C better than $\Omega(n)$.*

Proof. Consider n points with $s_i = 0$ for all i and let $X_1, \ldots, X_n$ be the locations chosen by an online algorithm at time 0. If there exist two points i, j with $dis(X_i, X_j) = d_{min}(0)$, then the adversary sets the departure time of i and j to be a large number T, and that of all the other points to be 1. Otherwise, there exists a point i with $dis(X_i, \partial P) = d_{min}(0)$, and the adversary sets $d_i = T$ and $d_j = 1$ for all $j \neq i$. We only analyze the first case as the second is almost the same. In the algorithm, $d_{min}(t) = d_{min}(0) \leq \frac{1}{n+1}$ for any t, and $\int_0^T d_{min}(t)dt \leq \frac{T}{n+1}$. However, by putting i and j at 1/3 and 2/3 respectively, we have $OPT_C(S; P) > \frac{T-1}{3}$. Thus the approximation ratio is $\Omega(n)$. □

Claim 2. (restated) *$\forall\ S = ((s_1, d_1), \ldots, (s_n, d_n))$, letting $m = \max_{t \leq T} |\{i : s_i \leq t \leq d_i\}|$, we have $OPT_A(S; P) = Disp(m; P)$.*

Proof. Let $X = (X_1, \ldots, X_n)$ be the optimal solution for the offline ATWC problem, and $t \in [0, T]$ be such that there exist exactly m points at time t. It is easy to see that

$$OPT_A(S; P) = \min_{t' \leq T} d_{min}(t'; X) \leq d_{min}(t; X) \leq Disp(m; P).$$

Next, let $Y = (Y_1, \ldots, Y_m)$ be the optimal solution for $Disp(m; P)$ and consider the following algorithm for the offline ATWC problem with input S: when a point arrives, arbitrarily pick a location Y_i that is not currently occupied and put it there; when a point leaves, the Y_i occupied by it becomes vacant again. Note that this is an offline algorithm because it knows m (and thus Y). Also note that this algorithm produces a valid solution, because there are at most m points simultaneously present at any time, and m locations are sufficient. Abusing notation sightly and letting $d_{min}(t; Y)$ be the minimum distance produced by the algorithm at time t, we have $d_{min}(t; Y) \geq Disp(m; P)$ for all $t \leq T$, thus

$$Disp(m; P) \leq OPT_A(S; P).$$

Therefore Claim 2 holds. □

Claim 3. (restated) *For any $k \geq 1$, $x > 0$ and any k-dimensional polytope P with an insphere, letting x be the radius of the insphere, we have $Disp(n; P) = \frac{2xDP(n;P)}{x+DP(n;P)}$.*

Proof. Let c be the center of the insphere. On the one hand, given the optimal locations $X_1, \ldots, X_n$ corresponding to $Disp(n; P)$, let P' be the polytope obtained from P by moving each facet towards c for distance $\frac{Disp(n;P)}{2}$. As the distance from c to each facet of P is exactly x, P' can also be obtained by shrinking P by a factor of $\lambda = \frac{x}{x - \frac{Disp(n;P)}{2}}$ with respect to the origin at c. Consider the packing of n balls with radius $\frac{Disp(n;P)}{2}$ in P', centered at the X_i's. Indeed, as the distance of each X_i to the facets of P is at least $Disp(n; P)$, its distance to the facets of P' is at least $\frac{Disp(n;P)}{2}$ and the n balls are contained wholly in P'. Moreover, as the distance of any two locations X_i and X_j is at least $Disp(n; P)$, the n balls are not overlapping with each other. By scaling P' up by a factor of λ

with respect to c, we get a packing of n balls in P with radius $r = \frac{\lambda Disp(n;P)}{2} = \frac{xDisp(n;P)}{2x-Disp(n;P)}$. Thus $Pack(r;P) \geq n$. Accordingly, $DP(n;P) \geq r$ by definition, which implies

$$Disp(n;P) \leq \frac{2xDP(n;P)}{x+DP(n;P)}.$$

On the other hand, given the optimal solution for $DP(n;P)$, with the balls centered at $Y_1, \ldots, Y_n$, let P'' be the polytope obtained from P by moving each facet away from c for distance $DP(n;P)$. It is easy to see that $Y_1, \ldots, Y_n$ is a dispersion in P'' with distance $2DP(n;P)$. Again because the distance from c to each facet of P is exactly x, P'' can be obtained by scaling P up by a factor of $\lambda' = \frac{x+DP(n;P)}{x}$ with respect to c. By scaling P'' down by a factor of λ', we obtain a dispersion in P with distance $d = \frac{2DP(n;P)}{\lambda'} = \frac{2xDP(n;P)}{x+DP(n;P)}$. Therefore

$$Disp(n;P) \geq \frac{2xDP(n;P)}{x+DP(n;P)},$$

and Claim 3 holds. □

Claim 4. (restated) *For any $k \geq 1$ and any k-dimensional convex polytope P with an insphere, given the radius of the insphere,*

(1) any polynomial-time algorithm for $Disp(n;P)$ implies such an algorithm for $Pack(r;P)$, and

(2) any polynomial-time algorithm for $Pack(r;P)$ implies an FPTAS for $Disp(n;P)$.

Proof. Roughly speaking, given an algorithm for one of the two problems, we can use binary search to find a solution for the other. Without loss of generality, assume P has volume 1.

Let x be the radius of the insphere. For the first part of the claim, let $Ball(r)$ be the volume of the k-dimensional ball with radius r and $N = \lfloor \frac{1}{Ball(r)} \rfloor$. Clearly, $Pack(r;P) \leq N$. The binary search over the set $\{0, 1, \ldots, N\}$ works as follows. In each round, find the median of the current set, denoted by n; compute $d_n = Disp(n;P)$ using the polynomial-time algorithm and $r_n = \frac{d_n x}{2x-d_n}$. Note $r_n = DP(n;P)$ by Claim 3. If $r_n < r$ then continue searching from $n-1$ and below—that is, one cannot pack n balls of radius r in P; and if $r_n \geq r$ then continue searching from n and above—that is, one can pack at least n balls of radius r in P. When n is the only number left, output it. The correctness of the algorithm follows from the fact that $Disp(n;P)$ and thus $DP(n;P)$ are non-increasing in n. Accordingly, there exists a unique n^* such that $r_n \geq r$ for all $n \leq n^*$ and $r_n < r$ for all $n > n^*$. It is easy to see that $Pack(r;P) = n^*$.

For the second part, using x as an upper bound for $Disp(n;P)$, the idea is almost the same. The only difference is that $Disp(n;P)$ may not have finite representations and the algorithm has to stop when the length of the interval is no larger than some small constant ϵ, leading to an FPTAS rather than an exact solution. □

Note that finding the radius of the insphere in an irregular convex polytope is a non-trivial computation problem. Thus we require it is given as an input. Finding the radius is easy for regular polytopes.

B Proofs for Section 3

B.1 Proof for Theorem 1

Theorem 1. (restated) *No algorithm achieves an approximation ratio better than* $2\ln 2$ *(*≈ 1.386*) for the 1-dimensional online ATWC problem.*

Proof. For any positive integer r, let $\sigma'_r = \sum_{i=r+1}^{2r} \frac{1}{i}$. We show that no algorithm achieves an approximation ratio better than $2\sigma'_r$. For the sake of contradiction, assume there exists an r and an online algorithm $\mathcal{A}$ with approximation ratio $\alpha < 2\sigma'_r$. We construct an instance of the online ATWC problem with three stages. In the first stage, $r-1$ points arrive simultaneously; in the second stage, r new points arrive one by one; and then all $2r-1$ points depart simultaneously.

Since no point departs before the last point arrives, by Claim 2 we have $OPT_A(S_i; P) = Disp(i; P)$ for any point i, where S_i is the instance containing only the first i points. Since the online algorithm is an α-approximation to OPT_A, it must ensure that for each point i, after its arrival, its distance to all the other present points is at least $\frac{OPT_A(S_i;P)}{\alpha} = \frac{Disp(i;P)}{\alpha}$: otherwise the adversary simply stops adding new points and the approximation ratio is violated for the instance S_i.

Nevertheless, we show that after the second stage, the claimed approximation ratio must be violated. To do so, note that $Disp(i; P) = \frac{1}{i+1}$ for any point i, because the optimal dispersion without time is to locate the points evenly on the segment, resulting in $i+1$ equal-length intervals. Denote by $Q = \{q_\ell | 1 \le \ell \le r-1\}$ the positions of the first $r-1$ points given by the algorithm, and let $q_0 = 0$ and $q_r = 1$. We claim that, in the second stage, no two points can be put into the same interval generated by Q. Assume otherwise, and assume exactly two points i and j are put into the same interval, with $r \le i < j \le 2r-1$. There must exist another interval $(q_\ell, q_{\ell+1})$ with $\ell \ge 0$, which does not contain any new point from stage two. Any other interval contains exactly one new point. Thus the total length of all the intervals generated by Q, denoted by $\mathcal{L}$, is

$$\mathcal{L} > (q_{\ell+1} - q_\ell) + \frac{Disp(i;P)}{2\sigma'_r} + 2 \times \frac{Disp(j;P)}{2\sigma'_r} + \sum_{r \le h \le 2r-1,\ h \notin \{i,j\}} 2 \times \frac{Disp(h;P)}{2\sigma'_r}. \tag{1}$$

To see why Equation 1 is true, first note that $q_{\ell+1} - q_\ell$ is the length of the interval which does not contain any new point. Second, the length of the interval split by i and j must be larger than

$$\frac{Disp(i;P)}{2\sigma'_r} + 2 \times \frac{Disp(j;P)}{2\sigma'_r}.$$

Indeed, upon arrival, point i first splits this interval into two smaller intervals, and one of them is further split by j; thus the sub-interval between i and the adjacent end-point of the interval is at least $\frac{Disp(i;P)}{\alpha} > \frac{Disp(i;P)}{2\sigma'_r}$, and each of the two sub-intervals created by j is at least $\frac{Disp(j;P)}{\alpha} > \frac{Disp(j;P)}{2\sigma'_r}$, due to the algorithm's claimed approximation ratio. Moreover, for each point $h \in \{r, r+1, \ldots, 2r-1\} \setminus \{i, j\}$, it splits one of the remaining intervals and each of the two sub-intervals is at least $\frac{Disp(h;P)}{\alpha} > \frac{Disp(h;P)}{2\sigma'_r}$.

Again because algorithm $\mathcal{A}$'s approximation ratio is $\alpha < 2\sigma'_r$, we have

$$q_{\ell+1} - q_\ell \ge \frac{Disp(r-1;P)}{\alpha} = \frac{1}{\alpha r} > \frac{1}{2\sigma'_r r} > \frac{1}{2\sigma'_r(i+1)} = \frac{Disp(i;P)}{2\sigma'_r}. \tag{2}$$

Combining Equations 1 and 2, we have

$$\mathcal{L} > \sum_{r \le h \le 2r-1} 2 \times \frac{Disp(h;P)}{2\sigma'_r} = \frac{1}{\sigma'_r} \sum_{r \le h \le 2r-1} \frac{1}{h+1} = \frac{1}{\sigma'_r} \sum_{r+1 \le h \le 2r} \frac{1}{h} = 1. \tag{3}$$

That is, the total length is larger than 1, a contradiction.

In general, having more than two points in the same interval and having more than one interval containing at least two points lead to the same contradiction. More specifically, for all new points i that contributes one copy of $\frac{Disp(i;P)}{2\sigma'_r}$ in the lower bound of $\mathcal{L}$ as in Equation 1, there exists exactly the same number of intervals that are not split by any new point. Thus we can arbitrarily fix a bijection between those points and intervals, such that each unsplit interval contributes another copy of $\frac{Disp(i;P)}{2\sigma'_r}$ for its corresponding point i as in Equation 2, and Equation 3 holds again. Accordingly, we conclude that each interval generated by Q is split by exactly one new point in stage two.

However, in this case, by a similar argument, the total length of these intervals is

$$\mathcal{L} > \sum_{r \le h \le 2r-1} 2 \times \frac{Disp(h;P)}{2\sigma'_r} = \frac{1}{\sigma'_r} \sum_{r \le h \le 2r-1} \frac{1}{h+1} = \frac{1}{\sigma'_r} \sum_{r+1 \le h \le 2r} \frac{1}{h} = 1,$$

the same contradiction. Therefore such an algorithm $\mathcal{A}$ does not exist.

In sum, for any integer $r > 0$, no algorithm achieves an approximation ratio better than $2\sigma'_r$. Since σ'_r is strictly increasing in r and $\lim_{r\to\infty} 2\sigma'_r = 2\ln 2$, no algorithm achieves an approximation ratio better than $2\ln 2$ and Theorem 1 holds. □

B.2 Proof for Lemma 1

Lemma 1. (restated) *Given any r and τ, Algorithm 1 is a $2\sigma_r$-approximation to OPT_A for any $S = ((s_1, d_1), \ldots, (s_n, d_n))$ with $m > r$, where m is maximum number of points simultaneously present at some time t.*

Proof. Note that for each arriving point Algorithm 1 first checks the existing positions in $\hat{Q}$. Only when there is no *vacant* position will it create a new one, with the pre-fixed positions in Q having the highest priority. In particular, whenever a new position is added to $\hat{Q}$ at time $\hat{t}$, $\min_{t<\hat{t}} d_{min}(t) = d_{min}(\hat{Q})$, where $d_{min}(\hat{Q}) \triangleq \min_{q,q' \in \hat{Q}} \{dis(q, \partial P), dis(q, q')\}$ is the minimum distance incurred by positions in $\hat{Q}$. Accordingly, the minimum distance through out Algorithm 1 happens when there are m points on the line. Note $OPT_A(S;P) = Disp(m;P) = \frac{1}{m+1}$ by Claim 2. Thus, in order to lower bound the approximation ratio of the algorithm, it suffices to consider the case where each point i arrives at time i and all points depart at the same time $m+1$— that is, $S = ((1, m+1), (2, m+1), \ldots, (m, m+1))$. By the hypothesis of Lemma 1, $m > r$.

