The Most Likely Object to be Seen Through a Window

Paz Carmi[1], Farah Chanchary[2], Anil Maheshwari[2], and Michiel Smid[2]

[1] Department of Computer Science, Ben-Gurion University
Beer-Sheva 84105, Israel
[2] School of Computer Science, Carleton University,
Ottawa, ON, K1S 5B6, Canada
carmip@cs.bgu.ac.il,farah.chanchary@carleton.ca,anil@scs.carleton.ca,michiel@scs.carleton.ca

Abstract. We study data structures to answer *window queries* using stochastic input sequences. The first problem is the *most likely maximal point* in a query window: Let $\alpha_1, \ldots, \alpha_c$ be constants, with $0 < \alpha_1 < \alpha_2 < \ldots < \alpha_c < 1$. Let $P = P_1 \cup P_2 \cup \ldots \cup P_c$ be a set of n points in $\mathbb{R}^d$, for some fixed d. For $i = 1, 2, \ldots, c$, each point in P_i is associated with a probability α_i of existence. A point $p = (x_1, \ldots, x_d)$ in P is on the maximal layer of P if there is no other point $q = (x'_1, \ldots, x'_d)$ in P such that $x'_1 > x_1, x'_2 > x_2, \ldots,$ and $x'_d > x_d$. Consider a random subset of P obtained by including, for $i = 1, 2, \ldots, c$, each point of P_i independently with probability α_i. For a query interval $[i, j]$, with $i \leq j$, we report the point in $P_{i,j} = (p_i, \ldots, p_j)$ that has the highest probability to be on the *maximal layer* of $P_{i,j}$ in $O(1)$ time using $O(n \log n)$ space. We solve a special problem as follows. A sequence P of n points in $\mathbb{R}^d$ is given ($d \geq 2$), where each point P has a probability $\in (0, 1]$ of existence associated with it. Given a query interval $[i, j]$ and an integer t with $i \leq t \leq j$, we report the probability of p_t to be on the maximal layer of $P_{i,j}$ in $O(\log^d n)$ time using $O(n \log^d n)$ space.

The second problem we consider is the *most likely common element* problem. Let $\mathcal{U} = \{1, 2, \ldots, n\}$ be the universe. Let $S_1, S_1, \ldots, S_m$ be a sequence of random subsets of $\mathcal{U}$ such that for $p = 1, \ldots, m$ and $i = 1, \ldots, n$, element i is added to S_p with probability α_{pi} (independently of other choices). Let τ be a fixed real number with $0 < \tau \leq 1$. For query indices p, q, i and j, with $1 \leq p \leq q \leq m$ and $1 \leq i \leq j \leq n$, we decide whether there exists an element k with $i \leq k \leq j$ such that $\Pr(k \in \cap_{r=p}^{q} S_r) \geq \tau$ in $O(1)$ time using $O(mn)$ space and report these elements in $O(\log n + w)$ time, where w is the size of the output.

Keywords: maximal point, most likely common element, window query, set membership

1 Introduction

Recent developments in real-world data acquisition methods (for example, reading data through sensor networking), other scientific measurements, and subsequent data management techniques (such as cleaning and integrating datasets) have led to a massive data generation with some inherent uncertainty. Various models of uncertainty have been devised to categorize and study these data. These models are studied not just in the field of data structures and algorithms, but also in data mining [6, 31], database management [18], statistics [28], physics [26], GIS and remote sensing [19].

In a *window query* we are given a sequence of input data and a predicate $\mathcal{P}$. We want to preprocess the input into a suitable data structure such that given a query window, it can efficiently answer the query based on only the input elements that lie in the query window and matches $\mathcal{P}$. Let i and j be two positive integers, with $i \leq j$, such that the interval $[i, j]$ represents a *query window*. In this paper we study *window queries* for *stochastic* input sequences such as geometric objects like points and elements of sets. We present data structures for solving two window problems, namely *the most likely maximal point problem* and *the most likely common element problem*. The problem definitions are as follows.

MLMP: Most Likely Maximal Point problem. Let $P = (p_1, \ldots, p_n)$ be a sequence of n points in $\mathbb{R}^d$ with $d \geq 1$. For a point $p = (x_1, \ldots, x_d) \in P$, we define $\mathcal{D}(p)$ to be the set of points of P that dominate p, i.e., $\mathcal{D}(p) = \{q = (x'_1, \ldots, x'_d)$ of P: $x'_1 > x_1, \ldots, x'_d > x_d\}$. A point p is dominated by q if $q \in \mathcal{D}(p)$. A point $p \in P$ is a maximal point if $\mathcal{D}(p) \cap P = \emptyset$. The maximal layer is the set of all maximal points in P.

Let α_1 and α_2 be constants with $0 < \alpha_1 < \alpha_2 < 1$. Recall that P is a sequence of n points in $\mathbb{R}^d$. Let each point of P be either colored blue or red. Let $B \subseteq P$ be the set of all blue points and let $R \subseteq P$ be

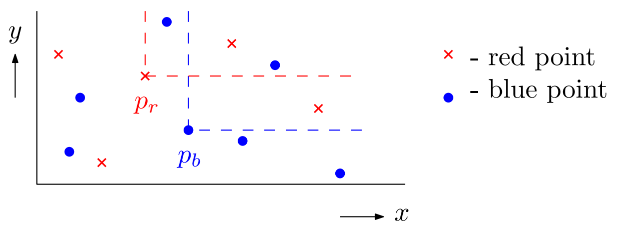


Fig. 1. An instance of $\mathrm{MLMP}(B, \alpha_1; R, \alpha_2)$ in $\mathbb{R}^2$ is given. $\Pr(p_b$ is on maximal layer in $P) = \alpha_1(1-\alpha_1)(1-\alpha_2)^2$ and $\Pr(p_r$ is on maximal layer in $P) = \alpha_2(1-\alpha_1)^2(1-\alpha_2)$.

the set of all red points. Furthermore, each blue point is associated with a probability α_1 of existence and each red point is associated with a probability α_2 of existence. Note that $P = B \cup R$ and we denote this instance of P as $(B, \alpha_1; R, \alpha_2)$. Let Z be a random subset of P given by including each blue point of P in Z independently with probability α_1 and each red point of P in Z independently with probability α_2.

For a point $p' \in P$, let s be the number of blue points in P in $\mathcal{D}(p')$. Similarly let t be the number of red points in P in $\mathcal{D}(p')$. If p' itself is a blue point then the probability that p' is on the maximal layer in Z is $\alpha_1(1-\alpha_1)^s(1-\alpha_2)^t$. Similarly if p' is a red point then the probability that p' is on the maximal layer in Z is $\alpha_2(1-\alpha_1)^s(1-\alpha_2)^t$. See Fig. 1 for an example.

