A Bouquet of Results on Maximum Range Sum: General Techniques and Hardness Reductions

RACHANA GUSAIN, SALADI RAHUL, and ADITYA SUBRAMANIAN, Indian Institute of Science, Bangalore, India

In this work we revisit the *maximum range sum* (MaxRS) problem which is well studied by the database and the computational geometry communities. The input is a set P of n weighted points in $\mathbb{R}^d$ and a geometric range Q (typically either an axis-aligned d-box or a d-ball). The goal is to design a fast algorithm to place Q in $\mathbb{R}^d$ so that the total weight of the points of P inside Q is maximized. We consider three natural variations of the MaxRS problem:

(1) In the *dynamic* MaxRS problem, points are inserted and deleted, and the goal is to efficiently update the placement of a d-ball. In $\mathbb{R}^d$ we present a randomized $(\frac{1}{2} - \varepsilon)$-approximation algorithm with update time $O_\varepsilon(\log n)$. The approximation factor holds with high probability. To the best of our knowledge, this problem was not studied before in the literature.

(2) In the *batched* MaxRS problem in $\mathbb{R}^1$, along with the points in P we are given m intervals of different lengths. The goal is to solve the MaxRS problem for each interval. We establish a conditional lower bound of $\Omega(mn)$ time for this problem, assuming the hardness of $(\min, +)$-convolution problem. Interestingly, this implies that the trivial upper bound of $O(mn \log n)$ for batched MaxRS in $\mathbb{R}^2$ is almost-tight. A similar lower bound is established for a related problem of *batched smallest k-enclosing interval*.

(3) In the *colored* MaxRS problem in $\mathbb{R}^d$, each point in P is assigned a color from $\{1, 2, \ldots, m\}$ and the goal is to find the placement of a d-ball Q that maximizes the number of *uniquely colored* points in $P \cap Q$. Prior work on this problem was limited to axis-aligned rectangle Q in $\mathbb{R}^2$. We obtain two new results for d-balls. The first result is a randomized $(\frac{1}{2} - \varepsilon)$-approximation algorithm with running time $O_\varepsilon(n \log n)$. Interestingly, the exponential dependence of $\log n$ on d is avoided in the running time. The second result improves upon the first result in $\mathbb{R}^2$ by providing a $(1 - \varepsilon)$-approximation algorithm with expected running time $O_\varepsilon(n \log n)$. The approximation factor holds with high probability for both results.

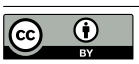
Our algorithms are obtained via two general techniques which we believe will be useful for solving other variants of MaxRS. The first technique provides a $(\frac{1}{2} - \varepsilon)$-approximation guarantee. The analysis relies on a volume argument involving d-balls and a randomized game. The second technique provides a $(1 - \varepsilon)$-approximation guarantee and works in two phases. In the first phase, we design an exact output-sensitive algorithm, and in the second phase, we speed up the exact algorithm by random sampling on colors.

CCS Concepts: • **Theory of computation → Computational geometry**; **Approximation algorithms analysis**.

Additional Key Words and Phrases: Computational geometry; Spatial databases; Randomized algorithms; Approximation algorithms

ACM Reference Format:
Rachana Gusain, Saladi Rahul, and Aditya Subramanian. 2025. A Bouquet of Results on Maximum Range Sum: General Techniques and Hardness Reductions. *Proc. ACM Manag. Data* 3, 5 (PODS), Article 273 (November 2025), 21 pages. https://doi.org/10.1145/3767709

Authors' Contact Information: Rachana Gusain, rachanag@iisc.ac.in; Saladi Rahul, saladi@iisc.ac.in; Aditya Subramanian, adityasubram@iisc.ac.in, Indian Institute of Science, Bangalore, Bangalore, India.



ACM 2836-6573/2025/11-ART273
https://doi.org/10.1145/3767709

1 Introduction

Maximum range sum (MaxRS) is a popular geometric placement problem that has been widely studied by the spatial database community [3, 4, 16, 17, 29, 38, 41, 43, 47, 52, 53, 58] and the computational geometry community [1, 8, 10, 12, 20, 32, 34, 48, 59]. The input is a set P of n weighted points in $\mathbb{R}^d$ and a geometric range Q (typically either an axis-aligned d-box or a d-ball). The goal is to design a fast algorithm to place Q in $\mathbb{R}^d$ so that the total weight of the points of P inside Q is maximized. In the unweighted setting, the goal is to place Q to cover the maximum number of points. See Figure 1a. In our work, the dimension d is small (say 5 or 8) and treated as a constant.

(a) MaxRS with unit weights. opt = 6.

(b) Colored MaxRS with opt = 3.

Fig. 1. MaxRS with Q being disks.

MaxRS shows up naturally in many spatial data analysis tasks, which involve detecting hotspots or regions of high activity or concentration. For example, during COVID times there were applications built by government agencies which had real-time data of the geographic location of the people currently infected and would display "hotspots" which had maximum patients. MaxRS can help political parties and companies make policy and infrastructure decisions. For example, Walmart has spatial data which contains the geographic location of its costumers. A MaxRS query can reveal the region with maximum number of costumers and hence, the decision of opening a new outlet can be made.

The MaxRS problem has primarily been studied for two classes of geometric ranges: axis-aligned d-dimensional hyper-rectangles (d-boxes for brevity) and d-dimensional euclidean balls (d-balls for brevity). The problem can trivially be solved in polynomial time (in terms of n) for d-boxes and d-balls, where d is a constant. For example, for 2-box (a.k.a. rectangle in $\mathbb{R}^2$) the optimal placement of Q is defined by at most four points, one point on each edge of the rectangle. The work on MaxRS was first initiated by [32] who presented an exact optimal $O(n \log n)$ time algorithm for 2-box (and later by [48]). The extension to d-box ($d \geq 3$) was done in [10] with running time $\widetilde{O}(n^{d/2})$. For 2-ball (a.k.a. disk in $\mathbb{R}^2$), Chazelle and Lee presented an exact $O(n^2)$ time algorithm [12], which was later shown to be optimal [5] (assuming hardness of 3SUM problem). Choi, Chung and Tao [17] perform an extensive study of the MaxRS problem in the I/O-model. [8, 11, 22–24, 59] are some other relevant papers on this problem.

Parallel efforts have been made to design $(1-\varepsilon)$-approximate solutions with near-linear running time in terms of n. Given an ε satisfying $0 < \varepsilon < 1$, the goal in the $(1-\varepsilon)$-approximate MaxRS problem is to place Q so that the total weight of points in $P \cap Q$ is at least $(1-\varepsilon) \cdot \mathtt{opt}$, where $\mathtt{opt}$ is the total weight in the optimal placement of Q. These include random sampling based algorithms [1, 2, 5, 53] and deterministic algorithms based on ε-approximations [20, 34].

In this work we consider three natural variants of the MaxRS problem: dynamic MaxRS, batched MaxRS and colored MaxRS.

1.1 Dynamic MaxRS

In the *dynamic* MaxRS problem, a new point is inserted into P or an existing point from P is deleted in each round, and the goal is to design an algorithm which can quickly update the optimal placement of Q. The naive approach would be to re-compute the solution from scratch which requires $\Omega(n)$ time, whereas the goal is to re-compute the solution in sub-linear time. Note that Q does not change in this setting. Going back to the COVID example, location of recovered patients are deleted and location of newly infected patients are inserted, and the authorities need to be reported about the updated hotspot in real-time. To the best of our knowledge, dynamic MaxRS has not been explicitly studied before. We obtain the following result.

Theorem 1.1. *For dynamic MaxRS with Q being a d-ball, there is a randomized $(\frac{1}{2}-\varepsilon)$-approximation algorithm with amortized update time $O(\varepsilon^{-2d-2}\log n)$, for $0 < \varepsilon < 1/2$. The approximation factor holds with high probability, i.e., $1 - n^{-\Omega(1)}$. Here n is the number of points of P in a given round.*

As a consequence of the above theorem, we obtain a new result for static MaxRS (where the set P is known upfront) with Q being a d-ball.

Theorem 1.2. *For static MaxRS with Q being a d-ball, there is a randomized $(\frac{1}{2}-\varepsilon)$-approximation algorithm with running time $O(\varepsilon^{-2d-2} n \log n)$, for $0 < \varepsilon < 1/2$. The approximation factor holds with high probability.*

Previous constructions can be extended to obtain a $(1-\varepsilon)$-approximation algorithm for static MaxRS with Q being a d-ball, but they have a running time of $O_\varepsilon(n \log^{\Theta(d)} n)$. Interestingly, in Theorem 1.2 the exponential dependence of $\log n$ on d is avoided in the running time (at the expense of approximation factor).

1.2 Batched MaxRS

In the *batched* MaxRS problem, along with the point set P, we are given a set $\{Q_1, Q_2, \ldots, Q_m\}$ of m geometric ranges, and the goal is to find a placement for each range Q_i which maximizes the weight of points of P covered by it. Our original motivation for studying batched MaxRS was Q_i's being axis-aligned rectangles in $\mathbb{R}^2$. By running the $O(n \log n)$ time algorithm [32, 48] for MaxRS for each Q_i, the batched MaxRS problem can be solved in $O(mn \log n)$ time. The following natural question arises:

Does there exist an $o(mn)$ time algorithm for batched MaxRS with axis-aligned rectangles in $\mathbb{R}^2$?

The punchline is that even in $\mathbb{R}^1$ this looks unlikely. Specifically, we establish an $\Omega(mn)$ time conditional lower bound for batched MaxRS in $\mathbb{R}^1$ assuming hardness of the popular $(\min,+)$-*convolution* problem.

The $(\min,+)$-convolution problem. Given two sequences $A = (A_0, A_1, \ldots, A_{n-1})$ and $B = (B_0, B_1, \ldots, B_{n-1})$, their $(\min,+)$-convolution produces a new sequence C, where each element is defined as:

$$C_k = \min_{i+j=k} (A_i + B_j),$$

for $k \in \{0, \ldots, n-1\}$.

This problem admits a trivial quadratic time algorithm, but has been conjectured to be quadratic hard, i.e., there exists no $O(n^{2-\epsilon})$ algorithm for it, for any constant $\epsilon > 0$ [19]. Till date the conditional hardness of sevaral problems has been established via the $(\min,+)$-convolution problem [7, 11, 19, 33, 36, 37].

Theorem 1.3. *Assuming that $(\min,+)$-convolution on sequences of length n requires $\Omega(n^2)$ time, the batched MaxRS problem on n points and m query interval lengths requires $\Omega(mn)$ time.*

A similar lower bound is established for a related problem of *batched smallest k-enclosing interval*. In the smallest k-enclosing interval (SEI) problem, the input is a set P of n points on the real-line and and an integer k in the range $[1, n]$. The goal is to find the smallest interval which contains k points of P. In the batched smallest k-enclosing interval problem, the goal is to solve the SEI problem for *all* integer values of k from 1 till n. Once again the problem can be solved trivially in $O(n^2)$ time. We establish a conditional lower bound of $\Omega(n^2)$ for this problem as well.

THEOREM 1.4. *Assuming that* $(\min, +)$*-convolution on sequences of length n requires* $\Omega(n^2)$ *time, the batched smallest k-enclosing interval problem on n points requires* $\Omega(n^2)$ *time.*

1.3 Colored MaxRS

In the *colored* MaxRS problem in $\mathbb{R}^d$, each point in P is assigned a color from $\{1, 2, \ldots, m\}$ and the goal is to find a placement of Q that maximizes the number of *uniquely colored* points in $P \cap Q$. Note that the points no longer have a weight associated. See Figure 1b for an example. Colored MaxRS was recently studied in the context of trajectory data [58]. They present an exact $O(n \log n)$ time algorithm for Q being an axis-aligned rectangle.