By the construction of the algorithm, upto time r, only the positions in Q may be used for the points. At time r, $\hat{Q} = Q$, all r positions in Q are *occupied* and the segment is sliced into $r+1$ intervals. After that, at time $i > r$, the arriving point i will split the current largest interval into two equal sub-intervals and create a new position in $\hat{Q}$. In fact, since the lengths of the intervals created by Q strictly decrease from left to right, the position creation procedure can be described by "rounds" as follows. Round 0 is $\tau_1, \ldots, \tau_r$ (from time 1 to time r); round 1 splits existing intervals one by one into halves, from the leftmost to the rightmost (from time $r+1$ to time $2(r+1)$); round 2 again splits existing intervals from the leftmost to the rightmost (from time $2(r+1)+1$ to time $4(r+1)$); so on and so forth. In particular, by the end of each round l, the number of sub-intervals sliced by $\hat{Q}$ is $2^l(r+1)$ and the number of points is $|\hat{Q}| = 2^l(r+1) - 1$. Also, for each interval (q_{i-1}, q_i) with $i \in [r+1]$, it has been sliced into 2^l sub-intervals after round l. Moreover, because the maximum interval (q_0, q_1) is less than twice of the minimum interval (q_r, q_{r+1}), whenever a point $i > r$ is added and $i \in (q_{j-1}, q_j)$ for some j, the resulting minimum distance occurs between i and its right neighbor in $\hat{Q}$ (and is also the length of all sub-intervals in (q_{j-1}, q_j) before i).

Let point m appear during round $(l+1)$ for some $l \geq 0$, then there exits $1 \leq i \leq r+1$ such that

$$2^l(r+1) + 2^l(i-1) \leq m < 2^l(r+1) + 2^l i.$$

That is, m appears in the interval (q_{i-1}, q_i). Accordingly, after point m is located, the minimum distance equals the length of every sub-interval in (q_{i-1}, q_i) before point m, as well as the sub-interval immediately after it. And this length is

$$d_{min}(m) = \frac{1}{2^{l+1}\sigma_r(r+i)}.$$

Therefore the approximation ratio is

$$\frac{OPT_A(S;P)}{d_{min}(m)} = \frac{\frac{1}{m+1}}{\frac{1}{2^{l+1}\sigma_r(r+i)}} = \frac{2^{l+1}\sigma_r(r+i)}{m+1} \leq \frac{2^{l+1}\sigma_r(r+i)}{2^l(r+1)+2^l(i-1)+1} = \frac{2^{l+1}\sigma_r(r+i)}{2^l(r+i)+1} < 2\sigma_r,$$

and Lemma 1 holds. □

B.3 Proof for Lemma 2

Lemma 2. (restated) *For any positive integer l and $r = 2^l - 1$, there exists an ordering τ for the corresponding set Q, such that Algorithm 1 is a $2\sigma_r$-approximation to OPT_A for any $S = ((s_1, d_1), \ldots, (s_n, d_n))$ with $m \leq r$.*

Proof. As shown in Algorithm 1, for each arriving point, when $m \leq r$, it always assigns the point to the existing positions in $\hat{Q}$, and when there is no vacant position, it creates a new one at the $(|\hat{Q}|+1)$st position in τ. Thus, it is again sufficient to consider all the instances $S = ((1, m+1), (2, m+1), \ldots, (m, m+1))$ with $m \leq r$: whenever a new position is created, the minimum distance so far is incurred by $\hat{Q}$, and the size of $\hat{Q}$ only increases. In other words, we can focus on the instance $S = ((1, r+1), (2, r+1), \ldots, (r, r+1))$ and prove that the approximation ratio at any time $d \in [r]$ is smaller than $2\sigma_r$.

Below we construct the desired ordering τ for Q by filling in a complete binary tree with r nodes. We do so in l rounds, with round $j \in \{0, \ldots, l-1\}$ filling in level j of the tree and level 0 being the root. In round 0, letting the *left end-point* be 0 and the *right end-point* be 2^l (corresponding to $q_0 = 0$ and $q_{2^l} = 1$), fill the root with the average of the two, namely, 2^{l-1}. In each round $j > 0$, process the nodes in level j from left to right. For each node x, letting its two neighbors in the current tree be filled with x_{left} and x_{right}, fill node x with $\frac{x_{left}+x_{right}}{2}$. If node x is the leftmost (respectively, rightmost) node in the current tree, then take $x_{left} = 0$ (respectively, $x_{right} = 2^l$). After the whole tree being filled, τ is obtained by traversing it in a breadth-first manner starting from the root: letting the j-th node visited being filled with x_j, then $\tau_j = q_{x_j}$. Figure 2 illustrates the structure of the tree and the ordering τ, for general l and for $l = 3$. We refer to the resulting τ as the *binary ordering* and we have the following claim.

Claim 5. *For any positive integer l and $r = 2^l - 1$, for any $d \in [r]$, writing $d = 2^i + s$ with $i \in \{0, 1, \ldots, l-1\}$ and $s \in \{0, 1, \ldots, 2^i - 1\}$, then the binary ordering τ is such that*

$$\tau_d = q_{2^{l-i-1}(2s+1)},$$

and the minimum distance according to Algorithm 1 with parameter r and τ at time d, $d_{min}(d)$, is

$$d_{min}(d) = \frac{1}{\sigma_r} \sum_{j=2^l+2^{l-i-1}(2s+1)}^{2^l-1+2^{l-i-1}(2s+2)} \frac{1}{j}.$$

We prove Claim 5 after the proof of Lemma 2. Note that $OPT_A(S_d; P) = Disp(d; P) = \frac{1}{d+1} = \frac{1}{2^i+s+1}$ for $d = 2^i + s$, where S_d contains only the first d points in S. Thus, by Claim 5 the approximation ratio at time d, $apx(d)$, is

$$apx(d) = \frac{OPT_A(S_d; P)}{d_{min}(d)} = \frac{\sigma_r}{(2^i + s + 1) \cdot \sum_{j=2^l+2^{l-i-1}(2s+1)}^{2^l-1+2^{l-i-1}(2s+2)} \frac{1}{j}}.$$

To show $apx(d) < 2\sigma_r$, we lower bound the denominator in two steps, by the following two claims, which are also proved after the proof of Lemma 2.

Claim 6. *Arbitrarily fixing $i \in \{0, 1, \ldots, l-1\}$ and letting*

$$f(s) = (2^i + s + 1) \cdot \sum_{j=2^l+2^{l-i-1}(2s+1)}^{2^l-1+2^{l-i-1}(2s+2)} \frac{1}{j}$$

for all $s \in \{0, 1, \ldots, 2^i - 1\}$, we have that $f(s)$ is strictly decreasing in s.

By Claim 6, $apx(d) = \frac{\sigma_r}{f(s;i)} \le \frac{\sigma_r}{f(2^i-1;i)} = \frac{\sigma_r}{(2^i+2^i)\cdot\sum_{j=2^l+2^{l-i-1}(2^{i+1}-1)}^{2^l-1+2^{l-i-1}2^{i+1}} \frac{1}{j}} = \frac{\sigma_r}{2^{i+1}\cdot\sum_{j=2^{l+1}-2^{l-i-1}}^{2^{l+1}-1} \frac{1}{j}}$.

Claim 7. *Letting*

$$g(i) = 2^{i+1} \cdot \sum_{j=2^{l+1}-2^{l-i-1}}^{2^{l+1}-1} \frac{1}{j}$$

for all $i \in \{0, 1, \cdots, l-1\}$, we have that $g(i)$ is strictly decreasing in i.

By Claim 7, for any $d \in [r]$, $apx(d) \le \frac{\sigma_r}{g(l-1)} = \frac{\sigma_r}{2^l\cdot\sum_{j=2^{l+1}-1}^{2^{l+1}-1} \frac{1}{j}} = \frac{\sigma_r}{2^l/(2^{l+1}-1)} = \sigma_r(2 - \frac{1}{2^l}) < 2\sigma_r$, and Lemma 2 holds. □

Remark. Note the denominator of $apx(d)$ may not be decreasing in d, and the jump may happen from $d = 2^i + (2^i - 1)$ to 2^{i+1}. We bypass this problem by breaking the analysis into two steps, as above.

We now prove the three claims.

Proof of Claim 5. We first provide some simple facts about τ and Algorithm 1. Since the algorithm adds positions to $\hat{Q}$ according to τ, the whole procedure can also be considered as l rounds, same as the construction of the binary tree. By induction, for each $i = 0, 1, \ldots, l-1$, the number of positions added to $\hat{Q}$ in round i is 2^i (i.e., the number of nodes in level i of the tree), and the number of intervals created by $\hat{Q}$ by the end of round i is 2^{i+1}. We denote these intervals by $I_0^i, I_1^i, \ldots, I_{2^{i+1}-1}^i$ from left to right, and refer to them as the *round-i intervals*. Moreover, referring to the intervals (q_{j-1}, q_j) with $j \in [r+1]$ as the *pre-fixed intervals*, we have that each round-i interval contains 2^{l-i-1} pre-fixed intervals: by construction, each position added in round i split the corresponding round-$(i-1)$ interval into two sub-intervals, not with the same length but with the same number of pre-fixed intervals, thus all round-i intervals contain the same number of pre-fixed intervals.

Next, it is easy to see that the points that arrive in round i are points $2^i+0, 2^i+1, \ldots, 2^i+(2^i-1)$. Thus point $d = 2^i + s$ arrives in round i and the corresponding position τ_d is in the round-$(i-1)$ interval I_s^{i-1}. Accordingly, there are $(2s+1)$ round-i intervals to the left of τ_d, corresponding to a total of $2^{l-i-1}(2s+1)$ pre-fixed intervals. That is, $\tau_d = q_{2^{l-i-1}(2s+1)}$ as we wanted to show.

Below we compute the minimum distance of the algorithm at time d. Note that, after point d is located, the intervals incurred by $\hat{Q}$ are $I_0^i, I_1^i, \ldots, I_{2s+1}^i, I_{2s+2}^i, I_{s+1}^{i-1}, \ldots, I_{2^i-1}^{i-1}$. By induction, we have that

- the lengths of $I_0^i, I_1^i, \ldots, I_{2s+2}^i$ are strictly decreasing,
- the lengths of $I_{s+1}^{i-1}, \ldots, I_{2^i-1}^{i-1}$ are also strictly decreasing,
- $|I_s^{i-1}| = |I_{2s+1}^i| + |I_{2s+2}^i|$, and
- $|I_{2^i-1}^{i-1}| < |I_s^{i-1}| < 2|I_{2^i-1}^{i-1}|$,

where the last inequality is because, for any two pre-fixed intervals (q_{j-1}, q_j) and $(q_{j'-1}, q_{j'})$ with $j < j'$, we have $|(q_{j'-1}, q_{j'})| < |(q_{j-1}, q_j)| < 2|(q_{j'-1}, q_{j'})|$. Accordingly, $|I_{2^i-1}^{i-1}| > \frac{|I_s^{i-1}|}{2} > |I_{2s+2}^i|$, and the minimum distance at time d is $d_{min}(d) = |I_{2s+2}^i|$. As the left end-point of I_{2s+2}^i is τ_d and the right end-point is $q_{2^{l-i-1}(2s+2)}$, we have

$$\begin{aligned} d_{min}(d) &= q_{2^{l-i-1}(2s+2)} - q_{2^{l-i-1}(2s+1)} \\ &= \left(\frac{1}{\sigma_r} \sum_{j=2^l}^{2^l-1+2^{l-i-1}(2s+2)} \frac{1}{j}\right) - \left(\frac{1}{\sigma_r} \sum_{j=2^l}^{2^l-1+2^{l-i-1}(2s+1)} \frac{1}{j}\right) = \frac{1}{\sigma_r} \sum_{j=2^l+2^{l-i-1}(2s+1)}^{2^l-1+2^{l-i-1}(2s+2)} \frac{1}{j}, \end{aligned}$$

and Claim 5 holds. □

Proof of Claim 6. For any $s < 2^i - 1$, we have

$$\begin{aligned} f(s+1) &= (2^i+s+2) \cdot \sum_{j=2^l+2^{l-i-1}(2s+3)}^{2^l-1+2^{l-i-1}(2s+4)} \frac{1}{j} \\ &= (2^i+s+2) \cdot \sum_{j=2^l+2^{l-i-1}(2s+1)+2^{l-i}}^{2^l-1+2^{l-i-1}(2s+2)+2^{l-i}} \frac{1}{j} \\ &= (2^i+s+2) \cdot \sum_{j=2^l+2^{l-i-1}(2s+1)}^{2^l-1+2^{l-i-1}(2s+2)} \frac{1}{j+2^{l-i}}. \end{aligned}$$

It suffices to show that for all $s < 2^i - 1$, $f(s+1) - f(s) < 0$: that is,

$$f(s+1) - f(s) = \sum_{j=2^l+2^{l-i-1}(2s+1)}^{2^l-1+2^{l-i-1}(2s+2)} \left(\frac{2^i+s+2}{j+2^{l-i}} - \frac{2^i+s+1}{j}\right) < 0. \tag{4}$$

Below we show that for each j in the range of the summation,

$$\frac{2^i+s+2}{j+2^{l-i}} - \frac{2^i+s+1}{j} < 0,$$

which is equivalent to $(2^i + s + 2)j < (j + 2^{l-i})(2^i + s + 1)$, or

$$j < 2^{l-i}(2^i + s + 1) = 2^l + 2^{l-i}(s+1) = 2^l + 2^{l-i-1}(2s+2).$$

However, notice that the maximum value of j in Equation 4 is $2^l - 1 + 2^{l-i-1}(2s+2)$, thus all the inequalities above hold immediately. Accordingly, Claim 6 holds. □

Proof of Claim 7. By definition,

$$\begin{aligned}
g(i) &= 2^{i+1} \cdot \sum_{j=2^{l+1}-2^{l-i-1}}^{2^{l+1}-1} \frac{1}{j} \\
&= 2^{i+1} \cdot \left(\sum_{j=2^{l+1}-2^{l-i-1}}^{2^{l+1}-2^{l-i-2}-1} \frac{1}{j} + \sum_{j=2^{l+1}-2^{l-i-2}}^{2^{l+1}-1} \frac{1}{j} \right) \\
&= 2^{i+1} \cdot \left(\sum_{j=2^{l+1}-2^{l-i-2}}^{2^{l+1}-1} \frac{1}{j - 2^{l-i-2}} + \sum_{j=2^{l+1}-2^{l-i-2}}^{2^{l+1}-1} \frac{1}{j} \right) \\
&> 2^{i+1} \cdot \sum_{j=2^{l+1}-2^{l-i-2}}^{2^{l+1}-1} \frac{2}{j} = 2^{i+2} \cdot \sum_{j=2^{l+1}-2^{l-i-2}}^{2^{l+1}-1} \frac{1}{j} = g(i+1),
\end{aligned}$$

where the inequality is because $\frac{1}{j-2^{l-i-2}} > \frac{1}{j}$ for all j in the range of the summation. Therefore Claim 7 holds. □

B.4 Proof for Theorem 2

Theorem 2. (restated) *There exists a deterministic polynomial-time online algorithm for the ATWC problem, whose approximation ratio can be arbitrarily close to* $2\ln 2$. *Moreover, the running time is polynomial in* $\frac{1}{\epsilon}$ *for approximation ratio* $2\ln 2 + \epsilon$.

