Efficient Approximations for the Online Dispersion Problem*†

Jing Chen[1], Bo Li[2], and Yingkai Li[3]

1 **Department of Computer Science, Stony Brook University, Stony Brook, NY, USA**
jingchen@cs.stonybrook.edu
2 **Department of Computer Science, Stony Brook University, Stony Brook, NY, USA**
boli2@cs.stonybrook.edu
3 **Department of Computer Science, Stony Brook University, Stony Brook, NY, USA**
yingkli@cs.stonybrook.edu

Abstract

The *dispersion* problem has been widely studied in computational geometry and facility location, and is closely related to the packing problem. The goal 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 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 in Euclidean space.

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 cumulative distance, this remains 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 $(2\ln 2+\epsilon)$-competitive, 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-competitive algorithm and a 1.183 lower-bound. Furthermore, for arbitrary k-dimensional polytopes with $k \geq 2$, we provide a $\frac{2}{1-\epsilon}$-competitive algorithm 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. Interestingly, 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 competitive ratio increases by a factor of at most 2. Our techniques also apply to online dispersion problems with different boundary conditions.

1998 ACM Subject Classification F.2.2 Nonnumerical Algorithms and Problems

Keywords and phrases dispersion, online algorithms, geometric optimization, packing, competitive algorithms

Digital Object Identifier 10.4230/LIPIcs.ICALP.2017.11

* A full version of this extended abstract is available at http://arxiv.org/abs/1704.06823.

† The authors thank Joseph Mitchell for motivating us to study the online dispersion 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 helpful comments.

44th International Colloquium on Automata, Languages, and Programming (ICALP 2017).
Editors: Ioannis Chatzigiannakis, Piotr Indyk, Fabian Kuhn, and Anca Muscholl;
Article No. 11; pp. 11:1–11:15

Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

1 Introduction

The problem of assigning elements to locations in a given area comes up only too often in real life: where to seat the customers in a restaurant, where to put certain facilities in a city, where to build 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 environmental safety, 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 many 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., [32, 3, 2, 4]. In particular, in the *dispersion* problem defined by [2], there is a k-dimensional polytope P and an integer n, 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, there is another important feature in all the scenarios mentioned above and many other real-world scenarios: 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. Indeed, 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. Online dispersion and facility location problems have been studied when the underlying locations are vertices of a graph [28, 17, 27]. In this paper we consider, for the first time in the literature, the online dispersion problem in Euclidean space. 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 throughout 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. As we will show, solutions for the online problem are already complex even on a segment. For 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.

The most technical parts are the online ATWC problem and the offline time-dependent CD problem. Interestingly, we provide an efficient reduction from the offline CD problem to the online ATWC problem, and show that in order to solve the former, one can use an algorithm for the latter as a black-box. For the online ATWC problem, it is not hard to see that a natural greedy algorithm provides a 2-competitive ratio. Our main contributions for this problem are to provide an efficient algorithm that is optimal for the 1-dimensional case, improve the competitive ratio and prove a lower-bound for squares, and provide an efficient implementation of the greedy algorithm for the general case. We also prove a simple lower-bound for the general case.

To establish our results, we show interesting new connections between dispersion and ball-packing – both *uniform* packing (i.e., with balls of identical radius) and *non-uniform*

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

	Online	Offline time-dependent
ATWC	$k = 1$: a $2\ln 2$ (≈ 1.386) lower-bound and an optimal algorithm; see Theorems 5 and 8. $k = 2$: a 1.183 lower-bound and a 1.591-competitive algorithm for squares; see Theorems 11 and 12. $k \geq 2$: a $\frac{7}{6}$ lower-bound and a $\frac{2}{1-\epsilon}$-competitive algorithm for arbitrary polytopes P; see Theorems 13 and 14.	Equivalent to dispersion without time; see Claim 2.
CD	No constant competitive algorithm even when $k = 1$; see Claim 1.	A black-box reduction to online ATWC for arbitrary k and P, with the competitive ratio scaling up by at most 2; see Theorem 15.

packing (i.e., with balls of different radii). 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 inapproximability results hold even when running time is not a concern. Due to lack of space, most proofs are given in the full version [8].