In the context of wildlife conservation, consider m trajectories each corresponding to say an endangered animal. Points are sampled from each trajectory and assigned a unique color. The objective is to optimally position a tracking device to monitor the maximum number of animals. See [58] for more examples. Colored MaxRS is also useful for finding neighborhood with maximum number of distinct facilities (such as restaurants, schools, hospitals, parks and fire stations).

In this work we primarily focus on colored MaxRS with Q being d-balls. Recall that even (uncolored) MaxRS for Q being a 2-ball has a conditional lower bound of $\Omega(n^2)$ [5] (and in fact we conjecture a conditional lower bound of $\Omega(n^d)$ for colored MaxRS for Q being a d-ball). Therefore, we focus on designing approximation algorithms with near-linear running time. The goal in the α-approximate colored MaxRS problem is to place Q so that the number of uniquely colored points in $P \cap Q$ is at least $\alpha \cdot \mathtt{opt}$, where $\mathtt{opt}$ is the number of uniquely colored points in the optimal placement of Q. The first result is the following.

THEOREM 1.5. *For colored MaxRS with Q being a d-ball, there is a randomized* $(\frac{1}{2} - \varepsilon)$*-approximation algorithm with running time* $O(\varepsilon^{-2d-2} n \log n)$*, for* $0 < \varepsilon < 1/2$*. The approximation factor holds with high probability.*

The second result improves upon the first result in $\mathbb{R}^2$ in terms of approximation factor.

THEOREM 1.6. *For colored MaxRS with Q being a* 2*-ball, there is a randomized* $(1 - \varepsilon)$*-approximation algorithm with expected running time* $O(\varepsilon^{-2} n \log n)$*, for* $0 < \varepsilon < 1$*. The approximation factor holds with high probability.*

1.4 The dual setting: computing the maximum depth

Till now we have described the MaxRS problem in the *primal* setting. By appropriate scaling, we can assume that the d-ball Q has unit radius. We now state the MaxRS problem in the *dual* setting. Replace each point $p \in P$ (with weight w_p) with a unit radius d-ball $B(p)$ (with weight w_p) centered at p. Let $\mathcal{B} \leftarrow \bigcup_{p \in P} B(p)$ be the collection of these n weighted unit balls. Then the MaxRS problem for Q being a d-ball is equivalent to the problem of finding the point in $\mathbb{R}^d$ with the maximum *weighted depth* w.r.t. $\mathcal{B}$. Similarly, colored MaxRS in the dual setting will be a set $\mathcal{B}$ of n colored unit d-balls and the goal is to report the point in $\mathbb{R}^d$ with the maximum *colored depth*. The colored-depth of a point p, say $\mathtt{cd}_{\mathcal{B}}(p)$, is the number of uniquely colored balls in $\mathcal{B}$ that contain p.

1.5 An overview of our techniques

MaxRS is amenable to sampling, i.e., in the dual setting a point in $\mathbb{R}^d$ with large (resp., small) depth w.r.t. $\mathcal{B}$ is expected to have a large (resp., small) depth wr.r.t. a uniform random sample in $\mathcal{B}$. Prior works obtained a $(1-\varepsilon)$-approximation by exploiting this property. The strategy works in two stages: ignoring the dependency on ε and assuming the value of $\texttt{opt}$ is known, first compute a sample of size roughly $O(\log n)$, and then run the exact algorithm on the sample. Unfortunately, for a d-ball in the dynamic setting, this approach leads to a $(1-\varepsilon)$-approximation algorithm with an update time that has exponential dependence on d (i.e., $\log^{O(d)} n$), since exact MaxRS with a d-ball on n points requires $O(n^d)$ time. In contrast, our first technique achieves an update time that does *not* have an exponential dependence on d with respect to $\log n$.

Our first general technique which obtains a $(\frac{1}{2}-\varepsilon)$-approximation follows a different approach. Instead of sampling the input objects, the algorithm samples a bunch of points in $\mathbb{R}^d$, maintains their (weighted or colored) depth w.r.t. $\mathcal{B}$ and reports the maximum among them. The algorithm is simple but the analysis entails a randomized game involving a set of fixed objects in $\mathbb{R}^d$ and a random sample of points, and a "volume" argument involving d-balls. This technique is used to obtain Theorem 1.1, 1.2, and 1.5.

Our second technique provides a $(1-\varepsilon)$-approximation guarantee for colored MaxRS problem (previous works gave a $(1-\varepsilon)$-approximation guarantee for un-colored MaxRS). The algorithm works in two phases. In the first phase, we design an exact output-sensitive algorithm, and in the second phase, we speed up the exact algorithm by random sampling on colors. As an application, we consider colored MaxRS with Q being a disk in $\mathbb{R}^2$. There is a straightforward $O(n^2 \log n)$ time algorithm to solve the problem. We first design an algorithm whose running time is proportional to the value of $\texttt{opt}$. Specifically, the running time is $O(n \log n + n \cdot \texttt{opt})$ (Theorem 4.6). Using this output-sensitive algorithm, we then design the $(1-\varepsilon)$-approximation algorithm with $O_\varepsilon(n \log n)$ running time (Theorem 1.6).

1.6 Other related work

The study of the MaxRS problem was initiated by the computational geometry community [1, 8, 12, 20, 32, 34, 48, 59], and more recently it has attracted significant interest in the database community. In the last decade, several works have appeared that study MaxRS and its variants in different models and application settings. For example, the MaxRS problem has been extensively studied in the I/O-model [16, 17, 53], as well as in spatial data streams [3, 4, 43]. An index-based method has been proposed to support queries with varying parameters efficiently [61]. To address inherent uncertainty in location data, the probabilistic MaxRS problem has also been explored [38, 47], and more recently extended to dynamic (varying locations of objects over time) and trajectory data [29, 41, 58]. A different dynamic variant, where object weights decay over time, has been studied in [52]. [30] studied MaxRS for optimizing area coverage of polygons. Variants of the MaxRS problem have also been considered in road networks [9, 14, 15, 39, 57, 60], mobile ad hoc networks [46] and wireless sensor networks [31].

A closely related problem is the *best region search* (BRS) problem introduced by [25], where the goal is to find an optimal rectangle of a specified size that maximizes a user-defined scoring function over the enclosed points. Extensions such as selecting the top-k best regions have also been studied [49, 51]. Other region search problems have also been considered in the literature [13, 26, 27, 40]. Some of the classical problems in computational geometry and databases, such as range searching [42] and range aggregation [28, 50], are related but fundamentally different from the MaxRS problem that we study in this paper (in range searching or range aggregation the placement of Q is fixed and summary of $P \cap Q$ is the query).

2 Preliminaries

For any point $x \in \mathbb{R}^d$, we refer to its coordinates as $x_1, \ldots, x_d$. Denote by $G_s(c)$ the uniform grid with cell side length s and the boundaries are defined by d families of hyperplanes. For each $1 \leq i \leq d$, the i-th family of hyperplanes are $\{x \in \mathbb{R}^d \mid x_i = c_i + k \cdot s, \text{ where } k \in \mathbb{Z}\}$ (perpendicular to the i-th dimension and separated by distance s).

For any point $p \in \mathbb{R}^d$, consider the cell in $G_s(c)$ containing p and let p' be the center of the cell. Then p is defined to be Δ-*near* if the euclidean distance between p and p' is at most Δ. The following is a simple lemma which states that if we perform enough "shifts" of the grid, then in at least one of the shifted grids, p will be Δ-near.

Lemma 2.1. *Consider a collection of grids* $\mathcal{G} = \left\{G_s\left(\frac{\Delta}{\sqrt{d}}z\right) \mid z \in \{0, 1, \ldots, \frac{s\sqrt{d}}{\Delta} - 1\}^d\right\}$. *For any point* $p \in \mathbb{R}^d$, *there exists at least one grid in* $\mathcal{G}$ *in which* p *is* Δ*-near.*

3 Technique 1: Sampling points in $\mathbb{R}^d$

In this section we present a general technique for obtaining fast approximations for the MaxRS problem for d-balls. We describe our algorithms in the dual setting, i.e., we are given an input set of n unit balls $\mathcal{B}$, and the goal is to find a point that has the maximum possible depth. Our technique is largely based on the idea of sampling a small set of points S, and estimating depth as the deepest point in this sample. We show that it is sufficient to consider a sample of $O_\varepsilon(n \log n)$ points to obtain a $(\frac{1}{2} - \varepsilon)$-approximation. This small sample size allows us to avoid the exponential dependence of $\log n$ on d for static algorithms, and also allows us to get fast dynamic update times, since depth of only a small number of points have to be modified for each update.

3.1 Dynamic MaxRS with d-ball

3.1.1 Algorithm. In the dynamic setting, balls can be inserted to, or deleted from $\mathcal{B}$. The goal of the algorithm is to maintain a point of (approximate) maximum depth. We first construct a collection of grids $\mathcal{G}$, by applying Lemma 2.1 with $s = \frac{2\varepsilon}{\sqrt{d}}$ and $\Delta = \varepsilon^2$. We say a grid cell is *non-empty* if at least one d-ball in $\mathcal{B}$ intersects the cell.

Sampling step. Pick a grid $G \in \mathcal{G}$. For each non-empty cell X, let $C(X)$ be the circumsphere of X. We sample a set S_X of some $t = \Theta(\varepsilon^{-2} \log |\mathcal{B}|)$ points on $C(X)$. Specifically, each point is sampled independently and uniformly at random from $C(X)$ [44, 55]. Iterate over all non-empty cells X to get $S_G \leftarrow \bigcup S_X$. Finally, iterate over all grids $G \in \mathcal{G}$, to obtain the set $S \leftarrow \bigcup S_G$.

Epochs and handling updates. Our algorithm proceeds in epochs. At the beginning of the j-th epoch, let the balls in $\mathcal{B}$ be denoted by $\mathcal{B}_j$. The j-th epoch *ends* when the number of balls in $\mathcal{B}$ go outside the range $[|\mathcal{B}_j|/2, 2|\mathcal{B}_j|]$, and the $(j{+}1)$-th epoch starts.

At the *start* of each epoch, we do a sampling step on $\mathcal{B}_j$ to obtain a set of points S_j. We then compute the weighted depth of each point in S_j. We will show in the analysis that this re-sampling, and re-computation of depth takes only $O_\varepsilon(|\mathcal{B}_j| \log |\mathcal{B}_j|)$ time.

During the *course* of an epoch, we need to handle insertions and deletions of balls. Let B be the unit d-ball inserted in the current round. For each cell, say X, intersected by B, we have two cases. If X is non-empty then the weighted depth of each point in $S_X \cap B$ is increased by w_B, the weight of ball B. Otherwise, if X is empty, then sample a set S_X of $t = \Theta(\varepsilon^{-2} \log |\mathcal{B}_j|)$ points, and any point in S_X contained in B will have weighted depth w_B. The case of deletion can be handled similarly.

3.1.2 Analysis. We will start the analysis by analyzing a randomized game (Lemma 3.1) involving a fixed collection of `opt` balls and a random sample of points. We believe that this game can have wider applications. For a closed set $Q \subseteq \mathbb{R}^d$, we will use ∂Q to denote its boundary.