Proof. Note that σ_r is strictly decreasing in r and $\lim_{l\to\infty} 2\sigma_{2^l-1} = 2\ln 2$. Thus for any small constant $\epsilon > 0$, there exists l such that $2\sigma_{2^l-1} < 2\ln 2 + \epsilon$. In particular, it suffices to take $l = \lceil \log_2(\frac{2}{\epsilon} + 1) - 1 \rceil$. The desired online algorithm first computes l and $r = 2^l - 1$ ($\leq \frac{2}{\epsilon}$), then computes τ and runs Algorithm 1 with r and τ on any online instance.

Since the selection of l only depends on ϵ and not on the input sequence, l is a constant, and so is r. Given l and r, the binary ordering τ can be constructed in time $O(r) = O(\frac{2}{\epsilon})$. Given r and τ, when a point arrives or departs, the running time of Algorithm 1 is polynomial in $|\hat{Q}|$, thus polynomial in the size of the input so far. Accordingly, Theorem 2 holds. □

B.5 Proof for Theorem 3

Theorem 3. (restated) *For any integer* $d = 2^i + s$ *with* $i \geq 0$ *and* $0 \leq s \leq 2^i - 1$, *let* $\tau_d = \frac{1}{\ln 2}\ln(1 + \frac{2s+1}{2^{i+1}}) = \log_2(1 + \frac{2s+1}{2^{i+1}})$. *If Algorithm 1 creates the d-th new position in* $\hat{Q}$ *to be* τ_d, *the approximation ratio is exactly* $2\ln 2$.

Proof. Similar to the proof of Lemma 1, the worst case happens when each point arrives at different time and all points depart at the same time. Arbitrarily fixing $d \geq 1$, we consider the minimum distance incurred by $\{\tau_1, \tau_2, \ldots, \tau_d\}$ (at time d), $d_{min}(d)$, and the approximation ratio at time d,

$$apx(d) = \frac{1}{(d+1)d_{min}(d)}.$$

Similar to the proof of Claim 5, as d grows, $\frac{2s+1}{2^{i+1}}$ takes values $\frac{1}{2}, \frac{1}{4}, \frac{3}{4}, \frac{1}{8}, \frac{3}{8}, \frac{5}{8}, \frac{7}{8}, \ldots$ sequentially, following the breadth-first search on the infinite complete binary tree. Therefore, when point d is located, the positions adjacent to $\tau_d = \frac{1}{\ln 2}\ln(1 + \frac{2s+1}{2^{i+1}})$ are $d_{left} = \frac{1}{\ln 2}\ln(1 + \frac{2s}{2^{i+1}})$ and $d_{right} = \frac{1}{\ln 2}\ln(1 + \frac{2s+2}{2^{i+1}})$, with the minimum distance incurred by τ_d being either $d_{right} - \tau_d$ or $\tau_d - d_{left}$. Note that, if $s = 0$ then $d_{left} = 0$, which is the left end-point of the segment; if $s = 2^i - 1$ then $d_{right} = 1$, which is the right end-point of the segment. Moreover, if $1 \leq s \leq 2^i - 2$, then $\frac{2s}{2^{i+1}}$ and $\frac{2s+2}{2^{i+1}}$ can each be written into the form of $\frac{2s'+1}{2^{j+1}}$ with $j \geq 0$ and $0 \leq s' \leq 2^j - 1$, by taking out the greatest common diviser of the denominator and the enumerator. Thus each of the two positions, d_{left} and d_{right}, corresponds to some position $\tau_{d'}$ where $d' = 2^j + s' < d$. Since

$$\begin{aligned} d_{right} - \tau_d &= \frac{1}{\ln 2}\ln(1 + \frac{2s+2}{2^{i+1}}) - \frac{1}{\ln 2}\ln(1 + \frac{2s+1}{2^{i+1}}) = \frac{1}{\ln 2}\ln(1 + \frac{1}{2^{i+1} + 2s + 1}) \\ < \quad & \frac{1}{\ln 2}\ln(1 + \frac{1}{2^{i+1} + 2s}) = \frac{1}{\ln 2}\ln(1 + \frac{2s+1}{2^{i+1}}) - \frac{1}{\ln 2}\ln(1 + \frac{2s}{2^{i+1}}) = \tau_d - d_{left}, \end{aligned}$$

we have

$$d_{min}(d) = d_{right} - \tau_d = \frac{1}{\ln 2}\ln(1 + \frac{1}{2^{i+1} + 2s + 1}).$$

Thus

$$apx(d) \quad = \quad \frac{\ln 2}{(2^i + s + 1)\ln(1 + \frac{1}{2^{i+1}+2s+1})} < \frac{\ln 2}{(2^i + s + 1)/(2^{i+1} + 2s + 2)} = 2\ln 2,$$

where the inequality is because $\ln(1 + \frac{1}{x}) > \frac{1}{x+1}$ for any $x \geq 1$. Moreover, when $d = 2^i$ and $i \to \infty$,

$$\lim_{i \to \infty} apx(2^i) = \frac{\ln 2}{(2^i + 1)\ln(1 + \frac{1}{2^{i+1}+1})} = 2\ln 2.$$

Therefore the desired approximation ratio is $\sup_d apx(d) = 2\ln 2$ and Theorem 3 holds. □

Remark: Not only the algorithm in Theorem 3 cannot be computed in polynomial time, but it also cannot be properly approximated by just rounding each created position to a fixed precision. The reason is that the online algorithm does not know m beforehand and cannot adjust the precision according to it. When the number of simultaneously present points grows, the optimal minimum distance may be much smaller than the precision, the relative positions of the points may be completely different, and the approximation ratio may be arbitrarily bad. Instead, the polynomial-time algorithm in Algorithm 1 deals with rational numbers and can adjust the precision as the number of present points grows.

C Proofs for Section 4

C.1 Proof for Lemma 3

Lemma 3. (restated) *For any $n \geq 1$, $\frac{2}{5+\sqrt{2\sqrt{3}n}} \leq Disp(n; P) \leq \frac{2}{2+\sqrt{2\sqrt{3}n}}$, where $P = [0,1]^2$ is the unit square.*

Proof. We first prove the upper-bound. Recall that $DP(n;P)$ is optimal radius for the dispersal packing problem with n balls. By Claim 3,

$$DP(n;P) = \frac{Disp(n;P)}{2(1-Disp(n;P))}, \tag{5}$$

as the radius of the insphere in the unit square P is $x = \frac{1}{2}$. Because the disk packing density in dimension 2 is at most $\frac{\pi}{2\sqrt{3}}$ [10], we have $DP(n;P) \leq \frac{1}{\sqrt{2\sqrt{3}n}}$ by comparing the total area of the n disks packed in P with P's own area. Accordingly,

$$Disp(n;P) \leq \frac{2}{2+\sqrt{2\sqrt{3}n}}. \tag{6}$$

In fact, $DP(n;P) \to \frac{1}{\sqrt{2\sqrt{3}n}}$ as $n \to \infty$, thus Equation 6 is tight as $n \to \infty$.

Next, we prove the lower-bound using hexagonal packing in P, as illustrated in Figure 4. In particular, we would like to find a radius r^* such that, each row of the packing contains $\left\lceil \sqrt{\frac{\sqrt{3}n}{2}} \right\rceil$ disks and there are $\left\lceil \sqrt{\frac{2n}{\sqrt{3}}} \right\rceil$ rows in total. If so then we can pack n disks in P with radius r^*, which implies $DP(n;P) \geq r^*$.

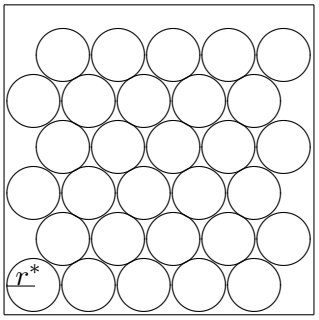

Figure 4: Hexagonal packing.

Notice that, in order to pack $\left\lceil \sqrt{\frac{\sqrt{3}n}{2}} \right\rceil$ disks in a row, the radius r^* must satisfy

$$\left(2\left\lceil \sqrt{\frac{\sqrt{3}n}{2}} \right\rceil + 1\right) r^* \leq 1;$$

in order to pack $\left\lceil \sqrt{\frac{2n}{\sqrt{3}}} \right\rceil$ rows, the radius r^* must satisfy

$$\left(\sqrt{3}\left(\left\lceil \sqrt{\frac{2n}{\sqrt{3}}} \right\rceil - 1\right) + 2\right) r^* \leq 1.$$

It is easy to verify that both inequalities are satisfied by $r^* = \frac{1}{3+\sqrt{2\sqrt{3}n}}$. Indeed,

$$\left(2\left\lceil \sqrt{\frac{\sqrt{3}n}{2}} \right\rceil + 1\right) \cdot \frac{1}{3+\sqrt{2\sqrt{3}n}} \leq \frac{2\sqrt{\frac{\sqrt{3}n}{2}}+3}{3+\sqrt{2\sqrt{3}n}} = 1$$

and

$$\left(\sqrt{3}\left(\left\lceil\sqrt{\frac{2n}{\sqrt{3}}}\right\rceil-1\right)+2\right)\cdot\frac{1}{3+\sqrt{2\sqrt{3}n}}\leq\frac{\sqrt{3}\cdot\sqrt{\frac{2n}{\sqrt{3}}}+2}{3+\sqrt{2\sqrt{3}n}}<1.$$

Accordingly, $DP(n;P)\geq\frac{1}{3+\sqrt{2\sqrt{3}n}}$ and

$$Disp(n;P)\geq\frac{2}{5+\sqrt{2\sqrt{3}n}} \tag{7}$$

by Equation 5. Thus Lemma 3 holds. □

C.2 Proof for Theorem 4

Theorem 4. (restated) *No online algorithm achieves an approximation ratio better than 1.183 for the 2-dimensional ATWC problem in a square.*

Proof. Arbitrarily fixing an online algorithm $\mathcal{A}$ and denoting its approximation ratio by σ, we show that $\sigma\geq 1.183$. To do so, arbitrarily fix a positive integer r and consider the set of positive integers

$$\Delta=\left\{\delta'\ :\ \frac{2}{5+\sqrt{2\sqrt{3}(r+\delta')}}\geq\frac{1}{2+\sqrt{2\sqrt{3}r}}\right\}.$$

Solving the inequality for δ', we have

$$\delta'\ \leq\ \left\lfloor\frac{(2\sqrt{2\sqrt{3}r}-1)^2}{2\sqrt{3}}-r\right\rfloor=\left\lfloor 3r-4\sqrt{\frac{r}{2\sqrt{3}}}+\frac{1}{2\sqrt{3}}\right\rfloor.$$

Letting $\delta=\max\Delta$, we have

$$\delta=\left\lfloor 3r-4\sqrt{\frac{r}{2\sqrt{3}}}+\frac{1}{2\sqrt{3}}\right\rfloor\ \text{ and }\ \Delta=\{1,\ldots,\delta\}.$$

Following Lemma 3, it is easy to see that $Disp(r+\delta;P)\geq\frac{Disp(r;P)}{2}$. We now consider the input sequence $S=((1,r+\delta+1),(2,r+\delta+1),\ldots,(r+\delta,r+\delta+1))$ and prove

$$r\pi\left(\frac{Disp(r;P)}{2\sigma}\right)^2+\sum_{i=1}^{\delta}\pi\left(\frac{Disp(r+i;P)}{\sigma}-\frac{Disp(r;P)}{2\sigma}\right)^2\leq\left(1-\frac{Disp(r;P)}{\sigma}\right)^2. \tag{8}$$

Indeed, after the first r points arrive, $d_{min}(r)\geq\frac{Disp(r;P)}{\sigma}$ by assumption, thus we can pack r disks with radius $\frac{Disp(r;P)}{2\sigma}$ in the square, centered at the r positions where the points are located. Their total area is

$$S_1=r\pi\left(\frac{Disp(r;P)}{2\sigma}\right)^2.$$

Next, for each arriving point $r+i$ with $1\leq i\leq\delta$, the distance between itself and any other point j with $1\leq j\leq r$ is at least

$$d_{min}(r+i)\geq\frac{Disp(r+i;P)}{\sigma}=\left(\frac{Disp(r+i;P)}{\sigma}-\frac{Disp(r;P)}{2\sigma}\right)+\frac{Disp(r;P)}{2\sigma}, \tag{9}$$

and the distance between itself and any other point $r+j$ with $1 \leq j < i$ is at least

$$\begin{aligned} d_{min}(r+i) &\geq \frac{Disp(r+i;P)}{\sigma} \\ &= \left(\frac{Disp(r+i;P)}{\sigma} - \frac{Disp(r;P)}{2\sigma}\right) + \frac{Disp(r;P)}{2\sigma} \\ &\geq \left(\frac{Disp(r+i;P)}{\sigma} - \frac{Disp(r;P)}{2\sigma}\right) + \left(\frac{Disp(r+j;P)}{\sigma} - \frac{Disp(r;P)}{2\sigma}\right), \end{aligned} \quad (10)$$

where the last inequality is because $Disp(n;P)$ is non-increasing in n and $Disp(r;P) \geq Disp(r+j;P)$. By the definition of δ, for any $i \leq \delta$ we have

$$\frac{Disp(r+i;P)}{\sigma} - \frac{Disp(r;P)}{2\sigma} \geq \frac{Disp(r+\delta;P)}{\sigma} - \frac{Disp(r;P)}{2\sigma} \geq 0,$$

and the left-hand side of the first inequality is a well-defined radius. Accordingly, by Equations 9 and 10, if we put a disk with radius $\frac{Disp(r+i;P)}{\sigma} - \frac{Disp(r;P)}{2\sigma}$ centered at the position of point $r+i$ for each $1 \leq i \leq \delta$, then the disk $r+i$ does not overlap with the first r disks whose radius is $\frac{Disp(r;P)}{2\sigma}$, neither does it overlap with any disk $r+j$ with $1 \leq j < i$, whose radius is $\frac{Disp(r+j;P)}{\sigma} - \frac{Disp(r;P)}{2\sigma}$. Moreover, since the radius of disk $r+i$ is at most $d_{min}(r+i)$, it does not overlap with the boundary of P either. By induction, all $r+\delta$ disks do not overlap with each other or with the boundary of P, and they are a *non-uniform packing* in P. The total area of all the disks $r+i$ with $1 \leq i \leq \delta$ is

$$S_2 = \sum_{i=1}^{\delta} \pi \left(\frac{Disp(r+i;P)}{\sigma} - \frac{Disp(r;P)}{2\sigma}\right)^2.$$

Finally, since

$$d_{min}(r) - \frac{Disp(r;P)}{2\sigma} \geq \frac{Disp(r;P)}{2\sigma}$$

and

$$d_{min}(r+i) - \left(\frac{Disp(r+i;P)}{\sigma} - \frac{Disp(r;P)}{2\sigma}\right) \geq \frac{Disp(r;P)}{2\sigma} \quad \forall 1 \leq i \leq \delta,$$

these $r+\delta$ disks are actually packed within the square $[\frac{Disp(r;P)}{2\sigma}, 1 - \frac{Disp(r;P)}{2\sigma}]^2$. Thus

$$S_1 + S_2 \leq \left(1 - \frac{Disp(r;P)}{\sigma}\right)^2$$

and Inequality 8 holds.