For $1 \leq i < j \leq n$, let $P_{i,j}$ denote the subsequence of stochastic points $(p_i, p_{i+1}, \ldots, p_j)$. Given a query interval $[i, j]$, with $i \leq j$, we report the most likely maximal point (MLMP) defined to be the point with the highest probability to be on the maximal layer of the sequence of stochastic points in $P_{i,j}$.

Given a query interval $[i, j]$, we define $P_{i,j} = B_{i,j} \cup R_{i,j}$, where $B_{i,j} = B \cap P_{i,j}$ and $R_{i,j} = R \cap P_{i,j}$. We present data structures that can answer window queries for the most likely maximal point in the above mentioned instance of P. Now the window problem of the most likely maximal point is denoted as $\mathrm{MLMP}(B_{i,j}, \alpha_1; R_{i,j}, \alpha_2)$.

Next we show how to extend this technique to solve the c-colored MLMP problem defined as follows. Let $\alpha_1, \ldots, \alpha_c$ be constants, with $0 < \alpha_1 < \alpha_2 < \ldots < \alpha_c < 1$. Given a sequence of stochastic points $P = P_1 \cup P_2 \cup \ldots \cup P_c$ in $\mathbb{R}^d$, where c is a constant and for $r = 1, \ldots, c$, each r-colored point belongs to P_r has a probability α_r associated with it and $d \geq 1$. P can be preprocessed into a data structure that can report the most likely maximal point within $P_{i,j}$, with $1 \leq i \leq j \leq n$.

We also answer the following query. Suppose a sequence of stochastic points $P = (p_1, p_2, \ldots, p_n)$ in $\mathbb{R}^2$ is given such that for $k = 1, 2, \ldots, n$, point p_k exists with probability γ_k, with $0 \leq \gamma_k \leq 1$, independently of the other points. Given a query interval $[i, j]$, with $i \leq j$, and an additional integer $t \in [i, j]$, we can report the probability of the point p_t is on the maximal layer in $P_{i,j}$.

MLCE: Most Likely Common Element problem. Let $\mathcal{U} = \{1, 2, \ldots, n\}$ be the universe. Let $S_1, S_2, \ldots, S_m$ be a sequence of random subsets of $\mathcal{U}$ such that for $p = 1, \ldots, m$ and $i = 1, \ldots, n$, element i is added to S_p with probability α_{pi} (independently of other choices). Let τ be a fixed real number with $0 < \tau \leq 1$. We answer queries of the following type: Given query indices p, q, i and j, with $1 \leq p \leq q \leq m$ and $1 \leq i \leq j \leq n$, decide if there exists an element $k \in \{i, i+1, \ldots, j\}$ that is contained in $S_p \cap S_{p+1} \cap \ldots, \cap S_q$ with probability at least τ.

Observe that $\Pr(k \in \cap_{r=p}^{q} S_r) = \prod_{r=p}^{q} \alpha_{rk}$. Thus, the problem is equivalent to the following: Let $\mathcal{S}$ be an $m \times n$ matrix in which each entry $\mathcal{S}[p, i]$ is a real number $\alpha_{pi} \in [0, 1]$. We want to preprocess $\mathcal{S}$ into a data structure such that for any query values p, q, i and j, we can decide if there exists some integer k for which $i \leq k \leq j$ and $\prod_{r=p}^{q} \alpha_{rk} \geq \tau$.

1.1 Related Work

The window data structures have been considered in many recent studies including [8, 9, 12–17]. These studies solve window queries for various geometric and graph problems and use sequences of geometric objects (such as, points, line segments and polygons) or graph edges as input. To the best of our knowledge, none of these papers have considered any *model of uncertainty* in their studies. Typically, models of uncertainty describe a distribution or regions for each point's location in the input sequence. Many papers have studied (offline) geometric problems with one or more of these models. The most widely studied models of uncertainty are *stochastic points*, where each point p_k has a fixed location which only exists with a probability α_k, with $0 < \alpha_k \leq 1$ (see [7, 22, 23, 32]); *uncertain points*, where each point p_k's location is described by a probability distribution α_k (see [21, 2, 29]); *indecisive points* or *multipoint* model, where each point can take one of a finite number of locations (see [4, 21, 30]) and *imprecise points*, where each point's location is not known precisely but it is restricted to a region (see [20, 25]).

Although, again to the best of our knowledge, these uncertainty models have not been studied so far in the window query setting, various authors presented range searching data structures on stochastic or uncertain geometric objects (e.g., points and line segments) such as range counting coresets for uncertain data [1], range reporting with a query interval and a probability threshold [3], range-maximum query on uncertain data [5], range queries (i.e., report top-k points with highest probabilities) given some query interval [24].

1.2 New Results

The main contributions of this paper are listed below.

1. *Window queries for c-colored MLMP problem:* We show how to preprocess a sequence of n stochastic points in $\mathbb{R}^d$, for a fixed $d \geq 1$, into a data structure of size $O(n \log n)$ so that given a query interval $[i, j]$ with $1 \leq i \leq j \leq n$, it can report the most likely maximal point of $P_{i,j}$ in $O(1)$ time.
2. *Window queries for MLCE problem:* We show how to preprocess a matrix $\mathcal{S}$ of size $m \times n$ with a fixed real number $0 < \tau \leq 1$ into a data structure of size $O(mn)$ in $O(mn)$ time such that for any query values p, q, i and j, we can decide if there exists some integer k for which $i \leq k \leq j$ and $\prod_{r=p}^{q} \alpha_{rk} \geq \tau$ in $O(1)$ time. The reporting version of the query can be answered in $O(\log n + w)$ time, where w is the total number of all elements k for which $i \leq k \leq j$ and $\prod_{r=p}^{q} \alpha_{rk} \geq \tau$.

1.3 Organization

The rest of this paper is organized as follows. Section 2 presents results for the c-colored windowed most likely maximal point problem. This section also outlines a data structure that can report the probability with which a given point can exist on the maximal layer of the queried points. In Section 3, we present algorithms and data structures for the most likely common element problem. Section 4 concludes this paper.

2 Most Likely Maximal Points

The point that has the highest probability to be on the maximal layer in P is defined to be the *most likely maximal point* in P. The window problem of finding the most likely maximal point is as follows. Given a query interval $q = [i, j]$, with $1 \leq i < j \leq n$, report the point that has the highest probability to be on the maximal layer in $P_{i,j}$.

In this section, we initially present the MLMP problem where the points of P are only colored by two colors. We later extend to c-colored MLMP, where $c \geq 2$ is a constant. We further present the data structure for computing the probability for a point to be on the maximal layer in $P_{i,j}$.

2.1 Blue-Red Stochastic Points

Let α_1 and α_2 be constants with $0 < \alpha_1 < \alpha_2 < 1$. Given a query interval $[i, j]$, $1 \leq i \leq j \leq n$, we define $P_{i,j} = \{p_i, p_{i+1}, \ldots, p_j\}$, $B_{i,j}$ is the set of blue points in $B \cap P_{i,j}$ and $R_{i,j}$ is the set of blue points in $R \cap P_{i,j}$. We denote the windowed version of the most likely maximal point problem as MLMP$(B_{i,j}, \alpha_1; R_{i,j}, \alpha_2)$.