Lemma 3.1. *Let C be a d-ball, and let $\mathcal{B}' \subseteq \mathcal{B}$ be a collection of k weighted d-balls, such that the $(d-1)$-dimensional measure of $\partial C \cap B$ relative to the $(d-1)$-dimensional measure of C is $\frac{1}{2} - \Theta(\varepsilon)$, for all $B \in \mathcal{B}'$[1]. Let $S = \{p_1, p_2, \ldots, p_t\}$ be a collection of points such that each point in S is independently and uniformly at random picked from ∂C, where $t = c\varepsilon^{-2} \log n$ (for a sufficiently large constant c). Then with high probability (i.e., $\geq 1 - n^{-\Omega(1)}$) there exists a point in S whose weighted depth is at least $\left(\frac{1}{2} - \Theta(\varepsilon)\right) W$, where $W = \sum_{B \in \mathcal{B}'} w_B$.*

Proof. We will use an indirect argument to establish the lemma. Consider any ball $B \in \mathcal{B}'$. We will first establish that with high probability at least $\left(\frac{1}{2} - \Theta(\varepsilon)\right) t$ points of S will lie inside B. For $1 \leq i \leq t$, let X_i be the indicator random variable which is equal to 1 if $p_i \in B$. Let $Y_B = \sum_{i=1}^{t} X_i$ be the number of points in S contained in B. Then, $\mathbb{E}[Y_B] \geq \left(\frac{1}{2} - \Theta(\varepsilon)\right) t$. Since $X_1, X_2, \ldots, X_t$ are independent random variables, via standard application of Chernoff bound, with high probability (w.h.p.), we have $Y_B \geq (1-\varepsilon)\left(\frac{1}{2} - \Theta(\varepsilon)\right) t \geq \left(\frac{1}{2} - \Theta(\varepsilon)\right) t$. Next, by union bound on $k \leq n$ balls, $Y_B \geq \left(\frac{1}{2} - \Theta(\varepsilon)\right) t$ holds w.h.p. for all $B \in \mathcal{B}'$.

Now we will connect the random variables Y_B with the weighted depth of points in S. Let $\mathtt{wd}(p)$ be the weighted depth of any point $p \in S$ w.r.t. $\mathcal{B}'$. Then, observe that $\sum_{p \in S} \mathtt{wd}(p) = \sum_{B \in \mathcal{B}'} w_B Y_B$, which by the previous argument is at least $W\left(\frac{1}{2} - \Theta(\varepsilon)\right) t$ w.h.p. This finishes the proof. □

Motivated by the previous lemma, we now show that if a unit ball passes *close to* the center of a ball of radius ε, then almost half the "surface area" of the smaller ball is covered.

Lemma 3.2. *Let B be a unit radius d-ball and C be a d-ball of radius ε. Suppose ∂B passes through a point that lies at distance ε^2 from the center of C. Then the $(d-1)$-dimensional measure of $B \cap \partial C$ relative to the $(d-1)$-dimensional measure of C is at least $\frac{1}{2} - \Theta(\varepsilon)$.*

Proof. Via suitable rotations of the d-balls, we can assume that the center of C is at the origin and the center of B is $(0, 0, \ldots, 0, 1+\varepsilon^2)$. See Figure 2(a). For any point $x = (x_1, \ldots, x_d) \in \partial B \cap \partial C$, $\|x\| = \varepsilon^2$ and $\|x - (0, \ldots, 0, 1+\varepsilon^2)\| = 1$. Expanding these two quantities, we get the following equations:

$$\sum_{i=1}^{d} x_i^2 = \varepsilon^2 \quad \text{and} \quad \sum_{i=1}^{d-1} x_i^2 + (x_d - 1 - \varepsilon^2)^2 = 1,$$

which solves to $x_d = \dfrac{3\varepsilon^2 + \varepsilon^4}{2 + 2\varepsilon^2}$. We denote the expression on the right by b. It can be verified that $\varepsilon^2 \leq b \leq 2\varepsilon^2$ for all $\varepsilon \in (0, 1)$.

In $d = 2$, suppose $B \cap \partial C$ subtends an angle 2θ at the center of C (ref Figure 2(a)). Since $\cos\theta = b/r$, the arc length of $B \cap \partial C$ relative to the circumference of C is

$$\frac{2\theta r}{2\pi r} = \frac{1}{\pi} \arccos\left(\frac{3\varepsilon + \varepsilon^3}{2 + 2\varepsilon^2}\right) \geq \frac{1}{\pi} \arccos(2\varepsilon) \geq \frac{1}{2} - 2\varepsilon,$$

for all $0 < \varepsilon < 1/2$.

For $d \geq 3$, we use a known result of computing the surface area of a spherical cap as follows. Let h denote the height of the spherical cap $B \cap \partial C$ determined by the hyperplane $x_d = b$. Define $q = 1 - h/r = b/r = \Theta(\varepsilon)$, since $b = \Theta(\varepsilon^2)$. By an estimate of [18, 56], the surface area of $B \cap \partial C$ relative to the surface area of C is given by

$$\frac{1}{2} - \frac{G_{d-2}(q)}{2\,G_{d-2}(1)},$$

[1] In $\mathbb{R}^2$, C will be a disk and any disk in $\mathcal{B}'$ will cover at least $\left(\frac{1}{2} - \Theta(\varepsilon)\right)$ fraction of C's circumference. We also remark that the lemma holds in a more general setting.

where $G_d(x) = \int_0^x (1-t^2)^{(d-1)/2}\,dt$ for all $x \in [0,1]$. The quantity $G_{d-2}(q)$ can be trivially upper bounded by $q = \Theta(\varepsilon)$, and the quantity $G_{d-2}(1)$ can be lower bounded by $\int_0^1 (1-t^2)^{(d-3)/2}\,dt \geq \int_0^{1/2} (3/4)^{(d-3)/2}\,dt = \Omega(1)$. This finishes the proof. □

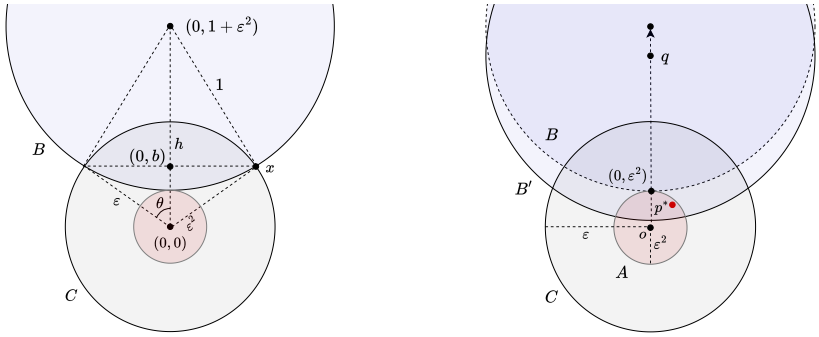


Fig. 2. (a) Intersection between a unit disk B centered at $(0, 1+\varepsilon^2)$ and a circle C centered at $(0,0)$ of radius ε. (b) Shifting the ball B' towards q.

Using the above two lemmas, we will now establish the approximation factor of our algorithm.

Lemma 3.3. *With high probability, the approximation factor of the algorithm is* $\frac{1}{2} - \varepsilon$.

Proof. Consider the grid in $\mathcal{G}$ in which the optimal point p^* is ε^2-near (guaranteed by Lemma 2.1). Let X^* be the cell in the grid containing p^*. W.l.o.g., assume that the center of X^* is at the origin o. Consider a d-ball A of radius ε^2 centered at the origin. Then p^* will lie inside A (see Figure 2(b)). The circumsphere $C := C(X^*)$ of the grid cell has radius ε. Let $B' \in \mathcal{B}$ be a ball containing p^* with center q. Translate B' in the direction $\overrightarrow{oq}$ to the ball B until it is tangential to A. In this case, the spherical cap $B \cap \partial C \subseteq B' \cap \partial C$, and therefore, the $(d-1)$-dimensional measure of the cap can only decrease. This is the setting considered in Lemma 3.2, which implies that the minimum measure is at least $\frac{1}{2} - \Theta(\varepsilon)$ relative to the measure of C. We then invoke Lemma 3.1 on the ball C and the set of balls $\mathcal{B}'$ which cover p^*, which shows the existence of a randomly sampled point in ∂C with weighted depth at least $\mathtt{opt}(\frac{1}{2} - \Theta(\varepsilon))$ with high probability. Scaling ε appropriately gives the desired approximation factor. □

Lemma 3.4. *The amortized update time of the algorithm is* $O(\varepsilon^{-2d-2} \log |\mathcal{B}|)$.

Proof. Consider the j-th epoch. First consider the cost of an insertion or deletion of a ball. In any fixed grid $G \in \mathcal{G}$, a unit ball intersects at most $(2/s)^d = \left(\frac{2\sqrt{d}}{\varepsilon}\right)^d = O(\varepsilon^{-d})$ cells and the collection $\mathcal{G}$ contains $\left(\frac{s\sqrt{d}}{\Delta}\right)^d = O(\varepsilon^{-d})$ grids. Therefore, the number of sample points in S updated is $O(\varepsilon^{-2d}) \times O(\varepsilon^{-2} \log |\mathcal{B}_j|) = O(\varepsilon^{-2d-2} \log |\mathcal{B}_j|)$. Hence each update costs $O_\varepsilon(\log |\mathcal{B}_j|)$.

We now consider the sampling step that will happen in the $(j+1)$-th epoch. The number of points sampled will be $O_\varepsilon(|\mathcal{B}_{j+1}| \log |\mathcal{B}_{j+1}|)$. Inserting the balls in $\mathcal{B}_{j+1}$ one-by-one, and updating the weighted depth of $O_\varepsilon(\log |\mathcal{B}_j|)$ sampled points per ball, takes $O_\varepsilon(|\mathcal{B}_{j+1}| \log |\mathcal{B}_{j+1}|)$ time. We will charge the cost of this sampling step to the updates which happened in the j-th epoch. Since $|\mathcal{B}_j| = \Theta(|\mathcal{B}_{j+1}|)$ and the number of updates in the j-th epoch are at least $|\mathcal{B}_j|/2$, each update in the j-th epoch is charged only $O_\varepsilon(\log |\mathcal{B}_j|)$. □

3.2 Colored MaxRS with d-ball

In the dual version of the colored MaxRS problem, we are given a set $\mathcal{B}$ of n colored unit balls, and we have to find a point that is covered by maximum number of distinctly colored balls. Our algorithm is very similar to the earlier described dynamic algorithm. We construct a collection of grids $\mathcal{G}$, and sample $O_\varepsilon(\log n)$ points for each non-empty cell. Only the depth computation step differs slightly.

Let S be the set of sampled points. For each point $p \in S$ maintain a temporary flag (initialized to -1), which records the most recent color that contributed to its coverage. To compute the colored depth of each point in S, we process the input balls grouped by their color. Specifically, we order the set $\mathcal{B}$ by color index using any sorting algorithm. Now iterate over the input balls in this order, updating the colored depth of points corresponding to cells which intersect the particular ball.

Let $\mathcal{B}_j$ denote the set of balls of color $j \in [m]$. For each ball in $\mathcal{B}_j$, if a point $p \in S$ is found to lie inside the ball, and the temporary flag of p is not already equal to j, then we update the flag to j, and increment the colored depth $\mathtt{cd}_{\mathcal{B}}(p)$ by 1. This ensures that the colored depth counts the number of distinct colors, for which the point is covered by at least one ball.