Replacing $Disp(r+i;P)$ and $Disp(r;P)$ in the left-hand side of Inequality 8 by proper lower-

and upper-bounds from Lemma 3 and moving σ to the right-hand side of the inequality, we have

$$\begin{aligned}
& r\pi\left(\frac{1}{5+\sqrt{2\sqrt{3}r}}\right)^2+\sum_{i=1}^{\delta}\pi\left(\frac{2}{5+\sqrt{2\sqrt{3}(r+i)}}-\frac{1}{2+\sqrt{2\sqrt{3}r}}\right)^2 \\
= \; & r\pi\left(\frac{1}{5+\sqrt{2\sqrt{3}r}}\right)^2+\sum_{i=1}^{\delta}\pi\left(\frac{2}{5+\sqrt{2\sqrt{3}(r+i)}}\right)^2+\sum_{i=1}^{\delta}\pi\left(\frac{1}{2+\sqrt{2\sqrt{3}r}}\right)^2 \\
& -\sum_{i=1}^{\delta}\frac{4\pi}{\left(5+\sqrt{2\sqrt{3}(r+i)}\right)\left(2+\sqrt{2\sqrt{3}r}\right)} \\
\leq \; & \sigma^2\left(1-\frac{Disp(r;P)}{\sigma}\right)^2.
\end{aligned}$$

Replacing δ with $\left\lfloor 3r-4\sqrt{\frac{r}{2\sqrt{3}}}+\frac{1}{2\sqrt{3}}\right\rfloor$ and letting $r\to\infty$, we calculate the limits term by term as follows.

$$\begin{aligned}
& \lim_{r\to\infty} r\pi\left(\frac{1}{5+\sqrt{2\sqrt{3}r}}\right)^2=\frac{\pi}{2\sqrt{3}}; \\
& \lim_{r\to\infty}\sum_{i=1}^{\delta}\pi\left(\frac{2}{5+\sqrt{2\sqrt{3}(r+i)}}\right)^2\geq\lim_{r\to\infty}\int_{r+1}^{r+\delta+1}\pi\left(\frac{2}{5+\sqrt{2\sqrt{3}x}}\right)^2dx \\
& \quad=\lim_{r\to\infty}\frac{4\pi}{\sqrt{3}}\ln\left(\frac{\sqrt{2\sqrt{3}(r+\delta+1)}+5}{\sqrt{2\sqrt{3}(r+1)}+5}\right)=\lim_{r\to\infty}\frac{4\pi}{\sqrt{3}}\ln\left(\frac{\sqrt{2\sqrt{3}\cdot 4r}}{\sqrt{2\sqrt{3}r}}\right)=\frac{4\pi\ln 2}{\sqrt{3}}; \\
& \lim_{r\to\infty}\sum_{i=1}^{\delta}\pi\left(\frac{1}{2+\sqrt{2\sqrt{3}r}}\right)^2=\lim_{r\to\infty}\pi\delta\left(\frac{1}{2+\sqrt{2\sqrt{3}r}}\right)^2=\lim_{r\to\infty}\frac{3\pi r}{2\sqrt{3}r}=\frac{\sqrt{3}\pi}{2}; \\
& \lim_{r\to\infty}\sum_{i=1}^{\delta}\frac{4\pi}{\left(5+\sqrt{2\sqrt{3}(r+i)}\right)\left(2+\sqrt{2\sqrt{3}r}\right)}\leq\lim_{r\to\infty}\sum_{i=1}^{\delta}\frac{4\pi}{\sqrt{2\sqrt{3}(r+i)}\cdot\sqrt{2\sqrt{3}r}} \\
& \quad=\frac{2\pi}{\sqrt{3}}\lim_{r\to\infty}\sum_{i=1}^{\delta}\frac{1}{\sqrt{r(r+i)}}\leq\frac{2\pi}{\sqrt{3}}\lim_{r\to\infty}\int_0^{\delta}\frac{1}{\sqrt{r(r+x)}}dx=\frac{2\pi}{\sqrt{3}}\lim_{r\to\infty}\frac{2}{\sqrt{r}}(\sqrt{r+\delta}-\sqrt{r}) \\
& \quad=\frac{2\pi}{\sqrt{3}}\lim_{r\to\infty}2\left(\sqrt{1+\frac{\delta}{r}}-1\right)=\frac{4\pi}{\sqrt{3}}; \\
& \lim_{r\to\infty}\sigma^2\left(1-\frac{Disp(r;P)}{\sigma}\right)^2=\sigma^2.
\end{aligned}$$

Accordingly,

$$\frac{\pi}{2\sqrt{3}}+\frac{4\pi\ln 2}{\sqrt{3}}+\frac{\sqrt{3}\pi}{2}-\frac{4\pi}{\sqrt{3}}=\frac{2(2\ln 2-1)\pi}{\sqrt{3}}\leq\sigma^2$$

and $\sigma\geq\sqrt{\frac{2(2\ln 2-1)\pi}{\sqrt{3}}}\approx 1.183$. Therefore Theorem 4 holds. □

C.3 Algorithm 4

In this section we define Algorithm 4, the Position Creation Phase used by Algorithm 2. It proceeds in rounds $i = 0, 1, \ldots$, and Figure 5 illustrates the colored areas after round 0.

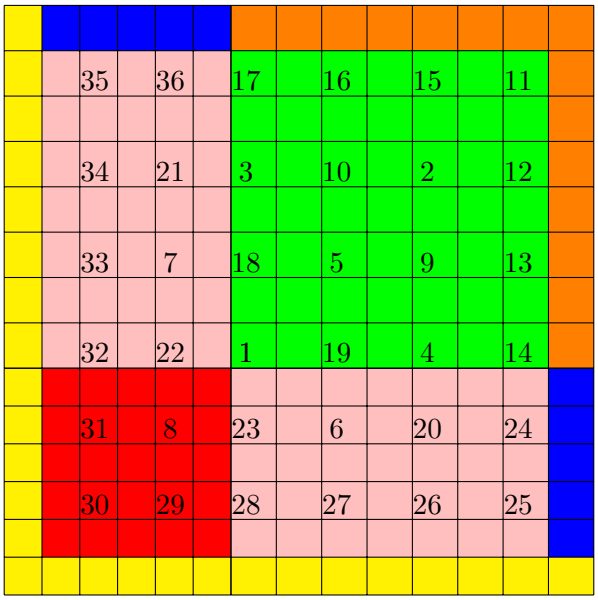

Figure 5: The round-0 rectangles and colored areas for Algorithm 4. The positions $q_1, \ldots, q_{36}$ are still labelled, in order to compare with Figure 3.

C.4 Some Intuition for the Set Q of Pre-fixed Positions

To help understanding the performance of Algorithm 2, we first provide some intuition for the selection of the first 36 positions in Q. Similar to those for Algorithm 1, they cannot be too unevenly distributed, and cannot be too evenly distributed either. That is why q_1 is not at the center of the square. Without loss of generality, it is closer to the left and the bottom boundary. Letting y be its distance to the left (and the bottom) boundary, $\frac{Disp(1;P)}{y} = \frac{1}{2y}$ will be roughly the approximation ratio. Again similar to Algorithm 1, q_1's distance to the right and the top boundary, $1 - y$, is larger than y but smaller than $2y$.

To make the minimum distance shrink as little as possible, q_2 is put in the center of the larger square, to the upper-right of q_1. Note that the best position for q_2 given q_1 is of equal distance to q_1 and the boundary, as illustrated by Figure 6. We instead put it in the center of the upper-right square so as to leave enough space for q_3 and q_4. Indeed, q_3 and q_4 are symmetric and at the grid vertices induced by q_1 and q_2, so that adding them to $\hat{Q}$ does not shrink the minimum distance.

After the first four positions, the square is split into 9 small rectangles, from A to I, as shown in Figure 7. Let us now consider q_5. Notice that among the four upper-right squares, E is better than B, C and F, as the latter three have the same size with E but are adjacent to the boundaries. If we put q_5 at the center of E, then eventually we will slice the area to the upper-right of q_1 into four intervals in each dimension, all of length $\frac{1-y}{4}$.

Another possibility is to put q_5 in G. However, note that the center of G is not the optimal position in G, for the same reason as the choice of q_2. We will choose y such that the center of E is better than the optimal position in G. In fact, a better approximation ratio is achieved if we slice

Algorithm 4. The Position Creation Phase.

In this phase, our algorithm repeatedly splits the rectangles into smaller ones by creating a position at its center. The internal areas (green, pink and red) will expand and the outer areas (orange, yellow and blue) will still be the strips with width of a single rectangle.

After q_{36} is added to $\hat{Q}$, the position creation phase proceeds in rounds $i = 0, 1, \ldots$ such that, after each round, each of the previous rectangles has been divided into four sub-rectangles by a position at its center. Accordingly, the number of rectangles created in round $i \geq 0$, referred to as *round-i* rectangles, is $49 \cdot 4^{i+1}$; the number of intervals in each dimension is $7 \cdot 2^{i+1}$; and the number of positions in $\hat{Q}$ is $(7 \cdot 2^{i+1} - 1)^2$, corresponding to the grid vertices not on the boundary. Moreover, the positions created in round i are positions $(7 \cdot 2^i - 1)^2 + 1 \leq n \leq (7 \cdot 2^{i+1} - 1)^2$.

The algorithm keeps the round number i and updates as it proceeds. (Or, given $n > 36$, we can find in time $O(\log n)$ the round i in which position n is created from the formula above.) The location of position n is decided by the following five cases. We provide the ranges of position n for each case, so that it can be easily decided which case position n belongs to.

Case 1. If there is a round-$(i-1)$ square in the green area whose center is not in $\hat{Q}$, arbitrarily pick such a rectangle and put position n at its center, splitting it into four round-i rectangles.

We claim these are positions $(7 \times 2^i - 1)^2 + 1 \leq n \leq= 65 \times 2^{2i} - 22 \times 2^i + 2$.

Case 2. Else, if there is a round-$(i-1)$ rectangle in the pink areas whose center is not in $\hat{Q}$, arbitrarily pick such a rectangle and put position n at its center, splitting it into four round-i rectangles.

We claim these are positions $65 \times 2^{2i} - 22 \times 2^i + 3 \leq n \leq 89 \times 2^{2i} - 36 \times 2^i + 4$.

Case 3. Else, if there exists a round-$(i-1)$ rectangle in the red area (which is actually a square) whose center is not in $\hat{Q}$, arbitrarily pick such a rectangle and create position n at its center, splitting it into four round-i rectangles (again, squares).

We claim these are positions $89 \times 2^{2i} - 36 \times 2^i + 5 \leq n \leq 98 \times 2^{2i} - 42 \times 2^i + 5$.

Case 4. Else, we distinguish two sub-cases.

First, if there exists a round-$(i-1)$ rectangle in the orange area and the two blue areas whose center is not in $\hat{Q}$, arbitrarily pick such a rectangle and create position n at its center, splitting it into four round-i rectangles.

Otherwise, if there exists a vertex of a round-i rectangle in the green and the orange areas which is (a) not in $\hat{Q}$, (b) not adjacent to the blue or the pink areas and (c) not on the boundary, then arbitrarily pick such a vertex for position n.

We claim these are positions $98 \times 2^{2i} - 42 \times 2^i + 6 \leq n \leq 130 \times 2^{2i} - 36 \times 2^i + 2$.

Case 5. Else, among all the centers of the round-$(i-1)$ rectangles in the yellow area and all the vertices of the round-i rectangles, which is not in $\hat{Q}$ and not on the boundary, arbitrarily pick one for position n.

We claim these are positions $130 \times 2^{2i} - 36 \times 2^i + 3 \leq n \leq (7 \times 2^{i+1} - 1)^2$.

After all positions in round i are created, the yellow, blue, and orange areas shrink by one round-i rectangle towards the boundary: they only contain rectangles adjacent to the boundary. The area released by orange is taken by green; that released by blue is taken by pink; and that released by yellow is taken by blue, pink, and red. Figure 5 illustrates the colored areas after round 0.

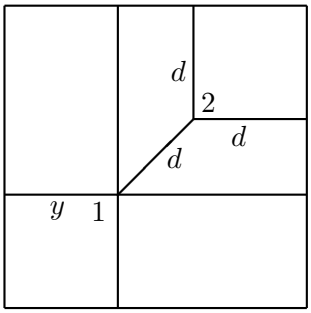

Figure 6: The best position for q_2 given q_1.

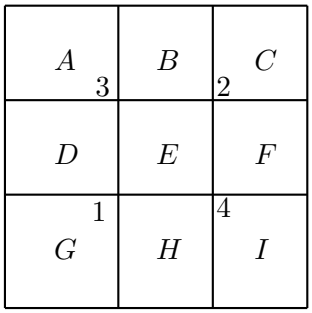

Figure 7: After the first four positions.

G into three intervals instead of four in each dimension, all of length $\frac{y}{3}$. Let $x = \frac{y}{3}$ and $cx = \frac{1-y}{4}$ be the corresponding lengths of the intervals after all the slicing, as illustrated by Figure 3. We have $x = \frac{1}{3+4c}$ and the minimum distance incurred by putting q_5 at the center of E is $\sqrt{2}cx$.

After the grid with 36 internal vertices is created, the positions on the grid appear naturally by always selecting the next optimum. In order to find this optimal position and compute the corresponding minimum distance easily, in a 1-dimensional space we would choose $x < cx < 2x$ as in Algorithm 1. In a 2-dimensional space we instead want $x < cx < \sqrt{2}x$, that is, $1 < c < \sqrt{2}$, because the distances are L_2-norm. The resulting ordering for Q is shown in Figure 3 and the minimum distances after each position n are shown in Table 2, grouped into 7 cases.

Case	1	2	3	4	5	6	7
n	1	$2, 3, 4$	5	$6, 7$	8	$9, \cdots, 17$	$18, \cdots, 36$
$d_{min}(n)$	$3x$	$2cx$	$\sqrt{2}cx$	$\sqrt{c^2+1}x$	$\sqrt{2}x$	cx	x

Table 2: The minimum distances as positions in Q are created.