Lemma 1. *Consider a query interval $[i, j]$ with $i < j$. If the answer to the most likely maximal point problem is a blue point then it is a maximal point of all points in $P_{i,j}$. If the answer to the most likely maximal point problem is a red point then it is a maximal points in $R_{i,j}$.*

Proof. Suppose the answer to our query is a blue point p_b. We show, by contradiction, that p_b is a maximal point of $P_{i,j}$. Suppose it is not. Then $\mathcal{D}(p_b) \cap P_{i,j} \neq \emptyset$ and assume that there are s points of color blue and t points of color red in $\mathcal{D}(p_b)$, such that $s + t > 0$. Then the probability that p_b is the most likely maximal point is $\alpha_1(1 - \alpha_1)^s(1 - \alpha_2)^t$ but that is strictly less than α_1. It contradicts the assumption that p_b is the most likely maximal point in $P_{i,j}$.

By a similar argument the second statement can also be proved. Suppose the answer to our query is a red point p_r but there exists t other red points in $\mathcal{D}(p_r)$ in $R_{i,j}$. Then the probability that p_r is the most likely maximal point in $P_{i,j}$ is $\alpha_2(1 - \alpha_2)^t$ which is strictly less than α_2.

First we discuss the case where points of $P = B \cup R$ lie on the line. Without loss of generality, we assume P lies on a horizontal line.

Points are in 1-dimension. Let $\hat{P}[1..n]$ be an array such that for $k = 1$ to n, $\hat{P}[k] = p_{k,x}$, where $p_{k,x}$ is the x-coordinate of the point p_k. Fig. 2 illustrates an example with 10 points on the line. Now given a query interval $[i, j]$ the problem of computing the most likely maximal point in $P_{i,j}$ reduces to the problem of computing the most likely maximum value in $\hat{P}[i], \ldots, \hat{P}[j]$.

Data structure: We can directly use the data structure of Agarwal et al.'s result for a set of uncertain points [5] to find the most likely maximum value in the range $[i, j]$ in $O(n^{1-t} + \log n)$ time using $O(n^{1+t})$ space for any $t \in [0, 1]$. Let the most likely maximum value be X. Then the point at position X is the most likely maximal point in $P_{i,j}$. However, we present a simpler data structure here.

Recall that we consider $(B, \alpha_1; R, \alpha_2)$ to be an instance of P, where B contains all the blue points of P and R contains all the red points of P. Let $\hat{B}$ be the array similar to $\hat{P}$ for all blue points and let $\hat{R}$ be the array similar to $\hat{P}$ for all red points. We build two separate 1-dimensional range maximum data structures on $\hat{B}$ and $\hat{R}$. Given a 1-dimensional array A with N entries, the range maximum (RM) query asks for the maximum element within a contiguous subarray of A. Linear time and space preprocessing algorithms are known for the 1-dimensional case that can answer queries in constant time (see for example [10]). Let RM_B be the 1-dimensional RM structure built on $\hat{B}$ and let RM_R be the 1-dimensional RM structure built on $\hat{R}$. Both RM_B and RM_R require $O(n)$ space in total. In addition, we map each blue point p_b in $\hat{B}$ to a 2-dimensional point $(b, \hat{B}[b])$, where b is the position of p_b in P. Suppose $\bar{B}$ is the set of all 2-dimensional points obtained in this way. Now we build a 2-dimensional range tree $T_{\bar{B}}$ on these points. Total space and time required for constructing this range tree is $O(n \log n)$ [11].

Now we discuss how we answer a query. Recall that $0 < \alpha_1 < \alpha_2 < 1$. Given a query interval $[i, j]$ in $\hat{P}$, we first search RM_R data structure to find the maximum value for a red point in the range $[i, j]$. Let this point be $\mathcal{R}_{max}$ with the value $\mathcal{V}_{max}$. Then we search $T_{\bar{B}}$ with a three sided range query $[i, j] \times [\mathcal{V}_{max}, +\infty)$ and count the total number of blue points in this range. Let this number be m. That is, these are the m blue points that exist in the interval $[i, j]$ and have values greater than $\mathcal{V}_{max}$. In other words, these m blue points in $P_{i,j}$ have higher x-coordinate value than that of the red point $\mathcal{R}_{max}$. Then the probability that the point $\mathcal{R}_{max}$ is the most likely maximum point in $P_{i,j}$ is $\alpha_2(1 - \alpha_1)^m$. Now we have the following two cases to consider.

- *Case 1*: If $\alpha_2(1 - \alpha_1)^m > \alpha_1$ then the red point $\mathcal{R}_{max}$ at position max is the most likely maximal point with probability $\alpha_2(1 - \alpha_1)^m$. We report this point as the answer to our query.
- *Case 2*: Otherwise, the most likely maximum point in $P_{i,j}$ is the blue point with maximum x-coordinate in $[i, j]$. To find this blue point with the maximum value we search the range maximum data structure RM_B and report the output point to be the maximum point with probability α_1.

RM_R and RM_B can answer queries in $O(1)$ time. Range counting query can be answered in $O(\log n)$ time. So the total query time is bounded by $O(\log n)$. Therefore we obtain the following theorem.

Theorem 1. *Let* α_1 *and* α_2 *be constants with* $0 < \alpha_1 < \alpha_2 < 1$. *Given an instance of MLMP*$(B, \alpha_1; R, \alpha_2)$ *in 1-dimension,* $P = B \cup R$ *can be preprocessed into a data structure of size* $O(n \log n)$ *in* $O(n \log n)$ *time such that given a query interval* $[i, j]$, *with* $1 \le i < j \le n$, *it can report the most likely maximal point in* $P_{i,j}$ *in* $O(\log n)$ *time.*

Now we present an efficient data structure that can report the window query Q for the most likely maximal point, where the points $(B, \alpha_1; R, \alpha_2)$ are in $\mathbb{R}^d$, with $d \ge 1$, and Q*: Given a query interval*