Theorem 1.5. *For colored MaxRS with Q being a d-ball, there is a randomized $(\frac{1}{2}-\varepsilon)$-approximation algorithm with running time $O(\varepsilon^{-2d-2} n \log n)$, for $0 < \varepsilon < 1/2$. The approximation factor holds with high probability.*

Proof. Every unit d-ball, intersects $O(\varepsilon^{-2d})$ cells across all grids in $\mathcal{G}$ (as shown earlier in proof of Lemma 3.4), and for each cell we maintain a sample of $O(\varepsilon^{-2}\log n)$ points. Hence corresponding to each ball, there are $O(\varepsilon^{-2d-2}\log n)$ points that need to be considered. Sorting the set of balls takes $O(n \log n)$ time. Since corresponding to each ball, we update the colored depth of $O_\varepsilon(\log n)$ points, and each update takes $O(1)$ time, we get an overall running time of $O(\varepsilon^{-2d-2}\log n)$. □

4 Technique 2: Output-sensitivity and color sampling

In this section we present a $(1-\varepsilon)$-approximation algorithm for the colored disk MaxRS problem in $\mathbb{R}^2$ (Theorem 1.6). We have already shown in Section 2 that this problem is equivalent to finding a point of maximum colored depth in a given set of colored unit disks. So, the input to our problem is a set $\mathcal{D}$ of n unit disks, each of which is assigned a color from the set $[m]$. The goal is to find a point p such that $\mathtt{cd}_{\mathcal{D}}(p) \geq (1-\varepsilon)\mathtt{opt}$ (the notation $\mathtt{cd}_{\mathcal{D}}(p)$ refers to the colored depth of a point p w.r.t. a set of disks $\mathcal{D}$).

4.1 Preliminaries

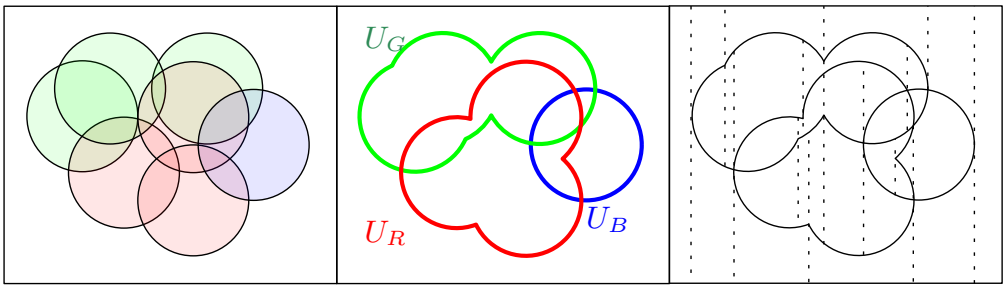
An x-monotone arc is a planar curve that is intersected by any vertical line in at most one point. Consider a set $\mathcal{A}$ of circular x-monotone arcs with k intersection points. A *vertical decomposition* of $\mathcal{A}$ partitions the plane into disjoint cells where each cell is a "pseudo-trapezoid" with its top and bottom segment being a portion from an arc in $\mathcal{A}$. The vertical decomposition is performed in three steps. First, enclose the arcs inside a large enough bounding box. Next, shoot a vertical ray upwards and downwards from the endpoints of each arc in $\mathcal{A}$ and the k intersection points. Finally, each ray travels until it first hits another arc in $\mathcal{A}$ or the top or the bottom of the bounding box. See Figure 3(c) for an example. The resulting planar subdivision is called the *trapezoidal map*. We will need the following result.

Lemma 4.1. *(Theorem 2 in [45]) The trapezoidal map a set of n circular x-monotone arcs with k intersection points can be computed in $O(n \log n + k)$ expected time. The map has $O(n+k)$ vertices, edges and faces.*

4.2 The first algorithm

Transformation to an uncolored problem. For a collection $\mathcal{D}'$ of unit disks, let $\mathcal{U}(\mathcal{D}')$ denote the *union* of the disks $\mathcal{D}'$, i.e., the set of points in the plane which are covered by at least one disk in $\mathcal{D}'$. Each $\mathcal{U}(\mathcal{D}')$ will be a (possibly disconnected) planar region, which may contain holes. For each color $c \in [m]$, let $U_c = \mathcal{U}(\mathcal{D}_c)$, where $\mathcal{D}_c$ is the collection of all disks of color c. The boundary of U_c, denoted ∂U_c, is composed of circular arcs from the corresponding disks of color c. The depth of a point p in this context is the number of U_i's, for all $1 \leq i \leq m$, containing p. See Figure 3(c).

We can now transform the original problem into the following new problem: find a point $p \in \mathbb{R}^2$ that maximizes its (uncolored) depth with respect to the set of regions $\{U_1, \ldots, U_m\}$.



(a) Colored input disks. (b) Merged objects U_c. (c) Cells in trapezoidal map.

Fig. 3. Reduction from max colored depth in disks to max (uncolored) depth among U_i's.

Traversal of the trapezoidal map. For each color $c \in [m]$, compute U_c and determine the circular arcs forming the boundary ∂U_c [6] (see Figure 3(b)). Each circular arc can be broken into two x-monotone arcs by breaking it at the point with the extreme x-coordinate. Next, apply Lemma 4.1 to compute the trapezoidal map of the circular x-monotone arcs (see Figure 3(c)).

Now we will traverse the trapezoidal map to compute the depth of each cell and identify the cell with the maximum depth. Any point inside this cell can be reported as the final output. For each cell C in the trapezoidal map, maintain a variable d_C denoting its depth. We will perform a breadth-first search type traversal of the map starting from a cell with depth zero (any cell near the boundary works). Let C be the current cell visited and let e be the common edge when traversing to an adjacent cell C'. If e belongs to one of the vertical rays shot, then $d_C = d_{C'}$. Otherwise, let D be the disk corresponding to edge e. If $C \in D$ and $C' \notin D$, set $d_{C'} = d_C - 1$; else, set $d_{C'} = d_C + 1$.

Lemma 4.2. *Let $\mathcal{D}$ be a collection of n colored disks in the plane. Given regions $U_1, \ldots, U_m$ corresponding to each color, define k to be the number of points such that the boundaries of any two regions intersect. Then there is an algorithm to answer colored disk MaxRS in $O(n \log n + k)$ expected time.*

Proof. For any color $c \in [m]$, the time taken to construct U_c is $O(|\mathcal{D}_c| \log |\mathcal{D}_c|)$ via power diagrams [6]. Hence, the time taken to construct $U_1, \ldots, U_m$ will be $O(n \log n)$. The expected time to construct the trapezoidal map is $O(n \log n + k)$. The traversal of the map can be performed in $O(n + k)$ time by using the standard doubly-connected edge list (DCEL) data structure [21, 54]. The check of $C \in D$ or not can be performed in constant time by testing if an arbitrary point in C lies inside D. Overall, the expected time is $O(n \log n + k)$. □

4.3 The second algorithm

In the second algorithm, we will break the original problem into several sub-problems with depth $O(\texttt{opt})$ and apply Lemma 4.2 on each sub-problem. This will lead to a running time of $O(n \log n + n \cdot \texttt{opt})$.

Algorithm. We first construct a collection of grids $\mathcal{G}$ by applying Lemma 2.1 with $s = 1$ and $\Delta = 0.25$. Fix any one shift of the grid. For each non-empty cell C in the grid, compute the disks in $\mathcal{D}$ intersecting C, say $\mathcal{D} \cap C$. Throw away the disks in $\mathcal{D} \cap C$ which do not contain any corner of cell C. Based on the remaining disks in $\mathcal{D} \cap C$, run the first algorithm (Lemma 4.2) to identify a point in this grid with the maximum depth. Repeat this for all the grid shifts and report the optimal point p^*.

Analysis. We first justify throwing away of disks, which allows us to bound the number of colors whose disks intersect any cell C.

Lemma 4.3. *(1) Consider a shift of the grid in which the optimal point p^* is 0.25-near, and lies in some cell C^*. A unit disk which does not contain any corner of C^* cannot contain p^*.*
(2) For any cell C, after throwing away disks, the number of uniquely colored disks in $\mathcal{D}$ intersecting C is at most $4 \cdot \texttt{opt}$.

Proof. Consider a disk D and a cell C^* as shown in Figure 4. Since $|bd| = |cd|$, we have $|bd| \leq 1/2$. Now $|ad| = \sqrt{|ab|^2 - |bd|^2} \geq \sqrt{1 - (1/2)^2} \geq \frac{\sqrt{3}}{2}$. Finally, $|ad| + |de| = 1$ implies $|de| \leq 1 - \frac{\sqrt{3}}{2} < 0.15$. This establishes that D cannot contain p^*, and completes the proof of (1).

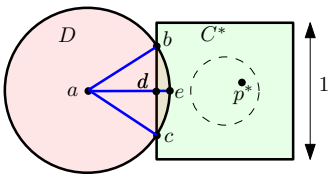


Fig. 4. A disk D not containing any corner of cell C^*, cannot contain the optimal point p^*.

Proof of (2) follows from the fact that the depth of each corner of C is at most $\texttt{opt}$. □

Now we bound the number of intersections among two color classes (Lemma 4.4), and then use this result to bound the total number of intersections between boundaries of different color classes (Lemma 4.5).

Lemma 4.4. *Let $\mathcal{D}_R$ and $\mathcal{D}_B$ be a collection of red and blue unit disks, respectively. Let $I(\mathcal{D}_R, \mathcal{D}_B)$ be the set of points where there is an intersection between an arc in ∂U_R and an arc in ∂U_B. Then $|I(\mathcal{D}_R, \mathcal{D}_B)| = O(|\mathcal{D}_R| + |\mathcal{D}_B|)$.*

Proof. Consider any point $p \in I(\mathcal{D}_R, \mathcal{D}_B)$. Let $D_R \in \mathcal{D}_R$ and $D_B \in \mathcal{D}_B$ be the disks corresponding to the arcs in ∂U_R and ∂U_B, respectively, intersecting at p. Since $p \in \partial U_R \cap \partial U_B$, it implies that p is at a distance of at least one from the center of any disk in $\mathcal{D}_R \cup \mathcal{D}_B$. Therefore, by definition of a boundary point, we claim that $p \in \partial(\mathcal{D}_R \cup \mathcal{D}_B)$.

For the sake of contradiction assume that p is not a vertex on $\partial(\mathcal{D}_R \cup \mathcal{D}_B)$. By general position assumption, only D_R and D_B contain p and hence, either the arc of disk D_R or D_B on $\partial(\mathcal{D}_R \cup \mathcal{D}_B)$ contains p. W.l.o.g. assume it is disk D_R. Then $\partial(\mathcal{D}_R \cup \mathcal{D}_B)$ will have a point infinitesimally close to p on either side which lie on D_R (as shown in Figure 5). Label them p_a and p_b.