C.5 Proof for Theorem 5

Theorem 5. (restated) *Algorithm 2 is of polynomial time and it is a* 1.591*-approximation for the 2-dimensional online ATWC problem in a square.*

We first show that the algorithms are well defined; see the claim below.

Claim 8. *The ranges of position n for each case in Algorithm 4 are all correct.*

Proof. For each $i \geq 0$, recall that the positions created in round i are $(7 \cdot 2^i - 1)^2 + 1 \leq n \leq (7 \cdot 2^{i+1} - 1)^2$. Below we show how the positions are distributed across the five cases of Algorithm 4, where for $i = 0$, round $i - 1$ refers to the creation of the first 36 positions in Q.

Case 1. Denote the number of round-$(i-1)$ squares in the green area by N_g^{i-1}. By construction, the number of round-i squares in the green area is $N_g^i = (2\sqrt{N_g^{i-1}} + 1)^2$: indeed, $\sqrt{N_g^{i-1}}$ is the number of intervals in each dimension in the green area after round $i-1$; $2\sqrt{N_g^{i-1}}$ is obtained because each one of them is split into two after round i; and the extra $+1$ is because, at the end of round i, the orange area shrinks and the green area grows by one round-i rectangle to the top and to the right.

Solving this inductive formula with initial condition $N_g^{-1} = 9$, we have

$$N_g^{i-1} = (4 \times 2^i - 1)^2$$

for each $i \geq 0$, which is exactly the number of positions created in Case 1 of round i, because one position is created at the center of every such square. Accordingly, in Case 1,

$$(7 \cdot 2^i - 1)^2 + 1 \leq n \leq (7 \cdot 2^i - 1)^2 + (4 \times 2^i - 1)^2 = 65 \times 2^{2i} - 22 \times 2^i + 2.$$

Case 2. Let N_p^{i-1} be the number of round-$(i-1)$ rectangles in the pink areas. In order to compute N_p^{i-1}, we first compute N_r^{i-1}, the number of round-$(i-1)$ squares in the red area. Similar to the induction above, we have $N_r^{-1} = 4$ and $N_r^i = (2\sqrt{N_r^{i-1}} + 1)^2$, thus

$$N_r^{i-1} = (3 \times 2^i - 1)^2.$$

Accordingly,

$$N_p^{i-1} = 2 \times \sqrt{N_r^{i-1}} \times \sqrt{N_g^{i-1}} = 2(3 \times 2^i - 1)(4 \times 2^i - 1),$$

which is exactly the number of positions created in Case 2 of round i, because one position is created at the center of every such square. Thus in Case 2,

$$(7 \cdot 2^i - 1)^2 + (4 \times 2^i - 1)^2 + 1 \leq n \leq (7 \cdot 2^i - 1)^2 + (4 \times 2^i - 1)^2 + 2(3 \times 2^i - 1)(4 \times 2^i - 1),$$

that is,

$$65 \times 2^{2i} - 22 \times 2^i + 3 \leq n \leq 89 \times 2^{2i} - 36 \times 2^i + 4.$$

Case 3. Similarly, in Case 3 of round i, the number of positions created is $N_r^{i-1} = (3 \times 2^i - 1)^2$ and

$$89 \times 2^{2i} - 36 \times 2^i + 5 \leq n \leq 89 \times 2^{2i} - 36 \times 2^i + 4 + (3 \times 2^i - 1)^2 = 98 \times 2^{2i} - 42 \times 2^i + 5.$$

Case 4. Furthermore, the number of round-$(i-1)$ orange rectangles and that of round-$(i-1)$ blue rectangles are, respectively,

$$N_o^{i-1} = (\sqrt{N_g^{i-1}} + 1)^2 - N_g^{i-1} = (4 \times 2^i)^2 - (4 \times 2^i - 1)^2 = 8 \times 2^i - 1$$

and

$$N_b^{i-1} = 2 \times \sqrt{N_r^{i-1}} = 2(3 \times 2^i - 1).$$

Thus the first sub-case in Case 4 of round i creates $N_o^{i-1} + N_b^{i-1} = (8 \times 2^i - 1) + (6 \times 2^i - 2) = 14 \times 2^i - 3$ positions.

In the second sub-case, there are $N_g^{i-1} + N_o^{i-1}$ round-$(i-1)$ squares in the green and the orange areas combined, thus the same number of centers created in them so far. For each of the upper-left N_g^{i-1} round-$(i-1)$ squares, its center induces 2 new positions, one to its right and one to its bottom. For each of the remaining $N_o^{i-1} - 1$ round-$(i-1)$ squares with the one at the bottom-right corner excluded, its center induces 1 new position, either to the bottom or to the right, because the other one is either on the boundary or is adjacent to the pink area below. Moreover, the center of the orange round-$(i-1)$ square at the bottom-right corner does not induce any new position, because the one to its right is on the boundary and the one to its bottom is adjacent to the blue area below. Accordingly, the number of

new positions created in this subcase is $2N_g^{i-1} + N_o^{i-1} - 1 = 2(4 \times 2^i - 1)^2 + 8 \times 2^i - 1 - 1 = 32 \times 2^{2i} - 8 \times 2^i$.

Combining the two sub-cases together, the positions created in Case 4 of round i are

$$98\times2^{2i}-42\times2^i+6 \le n \le 98\times2^{2i}-42\times2^i+5+(14\times2^i-3)+(32\times2^{2i}-8\times2^i) = 130\times2^{2i}-36\times2^i+2.$$

Case 5. Finally, all the remaining positions in round i are created in Case 5, with

$$130 \times 2^{2i} - 36 \times 2^i + 3 \le n \le (7 \cdot 2^{i+1} - 1)^2.$$

For completeness, we note that the number of round-$(i-1)$ yellow rectangles is

$$N_y^{i-1} = (7 \times 2^i)^2 - (7 \times 2^i - 1)^2 = 14 \times 2^i - 1.$$

Combining all the cases together, Claim 8 holds. □

Finally we are ready to prove Theorem 5.

Proof of Theorem 5. It is easy to see that the running times of Algorithms 2 and 4 are polynomial in $|\hat{Q}|$ and thus in the number of arrived points. For the approximation ratio, again we can focus on instances of the form $S = ((1, n+1), (2, n+1), \ldots, (n, n+1))$, with $n \ge 1$. Letting $apx(n; P) = \frac{Disp(n;P)}{d_{min}(n)}$, we would like to find $c = \arg\min_c \max_n apx(n; P)$. In the analysis below we distinguish three cases: first $n \le 36$, then n in rounds 0 and 1 of Algorithm 4, and at last n in round i for each $i \ge 2$ of Algorithm 4.

For $n \le 36$. Following Table 2 in Section C.4, the minimum distances within each of the seven cases are the same. Since $Disp(n; P)$ is non-increasing, the worst approximation ratios occur at the first position n in each case. For example, the worst approximation ratio in case 2 is $\frac{Disp(2;P)}{2cx}$.

For $n = 1, 2$ and 5, the exact solution for $Disp(n; P)$ can be easily found. That is, with one position in the center for $n = 1$; with two positions on a diagonal for $n = 2$, where the ratios of the three intervals are $\sqrt{2} : 1 : \sqrt{2}$; and with one position in the center and four positions in the four resulting squares for $n = 5$, where the four positions are all on the diagonals and the ratios of the resulting intervals on a diagonal is $\sqrt{2} : 1 : 1 : \sqrt{2}$.

Although there is no general closed-form for $Disp(n; P)$ in the literature, exact solutions for $DP(n; P)$ have been found for small n's, and we use them to get the worst approximation ratios of our algorithm for $6 \le n \le 36$. Extending Table 2, the approximation ratios are shown in Table 3. Note that here we do not need the upper bound of $Disp(n; P)$, as we know the exact solution.

Case	1	2	3	4	5	6	7
n	1	2, 3, 4	5	6, 7	8	$9, \cdots, 17$	$18, \cdots, 36$
$d_{min}(n)$	$3x$	$2cx$	$\sqrt{2}cx$	$\sqrt{c^2+1}x$	$\sqrt{2}x$	cx	x
$apx(n; P)$	$\frac{0.5}{3x}$	$\frac{0.36940}{2cx}$	$\frac{0.29290}{\sqrt{2}cx}$	$\frac{0.27292}{\sqrt{c^2+1}x}$ [29]	$\frac{0.25434}{\sqrt{2}x}$ [28]	$\frac{1}{4cx}$ [27]	$\frac{0.18769}{x}$ [23]

Table 3: Approximation ratios for $n \le 36$. Note that [29, 28, 27, 23] provide exact solutions for the corresponding $DP(n; P)$'s. We compute the corresponding $Disp(n; P)$'s by Claim 3.

Round 0. Our construction of Algorithm 4 guarantees that, in each round $i \geq 0$, $d_{min}(n)$ is the same for all n's in the same case. The general formulas of $d_{min}(n)$ in all five cases are shown in Table 4, and the approximation ratios are obtained by applying our upper-bound for $Disp(n; P)$ in Lemma 3. Again note that the worst approximation ratio in each case happens at the first position n, because $Disp(n; P)$ is non-decreasing. The corresponding ratios for round 0 can be obtained by setting $i = 0$.

Case	n	$d_{min}(n)$	$apx(n; P)$
1	$(7 \times 2^i - 1)^2 + 1 \leq n \leq 65 \times 2^{2i} - 22 \times 2^i + 2$	$\frac{\sqrt{2}cx}{2^{i+1}}$	$\frac{2\sqrt{2}\times 2^i}{cx(2+\sqrt{2\sqrt{3}((7\times 2^i-1)^2+1)})}$
2	$65 \times 2^{2i} - 22 \times 2^i + 3 \leq n \leq 89 \times 2^{2i} - 36 \times 2^i + 4$	$\frac{x\sqrt{1+c^2}}{2^{i+1}}$	$\frac{4\times 2^i}{x\sqrt{1+c^2}(2+\sqrt{2\sqrt{3}(65\times 2^{2i}-22\times 2^i+3)})}$
3	$89 \times 2^{2i} - 36 \times 2^i + 5 \leq n \leq 98 \times 2^{2i} - 42 \times 2^i + 5$	$\frac{\sqrt{2}x}{2^{i+1}}$	$\frac{2\sqrt{2}\times 2^i}{x(2+\sqrt{2\sqrt{3}(89\times 2^{2i}-36\times 2^i+5)})}$
4	$98 \times 2^{2i} - 42 \times 2^i + 6 \leq n \leq 130 \times 2^{2i} - 36 \times 2^i + 2$	$\frac{cx}{2^{i+1}}$	$\frac{4\times 2^i}{cx(2+\sqrt{2\sqrt{3}(98\times 2^{2i}-42\times 2^i+6)})}$
5	$130 \times 2^{2i} - 36 \times 2^i + 3 \leq n \leq (7 \times 2^{i+1} - 1)^2$	$\frac{x}{2^{i+1}}$	$\frac{4\times 2^i}{x(2+\sqrt{2\sqrt{3}(130\times 2^{2i}-36\times 2^i+3)})}$

Table 4: Approximation ratios for round $i \geq 0$.

Now we show the correctness of $d_{min}(n)$ in Table 4 for any $i \geq 0$. If n falls into Case 1, it is at the center of a round-$(i-1)$ square in the green area, and the length of the edge is $\frac{4cx}{\sqrt{N_g^{i-1}+1}} = \frac{4cx}{4\times 2^i} = \frac{cx}{2^i}$. Since the minimum distance is half of the diagonal of that square, we have

$$d_{min}(n) = \frac{cx}{\sqrt{2}2^i}.$$

If n falls into Case 2, it is at the center of a round-$(i-1)$ rectangle in the pink areas. The length of the rectangle is $\frac{cx}{2^i}$ and the width is $\frac{3x}{\sqrt{N_r^{i-1}+1}} = \frac{3x}{3\times 2^i} = \frac{x}{2^i}$. Since the minimum distance is half of the diagonal of that rectangle, we have

$$d_{min}(n) = \frac{\sqrt{1+c^2}}{2} \times \frac{x}{2^i} = \frac{x\sqrt{1+c^2}}{2^{i+1}}.$$

If n falls into Case 3, it is at the center of a round-$(i-1)$ square in the red area, and the length of the edge is $\frac{x}{2^i}$. Again, the minimum distance is half of the diagonal of that square and we have

$$d_{min}(n) = \frac{x}{\sqrt{2}2^i}.$$

If n falls into Case 4, then it is either at the center of a round-$(i-1)$ rectangle in the orange and the blue areas, or it is a vertex of a round-i rectangle in the green and the orange areas such that it is not on the boundary or adjacent to the pink or the blue areas. No matter which sub-case happens, the minimum distance incurred by position n is $\frac{1}{2} \times \frac{cx}{2^i}$, either to the boundary or to the center of a round-$(i-1)$ rectangle next to it. Thus

$$d_{min}(n) = \frac{cx}{2^{i+1}}.$$

Finally, if n falls into Case 5, it is either at the center of a round-$(i-1)$ rectangle in yellow area or it is a vertex of a round-i rectangle in the blue, yellow, pink or red areas. No matter which

sub-case happens, the minimum distance incurred by position n is $\frac{1}{2} \times \frac{x}{2^i}$, either to the boundary or to the center of the round-$(i-1)$ rectangle next to it. Therefore

$$d_{min}(n) = \frac{x}{2^{i+1}}.$$

In sum, the general formulas of $d_{min}(n)$ for any i and n are as shown in Table 4.

Round 1. By setting $i = 1$ in Table 4, similar formulas for $apx(n; P)$ can be obtained for all five cases of round 1.

Round $i \geq 2$. To find the worst approximation ratio for all rounds $i \geq 2$, the difficulty is that the upper-bounds for $apx(n; P)$ in Table 4 are not necessarily monotone, thus it is hard to tell where the worst ratio occurs. Instead, we find a universal upper-bound for the ratios and a novel way to analyze its monotonicity. More precisely, let $f(n; P) = \frac{2}{\sqrt{2\sqrt{3}n} \cdot d_{min}(n)}$. By Lemma 3, for any $n \geq 1$,

$$apx(n; P) = \frac{Disp(n; P)}{d_{min}(n)} \leq \frac{2}{(2 + \sqrt{2\sqrt{3}n}) d_{min}(n)} \leq \frac{2}{\sqrt{2\sqrt{3}n} \cdot d_{min}(n)} = f(n; P).$$

In Table 5, we replace $axp(n; P)$ by its upper-bound $f(n; P)$.