Discussion and future directions. An algorithm for high-dimensional polytopes may not be directly applicable in dimension 1, because all locations are on the boundary when a segment is treated as a high-dimensional polytope, and the minimum distance is always 0. Accordingly, we do not know whether the lower-bound for dimension 1 carries through to higher dimensions, and proving better lower-bounds will be an interesting problem for future studies. In [8], we consider online dispersion without the boundary constraint, where the lower-bound for dimension 1 indeed carries through. We show all our algorithms can be adapted for this setting. Another important future direction is to understand the role of randomized algorithms in the online dispersion problem. Finally, improving the (deterministic or randomized) algorithms' competitive ratios in various classes of polytopes is certainly a long-lasting theme for the online dispersion problem. Special classes such as regular polytopes and uniform polytopes may be reasonable starting points. Given the connections between dispersion and ball-packing, it is conceivable that new competitive algorithms for online dispersion may stem from and also imply new findings on ball-packing.

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 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

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

obstacles. [11] considers the problem of selecting n points in n given unit-disks, one per disk, and the objective is to maximize the minimum distance.

In discrete settings, [32, 5] show that, if the distances among the candidate locations do not satisfy triangle inequality, then there is no polynomial-time constant approximation for either objective, unless P=NP; while if triangle inequality is satisfied, then there are efficient 2-approximations for both objectives. If the goal is to maximize total distance and the candidate locations are in a k-dimensional space, [15] gives a PTAS under L_1-norm; and [6, 7] provide PTASes when locations need to satisfy matroid constraints. Finally, [3, 31, 25, 1] consider various dispersion problems in obnoxious facility allocation.

Packing without time. It is well known that dispersion and packing are "dual" problems of each other [26]. In this paper we show interesting new 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 work has been done on different variants of the problem; see [30, 21] for surveys on this topic.

One important problem is to *pack circles with identical radius, as many as possible, in a bounded region.* [18] shows this problem to be strongly NP-hard and [22] gives a PTAS for it. An APTAS for the *circle bin packing* problem is given in [29]. 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 a small constant; see [36, 35, 37, 30]. Heuristic methods have also been used in finding approximations when the number of circles gets large [24, 41]. 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 [14] and for dimension 3 in [20]. Very recently, [42] and [9] 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 [33].

Online geometric optimization. Many important geometric optimization problems have been studied in online settings, although the settings and objectives are quite different from ours. In particular, the seminal work of [38] provides a nearly-optimal competitive algorithm for the classic *online bin-packing* problem. Algorithms for variants of the problem have been considered ever since, such as a constant competitive ratio for packing circles in square bins [23], and constant competitive ratios for bin-packing in higher dimensions [12, 13].

In *online facility location* [28], 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 [28], when the demands arrive adversarially, there is a randomized polynomial-time $O(\log n)$-competitive algorithm, and a constant competitive ratio is impossible. A deterministic $O(\frac{\log n}{\log \log n})$-competitive algorithm and a matching lower-bound are provided in [17]. In *incremental facility location* [16], the facilities can be opened, closed or merged, depending on the arriving demands. In [27, 34], there is a cost for each location configuration and the goal is to minimize the cost when the facilities arrive online. A constant competitive algorithm for this problem is provided in [27], and [34] gives a reduction from the online problem to the offline version of the problem.

Dynamic resource division. Fair resource division is an important problem in economics [39, 10, 40]. When the resource is 1-dimensional and homogenous, dynamic fair division

is in some sense the "dual" of online dispersion: locating n points as far as possible from each other and from the boundary is the same as partitioning the segment into $n+1$ pieces as evenly as possible. [19] provides an optimal d-disruptive mechanism for 1-dimensional homogenous resource. Interestingly, our algorithm for the 1-dimensional case provides an optimal mechanism when $d = 1$, although the techniques are quite different. Optimal mechanisms for heterogenous or high-dimensional resource remain unknown. It would be interesting to see if our techniques for dispersion can be used in resource division in general.

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, $X_1, \ldots, X_n \in P$, for 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 [8], 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 an arrival/departure event. It must decide the location X_i for i upon its arrival, knowing neither the future events nor the number n. An adversary knows how the algorithm works and chooses future events after seeing the algorithm's output 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 defined to be the optimum of the corresponding *offline* problems, the same as the *ex-post* optimum for the online problems. Below we provide two simple observations about the objectives.