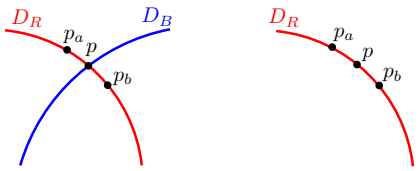


Fig. 5. Proof of Lemma 4.4

Now either p_a or p_b will lie at a distance of less than one from the center of D_B (in Figure 5 it is p_b). Therefore, one among p_a and p_b cannot be on $\partial(\mathcal{D}_R \cup \mathcal{D}_B)$. This is a contradiction. Therefore, p is a vertex on $\partial(\mathcal{D}_R \cup \mathcal{D}_B)$. Finally, notice that since $\mathcal{D}_R \cup \mathcal{D}_B$ is also a collection of disks, the number of vertices on $\partial(\mathcal{D}_R \cup \mathcal{D}_B)$ will be bounded by $O(|\mathcal{D}_R| + |\mathcal{D}_B|)$ [35]. □

Lemma 4.5. *Let $\mathcal{D}$ be a collection of n colored disks in the plane such that each disk has a color from $t = [4 \cdot \mathtt{opt}]$. Given regions $U_1, \ldots, U_t$ corresponding to each color, the number of intersection points among the boundaries of any two regions is bounded by $O(n \cdot \mathtt{opt})$.*

Proof. By Lemma 4.4 the number of intersection points, $|I(\mathcal{D}_i, \mathcal{D}_j)|$, among the boundaries of two regions ∂U_i and ∂U_j is $O(|\mathcal{D}_i| + |\mathcal{D}_j|)$. Summing this quantity over all the pairs, we get

$$\sum_{i=1}^{t} \sum_{j=1}^{t} O(|\mathcal{D}_i| + |\mathcal{D}_j|) = \sum_{i=1}^{t} O(|\mathcal{D}_i| \cdot \mathtt{opt} + n) = O(n \cdot \mathtt{opt}).$$ □

Theorem 4.6. *There is a randomized algorithm for exact colored disk MaxRS with $O(n \log n + n.\mathtt{opt})$ expected running time.*

Proof. Consider a cell C and let n_C be the number of disks intersecting C. The expected time taken to run the first algorithm (Lemma 4.2) on these disks is $O(n_C \log n_C + k_C)$, where $k_C = O(n_C \cdot \mathtt{opt})$ (by Lemma 4.5). Note that we can apply Lemma 4.5 since by Lemma 4.3, there are at most $4 \cdot \mathtt{opt}$ colored disks remaining in each cell. Since each disk intersects at most constant number of cells in the grid, the running time over all non-empty cells is $O(n \log n + n \cdot \mathtt{opt})$.

Now we argue about the correctness. Consider the grid in $\mathcal{G}$ where p^* is 0.25-near. Let C be the cell containing p^*. By Lemma 4.3, throwing away the disks will not change the colored depth of p^*. One subtle point is that when the first algorithm (Lemma 4.2) is run on the remaining disks in C, then a point not lying inside C might get reported. However, by definition of p^* being the optimal point, the reported point will also have the same colored depth as p^*. □

4.4 The final algorithm

In this section we will design a $(1-\varepsilon)$-approximation algorithm for colored disk MaxRS via random sampling on colors. Using Theorem 1.5 (with $\varepsilon = 1/4$), we first estimate $\mathtt{opt}$ to get a value $\mathtt{opt}'$ satisfying $\mathtt{opt}/4 \leq \mathtt{opt}' \leq \mathtt{opt}$. If $\mathtt{opt}' \leq c_1 \varepsilon^{-2} \log n$, then run the exact algorithm of Theorem 4.6 on $\mathcal{D}$.

Now we consider the interesting case of $\mathtt{opt}' > c_1 \varepsilon^{-2} \log n$. Sample each color in $[m]$ independently with probability $\lambda = \frac{c_1 \log n}{\varepsilon^2 \mathtt{opt}'}$. If a color is sampled, then add all disks in $\mathcal{D}$ of that color to the set R. Run the exact algorithm of Theorem 4.6 on the sampled set R to obtain a point $\hat{x}$ which is returned as the solution.

Lemma 4.7. *The algorithm has an expected running time of $O(\varepsilon^{-2} n \log n)$.*

Proof. If $\mathtt{opt}' \leq c_1\varepsilon^{-2}\log n$, then the expected running time is $O(n \cdot \mathtt{opt}) = O(\varepsilon^{-2} n \log n)$. Now consider the case where $\mathtt{opt}' > c_1\varepsilon^{-2}\log n$. For each color $j \in [m]$, let $\mathcal{D}_j$ be the disks of color j. Let X_j be an indicator random variable such that $X_j = 1$ if color j is sampled, and $X_j = 0$ otherwise. So we have $\mathbb{E}[X_j] = \lambda$.

The size of R can be written as $|R| = \sum_{j\in[m]} X_j \cdot |\mathcal{D}_j|$. So, its expected size is

$$\mathbb{E}[|R|] = \sum_{j\in[m]} |\mathcal{D}_j| \cdot \mathbb{E}[X_j] = \lambda \cdot \sum_{j\in[m]} |\mathcal{D}_j| = \frac{c_1 n \log n}{\varepsilon^2 \mathtt{opt}'}.$$

We run the exact algorithm on the set R, giving the desired runtime $\mathbb{E}[|R| \cdot \mathtt{opt}_R] \leq \mathtt{opt} \cdot \mathbb{E}[|R|] = O(\varepsilon^{-2} n \log n)$ in expectation. □

The arrangement of $\mathcal{D}$ contains $O(n^2)$ cells. The colored depth inside each cell remains the same. A cell is *shallow* if $\mathtt{cd}_{\mathcal{D}}(\cdot) < (1-\varepsilon)\mathtt{opt}$, for any $0 < \varepsilon < 1$. Pick an arbitrary point from each shallow cell and let P' be the collection of shallow points. The next lemma establishes that with high probability, any shallow point in P' will have colored depth $o(\log n)$ w.r.t. R. On the other hand, there exists at least one point (namely the optimum) that has colored depth $\Omega(\log n)$ w.r.t. R. Therefore, running the exact algorithm on R will not report a point from P' (with high probability), and hence the output is a $(1-\varepsilon)$-approximation.

Lemma 4.8. *Let p^* be the point with the optimal colored depth and let P' be the shallow points w.r.t. $\mathcal{D}$. Then, for $c = \frac{(1-\varepsilon/2)\mathtt{opt}}{\varepsilon^2 \mathtt{opt}'}$, we have with high probability,*

(1) $\mathtt{cd}_R(p^*) > c \cdot \log n$.

(2) for all $p \in P'$, $\mathtt{cd}_R(p) < c \cdot \log n$.

Proof. Let M be the set of colors which have a disk covering p^*, and for $j \in M$ let X_j be the random variable indicating whether color j was sampled. So, the colored depth of p^* w.r.t. R will be given by $X := \mathtt{cd}_R(p^*) = \sum_{j\in M} X_j$, and we can compute its expectation as

$$\mu := \mathbb{E}[\mathtt{cd}_R(p^*)] = \mathbb{E}\left[\sum_{j\in M} X_j\right] = \sum_{j\in M} \mathbb{E}[X_j] = \lambda \cdot \mathtt{opt}.$$

The concentration of the depth around the expectation can be given by,

$$\Pr\left[X \leq \left(1-\frac{\varepsilon}{2}\right)\mu\right] \leq \exp\left(-\frac{\mu(\varepsilon/2)^2}{2}\right) \leq \exp\left(-\frac{c_1 \mathtt{opt}}{8\mathtt{opt}'} \cdot \log n\right) \leq \frac{1}{n^k},$$

which holds for $k = \frac{c_1 \mathtt{opt}}{8\mathtt{opt}'} \geq \frac{c_1}{8} \in \Omega(1)$. So with the high probability of $(1 - 1/n^k)$ we have that $\mathtt{cd}_R(p^*) > (1-(\varepsilon/2))\mu \geq c \cdot \log n$. This proves (1).

Consider some point $p \in P'$. Let M' be the set of colors which have a disk covering p, and for $j \in M'$, let Y_j be the indicator random variable representing whether color j was sampled. So, the colored depth of p w.r.t. R will be given by $Y := \mathtt{cd}_R(p) = \sum_{j\in M'} Y_j$, and we can compute its expectation as

$$\mu' := \mathbb{E}[\mathtt{cd}_R(p)] = \mathbb{E}\left[\sum_{j\in M'} Y_j\right] = \sum_{j\in M'} \mathbb{E}[Y_j] < \lambda(1-\varepsilon)\mathtt{opt}.$$

The concentration of the depth around the expectation can be given by,

$$\Pr\left[Y \geq \left(1+\frac{\varepsilon}{2}\right)\mu'\right] \leq \exp\left(-\frac{\mu'(\varepsilon/2)^2}{3}\right) \leq \exp\left(-\frac{c_1(1-\varepsilon)\mathtt{opt}}{12\mathtt{opt}'} \cdot \log n\right) \leq \frac{1}{n^{k'}},$$

which holds for some $k' = \frac{c_1(1-\varepsilon)\mathtt{opt}}{12\mathtt{opt}'} \geq \frac{c_1(1-\varepsilon)}{12} = \Omega(1)$. So with the high probability of $(1 - 1/n^{k'})$ we have that $\mathtt{cd}_R(p) < (1+(\varepsilon/2))\mu' < (1-(\varepsilon/2))\mu \leq c \cdot \log n$.

We can now union bound over all $O(n^2)$ points $p \in P'$ to get that $\Pr[\forall p \in P', \mathsf{cd}_R(p) < c \cdot \log n] \geq 1 - \frac{1}{n^{k'-2}}$. But for a large enough constant c_1, we will have that $k' > 2$, and hence with high probability, for all points $p \in P'$ their colored depth w.r.t. R is $\mathsf{cd}_R(p) < c \cdot \log n$. This proves (2). □

The above two lemmas establish Theorem 1.6.

5 Lower bound for batched MaxRS

Consider the batched maximum range sum problem (batched MaxRS), where we are given a set of n weighted points $P = \{p_1, p_2, \ldots, p_n\}$, and a set of m intervals of length $L_1, L_2, \ldots, L_m$. For all $1 \leq i \leq m$, the goal is to find the placement of interval of length L_i so that the total weight of points of P covered is maximized. In this section, we show a $\Omega(mn)$ lower bound for this problem, conditioned on the conjectured hardness of the $(min, +)$-convolution problem. We achieve this via a series of reductions as illustrated in Figure 6.

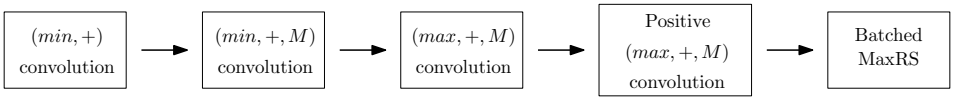


Fig. 6. Series of reductions leading to conditional hardness of batched MaxRS.

The reduction from (min, +)-convolution to positive (max, +, M)-convolution goes via a series of straightforward reductions, the details of which are given in Subsection 5.1 to 5.3. The technically interesting part of the reduction, which is from positive (max,+,M)-convolution to batched MaxRS is given in Subsection 5.4.

5.1 Reduction from (min,+)-convolution to (min,+,M)-convolution

The $(\min, +, M)$*-convolution problem.* Given two sequences $D = (D_0, D_1, \ldots, D_{n-1})$, $E = (E_0, E_1, \ldots, E_{n-1})$, and a set of m distinct indices $M = \{k_0, k_1, \ldots, k_{m-1}\}$, where each $k_s \in \{0, 1, \ldots, n-1\}$. The $(\min, +, M)$-convolution problem is to compute the sequence $F_M = (F_{k_s})_{s=0}^{m-1}$ where for each $k_s \in M$,

$$F_{k_s} = \min_{i+j=k_s} (A_i + B_j).$$

Reduction: Let input sequences A, B be given for an instance of (min, +)-convolution, and we are required to compute $C_k = \min_{i+j=k}(A_i + B_j)$ for all $k \in K_{\text{all}} = \{0, 1, \ldots, n-1\}$. We will solve this using an oracle for the $(\min, +, M)$-convolution problem.

We partition the set K_{all} into $p = \lceil n/m \rceil$ disjoint subsets $M_1, M_2, \ldots, M_p$. Each subset M_s contains at most m indices from K_{all}. For instance, we can define $M_s = \{i \in K_{\text{all}} \mid (s-1)m \leq i < sm \text{ and } i < n\}$. For each $s \in \{1, \ldots, p\}$, we make one call to the $(\min, +, M)$-convolution oracle with inputs A, B, and M_s. The union of the results from these p calls yields all the required C_k values for the original (min, +)-convolution problem.