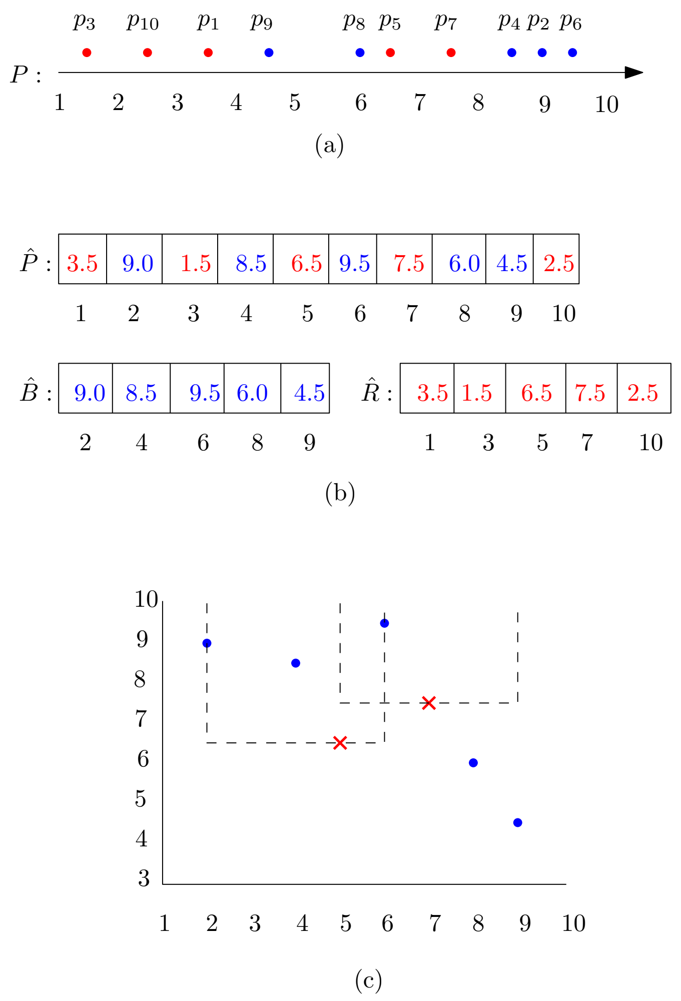

p_3 p_{10} p_1 p_9 p_8 p_5 p_7 p_4 p_2 p_6
$P:$
1 2 3 4 5 6 7 8 9 10
(a)
$\hat{P}:$ 3.5 9.0 1.5 8.5 6.5 9.5 7.5 6.0 4.5 2.5
1 2 3 4 5 6 7 8 9 10
$\hat{B}:$ 9.0 8.5 9.5 6.0 4.5
2 4 6 8 9
$\hat{R}:$ 3.5 1.5 6.5 7.5 2.5
1 3 5 7 10
(b)
(c)

Fig. 2. An example illustrating the 1-dimensional MLMP query. (a) Points $P = (p_1, \ldots, p_{10})$ are on the line. Each blue (respectively, red) point of P has an existential probability $\alpha_1 = 0.3$ (respectively, $\alpha_2 = 0.7$) associated with it. (b) Each array cell $\hat{P}[k]$ stores the x-coordinate value of the point p_k. $\hat{B}$ and $\hat{R}$ are respectively the arrays for all blue and red points. (c) Each blue point p_b is mapped to a 2-dimensional point $(b, \hat{B}[b])$. Given a query interval $[2, 6]$ we first find the red point p_5 ($\mathcal{R}_{max}$) with the maximum value 6.5 ($\mathcal{V}_{max}$). Now we query the range tree with $[2, 6] \times [6.5, +\infty)$ and count that there are three blue points p_2, p_4 and p_6 in the query range with values larger than 6.5. Since $0.7 \cdot (1 - 0.3)^3 = 0.24 < 0.3$, we report that a blue point p_6 (with value 9.5) to be the MLMP in $P_{2,6}$ with probability 0.3. Similarly, we can show that for a query interval $[5, 9]$, point p_7 is the $\mathcal{R}_{max}$ with value $\mathcal{V}_{max} = 7.5$. There is exactly on blue point p_6 in the query range $[5, 9] \times [7.5, +\infty)$. Since $0.7 \cdot (1 - 0.3) = 0.49 > 0.3$, the red point p_7 is the MLMP in $P_{5,9}$ with probability 0.7.

$q = [i, j]$ with $1 \leq i < j \leq n$, report the most likely maximal point in $P_{i,j}$.

Points are in d-dimension. Let $A[1 \cdots n]$ be an array such that $A[k] = p_k$ for all $k = 1, \ldots, n$. We assume that all n points have distinct coordinates in all dimensions. Let $D(p; i, j)$ be the number of points in $\mathcal{D}(p)$ in $P_{i,j}$, i.e.,

$$D(p; i, j) = |\mathcal{D}(p) \cap \{p_i, p_{i+1}, \cdots, p_j\}|.$$

By Lemma 1, we have to consider two cases for answering the most likely maximal point query. Either a blue point that is on the maximal layer of all points in $P_{i,j}$ can be the most likely maximal point in $P_{i,j}$ with probability α_1 or a red point in $R_{i,j}$ with minimum m blue points to its $\mathcal{D}$-region can be the most likely maximal point in $P_{i,j}$ with probability $\alpha_2(1 - \alpha_1)^m$. Thus we report the answer to query Q as $\max\{\alpha_1, \alpha_2(1 - \alpha_1)^m\}$.

Now we present solutions for the case where the most likely maximal point is a red point. Observe that if the answer to query Q is a red point with probability $\alpha_2(1-\alpha_1)^m \geq \alpha_1$ then $m \leq \lfloor \log(\alpha_2/\alpha_1)/\log(1/1-\alpha_1) \rfloor$. Therefore, for a fixed integer λ, with $0 \leq \lambda \leq n$, we ask query Q' as follows. *Given ℓ and r with $1 \leq \ell \leq r \leq n$, is there a red point p in $A[\ell, r]$ for which $D(p; \ell, r) \leq \lambda$?*

We first solve two simpler queries.

1. Q'_1*: For fixed indices ℓ and k with $1 \leq \ell \leq k \leq n$, given an integer r with $r \geq k$, is there a red point p in $A[\ell, \cdots, k]$ for which $D(p; \ell, r) \leq \lambda$?*
2. Q'_2*: For fixed indices k and r with $1 \leq k \leq r \leq n$, given an integer ℓ with $\ell \leq k$, is there a red point p in $A[k, \cdots, r]$ for which $D(p; \ell, r) \leq \lambda$?*

Solution for Q'_1: Observe that if p is a red point in $A[\ell \cdots k]$ for which $D(p; \ell, k) \geq \lambda + 1$, then p does not satisfy the query Q'_1. Therefore, let p be a red point in $A[\ell \cdots k]$ for which $D(p; \ell, k) \leq \lambda$. We define

$$r_p = \max\{r : r \leq n \text{ and } D(p; \ell, r) \leq \lambda\}$$

i.e., for a fixed ℓ, $[\ell, r_p]$ is the largest query interval for which point p satisfies the query. Note that, we assume here the maximum of an empty set is $\max\{\} = -\infty$. We also define the following.

$$R_{\ell k} = \max\{r_p : p \text{ is a red point in } A[\ell \cdots k] \text{ and } D(p; \ell, k) \leq \lambda\}$$

We answer the query as follows. Given an integer r with $r \geq k$, if $r \leq R_{\ell k}$ then return *yes*, otherwise return *no*. Since we only store the value of $R_{\ell k}$ for fixed indices ℓ and k, the total space required is $O(1)$. We answer the query in $O(1)$ time.