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

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, $m = \max_{t\leq T} |\{i : s_i \leq t \leq d_i\}|$.

▶ **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, the online CD problem is highly inapproximable and the offline ATWC problem is equivalent to the dispersion problem without time. Thus we will focus on the online ATWC problem and the offline CD problem, especially the former. Our results imply a simple $O(n)$-competitive algorithm for the online CD problem (see [8]), matching the lower-bound in Claim 1.

Below we point out some connections between dispersion and ball-packing: they are not hard to show, and similar results for the dispersion problem without the boundary condition have been pointed out in [26]. More precisely, the *(uniform) ball-packing* problem [22] 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. We 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 balls with identical radius that can be packed in P, together with a corresponding packing. 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 it 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 $Pack(r; P)$;*
(2) *any polynomial-time algorithm for $Pack(r; P)$ implies an FPTAS for $Disp(n; P)$.*

To the best of our 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 [18], 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 not so clear and worth further investigation: for example, it would be interesting to know if there exists a counterpart of Claim 4 when the polytope does not have an insphere.

Finally, the *insert-only* model, where all points have the same departure time, is a special case of our general model. Interestingly, as will become clear in our analysis, the difficulty of the general online ATWC problem is captured by the problem under this special model. The insert-only model was also considered by [27, 34] in settings different from ours and with a different objective function. We further discuss this model in [8].

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 competitive ratio of any algorithm, even computationally unbounded ones.

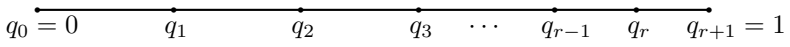

Figure 1 The pre-fixed positions in Q for dimension 1.

3.1 The Lower Bound

▶ **Theorem 5.** *No online algorithm achieves a competitive ratio better than* $2\ln 2$ *(*≈ 1.386*) for the 1-dimensional ATWC problem.*

Proof Ideas. Letting $\sigma'_r = \sum_{i=r+1}^{2r} \frac{1}{i}$ for any positive integer r, we show that no algorithm achieves a competitive 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 α-competitive to OPT_A with $\alpha < 2\sigma'_r$, it must be α-competitive after the arrival of each point, as it does not know the total number of points. Thus for each arriving point, there must exist an interval long enough such that putting the new point inside the interval does not violate the competitive ratio. We show that in order for $\mathcal{A}$ to do so, the segment must be longer than P itself, a contradiction. Theorem 5 holds by setting $r \to \infty$. The complete proof is in [8]. ◀

3.2 A Polynomial-Time Online Algorithm

Next, we provide a deterministic polynomial-time online algorithm whose competitive 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 competitive 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 5 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 $|(q_{j'-1}, q_{j'})| < |(q_{j-1}, q_j)| < 2|(q_{j'-1}, q_{j'})|$.

Our algorithm, Algorithm 1, also takes as parameter an ordering for the positions in Q, denoted by $\tau = (\tau_1, \tau_2, \ldots, \tau_r)$. We have the following two lemmas, whose main ideas are sketched 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 6.** *$\forall$ r and τ, Algorithm 1 is $2\sigma_r$-competitive to OPT_A for any S with $m > r$.*

Proof Ideas. Since Algorithm 1 only creates a new position when the number of points simultaneously present on the line increases, for any time t, the number of positions created is exactly the maximum number of points that has appeared simultaneously on the line. Thus only m positions is created for instance S. We prove that when $m > r$, the minimum distance produced by our algorithm, denoted by $d_{min}(m)$, is $\frac{1}{2^{l+1}\sigma_r(r+i)}$, where l, i are the unique integers such that $l \geq 0, 0 \leq i \leq r+1$ and $2^l(r+1) + 2^l(i-1) \leq m < 2^l(r+1) + 2^l i$. Note that the minimum distance only depends on m. By comparing OPT_A with $d_{min}(m)$, we show that the competitive ratio $2\sigma_r$ holds for all $m > r$. ◀

▶ **Lemma 7.** *For any integer $l > 0$ and $r = 2^l - 1$, there exists an ordering τ for the corresponding set Q, s.t. Algorithm 1 is $2\sigma_r$-competitive to OPT_A for any S with $m \leq r$.*