If we could solve $(\min, +, M)$-convolution in $o(nm)$ time, then we would have solved the (min, +)-convolution problem in $o(nm) \times n/m = o(n^2)$ time. But the latter problem is conjectured to be $\Omega(n^2)$ hard, and hence the $(\min, +, M)$-convolution problem has a conditional lower bound of $\Omega(nm)$.

5.2 Reduction from (min,+,M)-convolution to (max,+,M)-convolution

The $(\max, +, M)$*-convolution problem.* Given two sequences $A = (A_0, A_1, \ldots, A_{n-1})$, $B = (B_0, B_1, \ldots, B_{n-1})$, and a set of m distinct indices $M = \{k_0, k_1, \ldots, k_{m-1}\}$, where each $k_s \in$

$\{0, 1, \ldots, n-1\}$. The $(\max, +, M)$-convolution problem is to compute the sequence $C_M = (C_{k_s})_{s=0}^{m-1}$ where for each $k_s \in M$,

$$C_{k_s} = \max_{i+j=k_s} (A_i + B_j).$$

Reduction: Suppose we are given an instance of the $(\min, +, M)$-convolution problem, consisting of sequences D, E (each of length n) and a set of m target indices M. We aim to compute $F_k = \min_{i+j=k}(D_i + E_j)$ for each $k \in M$. We construct two new sequences, A' and B', as follows:

$$A'_i = -D_i, \quad \text{and} \quad B'_j = -E_j, \quad \text{for } j \in \{0, \ldots, n-1\}.$$

Next, we invoke an oracle for the $(\max, +, M)$-convolution problem with the transformed sequences A', B', and the original set of indices M. Let the output from this oracle be denoted by C'_k for each $k \in M$. According to the definition of $(\max, +, M)$-convolution, for each $k \in M$:

$$C'_k = \max_{i+j=k}(A'_i + B'_j) = \max_{i+j=k}(-D_i - E_j) = -\left(\min_{i+j=k}(D_i + E_j)\right) = -C_k.$$

Therefore, the desired results C_k for the original $(\min, +, M)$-convolution instance are obtained by computing $C_k = -C'_k$ for every $k \in M$. Note that the construction of sequences A and B requires $O(n)$ time, and the recovery of C_k from C'_k takes $O(m)$ time. This is hence an efficient, linear-time reduction.

5.3 Reduction from (max,+,M)-convolution to Positive (max,+,M)-convolution

The positive $(\max, +, M)$*-convolution problem.* Given two sequences $A = (A_0, A_1, \ldots, A_{n-1})$ and $B = (B_0, B_1, \ldots, B_{n-1})$, such that $A_i \geq 0$ for all i and $B_j \geq 0$ for all j. Additionally, a set of m distinct indices $M = \{k_0, k_1, \ldots, k_{m-1}\}$ is provided, where each $k_s \in \{0, 1, \ldots, n-1\}$. The Positive $(\max, +, M)$-convolution problem is to compute the sequence $C_M = (C_{k_s})_{s=0}^{m-1}$ where for each $k_s \in M$,

$$C_{k_s} = \max_{i+j=k_s} (A_i + B_j).$$

Reduction: Suppose we are given an instance of the $(\max, +, M)$-convolution problem, with sequences A, B (which may contain negative values) and a set of m indices M. Our goal is to compute $C_k = \max_{i+j=k}(A_i + B_j)$ for each $k \in M$, using an oracle for Positive $(\max, +, M)$-convolution which requires non-negative input sequences.

First, find the minimum element Δ present in either sequence A or B:

$$\Delta = \min\left(\left(\min_{0 \leq i \leq n} A_i\right), \left(\min_{0 \leq j \leq n} B_j\right)\right).$$

If $\Delta \geq 0$, both sequences A and B are already non-negative. In this scenario, A, B, and M can be passed directly to the Positive $(\max, +, M)$-convolution oracle, and its output will be the desired C_M.

If $\Delta < 0$, we construct two new non-negative sequences A' and B' as follows:

$$A'_i = A_i - \Delta, \quad \text{and} \quad B'_j = B_j - \Delta, \quad \text{for } j \in \{0, \ldots, n-1\}.$$

Since $A_i \geq \Delta$ and $B_j \geq \Delta$ for all respective i, j, it is guaranteed that $A'_i \geq 0$ and $B'_j \geq 0$. We then call the Positive $(\max, +, M)$-convolution oracle with these non-negative sequences A', B', and the set of indices M. Let the results from the oracle be C'_k for $k \in M$. For each $k \in M$, the oracle computes:

$$C'_k = \max_{i+j=k}(A'_i + B'_j) = \left(\max_{i+j=k}(A_i + B_j)\right) - 2\Delta = C_k - 2\Delta.$$

Thus we can obtain the required sequence C_k as $C_k = C'_k + 2\Delta$. The computation of Δ takes $O(n)$ time. The construction of A' and B' also takes $O(n)$ time, and recovering the final results C_k from C'_k takes $O(m)$ time. This is hence an efficient, linear-time reduction.

5.4 From positive (max,+,M)-convolution to batched MaxRS

Reduction: Let an instance of Positive $(\max, +, M)$-convolution be given by two sequences A, B where $A_i \geq 0, B_j \geq 0$ for all i, j, and a set of m target indices $M = \{k_0, k_1, \ldots, k_{m-1}\}$, where each $k_s \in \{0, 1, \ldots, n-1\}$. We wish to compute $C_{k_s} = \max_{i+j=k_s}(A_i + B_j)$ for each $k_s \in M$.

In this subsection, we represent a weighted point p, located at x-coordinate x_p, and with weight w_p, as $p = (x_p, w_p)$. We construct an instance of batched MaxRS as follows:

(1) **Point set construction:** Create a set P_S of $4n$ weighted points.
- For each A_i where $i \in \{0, \ldots, n-1\}$, create a point $p_i^A = (i, A_i)$, and its 'guard' point $p_i'^A = (i - 0.5, -A_i)$.
- Let $X_{\text{offset}} = 2n - 1$. For each B_j where $j \in \{0, \ldots, n-1\}$, create a point $p_j^B = (X_{\text{offset}} - j, B_j)$, and its guard point $p_j'^B = (X_{\text{offset}} - j + 0.5, -B_j)$.

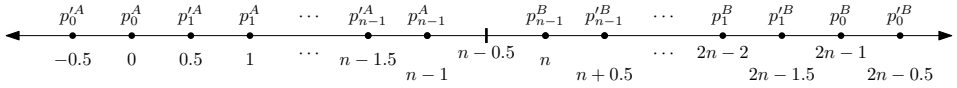


Fig. 7. Position of constructed (weighted) points on real line.

(2) **Query interval lengths:** For each index $k_s \in M$, we define a query interval length as $L_s = 2n - 1 - k_s$ for the batched MaxRS problem. Note that since $0 \leq k_s \leq n-1$, the interval length $L_s \geq n$.

We then query the batched MaxRS oracle with the point set P and the m interval lengths $\{L_s\}_{k_s \in M}$. Let W_s^* be the maximum sum returned by the oracle for the interval of length L_s. We want to claim that this value W_s^* is in fact the required value C_{k_s}.

Lemma 5.1. *For each $k_s \in M$, the maximum weight W_s^* found by the batched MaxRS oracle for an interval of length $L_s = 2n - 1 - k_s$, is equal to $C_{k_s}^* = \max_{i+j=k_s}(A_i + B_j)$.*

Proof Sketch. Since every point has an associated guard point which negates its weight, any placement of an interval can cover the weight of at most one A-point, and at most one B-point. Since placing an interval of length L_s, starting at the coordinate of some A-point p_i^A, covers weight $A_i + B_j$ for $j = k_s - i$ (proved in Lemma 5.2), we can discard all other placements of such a segment (since they will either cover negative weight, zero weight, or weight of a single element A_i or B_j). Using the batched MaxRS oracle gives us the maximum possible such sum, which is hence the maximum possible sum of $A_i + B_j$ where the indices satisfy $i + j = k_s$. □

We now formalize the above intuition and give a rigorous proof. Let $w(I)$ be the weight covered by some interval I.

Lemma 5.2. *The weight covered by an interval $I_{i,j} = [i, (2n-1) - j]$ for some $i, j \in \{0, \ldots, n-1\}$, is given by $w(I_{i,j}) = A_i + B_j$.*

Proof. Since $I_{i,j}$ starts at x-coordinate i, it includes point $p_i^A = (i, A_i)$, but not the corresponding guard point $p_i'^A = (i - 0.5, -A_i)$. Similarly, since $I_{i,j}$ ends at $(2n-1) - j$, it includes the point $p_j^B = ((2n-1) - j, B_j)$, but not its corresponding guard point $p_j'^B = ((2n-1) - j + 0.5, -B_j)$.

Further for all $i < i' \leq n-1$ and $0 \leq j' < j$, the interval $I_{i,j}$ covers the points $p_i^A, p_{i'}^{\prime A}, p_j^B, p_{j'}^{\prime B}$. By construction, if a point and its corresponding guard point are picked, then their weights sum to zero. Hence we get that $w(I_{i,j}) = A_i + B_j$. □

Proof of Lemma 5.1. Since $A_i, B_j \geq 0$ for all $i, j \in \{0, \ldots, n-1\}$, it follows that $C^*_{k_s} \geq 0$. Let (i_0, j_0) be a pair of indices such that $i_0, j_0 \in \{0, \ldots, n-1\}$, $i_0 + j_0 = k_s$, and $A_{i_0} + B_{j_0} = C^*_{k_s}$. Consider an interval $I_{i_0,j_0} = [i_0, (2n-1) - j_0]$. The length of this interval is $x_{end} - x_{start} = (2n-1) - (i_0 + j_0) = (2n-1) - k_s$, which is precisely L_s, the query interval length for the target index k_s. Further from Lemma 5.2 we know that $w(I_{i_0,j_0}) = A_{i_0} + B_{j_0} = C^*_{k_s}$

Since the batched MaxRS oracle returns W^*_s, the maximum possible sum for an interval of length L_s, we have $W^*_s \geq C^*_{k_s}$. To show that $W^*_s = C^*_{k_s}$, we argue that there exists an optimal interval $I^* = [x^*, x^* + L_s]$ which is of the form $[i, (2n-1) - j]$ for some $i, j \in \{0, \ldots, n-1\}$ such that $i + j = k_s$.

(1) If $x^* + L_s < n - 0.5$ (i.e., I^* **covers no** B**-points**): then the interval covers all points starting from $p_0^{\prime A}$ up to some point p_a^A or $p_a^{\prime A}$ for some $a \in \{0, \ldots, n-1\}$ (this is because $L_s \geq n$). This interval hence either covers zero weight (in the former case, when all points covered are paired with their corresponding guard points), or negative weight (in the latter case, where only $p_a^{\prime A}$ is picked but not p_a^A).
(2) If $x^* \geq n - 0.5$ (i.e., I^* **covers no** A**-points**): then the interval covers all points starting from some point p_b^B or $p_b^{\prime B}$ for some $b \in \{0, \ldots, n-1\}$ up to the point $p_0^{\prime B}$. This interval hence either covers zero weight (in the former case, when all covered are paired with their corresponding guard points), or negative weight (in the latter case, where only $p_b^{\prime B}$ is picked but not p_b^B).
(3) We now know that the left endpoint lies to the left of $n - 0.5$, and the right end point lies to the right of $n - 0.5$. If the left endpoint satisfied $x^* < -0.5$ (i.e., I^* **covers all** A**-points**), then all points p_a^A for $a \in \{0, \ldots, n-1\}$ will be paired with their guard points, and hence contribute a total weight of zero. Further, at most one point p_b^B for $b \in \{0, \ldots, n-1\}$ can be left unpaired, and hence the weight covered by the interval is either zero or B_b for some b. Similarly, if the right endpoint satisfied $x^* + L_s > 2n - 0.5$ (i.e., I^* **covers all** B**-points**), then the cost covered by the interval can be either zero or A_a for some a.
(4) We are now left with the case where I^* **covers some** A**-points, and some** B**-points**. Consider an interval starting at an integral coordinate $i \in \{0, \ldots, n-1\}$. Given a length L_s, the interval would end at some $i + L_s = (2n-1) - (k_s - i) = (2n-1) - j$ (for $j \in \{0, \ldots, n-1\}$ satisfying $i + j = k_s$). By Lemma 5.2 this interval covers weight $A_i + B_j$. Now, if we shift the interval left by some $\delta \in (0, 0.5]$, the interval will no more cover point p_j^B, and could potentially additionally cover the point $p_i^{\prime A}$, i.e., drop a positive weight point, and include a negative weight point. This operation can hence not lead to an optimal placement. Similarly, if we move the interval right by some $\delta \in (0, 0.5]$, the interval will no more cover point p_i^A, and potentially cover point $p_j^{\prime B}$ leading again to a sub-optimal placement. Hence, there exists an optimal placement of the interval starting (and ending) at integral coordinates.