Solution for Q'_2: The solution is symmetric to the one discussed above for Q'_1. Let p be a red point in $A[k \cdots r]$ for which $D(p; k, r) \leq \lambda$. We define

$$\ell_p = \min\{\ell : \ell \geq 1 \text{ and } D(p; \ell, r) \leq \lambda\}$$

We assume that minimum of an empty set $\min\{\} = \infty$. We define

$$L_{kr} = \min\{\ell_p : p \text{ is a red point in } A[k \cdots r] \text{ and } D(p; k, r) \leq \lambda\}$$

We answer the query as follows. Given an integer ℓ with $\ell \leq k$, if $L_{kr} \leq \ell$ then return *yes*, otherwise return *no*. For the similar reasoning, both the space and the query bound is $O(1)$.

Answering Q': We now describe how we answer query Q' using results of queries Q'_1 and Q'_2. For all $\ell = 1, 2, \cdots, n$, we store the solutions for Q'_1 for intervals of width $j = 0, 1, \cdots, \lfloor \log(n - \ell + 1) \rfloor$. More formally, we store the following.

```
for $\ell = 1, 2, \cdots, n$:
    for $j = 0, 1, \cdots, \lfloor \log(n - \ell + 1) \rfloor$:
        store $R_{\ell, \ell+2^j-1}$
```

Similarly, we also store the following.

```
for $r = 1, 2, \cdots, n$:
    for $j = 0, 1, \cdots, \lfloor \log r \rfloor$:
        store $L_{r-2^j+1, r}$
```

For every point in $A[1 \cdots n]$ we store at most $O(\log n)$ many L and R values. Therefore the total space required is $O(n \log n)$. For a given query with integers ℓ and r, with $1 \leq \ell \leq r \leq n$, we answer the query as follows.

1. If $\ell = r$: if p_ℓ is a red point then we return *yes*, else we return *no*.
2. If $\ell < r$: let $h = \lfloor \log(r - \ell) \rfloor$. Observe that $\{\ell, \ell+1, \cdots, r\} = \{\ell, \cdots, \ell + 2^h - 1\} \cup \{r - 2^h + 1, \cdots, r\}$. If $r \leq R_{\ell, \ell+2^h-1}$ or $L_{r-2^h+1, r} \leq \ell$ then we return *yes*, else we return *no*.

Algorithm 1: Compute Log Values

```
    Input  : An array $L[1 \dots n]$.
1   Initialize $\ell = 0$ and $x = 1$.                  $//x = 2^\ell$.
2   while $x \le n$ do
3       for $i = x$ to $\min(2x - 1, n)$ do
4           Set $L[i] = \ell$.                          $//L[i] = \lfloor \log i \rfloor$
5       Set $x = 2x$.
6       Set $\ell = \ell + 1$.                          $//\ x = 2^\ell$
```

The query Q' can be answered in $O(1)$ time.

We also present an alternative method for computing $\lfloor \log(r - \ell) \rfloor$ in $O(1)$ time. We precompute the values of $\lfloor \log(r - \ell) \rfloor$ and store them in an array so that the corresponding value can be fetched when required. Let $L[1 \dots n]$ be an array that is initially empty. We initialize $\ell = 0$ and $x = 2^\ell$. See Algorithm 1 for details on how to compute $L[i]$ for $i = 1, \dots, n$.

We summarize the result for querying Q' in the following theorem.

Theorem 2. *Let α_1 and α_2 be constants with $0 < \alpha_1 < \alpha_2 < 1$, and let λ be a fixed integer with $0 \le \lambda \le n$. Suppose $A[1 \cdots n]$ is an array such that for k = 1 to n each $A[k]$ is either a blue point with existential probability α_1 or a red point with existential probability α_2. A can be preprocessed into a data structure of size $O(n \log n)$ such that given a query interval $[\ell, r]$ with $1 \le \ell \le r \le n$, one can report whether there exists a red point in $A[\ell \cdots r]$ with at most λ blue points in its $\mathcal{D}$-region in $O(1)$ time.*

Answering Q: Finally given a query interval $[i, j]$, with $1 \le i < j \le n$, we answer query Q as follows. Recall that m is a constant with value at most $\beta = \lfloor \log(\alpha_2/\alpha_1)/\log(1/(1 - \alpha_1)) \rfloor$. We build our data structures for solving query Q' for $\lambda = 0, 1, \cdots, \beta$. Given a query interval $[i, j]$, we query these data structures sequentially starting with $\lambda = 0$ and onwards. If any of these queries with $0 \le \lambda \le \lfloor \log(\alpha_2/\alpha_1)/\log(1/1 - \alpha_1) \rfloor$ returns *yes*, then we stop and report that a red point to be the most likely maximal point with probability $\alpha_2(1 - \alpha_1)^\lambda$. Otherwise we report that a blue point is the most likely maximal point with probability α_1. Thus we obtain the following result.

Theorem 3. *Let P be a sequence of n points in $\mathbb{R}^d$. Let each point of P be either colored blue or red. Let $B \subseteq P$ be the set of all blue points and let $R \subseteq P$ be the set of all red points. Let α_1 and α_2 be constants with $0 < \alpha_1 < \alpha_2 < 1$. Given an instance of MLMP$(B, \alpha_1; R, \alpha_2)$, P can be preprocessed into a data structure of size $O(n \log n)$, such that given a query interval $[i, j]$ with $1 \le i < j \le n$, one can report the most likely maximal point in $P_{i,j}$ in $O(1)$ time.*

2.2 c-colored MLMP

In this section, we generalize our algorithm so that it can answer the c-colored MLMP query defined as follows. Let $\alpha_1, \dots, \alpha_c$ be constants, with $0 < \alpha_1 < \alpha_2 < \dots < \alpha_c < 1$. Let $P = P_1 \cup P_2 \cup \dots \cup P_c$ be a set of n points in $\mathbb{R}^d$, for some fixed d. For $i = 1, 2, \dots, c$, let all points in P_i be colored by the ith color. Furthermore, for $i = 1, 2, \dots, c$, each point in P_i is associated with a probability α_i of existence.

We first describe the extension for $c = 3$. Let α_1, α_2 and α_3 be constants with $0 < \alpha_1 < \alpha_2 < \alpha_3 < 1$. Let $P = B \cup R \cup G$ be a subsequence of B blue, R red and G green points, where each blue point has a probability α_1 associated with it, each red point has a probability α_2 associated with it and each green point has a probability α_3 associated with it. Let Z be a random subset of P where each blue (respectively, red and green) point of P is added to Z with probability α_1 (respectively, α_2 and α_3) independently of other points. For a query window $[i, j]$, we define $P_{i,j} = \{p_i, \dots, p_j\}$, $B_{i,j} = B \cap P_{i,j}$, $R_{i,j} = R \cap P_{i,j}$ and $G_{i,j} = G \cap P_{i,j}$. Lemma 1 can be extended as follows.