Proof Ideas. Interestingly, due to the structure of Algorithm 1, we only need to consider the instance $S = ((1, r+1), (2, r+1), \ldots, (r, r+1))$. We construct an ordering $\tau = \{\tau_d\}_{d \in [r]}$ for Q such that the competitive 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. Given any $d = 2^i + s$ with $i \in \{0, 1, \ldots, l-1\}$ and $s \in \{0, 1, \ldots, 2^i - 1\}$, we have $\tau_d = q_{2^{l-i-1}(2s+1)}$. Denoting the competitive ratio at time d by $apx(d)$, we prove that

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

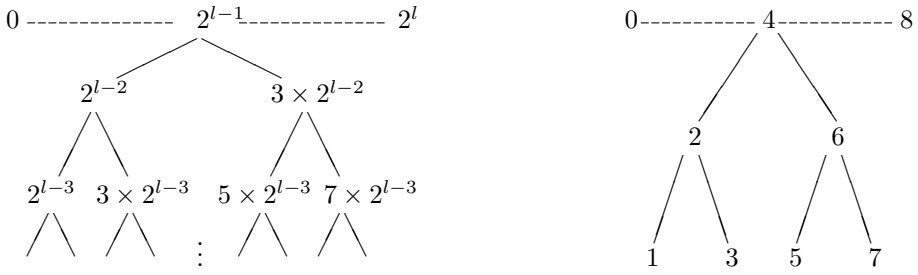

Figure 2 The left-hand side shows the top three levels of the binary tree for a general l, with $\tau = (q_{2^{l-1}}, q_{2^{l-2}}, q_{3\times 2^{l-2}}, q_{2^{l-3}}, q_{3\times 2^{l-3}}, q_{5\times 2^{l-3}}, q_{7\times 2^{l-3}}, \dots)$. The right-hand side shows the complete binary tree for $l = 3$,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

Writing $apx(d)$ as $apx(i, s)$, we prove that, fixing i, $apx(i, s)$ is strictly increasing in s; and letting $s = 2^i - 1$, $apx(i, s)$ is strictly increasing in i. Thus the worst competitive ratio occurs at $i = l-1$ and $s = 2^{l-1} - 1$. As $apx(l-1, 2^{l-1}-1) = (2 - \frac{1}{2^l})\sigma_r < 2\sigma_r$, Lemma 7 holds. ◀

The theorem below follows easily from the above two lemmas.

▶ **Theorem 8.** *There exists a deterministic polynomial-time online algorithm for the ATWC problem, whose competitive ratio can be arbitrarily close to* $2\ln 2$. *Moreover, the running time is polynomial in* $\frac{1}{\epsilon}$ *for competitive ratio* $2\ln 2 + \epsilon$.

Remark. When the number of 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 5, 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 competitive ratio is exactly $2\ln 2$: see Theorem 9. However, this solution involves irrational numbers and cannot be exactly computed in polynomial time.

▶ **Theorem 9.** *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 competitive ratio is* $2\ln 2$.

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 competitive algorithm must rely on some proper upper- and lower-bounds for $Disp(n; P)$, which is part of the reason why the resulting bounds are not tight. In particular, we have the following.

▶ **Lemma 10.** *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.

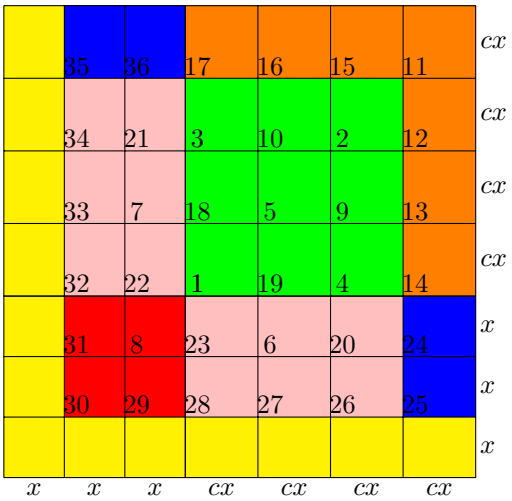

Figure 3 The set of pre-fixed positions, $Q = \{q_1, \ldots, q_{36}\}$, and the grid created by Q. A grid point labelled by i indicates the position q_i. The colored areas are used in the algorithm's description. More specifically, denoting a rectangle by the position in Q at its lower-left corner, the 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.