Thus, an optimal interval I^* must be of the form $I_{i^*,j^*} = [i^*, (2n-1) - j^*]$ for some $i^*, j^* \in \{0, \ldots, n-1\}$. The length constraint $L_s = (2n-1) - k_s$ implies $i^* + j^* = k_s$. The batched MaxRS oracle hence returns the cost $W^*_s = \max_{i^*+j^*=k_s} w(I_{i^*,j^*}) = \max_{i^*+j^*=k_s}(A_{i^*} + B_{j^*}) = C^*_{k_s}$. □

This chain of reductions establishes the proof of hardness of batched MaxRS (Theorem 1.3).

6 Lower bound for batched smallest k-enclosing interval

In this section, we study a closely related problem, that is the batched version of the smallest k-enclosing interval (BSEI). Given a set of points on the real line, the smallest k-enclosing interval problem requires us to find the smallest interval that encloses k points. More formally, in the batched version of the problem, we are given a set of n points $p_1, p_2, \ldots, p_n$ on the real line. The goal is to find the smallest interval enclosing k points, for all $k \in [n]$.

There is a quadratic time algorithm for the problem, and we show that there is also a matching quadratic lower bound conditioned on the conjectured hardness of the $(\min, +)$-convolution problem. We achieve the required result by the series of reductions: $(\min, +)$-convolution $\leq$ monotone $(\min, +)$-convolution $\leq$ BSEI. The problem definitions and the reductions are given below.

6.1 Reduction from (min,+)-convolution to monotone (min,+)-convolution

Definition 6.1 (Monotone $(\min, +)$*-convolution).* Given two sequences $D = (D_0, D_1, \ldots, D_{n-1})$ and $E = (E_0, E_1, \ldots, E_{n-1})$, such that $D_0 > D_1 > \ldots > D_{n-1}$ and $E_0 > E_1 > \ldots > E_{n-1}$, the monotone $(\min, +)$-convolution produces a new sequence F, where for each $k \in \{0, \ldots, n-1\}$,

$$F_k = \min_{i+j=k} (D_i + E_j).$$

Reduction: We are given unsorted sequences A, B, as part of the instance of $(\min, +)$-convolution. We will generate two monotone sequences D, E, to create an instance of the monotone $(\min, +)$-convolution problem, such that solving the monotone instance would give us a solution for the original instance.

(1) Let $\Delta = 1 + \max_{i \in [n-1]} \max((A_i - A_{i-1}), (B_i - B_{i-1}))$. So the new sequences are, for all $i \in \{0, \ldots, n-1\}$, $D_i = A_i - i \cdot \Delta$ and $E_i = B_i - i \cdot \Delta$. Note that for all $i \in [n-1]$, $D_{i-1} - D_i = A_{i-1} - A_i + \Delta > 0$, and hence the sequence is strictly decreasing (and similarly sequence E is strictly decreasing).

(2) Solve the monotone $(\min, +)$-convolution problem to obtain the sequence F such that

$$F_k = \min_{i+j=k} (D_i + E_j) = \min_{i+j=k} ((A_i - i \cdot \Delta) + (B_j - j \cdot \Delta)) = C_k - k \cdot \Delta.$$

(3) Obtain the solution to the $(\min, +)$-convolution problem as $C_k = F_k + k \cdot \Delta$. Note that the reduction only takes linear time, to compute Δ, and then subtract it from every element of the input sequences A and B.

6.2 Reduction from monotone (min,+)-convolution to BSEI

Reduction: We are given monotone sequences D, E as part of an instance of monotone $(\min, +)$-convolution. Consider the sequence P of $2n$ elements defined as follows: For each $i \in \{0, \ldots, n-1\}$, $P_i = -D_i + (D_{n-1} - 1)$, and $P_{n+i} = E_{(n-1)-i} + (1 - E_{n-1})$. Notice that the added offsets of $(D_{n-1} - 1)$ ($(1 - E_{n-1})$ resp.) ensure that P_i (P_{n+i} resp.) remains negative (positive resp.).

Now let us construct an instance of BSEI with $2n$ points, and the location of the i-th point being given by P_i. See Figure 8 to see the locations of points in this instance. Now, if we could solve the BSEI problem, we would obtain a sequence $G_1, G_2, \ldots, G_{2n}$, where G_k denotes the minimum length of an interval covering k points.

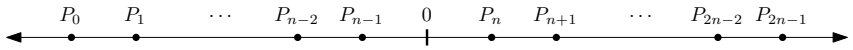


Fig. 8. Values of newly constructed sequence as points on the real line.

The distance between arbitrary (say i-th) point to its ℓ-th successor is $P_{\ell+i} - P_i$, and this covers $\ell + 1$ points. So for $k \in \{0, \ldots .n-1\}$, we can write the term F_k as,

$$\begin{aligned} F_k &= \min_{i\in\{0,\ldots,k\}} (E_{k-i} + D_i) = \min_{i\in\{0,\ldots,k\}} \left(E_{(n-1)-(n-1-k+i)} + D_i\right) \\ &= (D_{n-1} - 1) - (1 - E_{n-1}) + \min_{i\in\{0,\ldots,k\}} \left(P_{n+(n-1-k+i)} - P_i\right) \\ &= G_{2n-k} + D_{n-1} + E_{n-1} - 2. \end{aligned}$$

So, if we could solve BSEI, we would be able to get the solution to the instance of monotone $(min, +)$-convolution as $F_k = G_{2n-k} + D_{n-1} + E_{n-1} - 2$ for $k \in \{0, 1, \ldots, n-1\}$. Note that this reduction only takes linear time, as we only have to construct the new array by iterating over the given sequences once. This finishes the proof of Theorem 1.4.

7 Future work

We conclude the paper with a few open problems.

- Our technique based on output sensitivity and sampling colors in Section 4 to obtain $(1-\varepsilon)$-approximation for colored disk MaxRS in $\mathbb{R}^2$ (Theorem 1.6) is quite general, and we believe it will be useful for solving other variants of MaxRS. An immediate open problem is to extend this technique to answer colored MaxRS for boxes and balls in $\mathbb{R}^3$.
- For the batched MaxRS problem in $\mathbb{R}^1$, the conditional hardness result—lower bound of $\Omega(mn)$—holds for the weighted set of points (Theorem 1.3). It remains open whether a similar lower bound can be established for the unweighted case in $\mathbb{R}^1$. The best known upper bound for batched MaxRS for rectangles in $\mathbb{R}^2$ is still $O(mn \log n)$ in the unweighted setting (by running the exact $O(n \log n)$ time algorithm [32, 48] for each of the m ranges).
- The batched MaxRS problem for disks in $\mathbb{R}^2$ can be solved in $O(mn^2)$ time (by running the exact $O(n^2)$ time algorithm [12] for each of the m ranges). Can we prove a matching lower bound?
- Finally, for the static MaxRS for d-balls, a simple upper bound of $\widetilde{O}(n^d)$ can be obtained by computing the entire arrangement. It would be interesting to prove if there is a matching lower bound. Currently, such a conditional hardness result is known for disks [5].

Acknowledgments

Rachana Gusain is supported by the Prime Minister's Research Fellowship (PMRF). Saladi Rahul would like to thank "Walmart Center for Tech Excellence" and "Anusandhan National Research Foundation (ANRF) CRG/2023/005776.

References

[1] Pankaj K. Agarwal, Torben Hagerup, Rahul Ray, Micha Sharir, Michiel H. M. Smid, and Emo Welzl. 2002. Translating a Planar Object to Maximize Point Containment. In *10th Annual European Symp. on Algorithms, ESA 2002*. 42–53.

[2] Pankaj K. Agarwal and Alex Steiger. 2021. An Output-Sensitive Algorithm for Computing the Union of Cubes and Fat Boxes in 3D. In *48th Int'l Colloquium on Automata, Languages, and Programming, ICALP 2021*. 1–20.

[3] Daichi Amagata and Takahiro Hara. 2016. Monitoring MaxRS in Spatial Data Streams. In *23rd Int'l Conf. on Extending Database Technology, EDBT 2016*. 317–328.

[4] Daichi Amagata and Takahiro Hara. 2017. A General Framework for MaxRS and MaxCRS Monitoring in Spatial Data Streams. *ACM Trans. Spatial Algorithms Syst.* 3, 1 (2017), 1–34.

[5] Boris Aronov and Sariel Har-Peled. 2008. On Approximating the Depth and Related Problems. *SIAM J. Comput.* 38, 3 (2008), 899–921.

[6] Franz Aurenhammer. 1988. Improved Algorithms for Discs and Balls Using Power Diagrams. *Journal of Algorithms* 9, 2 (1988), 151–161.

[7] Arturs Backurs, Piotr Indyk, and Ludwig Schmidt. 2017. Better Approximations for Tree Sparsity in Nearly-Linear Time. In *28th Annual ACM-SIAM Symp. on Discrete Algorithms, SODA 2017*. 2215–2229.

[8] Gill Barequet, Matthew T. Dickerson, and Petru Pau. 1997. Translating a Convex Polygon to Contain a Maximum Number of Points. *Comput. Geom.* 8 (1997), 167–179.

[9] Xin Cao, Gao Cong, Christian S. Jensen, and Man Lung Yiu. 2014. Retrieving Regions of Interest for User Exploration. In *40th Int'l Conf. on Very Large Data Bases, VLDB 2014*. 733–744.

[10] Timothy M. Chan. 2010. A (slightly) faster algorithm for Klee's measure problem. *Comput. Geom.* 43, 3 (2010), 243–250.

[11] Timothy M. Chan and Sariel Har-Peled. 2021. Smallest k-Enclosing Rectangle Revisited. *Discret. Comput. Geom.* 66, 2 (2021), 769–791.

[12] Bernard Marie Chazelle and Der-Tsai Lee. 1986. On a Circle Placement Problem. *Computing* 36, 1 (1986), 1–16.

[13] Xuefeng Chen, Xin Cao, Yifeng Zeng, Yixiang Fang, and Bin Yao. 2020. Optimal Region Search with Submodular Maximization. In *29th Int'l Joint Conf. on Artificial Intelligence, IJCAI 2020*. 1216–1222.

[14] Zitong Chen, Yubao Liu, Raymond Chi-Wing Wong, Jiamin Xiong, Ganglin Mai, and Cheng Long. 2015. Optimal Location Queries in Road Networks. *ACM Trans. Database Syst.* 40, 3 (2015), 1–41.