Lemma 2.

1. *A blue point is the answer to our query with probability α_1 if it is on the maximal layer of all points in $P_{i,j}$.*

2. *A red point p' is the answer to our query with probability $\alpha_2(1-\alpha_1)^m$ if it is on the maximal layer of $R_{i,j}$ such that at most m blue points exist in $B_{i,j} \cap \mathcal{D}(p')$, with $0 \le m \le \lfloor \log(\alpha_2/\alpha_1)/\log(1/1-\alpha_1) \rfloor$, and no green points exist in in $G_{i,j} \cap \mathcal{D}(p')$.*
3. *A green point p'' is the answer to our query with probability $\alpha_3(1-\alpha_1)^k(1-\alpha_2)^\ell$ if it is on the maximal layer of $G_{i,j}$ such that at most k blue points exist in $B_{i,j} \cap \mathcal{D}(p'')$, with $0 \le k \le \lfloor \log(\alpha_3/\alpha_1)/\log(1/(1-\alpha_1)) \rfloor$, and at most ℓ red points exist in $R_{i,j} \cap \mathcal{D}(p'')$, with $0 \le \ell \le \lfloor \log(\alpha_3/\alpha_2)/\log(1/(1-\alpha_2)) \rfloor$.*

The proof is similar to that of Lemma 1 and hence omitted.

Answering queries: The final answer to our query in this setting can be obtained by computing the $\max\{\alpha_1, \alpha_2(1-\alpha_1)^m, \alpha_3(1-\alpha_1)^k(1-\alpha_2)^\ell\}$. Note that the data structures we presented in the previous section can answer for the first two cases of Lemma 2.

For case 3, similar to the analysis presented earlier, we can show that if $\ell = 0$ then $k \le \lfloor \log(\alpha_3/\alpha_1)/\log(1/(1-\alpha_1)) \rfloor$ and if $k = 0$ then $\ell \le \lfloor \log(\alpha_3/\alpha_2)/\log(1/(1-\alpha_2)) \rfloor$. Since $\alpha_1 < \alpha_2$, the maximum possible value of k is larger than the maximum possible value of ℓ. So, a green point p_g can be the most likely maximal point if there are at most $\lambda' = \lfloor \log(\alpha_3/\alpha_1)/\log(1/(1-\alpha_1)) \rfloor$ points in $\mathcal{D}(p_g)$. So we ask this query: '*Is there a green point p_g with at most λ' points in $\mathcal{D}(p_g)$ in $P_{i,j}$?*'. The data structure for Q' (see Section 2.1) can answer this query. Since we assume that all probability values α_1, α_2 and α_3 are constants, we obtain constant number of pairs (k, ℓ), with $k + \ell \le \lambda'$, to query for case 3. Now we query the data structure for Q' with $\lambda = 0, 1, \ldots, \lambda'$. As mentioned above, the final answer for the 3-colored MLMP point can be answered by computing the maximum of all probabilities obtained for all the three cases in constant time. Total space required is $O(n \log n)$ and query can be answered in $O(1)$ time. We can generalize this to a constant number of colors. The result is summarized as follows.

Theorem 4. *For a fixed integer $c > 0$, let $\alpha_1, \ldots, \alpha_c$ be constant real numbers with $0 < \alpha_1 < \ldots < \alpha_c < 1$. Let $P = P_1 \cup P_2 \cup \ldots \cup P_c$ be a sequence of n stochastic points such that for $r = 1, \ldots, c$, each r-colored point belongs to P_r and has a probability α_r associated with it. P can be preprocessed into a data structure so that given a query window $[i, j]$, with $1 \le i \le j \le n$, it can report the c-colored MLMP in $P_{i,j}$ in $O(1)$ time using $O(n \log n)$ space.*

2.3 Report Pr(p_t is on the Maximal Layer in $P_{i,j}$)

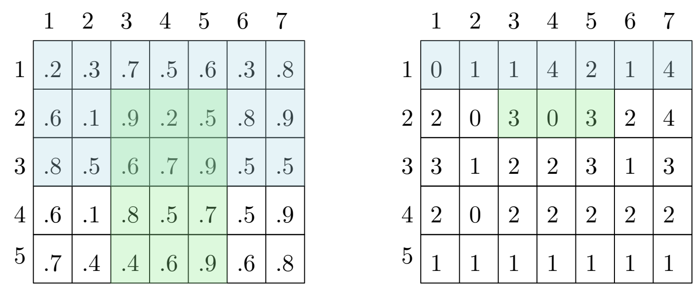
Let $P = (p_1, p_2, \ldots, p_n)$ be a sequence of n points in $\mathbb{R}^2$, where for $k = 1$ to n each point $p_k \in P$ has a probability $\gamma_k \in (0, 1]$ of existence associated with it. In this section we solve the following problem: '*Given three integers i, j and t, with $1 \le i \le t \le j \le n$, report the probability that p_t is on the maximal layer in $P_{i,j}$*).

For k=1 to n, let each point p_k be represented as (p_{k,x_1}, p_{k,x_2}), where x_1 and x_2 are the coordinates of p_k. We build a 3-dimensional range tree T on the points in P, where the first level is on the sequence of the points $1 \ldots n$ from left to right. At each of the canonical nodes of this tree we build two more levels based on, respectively, the x_1 and the x_2 coordinates of the points that are stored in this node. Now at each canonical node w of the third level of T, we store the probability that none of the points stored in the subtree T_w exist. We denote this value by $\rho(w) = \prod_{p_\ell \in T_w}(1-\gamma_\ell)$. The time and space required to build the range tree T is $O(n \log^2 n)$ [11].

Answering a query. Given i, j and t, with $1 \le i \le t \le j \le n$, we want to report the probability of p_t to be on the maximal layer in $P_{i,j}$. This requires us to find all points p_ℓ, with $i \le \ell \le j$, such that $p_\ell \in \mathcal{D}(p_t)$. Recall that $\mathcal{D}(p_t)$ is the set of points in $P_{i,j}$ that dominates p_t. We obtain the following lemma.