Case	n	$d_{min}(n)$	$f(n; P)$
1	$(7 \times 2^i - 1)^2 + 1 \leq n \leq 65 \times 2^{2i} - 22 \times 2^i + 2$	$\frac{\sqrt{2}cx}{2^{i+1}}$	$\frac{2\sqrt{2}\times 2^i}{cx\sqrt{2\sqrt{3}((7\times 2^i-1)^2+1)}}$
2	$65 \times 2^{2i} - 22 \times 2^i + 3 \leq n \leq 89 \times 2^{2i} - 36 \times 2^i + 4$	$\frac{x\sqrt{1+c^2}}{2^{i+1}}$	$\frac{4\times 2^i}{x\sqrt{1+c^2}\sqrt{2\sqrt{3}(65\times 2^{2i}-22\times 2^i+3)}}$
3	$89 \times 2^{2i} - 36 \times 2^i + 5 \leq n \leq 98 \times 2^{2i} - 42 \times 2^i + 5$	$\frac{\sqrt{2}x}{2^{i+1}}$	$\frac{2\sqrt{2}\times 2^i}{x\sqrt{2\sqrt{3}(89\times 2^{2i}-36\times 2^i+5)}}$
4	$98 \times 2^{2i} - 42 \times 2^i + 6 \leq n \leq 130 \times 2^{2i} - 36 \times 2^i + 2$	$\frac{cx}{2^{i+1}}$	$\frac{4\times 2^i}{cx\sqrt{2\sqrt{3}(98\times 2^{2i}-42\times 2^i+6)}}$
5	$130 \times 2^{2i} - 36 \times 2^i + 3 \leq n \leq (7 \times 2^{i+1} - 1)^2$	$\frac{x}{2^{i+1}}$	$\frac{4\times 2^i}{x\sqrt{2\sqrt{3}(130\times 2^{2i}-36\times 2^i+3)}}$

Table 5: Approximation ratios upper-bounded by $f(n; P)$.

Given Tables 4 and 5, for any $n \geq 37$, it falls into one entry based on its round number i and case number $j \in \{1, \ldots, 5\}$. Accordingly, we also use $apx(i, j)$ and $f(i, j)$ to denote $apx(n; P)$ and $f(n; P)$. When $1 \leq n \leq 36$, it falls into one of the seven cases in Table 3, and we denote the corresponding quantities by $apx(*, j)$ and $f(*, j)$ with $j \in \{1, \ldots, 7\}$. Since we want to find a proper parameter c to minimize $\max\{\max_{i\geq 0,\ j\leq 5} apx(i, j), \max_{j\leq 7} apx(*, j)\}$, we have to find where $apx(i, j)$ or $apx(*, j)$ is maximized.

When $i \geq 2$, given any $j \leq 5$, it is easy to see that $f(i, j)$ is decreasing with respect to i—that is, $f(n; P)$ may not be monotone overall, but it is monotone across the same case! Therefore we only need to compare $apx(*, j)$ for $j = 1, \cdots, 7$, $apx(i, j)$ for $i = 0, 1$ and $j = 1, \cdots, 5$, and $f(2, j)$ for $j = 1, \cdots, 5$. Note that $f(i, j)$ is also decreasing at $i = 0, 1$, but it is a loose upper-bound there, thus we use $apx(i, j)$ for $i = 0, 1$ in order to get a better bound.

What remains is simple. Indeed, there are in total 22 $apx(\cdot; \cdot)$'s and $f(\cdot; \cdot)$'s to consider, whose values can be directly calculated from the tables. More specifically, each one of them is less than or equal to $\max\{apx(*, 6), apx(0, 5)\}$. Notice that, so far, we haven't used the value of c. By choosing c

to minimize $\max\{apx(*,6), apx(0,5)\}$, we get $c = \frac{2+\sqrt{194\sqrt{3}}}{16} \approx 1.271 \in (1, \sqrt{2})$, [2] and the two values are both $\frac{1}{4cx}$. Accordingly, the approximation ratio of Algorithm 2 is $\max_n \frac{Disp(n;P)}{d_{min}(n)} = \frac{1}{4cx} < 1.591$, and Theorem 5 holds. □

D Proofs for Section 5

Theorem 6. (restated) *For any $k \geq 2$, no algorithm achieves an approximation ratio better than $\frac{7}{6}$ for the online ATWC problem for arbitrary polytopes.*

Proof. We show that even when the given k-dimensional "polytope" P is a sphere and there are only two points, no algorithm achieves an approximation ratio better than $\frac{7}{6}$. As a sphere can be approximated arbitrarily closely by a polytope, this implies our theorem.

Without loss of generality, let the center of the sphere be $(0, 0, \ldots, 0)$ and the radius be $\frac{1}{2}$. We first consider the optimal solution for the dispersion problem without time. Clearly, $Disp(1; P) = \frac{1}{2}$. When there are two points, we have the following.

Claim 9. *For any algorithm $\mathcal{A}$ locating 2 points in P, there exists another algorithm $\mathcal{A}'$ such that $\mathcal{A}'$ locates the two points on a diameter of the sphere and the approximation ratio of $\mathcal{A}'$ to $Disp(2; P)$ is no worse than $\mathcal{A}$.*

Proof. Without loss of generality, assume $\mathcal{A}$ locates point 1 at $p_1 = (x, 0, 0, \ldots, 0)$ with $x \leq 0$. Denote the position of point 2 according to $\mathcal{A}$ as $p_2 = (y_1, y_2, \ldots, y_k)$. It is easy to see that p_2 has distance $y' = \sqrt{\sum_i^k y_i^2}$ to the center and $\frac{1}{2} - y'$ to the boundary. Let algorithm $\mathcal{A}'$ locate point 1 at p_1 and point 2 at $p_2' = (y', 0, 0, \ldots, 0)$. Note that $dis(p_2', \partial P) = \frac{1}{2} - y' = dis(p_2, \partial P)$ and $dis(p_2', p_1) = y' - x \geq dis(p_2, p_1)$, by triangle inequality. Thus Claim 9 holds. □

Following Claim 9, it is not hard to see that the optimal solution has p_1 and p_2' with $x = -\frac{1}{6}$ and $y' = \frac{1}{6}$, thus $Disp(2; P) = \frac{1}{3}$. Now we consider online algorithms for ATWC. Following Claim 9, without loss of generality we focus on algorithms that locate the second point on the same diameter as the first. Again without loss of generality, the two positions are $p_1 = (x, 0, \ldots, 0)$ and $p_2 = (y, 0, \ldots, 0)$, with $x \leq 0 \leq y$. Moreover, given p_1, the best position for p_2 is to set y such that $dis(p_2, \partial P) = dis(p_2, p_1)$. That is, $y - x = \frac{1}{2} - y$, which implies $y = \frac{1+2x}{4}$. Accordingly, $d_{min}(2; P) = \min\left\{\frac{1}{2} + x, \frac{1-2x}{4}\right\}$ and the worst approximation ratio after the first two points is

$$\max\left\{\frac{\frac{1}{2}}{\frac{1}{2}+x}, \frac{\frac{1}{3}}{\min\left\{\frac{1}{2}+x, \frac{1-2x}{4}\right\}}\right\}.$$

Choosing x to minimize this maximum, we have $x = -\frac{1}{14}$ and the approximation ratio cannot be better than $\frac{7}{6}$. Thus Theorem 6 holds. □

Similarly, a lower-bound of $1 + \frac{1}{4\sqrt{k}+2}$ can be obtained for k-dimensional cubes for any $k \geq 2$. To construct an online algorithm for ATWC for arbitrary polytopes, we first consider the straightforward greedy algorithm, and we have the following. Recall that the geometric problems of finding the minimum bounding cube, deciding whether a position is in P, and finding the distance between a point in P and the boundary of P are given as oracles.

[2] In order to deal with irrational numbers, we round c to 1.271 and the result still holds.

Lemma 4. *For any $k \geq 2$, $n \geq 1$, k-dimensional polytope P and n positions $p_1, \ldots, p_n \in P$, there exists a position $p \in P$ such that $\min_{i\in[n]}\{dis(p, p_i), dis(p, \partial P)\} \geq \frac{Disp(n;P)}{2}$.*

Proof. Denote by $\mathcal{P}$ the set of points in P with distance larger than or equal to $\frac{Disp(n;P)}{2}$ from the boundary. By contradiction, assume that for every $p \in \mathcal{P}$, $\min_{i\in[n]}\{dis(p, p_i)\} < \frac{Disp(n;P)}{2}$. Letting $r = \sup_{p\in\mathcal{P}} \min_{i\in[n]}\{dis(p, p_i)\}$, we have $r < \frac{Disp(n;P)}{2}$, because $\mathcal{P}$ is closed. Accordingly, $\mathcal{P}$ is covered by n balls centered at $p_1, \ldots, p_n$ with radius r and

$$vol(\mathcal{P}) \leq nBall(r),$$

where $vol(\mathcal{P})$ is the volume of $\mathcal{P}$ and $Ball(r)$ is the volume of a ball with radius r. However, by the definition of $Disp(n; P)$, there exist n positions in P such that

$$\min_{i,j\in[n]}\{dis(p_i, p_j), dis(p_i, \partial P)\} \geq Disp(n; P).$$

Thus n balls with radius $\frac{Disp(n;P)}{2}$ can be packed into $\mathcal{P}$, which implies

$$vol(\mathcal{P}) \geq nBall(\frac{Disp(n; P)}{2})$$

and we get a contradiction, because $Ball(r) < Ball(\frac{Disp(n;P)}{2})$. Therefore Lemma 4 holds. □

Consider the greedy algorithm that, for each $n \geq 1$, creates the $(n + 1)$-st position at $p = \arg\max_{p\in P} \min_{i\in[n]}\{dis(p, p_i), dis(p, \partial P)\}$. (As Algorithm 1, it adds p to $\hat{Q}$ and creates a position only if positions in $\hat{Q}$ are all occupied.) Since $\frac{Disp(n;P)}{2} \geq \frac{Disp(n+1;P)}{2}$, by Lemma 4 and an inductive reasoning, it is easy to see that this online algorithm is a 2-approximation for the ATWC problem. However, finding the optimal position p may be time consuming even given the oracles, and we design a polynomial-time approximation algorithm which achieves an approximation ratio of $\frac{2}{1-\epsilon}$ for any $\epsilon > 0$.

Theorem 7. (restated) *For any constants $\gamma, \epsilon > 0$, for any integer $k \geq 2$ and any k-dimensional polytope P with covering rate at least γ, there exists a deterministic polynomial-time online algorithm for the ATWC problem, with approximation ratio $\frac{2}{1-\epsilon}$ and running time polynomial in $\frac{1}{(\gamma\epsilon)^k}$.*

Proof. Without loss of generality, the minimum bounding cube of P is the unit cube. Thus the edge-length of the maximum inscribed cube C is at least γ. The idea is to simply slice the unit cube into small cubes and exhaustively search all the cube-centers that are in P. The number of cubes need to be searched depends on the number n of existing positions, the approximation parameter ϵ, and a lower-bound for $Disp(n; P)$, which ultimately depends on γ. In particular, we have the following.

Claim 10. *For any $n \geq 1$, let $m = \left\lceil \frac{\sqrt{k}(n^{1/k}+2)}{\gamma\epsilon} \right\rceil$. If we uniformly slice the unit cube into m^k smaller cubes $\{C_1, C_2, \ldots, C_{m^k}\}$ with edge-length $\frac{1}{m}$, then for any n positions $p_1, \ldots, p_n \in P$, there exists a cube C_j whose center c_j is in P and $\min_{i\in[n]}\{dis(c_j, p_i), dis(c_j, \partial P)\} \geq \frac{(1-\epsilon)Disp(n;P)}{2}$.*

Proof. Let p be the optimal position chosen by the greedy algorithm given $p_1, \ldots, p_n$, C_j be the cube containing p, and c_j the center of C_j. We have that

$$\min_{i\in[n]}\{dis(p, p_i), dis(p, \partial P)\} \geq \frac{Disp(n; P)}{2} \tag{11}$$

and

$$dis(p, c_j) \leq \frac{\sqrt{k}}{2m} \leq \frac{\epsilon\gamma}{2(n^{1/k}+2)}. \tag{12}$$

Now we show $Disp(n; P) \geq \frac{\gamma}{n^{1/k}+2}$ by finding n positions $p'_1, \ldots, p'_n \in P$ such that

$$\min_{i,j\in[n]} \{dis(p'_i, p'_j), dis(p'_i, \partial P)\} \geq \frac{\gamma}{n^{1/k}+2}.$$

In particular, we uniformly slice cube C into $(\lceil n^{1/k} \rceil + 1)^k$ smaller cubes with edge-length at least $\frac{\gamma}{\lceil n^{1/k} \rceil + 1}$ and arbitrarily choose n vertices of them from the interior of C: note that there are $(\lceil n^{1/k} \rceil)^k \geq n$ such vertices. It is easy to see that

$$\min_{i,j\in[n]} \{dis(p'_i, p'_j), dis(p'_i, \partial P)\} \geq \min_{i,j\in[n]} \{dis(p'_i, p'_j), dis(p'_i, \partial C)\} \geq \frac{\gamma}{\lceil n^{1/k} \rceil + 1} \geq \frac{\gamma}{n^{1/k}+2},$$

which implies

$$Disp(n; P) \geq \frac{\gamma}{n^{1/k}+2} \tag{13}$$

as desired. Combining Equations 11, 12, 13, for any $\epsilon < 1$, we have $dis(p, c_j) < \frac{\gamma}{2(n^{1/k}+2)} \leq \frac{Disp(n;P)}{2} \leq dis(p, \partial P)$, thus $c_j \in P$. Moreover, for any $i \in [n]$, $dis(p, c_j) < \frac{Disp(n;P)}{2} \leq dis(p, p_i)$. By the triangle inequality,

$$dis(c_j, p_i) \geq dis(p, p_i) - dis(p, c_j) \geq \frac{Disp(n; P)}{2} - \frac{\gamma\epsilon}{2(n^{1/k}+2)} \geq \frac{(1-\epsilon)Disp(n; P)}{2}.$$

Similarly,

$$dis(c_j, \partial P) \geq dis(p, \partial P) - dis(p, c_j) \geq \frac{(1-\epsilon)Disp(n; P)}{2}.$$

Accordingly,

$$\min_{i\in[n]} \{dis(c_j, p_i), dis(c_j, \partial P)\} \geq \frac{(1-\epsilon)Disp(n; P)}{2}$$

as desired, and Claim 10 holds. □

Thus by going through all the c_j's in P and choosing the one that maximizes $\min_{i\in[n]}\{dis(c_j, p_i), dis(c_j, \partial P)\}$, we find the $(n+1)$-st position in time $O(m^k \cdot n) = O(\frac{n^2 k^{k/2}}{\gamma^k \epsilon^k})$. Again because $Disp(n; P) \geq Disp(n+1; P)$, applying Claim 10 to each round of the greedy algorithm, by induction we have that the resulting algorithm is of polynomial time and is a $\frac{2}{1-\epsilon}$-approximation. Therefore Theorem 7 holds. □

E Proofs for Section 6

Claim 11. *The output of Algorithm $\mathcal{A}_{\mathcal{I}}$ satisfies properties* $\Phi.1$ *and* $\Phi.2$ *and the running time of $\mathcal{A}_{\mathcal{I}}$ is $O(n^2)$.*

Proof. Given an input sequence $S = ((s_1, d_2), ..., (s_n, d_n))$, let $\mathcal{T}$ be the union of all the intervals that contain at least one point. Note that, for the original input sequence, we may assume $\mathcal{T} = [0, T]$ without loss of generality. However, this may change in later iterations.