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 competitive 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.

▶ **Theorem 11.** *No online algorithm achieves a competitive 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 1.591-competitive to OPT_A. Similar to Algorithm 1, we 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 in Figure 3. The q_i's indices specify the order according to which they should be occupied, thus we do not need an extra ordering τ. Note 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.

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 2 is similar to Algorithm 1. It uses in Step 8 a sub-routine to implement (1) and (2) above: the *Position Creation Phase,* as defined in [8], where we further provide some intuition on the choices of Q, x, and c. Setting $c = 1.271$, we have the following.

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 [8].
    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

▶ **Theorem 12.** *Algorithm 2 runs in polynomial time and is* 1.591*-competitive for the 2-dimensional online ATWC problem in a square.*

Note that the upper-bound for $Disp(n; P)$ in Lemma 10 is not tight when n is small. With better upper-bounds for $Disp(n; P)$, better competitive ratios for our algorithm can be directly obtained via a similar analysis. Moreover, we believe the competitive 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. Finally, similar techniques can be used when P is a rectangle, but the gap between the lower- and upper-bounds will be even larger, and the analysis will be more complicated without adding much new insight to the problem. Thus we leave a thorough study on rectangles for the future.

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 polynomial-time algorithm for the online ATWC problem. Below we only state the theorems.

▶ **Theorem 13.** *For any $k \geq 2$, no online algorithm achieves a competitive ratio better than $\frac{7}{6}$ for the ATWC problem for arbitrary polytopes.*

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. Note that, although a natural greedy algorithm provides a 2-competitive ratio, the exact greedy solution may not be computable in polynomial time. Here we show the greedy algorithm can be efficiently approximated arbitrarily closely. 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.

Algorithm 3. Algorithm $\mathcal{A}_{\mathcal{I}}$ for computing $\mathcal{I} = \{\mathcal{I}_1, \mathcal{I}_2\}$ 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 end-points of a "sliding window" for the arriving times under consideration.)
Let $index = 1$.
while d $\neq T$ **do**
  Let $\hat{S} = \{i | i \in S, s_i > s, s_i \leq d, d_i > d\}$.
  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 $\mathcal{I} = \{\mathcal{I}_1, \mathcal{I}_2\}$.

▶ **Theorem 14.** *For any constants $\gamma, \epsilon > 0$, integer $k \geq 2$ and k-dimensional polytope P with covering rate at least γ, there exists a deterministic polynomial-time online algorithm for the ATWC problem, with competitive ratio $\frac{2}{1-\epsilon}$ and running time polynomial in $\frac{1}{(\gamma\epsilon)^k}$.*

6 The General k-Dimensional Offline CD Problem

By Claim 1, no online algorithm provides a good competitive ratio for the CD problem, thus we focus on the offline problem. 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. Our algorithm will be such that, *in each time interval,* the minimum distance is a good approximation to the optimal dispersion problem without time, for the points present in this interval.

Interestingly, this is achieved by reducing the offline CD problem to the online ATWC problem, for any dimension k and polytope P. To carry out this idea, we first provide a polynomial-time algorithm $\mathcal{A}_{\mathcal{I}}$ (Algorithm 3) that, given a sequence S, selects a subset $\mathcal{I}$ of points from S. The set $\mathcal{I}$ satisfies the following properties, as proved in [8].

(Φ.1) $\mathcal{I}$ can be partitioned into two groups $\mathcal{I}_1$ and $\mathcal{I}_2$ such that the points in the same group have disjoint time intervals.

(Φ.2) For any time $0 \leq t \leq T$, if there are points in S present at time t, then at least one of them is selected to $\mathcal{I}$.