[15] Zijun Chen, Qin Yuan, and Wenyuan Liu. 2019. Monitoring best region in spatial data streams in road networks. *Data Knowl. Eng.* 120 (2019), 100–118.

[16] Dong-Wan Choi, Chin-Wan Chung, and Yufei Tao. 2012. A Scalable Algorithm for Maximizing Range Sum in Spatial Databases. *Proc. VLDB Endow.* 5, 11 (2012), 1088–1099.

[17] Dong-Wan Choi, Chin-Wan Chung, and Yufei Tao. 2014. Maximizing Range Sum in External Memory. *ACM Trans. Database Syst.* 39, 3 (2014), 1–44.

[18] Alexander Mikhailovich Chudnov. 1986. On Minimax Signal Generation and Reception Algorithms. *Problems of Information Transmission* 22, 4 (1986), 49–54.

[19] Marek Cygan, Marcin Mucha, Karol Wegrzycki, and Michal Wlodarczyk. 2019. On Problems Equivalent to (min, +)-Convolution. *ACM Trans. Algorithms* 15, 1 (2019), 1–25.

[20] Mark de Berg, Sergio Cabello, and Sariel Har-Peled. 2009. Covering Many or Few Points with Unit Disks. *Theory Comput. Syst.* 45, 3 (2009), 446–469.

[21] Mark de Berg, Marc van Kreveld, Mark Overmars, and Otfried Schwarzkopf. 1997. *Computational Geometry: Algorithms and Applications*. Springer.

[22] Celina M. H. de Figueiredo and Guilherme Dias da Fonseca. 2009. Enclosing weighted points with an almost-unit ball. *Inf. Process. Lett.* 109, 21-22 (2009), 1216–1221.

[23] Jonathan Eckstein, Peter L. Hammer, Ying Liu, Mikhail Nediak, and Bruno Simeone. 2002. The Maximum Box Problem and its Application to Data Analysis. *Comput. Optim. Appl.* 23, 3 (2002), 285–298.

[24] Alon Efrat, Micha Sharir, and Alon Ziv. 1994. Computing the Smallest K-enclosing Circle and Related Problems. *Computational Geometry* 4 (1994), 119–136.

[25] Kaiyu Feng, Gao Cong, Sourav S. Bhowmick, Wen-Chih Peng, and Chunyan Miao. 2016. Towards Best Region Search for Data Exploration. In *Int'l Conf. on Management of Data, SIGMOD 2016*. 1055–1070.

[26] Kaiyu Feng, Gao Cong, Christian S. Jensen, and Tao Guo. 2019. Finding Attribute-Aware Similar Region for Data Analysis. *Proc. VLDB Endow.* 12, 11 (2019), 1414–1426.

[27] Kaiyu Feng, Tao Guo, Gao Cong, Sourav S. Bhowmick, and Shuai Ma. 2020. SURGE: Continuous Detection of Bursty Regions Over a Stream of Spatial Objects. *IEEE Trans. Knowl. Data Eng.* 32, 11 (2020), 2254–2268.

[28] Sathish Govindarajan, Pankaj K. Agarwal, and Lars Arge. 2003. CRB-Tree: An Efficient Indexing Scheme for Range-Aggregate Queries. In *9th Int'l Conf. on Database Theory, ICDT 2003*. 143–157.

[29] Muhammed Mas-ud Hussain, Kazi Ashik Islam, Goce Trajcevski, and Mohammed Eunus Ali. 2017. Towards Efficient Maintenance of Continuous MaxRS Query for Trajectories. In *20th Int'l Conf. on Extending Database Technology, EDBT 2017*. 402–413.

[30] Muhammed Mas-ud Hussain and Goce Trajcevski. 2018. Maximizing area-range sum for spatial shapes (MAxRS^3). In *30th Int'l Conf. on Scientific and Statistical Database Management, SSDBM 2018*. 28:1–28:4.

[31] Muhammed Mas-ud Hussain, Panitan Wongse-ammat, and Goce Trajcevski. 2015. Demo: Distributed MaxRS in Wireless Sensor Networks. In *13th ACM Conf. on Embedded Networked Sensor Systems, SenSys 2015*. 479–480.

[32] Hiroshi Imai and Takao Asano. 1983. Finding the Connected Components and a Maximum Clique of an Intersection Graph of Rectangles in the Plane. *J. Algorithms* 4, 4 (1983), 310–323.

[33] Klaus Jansen and Lars Rohwedder. 2023. On Integer Programming, Discrepancy, and Convolution. *Math. Oper. Res.* 48, 3 (2023), 1481–1495.

[34] Kai Jin, Jian Li, Haitao Wang, Bowei Zhang, and Ningye Zhang. 2018. Near-linear time approximation schemes for geometric maximum coverage. *Theor. Comput. Sci.* 725 (2018), 64–78.

[35] Klara Kedem, Ron Livne, János Pach, and Micha Sharir. 1986. On the Union of Jordan Regions and Collision-Free Translational Motion Amidst Polygonal Obstacles. *Discret. Comput. Geom.* 1 (1986), 59–71.

[36] Marvin Künnemann, Ramamohan Paturi, and Stefan Schneider. 2017. On the Fine-Grained Complexity of One-Dimensional Dynamic Programming. In *44th Int'l Colloquium on Automata, Languages, and Programming, ICALP 2017*. 1–15.
[37] Eduardo Sany Laber, Wilfredo Bardales Roncalla, and Ferdinando Cicalese. 2014. On lower bounds for the Maximum Consecutive Subsums Problem and the (min, +)-convolution. In *IEEE Int'l Symp. on Information Theory, 2014*. 1807–1811.
[38] Qiyu Liu, Xiang Lian, and Lei Chen. 2019. Probabilistic Maximum Range-Sum Queries on Spatial Database. In *27th ACM SIGSPATIAL Int'l Conf. on Advances in Geographic Information Systems, GIS 2019*. 159–168.
[39] Ruifeng Liu, Ada Wai-Chee Fu, Zitong Chen, Silu Huang, and Yubao Liu. 2016. Finding multiple new optimal locations in a road network. In *24th ACM SIGSPATIAL Int'l Conf. on Advances in Geographic Information Systems, GIS 2016*. 1–10.
[40] Siyu Liu, Qizhi Liu, and Zhifeng Bao. 2020. DARS: Diversity and Distribution-Aware Region Search. In *Database Systems for Advanced Applications - 25th Int'l Conf. , DASFAA 2020*. 204–220.
[41] Yuanjun Liu, Zhengcao Zhang, Jianfeng Qu, Guanfeng Liu, and An Liu. 2024. Beyond SweepLine: Efficient MaxRS Queries over Inaccurate Location Data. In *Database Systems for Advanced Applications - 29th Int'l Conf., DASFAA 2024*. 119–135.
[42] Jiří Matoušek. 1994. Geometric Range Searching. *ACM Comput. Surv.* 26, 4 (1994), 421–461.
[43] Mir Imtiaz Mostafiz, S. M. Farabi Mahmud, Muhammed Mas-ud Hussain, Mohammed Eunus Ali, and Goce Trajcevski. 2017. Class-based Conditional MaxRS Query in Spatial Data Streams. In *29th Int'l Conf. on Scientific and Statistical Database Management, SSDBM 2017*. 13:1–13:12.
[44] Mervin E. Muller. 1959. A Note on a Method for Generating Points Uniformly on N-Dimensional Spheres. *Commun. ACM* 2, 4 (1959), 19–20.
[45] Ketan Mulmuley. 1991. A Fast Planar Partition Algorithm, II. *J. ACM* 38, 1 (1991), 74–103.
[46] Yuki Nakayama, Daichi Amagata, and Takahiro Hara. 2016. An Efficient Method for Identifying MaxRS Location in Mobile Ad Hoc Networks. In *Database and Expert Systems Applications - 27th Int'l Conf., DEXA 2016*. 37–51.
[47] Yuki Nakayama, Daichi Amagata, and Takahiro Hara. 2017. Probabilistic MaxRS Queries on Uncertain Data. In *Database and Expert Systems Applications - 28th Int'l Conf., DEXA 2017*. 111–119.
[48] Subhas C. Nandy and Bhargab B. Bhattacharya. 1995. A Unified Algorithm for Finding Maximum and Minimum Object Enclosing Rectangles and Cuboids. *Computers & Mathematics with Applications* 29, 8 (1995), 45–61.
[49] Hamid Shahrivari, Matthaios Olma, Odysseas Papapetrou, Dimitrios Skoutas, and Anastasia Ailamaki. 2020. A Parallel and Distributed Approach for Diversified Top-k Best Region Search. In *23rd Int'l Conf. on Extending Database Technology, EDBT 2020*. 265–276.
[50] Cheng Sheng and Yufei Tao. 2011. New Results on Two-dimensional Orthogonal Range Aggregation in External Memory. In *30th ACM SIGMOD-SIGACT-SIGART Symp. on Principles of Database Systems, PODS 2011*. 129–139.
[51] Dimitrios Skoutas, Dimitris Sacharidis, and Kostas Patroumpas. 2018. Efficient Progressive and Diversified Top-k Best Region Search. In *26th ACM SIGSPATIAL Int'l Conf. on Advances in Geographic Information Systems, GIS 2018*. 299–308.
[52] Ashraf Tahmasbi and Goce Trajcevski. 2022. Maximum Range-Sum for Dynamically Occurring Objects with Decaying Weights. In *Advances in Databases and Information Systems - 26th European Conf. , ADBIS 2022*. 238–252.
[53] Yufei Tao, Xiaocheng Hu, Dong-Wan Choi, and Chin-Wan Chung. 2013. Approximate MaxRS in Spatial Databases. In *39th Int'l Conf. on Very Large Data Bases, VLDB 2013*, Vol. 6. 1546–1557.
[54] Ron Wein, Eric Berberich, Efi Fogel, Dan Halperin, Michael Hemmer, Oren Salzman, and Baruch Zukerman. [n. d.]. *CGAL 6.0.1 - 2D Arrangements*. https://doc.cgal.org/latest/Arrangement_on_surface_2/index.html
[55] Eric W. Weisstein. [n. d.]. Hypersphere point picking. From MathWorld—A Wolfram Web Resource. https://mathworld.wolfram.com/HyperspherePointPicking.html
[56] Wikipedia. [n. d.]. Hyperspherical cap — Wikipedia, The Free Encyclopedia. https://en.wikipedia.org/wiki/Spherical_cap#Hyperspherical_cap
[57] Xiaokui Xiao, Bin Yao, and Feifei Li. 2011. Optimal Location Queries in Road Network Databases. In *27th IEEE Int'l Conf. on Data Engineering, ICDE 2011*. 804–815.
[58] Kaiqi Zhang, Hong Gao, Xixian Han, Jian Chen, and Jianzhong Li. 2022. Maximizing Range Sum in Trajectory Data. In *38th IEEE Int'l Conf. on Data Engineering, ICDE 2022*. 755–766.
[59] Kaiqi Zhang, Siyuan Zhang, Jirun Gao, Hongzhi Wang, Hong Gao, and Jianzhong Li. 2025. Approximation algorithms for finding maximum containing circle and sphere. *Theor. Comput. Sci.* 1023 (2025), 1–13.
[60] Xiaoling Zhou and Wei Wang. 2016. An Index-Based Method for Efficient Maximizing Range Sum Queries in Road Network. In *Databases Theory and Applications - 27th Australasian Database Conf., ADC 2016*. 95–109.
[61] Xiaoling Zhou, Wei Wang, and Jianliang Xu. 2016. General Purpose Index-Based Method for Efficient MaxRS Query. In *Database and Expert Systems Applications - 27th Int'l Conf., DEXA 2016*. 20–36.

Received June 2025; accepted August 2025