In each round of $\mathcal{A}_{\mathcal{I}}$'s **while** loop, the algorithm considers all the points that arrive within the sliding window $(s, d]$ and depart after the sliding window. If there are such points, then it selects

the point j with the latest departure time among them and moves the sliding window to $(d, d_j]$. It puts point j in $\mathcal{I}_{index}$, which alternates between $\mathcal{I}_1$ and $\mathcal{I}_2$ in different rounds. If there is no such point, then it finds the earliest arriving time after d, $s' = \min_{i \in S, s_i > d} s_i$, and moves the sliding window to $(d, s']$. Note that whichever case happens, the new sliding window satisfies $s < d$, thus is well defined.

Now we prove the output $\{\mathcal{I}_1, \mathcal{I}_2\}$ satisfies Φ.1. Indeed, note that the variable *index* has the same value for any two rounds l and $l+2$. Let $(s, d]$ be the sliding window in round l. If point j is added to $\mathcal{I}_{index}$ in round l and point j' is added to $\mathcal{I}_{index}$ in round $l+2$, then the three sliding windows in rounds $l, l+1$ and $l+2$ are, respectively, $(s, d], (d, d_j]$ and $(d_j, x]$ with some $x > d_j$, computed either in Step 7 or Step 9. By the definition of j', $s_{j'} > d_j$ and points j and j' do not overlap. By induction, we have that no two points in $\mathcal{I}_1$ overlap, neither do any two points in $\mathcal{I}_2$. Thus property Φ.1 holds.

Next, we prove the output $\{\mathcal{I}_1, \mathcal{I}_2\}$ satisfies Φ.2: in other words, the time intervals of points in $\mathcal{I} = \mathcal{I}_1 \cup \mathcal{I}_2$ cover $\mathcal{T}$. To do so, note that the sliding windows in Algorithm $\mathcal{A}_{\mathcal{I}}$ are all disjoint and their union covers $[0, T]$. Accordingly, for any time $t \in [0, T]$ with at least one point j' at time t in S, there exists a unique round i with t in its sliding window $(s^i, d^i]$: that is,

$$s_{j'} \leq t \leq d^i \text{ and } d_{j'} \geq t > s^i. \tag{14}$$

We distinguish three cases.

Case 1. If $s^i = -1$ (i.e., round i is the first round in the algorithm), then it must be $d^i = 0 = s_{j'} = t$ and $d_{j'} > 0$. By definition, $\hat{S} \neq \emptyset$ and the chosen point j is such that $s_j = 0$ and $d_j \geq d_{j'}$. Therefore at least one point at time t (i.e., point j) is in $\mathcal{I}$.

Case 2. $s^i \geq 0$ and the sliding window $(s^i, d^i]$ is obtained from round $i-1$ in Step 7. Letting j be the point chosen in round $i-1$ in Step 6, we have $s_j \leq s^i$ and $d_j = d^i$, thus j is a point present at time t in S and $j \in \mathcal{I}$.

Case 3. $s^i \geq 0$ and the sliding window $(s^i, d^i]$ is obtained from round $i-1$ in Step 9. Letting $(s^{i-1}, d^{i-1}]$ be the sliding window in round $i-1$. By the construction of the algorithm, we have $\hat{S} = \emptyset$ in round $i-1$ and $s^i = d^{i-1}$, thus $d_{j'} > d^{i-1}$ by the second part of Equation 14. Below we consider the three possible intervals for $s_{j'}$.

If $s_{j'} > d^{i-1}$, then $d^i \leq s_{j'}$ by Step 9, thus $d^i = s_{j'} = t$ by the first part of Equation 14. Since $d_{j'} > s_{j'}$, we have $j' \in \hat{S}$ in round i, thus some point j is chosen in Step 6 of round i with $s_j \leq d^i$ and $d_j > d^i$. Therefore j is present at time t and $j \in \mathcal{I}$, as desired.

If $s^{i-1} < s_{j'} \leq d^{i-1}$, then $j' \in \hat{S}$ in round $i-1$, which is a contradiction. Thus j' cannot be in this interval.

Finally, if $s_{j'} \leq s^{i-1}$, then there exists another round i' and sliding window $(s^{i'}, d^{i'}]$ with $i' < i-1$, which contains $s_{j'}$. Since $d^{i'} \leq s^{i-1} < d^{i-1}$, we have $d_{j'} > d^{i'}$ and $j' \in \hat{S}$ in round i'. Accordingly, there is a point j chosen in round i' and $d^{i'+1} = d_j$. However, $d_{j'} \geq t > s^i = d^{i-1} \geq d^{i'+1} = d_j$, contradicting the fact that j has the latest departure time among all points in $\hat{S}$ in round i'. Thus j' cannot be in this interval either.

Accordingly, in Case 3 we have $s_{j'} > d^{i-1}$ and there is a point in $\mathcal{I}$ which is present at t.

Combining all the cases, property Φ.2 holds.

Finally, because there are $2n$ arriving and departure times in the sequence S and the end-points of the sliding windows only occur at those times, algorithm $\mathcal{A}_{\mathcal{I}}$ uses at most $2n$ sliding windows

(including $(-1, 0]$) and at most $2n$ rounds. Since each round takes $O(n)$ time, the running time of $\mathcal{A}_{\mathcal{I}}$ is $O(n^2)$. Therefore Claim 11 holds. □

The offline CD algorithm $\mathcal{A}_{CD}$ repeatedly uses $\mathcal{A}_{\mathcal{I}}$ for the remaining points until all points are processed. Based on the partitions constructed by $\mathcal{A}_{\mathcal{I}}$, $\mathcal{A}_{CD}$ constructs an instance of the online ATWC problem and uses an online algorithm $\mathcal{A}_{ATWC}$ for the latter as a black-box, so as to decide how to locate the points. Algorithm $\mathcal{A}_{CD}$ is defined in Algorithm 5 and we have the following theorem.

Algorithm 5. $\mathcal{A}_{CD}$

Input: A sequence $S = ((s_1, d_1), \ldots, (s_n, d_n))$.

1: Let r = 0.
2: **while** $S \neq \emptyset$ **do**
3: Run $\mathcal{A}_{\mathcal{I}}$ on S to obtain two disjoint sets $\mathcal{I}_{2r+1}, \mathcal{I}_{2r+2} \subseteq S$.
4: $S = S \setminus (\mathcal{I}_{2r+1} \cup \mathcal{I}_{2r+2})$.
5: $r = r + 1$.
6: **end while**
7: Run $\mathcal{A}_{ATWC}$ on the following online sequence of $2r$ points: for all $i \in \{0, 1, \ldots, r-1\}$, points $2i+1$ and $2i+2$ arrive at time i. All points depart at time r.
8: Letting x_{2i+1}, x_{2i+2} be the two positions returned by $\mathcal{A}_{ATWC}$ at time i, assign all points in $\mathcal{I}_{2i+1}$ to x_{2i+1} and all points in $\mathcal{I}_{2i+2}$ to x_{2i+2}.

Theorem 8. (restated) *For any $k \geq 1$ and k-dimensional polytope P, given any polynomial-time online algorithm $\mathcal{A}_{ATWC}$ for the ATWC problem with approximation ratio σ, there is a polynomial-time offline algorithm $\mathcal{A}_{CD}$ for the CD problem with approximation ratio $\sigma \max_{i \geq 1} \frac{Disp(i;P)}{Disp(2i;P)} \leq 2\sigma$, using $\mathcal{A}_{ATWC}$ as a black-box.*

Proof. We first consider the running time of $\mathcal{A}_{CD}$. Each time it runs $\mathcal{A}_{\mathcal{I}}$, the number of points in S decreases by at least one, thus $\mathcal{A}_{CD}$ runs for at most n rounds in the **while** loop: that is, $r \leq n$ in the end. By Claim 11, it takes $O(n^3)$ time to finish the **while** loop of $\mathcal{A}_{CD}$. Therefore the total running time of $\mathcal{A}_{CD}$ is $O(n^3 + Time(\mathcal{A}_{ATWC}))$, where the second term is the running time of the blackbox algorithm $\mathcal{A}_{ATWC}$ on $2r \leq 2n$ points. Since $\mathcal{A}_{ATWC}$ runs in polynomial time, $\mathcal{A}_{CD}$ runs in polynomial time.

Below we analyze the approximation ratio of $\mathcal{A}_{CD}$. After the **while** loop of $\mathcal{A}_{CD}$, S is partitioned into $\{\mathcal{I}_1, \mathcal{I}_2\}, \ldots, \{\mathcal{I}_{2r-1}, \mathcal{I}_{2r}\}$. Note that for each $i \in \{0, 1, \ldots, r-1\}$, by Claim 11, $\{\mathcal{I}_{2i+1}, \mathcal{I}_{2i+2}\}$ satisfies property $\Phi.1$ and can be located at two positions, one for all points in $\mathcal{I}_{2i+1}$ and the other for all points in $\mathcal{I}_{2i+2}$. Given the online instance created in Step 7, at each time step $i \leq r-1$, algorithm $\mathcal{A}_{ATWC}$ creates 2 positions x_{2i+1} and x_{2i+2}, for $\mathcal{I}_{2i+1}$ and $\mathcal{I}_{2i+2}$.

Let $T_1, \cdots, T_l$ be the sequence of time intervals sliced by the arriving times and departure times of all points in S. For each $i \in [l]$, let $T_i = [left_i, right_i]$ and $|T_i| = right_i - left_i$ be the length of T_i. Note that $left_1 = 0$ and $right_l = T$. By construction, the set of present points only changes at the end-points of the intervals and stays the same within each interval. Accordingly, given the optimal offline positions $X_1, \ldots, X_n \in P$,

$$OPT_C(S; P) = \int_0^T d_{min}(t; X_1, \ldots, X_n)dt = \sum_{i \in [l]} |T_i| d_{min}(T_i; X_1, \ldots, X_n),$$

where $d_{min}(T_i; X_1, \ldots, X_n)$ is the minimum distance incurred by $X_1, \ldots, X_n$ at any time t within T_i (that is, $t \in (left_i, right_i)$).

Now we consider the minimum distance incurred by algorithm $\mathcal{A}_{CD}$ in each T_i. For each $i \in [l]$, let $S_i = \{j | j \in [n], s_j < right_i, d_j > left_i\}$ be the set of points that overlap with T_i, and $n_i = |S_i|$. Again by Claim 11 and in particular by property $\Phi.2$, all points in S_i are removed from S in the first n_i rounds of algorithm $\mathcal{A}_{CD}$ and

$$S_i \subseteq \mathcal{I}_1 \cup \mathcal{I}_2 \cup \cdots \cup \mathcal{I}_{2n_i}.$$

Accordingly, all points in S_i are located at the first $2n_i$ positions created by algorithm $\mathcal{A}_{ATWC}$ from time 0 to time $n_i - 1$. Since $\mathcal{A}_{ATWC}$ is a σ-approximation algorithm, it ensures that at time $n_i - 1$ in the online instance, the minimum distance incurred by positions $x_1, \ldots, x_{2n_i}$ is at least $\frac{Disp(2n_i;P)}{\sigma}$. Thus the cumulative distance in T_i according to $\mathcal{A}_{CD}$ is

$$|T_i| d_{min}(T_i; x_1, \ldots, x_{2n_i}) \geq \frac{|T_i| Disp(2n_i; P)}{\sigma},$$

and the total cumulative distance is

$$\int_0^T d_{min}(t; x_1, \ldots, x_{2r}) dt = \sum_{i \in [l]} |T_i| d_{min}(T_i; x_1, \ldots, x_{2n_i}) \geq \sum_{i \in [l]} \frac{|T_i| Disp(2n_i; P)}{\sigma}.$$

Again because the set of present points stays the same throughout T_i, we have $d_{min}(T_i; X_1, \ldots, X_n) \leq Disp(n_i; P)$ and

$$OPT_C(S; P) \leq \sum_{i \in [l]} |T_i| Disp(n_i; P).$$

Therefore the approximation ratio is no larger than

$$\frac{\sigma \sum_{i \in [l]} |T_i| Disp(n_i; P)}{\sum_{i \in [l]} |T_i| Disp(2n_i; P)} \leq \sigma \cdot \max_{i \in [l]} \frac{Disp(n_i; P)}{Disp(2n_i; P)} \leq \sigma \max_{i \geq 1} \frac{Disp(i; P)}{Disp(2i; P)},$$

as we wanted to show. Finally, all that remains is to prove the following claim.

Claim 12. *For any $k \geq 1$, $i \geq 1$ and k-dimensional polytope P,*

$$\frac{Disp(i; P)}{Disp(2i; P)} \leq 2.$$

Proof. Use $\{x_l\}_{l \in [i]}$ to denote the optimal positions for dispersing i points in P and let $\delta = \left(\frac{Disp(i;P)}{4}, 0, 0, \ldots, 0\right)$ be a k-dimensional vector. Now we can define a set X' of $2i$ positions,

$$X' = \{x'_j | x'_j = x_{\lceil j/2 \rceil} + \delta(-1)^{j \bmod 2}\}_{j \in [2i]}.$$

That is, $x'_{2\lceil j/2 \rceil - 1}$ and $x'_{2\lceil j/2 \rceil}$ are obtained by shifting the first coordinate of $x_{\lceil j/2 \rceil}$ by $\frac{Disp(i;P)}{4}$ in two directions. By the definition of X', for any j and j' in $[2i]$,

- if $\lceil j/2 \rceil = \lceil j'/2 \rceil$ then

$$dis(x'_j, x'_{j'}) = \frac{Disp(i; P)}{2}; \tag{15}$$

- if $\lceil j/2 \rceil \neq \lceil j'/2 \rceil$ then

$$dis(x'_j, x'_{j'}) \geq \frac{Disp(i;P)}{2}. \tag{16}$$

Indeed, Equation 15 is because x'_j and $x'_{j'}$ only differ at their first coordinates by $\frac{Disp(i;P)}{2}$; and Equation 16 is by applying the triangle inequality twice and because $dis(x_{\lceil j/2 \rceil}, x_{\lceil j'/2 \rceil}) \geq Disp(i;P)$:

$$\begin{aligned} & dis(x'_j, x'_{j'}) \geq dis(x'_j, x_{\lceil j'/2 \rceil}) - dis(x'_{j'}, x_{\lceil j'/2 \rceil}) = dis(x'_j, x_{\lceil j'/2 \rceil}) - \frac{Disp(i;P)}{4} \\ \geq \quad & dis(x_{\lceil j/2 \rceil}, x_{\lceil j'/2 \rceil}) - dis(x'_j, x_{\lceil j/2 \rceil}) - \frac{Disp(i;P)}{4} = dis(x_{\lceil j/2 \rceil}, x_{\lceil j'/2 \rceil}) - \frac{Disp(i;P)}{2} \geq \frac{Disp(i;P)}{2}. \end{aligned}$$

Moreover, for any $j \in [2i]$, since x'_j differs from $x_{\lceil j/2 \rceil}$ only in the first coordinate by $\frac{Disp(i;P)}{4}$,

$$dis(x'_j, \partial P) \geq dis(x_{\lceil j/2 \rceil}) - \frac{Disp(i;P)}{4} \geq Disp(i;P) - \frac{Disp(i;P)}{4} > \frac{Disp(i;P)}{2}. \tag{17}$$

By Equations 15, 16 and 17, and by the definition of $Disp(2i;P)$, we have

$$Disp(2i;P) \geq d_{min}(X') \geq \frac{Disp(i;P)}{2},$$

and Claim 12 holds. □

By Claim 12, $\max\limits_{i \geq 1} \frac{Disp(i;P)}{Disp(2i;P)} \leq 2$ and Theorem 8 holds. □

Combining Theorems 2, 5, 7 and 8, we immediately have the following.