The offline CD algorithm $\mathcal{A}_{CD}$ uses algorithm $\mathcal{A}_{\mathcal{I}}$ to select $\mathcal{I}$ from its input S, eliminates the selected points from S, and repeats on the remaining S. Recall that m is the maximum number of points simultaneously present at any time. By property Φ.2, this procedure ends in at most m iterations. Based on the partitions constructed by $\mathcal{A}_{\mathcal{I}}$, $\mathcal{A}_{CD}$ constructs an instance of the online ATWC problem and uses any 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 4 and we have the following theorem. Below we sketch the main ideas.

Algorithm 4. $\mathcal{A}_{CD}$

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 15.** *$\forall\ k \geq 1$ and k-dimensional polytope P, given any polynomial-time σ-competitive online algorithm $\mathcal{A}_{ATWC}$ for ATWC, there is a polynomial-time offline algorithm $\mathcal{A}_{CD}$ for CD with competitive 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 Ideas. Given an input sequence S, we slice the whole time interval $[0, T]$ into smaller ones according to the arriving time and the departure time of each point. Denote these small intervals by $T_1, \ldots, T_l$, where l is the number of small intervals created. For each interval T_i, let S_i be the set of points that overlap with T_i and $n_i = |S_i|$. By properties $\Phi.1$ and $\Phi.2$, all points in S_i are eliminated from S in the first n_i iterations of $\mathcal{A}_{\mathcal{I}}$, thus are located at the first $2n_i$ positions created by $\mathcal{A}_{ATWC}$. The minimum distance among points in T_i (and to the boundary) is at least $\frac{Disp(2n_i;P)}{\sigma}$, since algorithm $\mathcal{A}_{ATWC}$ has competitive ratio σ. Thus, within each T_i, the competitive ratio to the optimal solution is upper-bounded by $\frac{\sigma Disp(n_i;P)}{Disp(2n_i;P)}$. Taking summation over all T_i's, the competitive ratio is upper-bounded by $\sigma \max_{i \geq 1} \frac{Disp(i;P)}{Disp(2i;P)}$. Finally, we prove $\max_{i \geq 1} \frac{Disp(i;P)}{Disp(2i;P)} \leq 2$, finishing the proof of Theorem 15. ◀

References

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

2 Christoph Baur and Sándor P. Fekete. Approximation of geometric dispersion problems. *Algorithmica*, 30(3):451–470, 2001.

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

4 Marc Benkert, Joachim Gudmundsson, Christian Knauer, Esther Moet, René van Oostrum, and Alexander Wolff. A polynomial-time approximation algorithm for a geometric dispersion problem. In *International Computing and Combinatorics Conference*, pages 166–175. Springer, 2006.

5 Benjamin Birnbaum and Kenneth J. Goldman. An improved analysis for a greedy remote-clique algorithm using factor-revealing LPs. *Algorithmica*, 55(1):42–59, 2009.

6 Alfonso Cevallos, Friedrich Eisenbrand, and Rico Zenklusen. Max-sum diversity via convex programming. In *Proceedings of the 32nd Symposium on Computational Geometry (SoCG)*, pages 26:1–26:14. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2016.

7 Alfonso Cevallos, Friedrich Eisenbrand, and Rico Zenklusen. Local search for max-sum diversification. In *Proceedings of the Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 130–142. SIAM, 2017.

8 Jing Chen, Bo Li, and Yingkai Li. Efficient approximations for the online dispersion problem. *arXiv:1704.06823*, 2017. Full version.

9 Henry Cohn, Abhinav Kumar, Stephen D. Miller, Danylo Radchenko, and Maryna Viazovska. The sphere packing problem in dimension 24. *arXiv:1603.06518*, 2016.

10 Lester E. Dubins and Edwin H. Spanier. How to cut a cake fairly. *The American Mathematical Monthly*, 68(1):1–17, 1961.

11 Adrian Dumitrescu and Minghui Jiang. Dispersion in disks. *Theory of Computing Systems*, 51(2):125–142, 2012.

12 Leah Epstein and Rob Van Stee. Optimal online algorithms for multidimensional packing problems. *SIAM Journal on Computing*, 35(2):431–448, 2005.

13 Leah Epstein and Rob Van Stee. Bounds for online bounded space hypercube packing. *Discrete optimization*, 4(2):185–197, 2007.

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

15 Sándor P. Fekete and Henk Meijer. Maximum dispersion and geometric maximum weight cliques. *Algorithmica*, 38(3):501–511, 2004.