Lemma 3. *Suppose $1 \le i \le t \le j \le n$. The point p_t is on the maximal layer in $P_{i,j}$ if p_t exists and none of the points $p_\ell \in \mathcal{D}(p_t)$ exist, where $i \le \ell \le j$.*

Therefore, the probability that p_t belongs to the maximal layer in $P_{i,j}$ is the value $\gamma_t \cdot \prod_{p_\ell \in \mathcal{D}(p_t)}(1-\gamma_\ell)$. To find all points in the $\mathcal{D}$-region of a given point $p_t = (p_{t,x1}, p_{t,x2})$, we query T with $[i, j] \times [p_{t,x1}, +\infty) \times [p_{t,x2}, +\infty)$. As the result, we obtain $O(\log^2 n)$ number of ρ values from $O(\log^2 n)$ canonical nodes that



	1	2	3	4	5	6	7
1	.2	.3	.7	.5	.6	.3	.8
2	.6	.1	.9	.2	.5	.8	.9
3	.8	.5	.6	.7	.9	.5	.5
4	.6	.1	.8	.5	.7	.5	.9
5	.7	.4	.4	.6	.9	.6	.8

(a)

	1	2	3	4	5	6	7
1	0	1	1	4	2	1	4
2	2	0	3	0	3	2	4
3	3	1	2	2	3	1	3
4	2	0	2	2	2	2	2
5	1	1	1	1	1	1	1

(b)

Fig. 3. (a) $\mathcal{S}$ is a matrix of size 5×7 and $\tau = 0.3$. (b) The data structure $\mathcal{S}'$. An example illustrating the query with indices 1, 7, 1 and 3 (corresponding elements are blue in $\mathcal{S}$ and $\mathcal{S}'$). The range maximum data structure for row 1 of $\mathcal{S}'$ returns the maximum value 4, which is greater than $3 - 1 + 1 = 3$, so there exists some element (4 & 7) in $[1, 7]$ for which the answer is positive. Whereas, the query with indices 3, 5, 2 and 5 returns a negative result (highlighted with green). In row 2 of $\mathcal{S}'$ the maximum value $\mathcal{M}(x_{3,5}) = 3$, which is less than $5 - 2 + 1 = 4$.

cover all points in $P_{i,j}$. We denote these ρ-values by $\rho_1, \ldots, \rho_{\log^2 n}$. We report $\gamma_t \cdot (\prod_{m=1}^{\log^2 n} \rho_m)$ as the probability that point p_t is on the maximal layer in $P_{i,j}$. Therefore, it takes $O(\log^2 n)$ time to answer a query. We can generalize this data structure for points in d-dimension, for any fixed $d \geq 2$.

Theorem 5. *A sequence $P = (p_1, p_2, \ldots, p_n)$ of n points in $\mathbb{R}^d$ is given, with a fixed $d \geq 2$, where for $k = 1$ to n each point $p_k \in P$ has a probability $\gamma_k \in (0, 1]$ of existence associated with it. P can be preprocessed into a data structure of size $O(n \log^d n)$ in $O(n \log^d n)$ time such that given a query interval $[i, j]$ and an integer t with $i \leq t \leq j$, the probability that p_t to be on the maximal layer of $P_{i,j}$ can be reported in $O(\log^d n)$ time.*

3 Most Likely Common Elements

Let $\mathcal{S}$ be an $m \times n$ matrix in which each entry $\mathcal{S}[p, i]$ is a real number $\alpha_{pi} \in [0, 1]$. We want to preprocess $\mathcal{S}$ into a data structure such that for query values p, q, i and j, with $1 \leq p \leq q \leq m$ and $1 \leq i \leq j \leq n$, it can decide if there exists some integer k (the most likely common element) for which $i \leq k \leq j$ and $\prod_{r=p}^{q} \alpha_{rk} \geq \tau$. We also present a data structure that can report all common elements in the query window.

Preprocessing: Let $\mathcal{S}'$ be the matrix of size $m \times n$. For $p = 1, \ldots, m$ and for $i = 1, \ldots, n$, we store $S'[p, i] = r - p + 1$, where $r = \max\{h : \prod_{\ell=p}^{h} \alpha_{\ell i} \geq \tau\}$, i.e., the length of the maximum subsequence of rows such that the product of the entries $\mathcal{S}[\ell, i]$, with $p \leq \ell \leq h$, is greater than τ. Fig. 3 illustrates an example. For $i = 1, \ldots, n$, we scan each column i and find the values of $S'[p, i]$. See Algorithm 2 for the preprocessing step. For each row $p = 1, \ldots, m$, we build the range maximum data structure RM_p using linear space [10], so that for any query interval $[i, j]$, with $1 \leq i < j \leq n$, it can report the maximum value stored in $\mathcal{S}'[p, i], \ldots, \mathcal{S}'[p, j]$ in $O(1)$ time. The total preprocessing requires $O(mn)$ time.

To report all common elements that satisfy the condition, we build the data structure described as follows. For $p = 1, \ldots, m$ and for $i = 1, \ldots, n$, we map the values $\mathcal{S}'[p, i]$ to a point $(i, \mathcal{S}'[p, i]) \in \mathbb{R}^2$. Let X_p be the set of points obtained in this way for the row p of $\mathcal{S}'$. For $p = 1, \ldots, m$ we create a priority search tree (PST) (see [27]) T_p for the points in X_p. Hence the first level of T_p is based on the values of i. For each canonical node of this tree, we build a second level on the sorted values of $\mathcal{S}'[p, i]$. For each row p, this can be done in $O(n \log n)$ time and using $O(n)$ space. So the total space requirement becomes $O(mn)$.

Answering Queries: Given query indices p, q, i and j, we query the range maximum data structure RM_p with interval $[i, j]$. Suppose we obtain the maximum value and denote it by $\mathcal{M}_{p,i,j}$. If $\mathcal{M}_{p,i,j} \geq q - p + 1$ then there exists at least an element k in $[i, j]$ such that $\prod_{r=p}^{q} \alpha_{rk} \geq \tau$. Otherwise the answer is negative. To report the common elements for the same query indices i, j, p and q, we query T_p using a query range $[i, j] \times [q - p + 1, +\infty)$. The output of this query is the set of all those elements k in $\mathcal{S}[p, i], \ldots, \mathcal{S}[q, j]$ such that $\prod_{r=p}^{q} \alpha_{rk} \geq \tau$. Each query can be answered in $O(\log n + w)$ time, where w is the size of the output. Therefore, we obtain the following.