Corollary 1. *For the CD problem, there exists a polynomial-time offline algorithm that achieves approximation ratio arbitrarily close to* $4 \ln 2$ *in dimension 1; approximation ratio* 3.182 *for 2-dimensional squares; and approximation ratio* $\frac{4}{1-\epsilon}$ *for any* $k \geq 2$*, k-dimensional polytope* P *and* $\epsilon > 0$*.*

F Dispersion Without the Boundary Condition

Sometimes literature considers a similar problem when the distance to the boundary is not taken into consideration (which referred as the *spreading points problem* [5] or *facility dispersion* [25]). The objective is

$$SP(n;P) \triangleq \max_{X_1,\ldots,X_n \in P} \min_{i,j \in [n]} dis(X_i, X_j).$$

Recall that $T = \max_{i \in [n]} d_i$ and given locations $X = (X_1, \ldots, X_n)$, for any $t \leq T$, similarly let

$$d^{SP}_{min}(t;X) = \min_{i,j \in [n]: s_i \leq t \leq d_i, s_j \leq t \leq d_i} dis(X_i, X_j)$$

be the minimum distance among the points that are present at time t. When X is clear from the context, we may write $d^{SP}_{min}(t)$ for short. Here we also consider two natural objectives: the *all-time worst-case* (ATWC) problem, where the objective is

$$OPT^{SP}_A(S;P) \triangleq \max_{X_1,\ldots,X_n} \min_{t \leq T} d^{SP}_{min}(t);$$

and the *cumulative distance* (CD) problem, where the objective is

$$OPT_C^{SP}(S;P) \triangleq \max_{X_1,\ldots,X_n} \int_0^T d_{min}^{SP}(t)dt.$$

Similar to Claim 2, $\forall\ S = ((s_1,d_1),\ldots,(s_n,d_n))$, letting $m = \max_{t\le T} |\{i : s_i \le t \le d_i\}|$, we have $OPT_A^{SP}(S;P) = SP(m;P)$.

In the dispersion without boundary condition problem, the optimal solution and approximation may be completely different from the dispersion problem. For instance, consider the polytope shown in Figure 8 and when the number of points is small, the optimal dispersion without boundary condition solution allocates most points in the left part of the polytope while the optimal dispersion solution allocates most points in the right part.

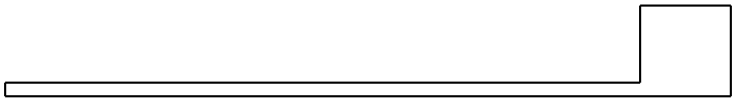

Figure 8: A 2-dimensional polytope

However, we show that most of the ideas for the dispersion problem can be carried through the dispersion without boundary condition problem. For the one dimensional case, by allocating the first two positions at the two end points of the segment, we achieve $2\ln 2$ approximation lower- and upper- bounds for the ATWC dispersion without boundary problem. For arbitrary k-dimensional polytope, by adopting the similar analysis in Appendix D, the dispersion without boundary condition problem has similar lower- and upper- bounds. Formally we have the following theorems.

Theorem 9. *For any $k \ge 2$, no algorithm achieves an approximation ratio better than $\frac{3\sqrt{2}}{4}$ for the online ATWC dispersion without boundary condition problem for arbitrary polytopes.*

Proof. Similar to the proof in Theorem 6, we only consider the case when the polytope is a k-dimensional sphere. Without loss of generality, let the center of the sphere be $(0,0,\ldots,0)$ and the radius be $\frac{1}{2}$. Clearly $SP(1;P) = \infty, SP(2;P) = 1$. Next we focus on the case allocating the third point in P and we have the following claim.

Claim 13. *For any algorithm $\mathcal{A}$ for $SP(3;P)$, there exists an algorithm $\mathcal{A}'$ such that $\mathcal{A}'$ locates the three points on the boundary of a 2-D circle centered at $(0,0,\ldots,0)$ and the approximation ratio of $\mathcal{A}'$ to $SP(3;P)$ is no worse than $\mathcal{A}$.*

The proof is similar to Claim 9 and quite intuitive, thus omitted here. With Claim 13, we can without loss of generality assume the three points are allocated at $p_1 = (-\frac{1}{2},0,0,\ldots,0)$, $p_2 = (x,y,0,\ldots,0)$, $p_3 = (x',y',0,\ldots,0)$, where $x^2+y^2 = x'^2+y'^2 = \frac{1}{4}$. Obviously the optimal distance for the offline problem is achieved by allocating three points evenly and hence $x = x' = \frac{1}{4}, y = -y' = \frac{\sqrt{3}}{4}$, $SP(3;P) = \frac{\sqrt{3}}{2}$. For the online problem, we assume without loss of generality that $y \ge 0$ and hence $p_2 = (x, \sqrt{\frac{1}{4}-x^2}, 0,\ldots,0)$. Given p_1, p_2 fixed, the optimal allocation for p_3 is on the vertical bisector of p_1p_2. Thus,

$$dis(p_1,p_2) = \sqrt{\frac{1}{2}+x}$$

$$dis(p_1,p_3) = \sqrt{\frac{1}{2}+\sqrt{\frac{1}{8}-\frac{x}{4}}}$$

By selecting x minimizing

$$\max\left\{\frac{SP(2;P)}{\sqrt{\frac{1}{2}+x}}, \frac{SP(3;P)}{\min\{\sqrt{\frac{1}{2}+x}, \sqrt{\frac{1}{2}+\sqrt{\frac{1}{8}-\frac{x}{4}}}\}}\right\},$$

we have $x = \frac{7}{18}$ and the approximation ratio cannot be better than $\frac{3\sqrt{2}}{4} \approx 1.06$. □

Theorem 10. *For any $k \geq 2$, polytope P with with covering rate at least e and $\epsilon > 0$, there exists a polynomial-time algorithm for the online ATWC dispersion without boundary condition problem, with approximation ratio $\frac{2}{1-\epsilon}$ and running time polynomial in $\frac{1}{(\gamma\epsilon)^k}$..*

Proof. We prove the 2-approximation for the greedy algorithm by induction. First when there is only one point in P, the optimal distance is equal to the maximal distance given by the greedy algorithm since the boundary is not taken into consideration. Suppose when there are n points in P and the allocation of the n points satisfies 2-approximation, we find a feasible position for the $(n+1)th$ point satisfying the requirement. Note that for arbitrarily fixed n points $\{p_1, \ldots, p_n\}$ where p_i is in P for $i \in [n]$, there exists a point p^* such that

$$dis(p_i, p^*) \geq \frac{SP(n+1;P)}{2}, \forall i \in [n].$$

This is true because otherwise n spheres centered at $\{p_1, \ldots, p_n\}$ with radius $\frac{SP(n+1;P)}{2}$ is a covering for P. Thus for $n+1$ points arbitrarily allocated in P, there exists $i \in [n+1]$ such that there are at least two different points allocated in the same sphere centered at p_i, excluding the boundary. Therefore, the minimum distance among those two points, and hence the minimum distance among $n+1$ points is less than $SP(n+1;P)$. Thus we derive a contradiction and the above inequality holds. By allocating the $(n+1)$th point at position p^* gives us the 2-approximation.

The method to approximate the greedy mechanism is similar to the proof of Theorem 7, but not exactly the same. Note that when the boundary is not taken into consideration, there may not exist a cube whose center is in P and its distance to the greedy solution (denoted by p^*) is small. However, the distance from p^* to any point within the cube which contains p^* is small. By brute force searching the cube which intersects with P and whose center has largest distance to the existing points, allocating any point in the intersection of the cube and P gives us a $\frac{2}{1-\epsilon}$-approximation with running time polynomial in $\frac{1}{(\gamma\epsilon)^k}$. □

The 2-D ATWC dispersion without boundary problem in a square can have a better approximation by carefully selecting the first few positions and then applying greedy to it. The analysis and intuition are similar to the proof of Theorem 5, thus omitted here.

The offline CD dispersion without boundary problem is similar to the dispersion problem. We can reduce the offline CD instance to the online ATWC instance with an increase of factor 2 in the approximation ratio. See Appendix E for details.

References

[1] S. Abravaya and M. Segal. Maximizing the number of obnoxious facilities to locate within a bounded region. *Computers & Operations Research*, 37(1):163–171, 2010.

[2] C. Baur and S. P. Fekete. Approximation of geometric dispersion problems. *Algorithmica*, 30(3):451–470, 2001.

[3] B. Ben-Moshe, M. J. Katz, and M. Segal. Obnoxious facility location: Complete service with minimal harm. *International Journal of Computational Geometry & Applications*, 10(06):581–592, 2000.

[4] M. Benkert, J. Gudmundsson, C. Knauer, E. Moet, R. van Oostrum, and A. Wolff. A polynomial-time approximation algorithm for a geometric dispersion problem. In *International Computing and Combinatorics Conference*, pages 166–175. Springer, 2006.

[5] S. Cabello. Approximation algorithms for spreading points. *Journal of Algorithms*, 62(2):49–73, 2007.

[6] H. Cohn, A. Kumar, S. D. Miller, D. Radchenko, and M. Viazovska. The sphere packing problem in dimension 24. *arXiv:1603.06518*, 2016.

[7] A. Dumitrescu and M. Jiang. Dispersion in disks. *Theory of Computing Systems*, 51(2):125–142, 2012.

[8] L. Epstein and R. Van Stee. Optimal online algorithms for multidimensional packing problems. *SIAM Journal on Computing*, 35(2):431–448, 2005.

[9] L. Epstein and R. Van Stee. Bounds for online bounded space hypercube packing. *Discrete optimization*, 4(2):185–197, 2007.

[10] L. Fejes. Über die dichteste kugellagerung. *Mathematische Zeitschrift*, 48(1):676–684, 1942.

[11] S. P. Fekete and H. Meijer. Maximum dispersion and geometric maximum weight cliques. *Algorithmica*, 38(3):501–511, 2004.

[12] D. Fotakis. Incremental algorithms for facility location and k-median. *Theoretical Computer Science*, 361(2):275–313, 2006.

[13] D. Fotakis. On the competitive ratio for online facility location. *Algorithmica*, 50(1):1–57, 2008.

[14] R. J. Fowler, M. S. Paterson, and S. L. Tanimoto. Optimal packing and covering in the plane are NP-complete. *Information processing letters*, 12(3):133–137, 1981.

[15] T. C. Hales. A proof of the Kepler conjecture. *Annals of mathematics*, 162(3):1065–1185, 2005.

[16] M. Hifi and R. M'hallah. A literature review on circle and sphere packing problems: models and methodologies. *Advances in Operations Research*, 2009, 2009.

[17] D. S. Hochbaum and W. Maass. Approximation schemes for covering and packing problems in image processing and VLSI. *Journal of the ACM (JACM)*, 32(1):130–136, 1985.

[18] P. Hokama, F. K. Miyazawa, and R. C. Schouery. A bounded space algorithm for online circle packing. *Information Processing Letters*, 116(5):337–342, 2016.

[19] W. Huang and T. Ye. Greedy vacancy search algorithm for packing equal circles in a square. *Operations Research Letters*, 38(5):378–382, 2010.

[20] M. J. Katz, K. Kedem, and M. Segal. Improved algorithms for placing undesirable facilities. *Computers & Operations Research*, 29(13):1859–1872, 2002.

[21] A. Meyerson. Online facility location. In *42rd Annual IEEE Symposium on Foundations of Computer Science*, pages 426–431. IEEE, 2001.

[22] F. K. Miyazawa, L. L. Pedrosa, R. C. Schouery, M. Sviridenko, and Y. Wakabayashi. Polynomial-time approximation schemes for circle packing problems. In *European Symposium on Algorithms*, pages 713–724. Springer, 2014.

[23] R. Peikert, D. Würtz, M. Monagan, and C. de Groot. *Packing circles in a square: A review and new results*, pages 45–54. Springer, 1992.

[24] Z. Qin, Y. Xu, and B. Zhu. On some optimization problems in obnoxious facility location. In *International Computing and Combinatorics Conference*, pages 320–329. Springer, 2000.

[25] S. S. Ravi, D. J. Rosenkrantz, and G. K. Tayi. Heuristic and special case algorithms for dispersion problems. *Operations Research*, 42(2):299–310, 1994.

[26] C. A. Rogers. Existence theorems in the geometry of numbers. *Annals of Mathematics*, pages 994–1002, 1947.

[27] J. Schaer. The densest packing of nine circles in a square. *Canad. Math. Bull*, 8:273–277, 1965.

[28] J. Schaer and A. Meir. On a geometric extremum problem. *Canad. Math. Bull*, 8:21–27, 1965.

[29] B. Schwartz. Separating points in a square. *J. Recr. Math*, 3:195–204, 1970.

[30] S. S. Seiden. On the online bin packing problem. *Journal of the ACM (JACM)*, 49(5):640–671, 2002.

[31] P. G. Szabó and E. Specht. Packing up to 200 equal circles in a square. In *Models and Algorithms for Global Optimization*, pages 141–156. Springer, 2007.

[32] M. Viazovska. The sphere packing problem in dimension 8. *arXiv:1603.04246*, 2016.