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

17 Dimitris Fotakis. On the competitive ratio for online facility location. *Algorithmica*, 50(1):1–57, 2008.

18 Robert J. Fowler, Michael S. Paterson, and Steven L. Tanimoto. Optimal packing and covering in the plane are NP-complete. *Information processing letters*, 12(3):133–137, 1981.

19 Eric Friedman, Christos-Alexandros Psomas, and Shai Vardi. Dynamic fair division with minimal disruptions. In *Proceedings of the sixteenth ACM conference on Economics and Computation*, pages 697–713. ACM, 2015.

20 Thomas C. Hales. A proof of the Kepler conjecture. *Annals of mathematics*, 162(3):1065–1185, 2005.

21 Mhand Hifi and Rym M'hallah. A literature review on circle and sphere packing problems: models and methodologies. *Advances in Operations Research*, 2009:150624:1–150624:22, 2009.

22 Dorit S. Hochbaum and Wolfgang Maass. Approximation schemes for covering and packing problems in image processing and VLSI. *Journal of the ACM (JACM)*, 32(1):130–136, 1985.

23 Pedro Hokama, Flávio K. Miyazawa, and Rafael C. S. Schouery. A bounded space algorithm for online circle packing. *Information Processing Letters*, 116(5):337–342, 2016.

24 Wenqi Huang and Tao Ye. Greedy vacancy search algorithm for packing equal circles in a square. *Operations Research Letters*, 38(5):378–382, 2010.

25 Matthew J. Katz, Klara Kedem, and Michael Segal. Improved algorithms for placing undesirable facilities. *Computers & Operations Research*, 29(13):1859–1872, 2002.

26 Marco Locatelli and Ulrich Raber. Packing equal circles in a square: a deterministic global optimization approach. *Discrete Applied Mathematics*, 122(1):139–166, 2002.

27 Ramgopal R. Mettu and C. Greg Plaxton. The online median problem. *SIAM Journal on Computing*, 32(3):816–832, 2003.

28 Adam Meyerson. Online facility location. In *42rd Annual IEEE Symposium on Foundations of Computer Science*, pages 426–431. IEEE, 2001.

29 Flávio K Miyazawa, Lehilton L. C. Pedrosa, Rafael C. S. Schouery, Maxim Sviridenko, and Yoshiko Wakabayashi. Polynomial-time approximation schemes for circle packing problems. In *European Symposium on Algorithms*, pages 713–724. Springer, 2014.

30 Ronald Peikert, Diethelm Würtz, Michael Monagan, and Claas de Groot. Packing circles in a square: a review and new results. In *System Modelling and Optimization: Proceedings of the 15th IFIP Conference, Zurich, Switzerland, September 2–6, 1991*, pages 45–54. Springer, 1992.

31 Zhongping Qin, Yinfeng Xu, and Binhai Zhu. On some optimization problems in obnoxious facility location. In *International Computing and Combinatorics Conference*, pages 320–329. Springer, 2000.

32 Sekharipuram S. Ravi, Daniel J. Rosenkrantz, and Giri K. Tayi. Heuristic and special case algorithms for dispersion problems. *Operations Research*, 42(2):299–310, 1994.

33 Claude Ambrose Rogers. Existence theorems in the geometry of numbers. *Annals of Mathematics*, pages 994–1002, 1947.

34 Daniel J. Rosenkrantz, Giri K. Tayi, and S. S. Ravi. Obtaining online approximation algorithms for facility dispersion from offline algorithms. *Networks*, 47(4):206–217, 2006.

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

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

37 B. L. Schwartz. Separating points in a square. *J. Recr. Math*, 3:195–204, 1970.

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

39 Hugo Steinhaus. The problem of fair division. *Econometrica*, 16:101–104, 1948.

40 Walter Stromquist. How to cut a cake fairly. *The American Mathematical Monthly*, 87(8):640–644, 1980.

41 Péter Gábor Szabó and Eckard Specht. Packing up to 200 equal circles in a square. In *Models and Algorithms for Global Optimization*, pages 141–156. Springer, 2007.

42 Maryna Viazovska. The sphere packing problem in dimension 8. *arXiv:1603.04246*, 2016.