Algorithm 2: Compute $\mathcal{S}'$

```
Input : A matrix $\mathcal{S}$ of size $m \times n$ and a fixed value $\tau \in [0,1]$
1  for $p$ = 1 to $m$ do
2      for $i$ = 1 to $n$ do
3          Initialize $\mathcal{S}'[p,i]$ with value $-1$.
4  for $i = 1$ to $n$ do
5      Set $p \leftarrow 1$ and $top \leftarrow 0$.
6      for $\ell = 1$ to $m$ do
7          case $\ell = 1$ : do
8              if $\mathcal{S}[\ell,i] < \tau$ then
9                  Set $\mathcal{S}'[\ell,i] = 0$.
10             else
11                 Set $p = p * \mathcal{S}[\ell,i]$.
12                 Set $top = 1$.
13             Set $\ell++$.
14         case $\ell \geq 2$ : do
15             if $p * \mathcal{S}[\ell,i] \geq \tau$ then
16                 Set $p \leftarrow p * \mathcal{S}[\ell,i]$ and $\ell++$.
17             else
18                 Set $\mathcal{S}'[top,i] = \ell - top$.
19                 Set $p \leftarrow (p/\mathcal{S}[top,i])$.
20                 Set $top++$.
```

Theorem 6. *Let $\mathcal{S}$ be a matrix of size $m \times n$, where m and n are positive integers. Suppose for $p = 1,\dots,m$ and $i = 1,\dots,n$, each element of $\mathcal{S}[p,i]$ is a real number $\alpha_{pi} \in (0,1)$. Let τ be a fixed threshold in $[0,1]$. $\mathcal{S}$ can be preprocessed into data structures of size $O(mn)$ so that given query indices i, j, p and q, with $1 \leq i < j \leq n$ and $1 \leq p < q \leq m$, it can answer the MLCE queries in $O(1)$ time and report these common elements in $O(\log n + w)$ time, where w is the size of the output.*

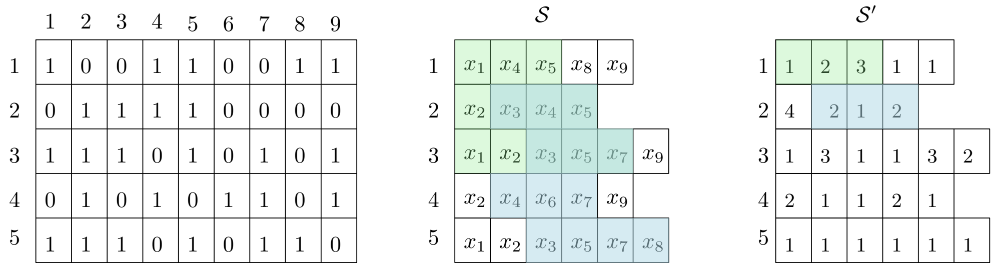
3.1 A Special Case

We consider a special case of MLCE problem where $\mathcal{S}$ is an $m \times n$ matrix and each entry $\mathcal{S}[p,i]$ is $\alpha_{pi} \in \{0,1\}$ and $\tau = 1$. We want to preprocess $\mathcal{S}$ into a data structure such that for query values p, q, i and j, with $1 \leq p \leq q \leq m$ and $1 \leq i \leq j \leq n$, it can decide if there exists some integer k for which $i \leq k \leq j$ and $\prod_{r=p}^{q} \alpha_{rk} = \tau$.

We remove all entries of $\mathcal{S}$ that contain zero. Let N be the total number of elements in $\mathcal{S}$ that have value 1, that is $\sum_{p=1}^{m}\sum_{i=1}^{n} \alpha_{p,i} = N$. Fig. 4 illustrates an example. After removal of all zeros the remaining elements in $\mathcal{S}$ are represented by x_k (see Fig. 4(b)), where k represents its original column index.

Preprocessing step: For each $p = 1$ to m and $x_k \in \mathcal{S}[p]$, we define $\beta_p(k) = r - p$, where r is the smallest integer such that $r \geq p+1$ and $x_k \notin \mathcal{S}[r]$. Let $\mathcal{S}'$ be a matrix of the same size as $\mathcal{S}$. For each row p and for each $x_k \in \mathcal{S}[p]$, let $\mathcal{S}'[p]$ store the value of $\beta_p(k)$. As before, for each row p of $\mathcal{S}'[p]$ we build the range maximum data structure RM_p so that for any query interval $[i,j]$, with $1 \leq i < j \leq |\mathcal{S}'[p]|$, it can report the maximum value stored in $\mathcal{S}'[p,i],\dots,\mathcal{S}'[p,j]$ in $O(1)$ time using linear space [10]. See Fig. 4(c).

Algorithm for computing $\beta_p(k)$: For each element $k \in \{1,2,\dots,n\}$, we maintain an array X_k that contains the sorted indices (in the non-decreasing order) of the rows p such that $x_k \in \mathcal{S}[p]$. For $k = 1$ to n, we scan each element of the array X_k sequentially to find the maximum subsequence of sets that contain element x_k. Then for each of these sets we compute $\beta_p(k)$ according to Algorithm 3. Initialization of m empty lists $\mathcal{S}'[p]$, with $1 \leq p \leq m$, (lines 1 and 2) takes $O(N)$ time. From line 3 to 13, every element of lists $X_1 \dots X_n$ is scanned exactly once to find the maximum subsequence of sets as discussed above. Therefore the total time required for this algorithm is $O(N)$. The total space required for $\mathcal{S}'$ is also



	1	2	3	4	5	6	7	8	9
1	1	0	0	1	1	0	0	1	1
2	0	1	1	1	1	0	0	0	0
3	1	1	1	0	1	0	1	0	1
4	0	1	0	1	0	1	1	0	1
5	1	1	1	0	1	0	1	1	0

(a)

$\mathcal{S}$						
1	x_1	x_4	x_5	x_8	x_9	
2	x_2	x_3	x_4	x_5		
3	x_1	x_2	x_3	x_5	x_7	x_9
4	x_2	x_4	x_6	x_7	x_9	
5	x_1	x_2	x_3	x_5	x_7	x_8

(b)

$\mathcal{S}'$						
1	1	2	3	1	1	
2	4	2	1	2		
3	1	3	1	1	3	2
4	2	1	1	2	1	
5	1	1	1	1	1	1

(c)

Fig. 4. (a) A matrix $\mathcal{S}$ of size 5×9. (b) $\mathcal{S}$ after removing all zero's. (c) The data structure $\mathcal{S}'$. Suppose, we query $\mathcal{S}$ with query indices 1, 7, 1 and 3. All corresponding elements are highlighted with green. The query range $[1, 7]$ in $\mathcal{S}'$ is also highlighted with green. We query the range maximum data structure in $\mathcal{S}'[1]$ for the maximum element within $[1, 7]$. Since $\mathcal{M}(x_{1,7}) = 3$ is equal to $3 - 1 + 1 = 3$, we answer: Yes, there exists an element x_5 such that $\prod_{r=1}^{7} \alpha_{r,5} = 1$. Similarly the answer to the query with indices 3, 8, 2 and 5 (corresponding elements are similarly highlighted with blue in $\mathcal{S}$ and in $\mathcal{S}'$) is negative. The maximum value $\mathcal{M}(x_{3,8})$ in $\mathcal{S}'[2]$ is 2 that is less than $5 - 2 + 1 = 4$, so there does not exist any element in $[3, 8]$ that satisfies the condition.

Algorithm 3: Compute $\beta_p(k)$

```
Input : A matrix $\mathcal{S}$ of size $N$
1  for $\ell = 1$ to $m$ do
2      Initialize empty rows $\mathcal{S}'[\ell]$ of size $|\mathcal{S}[\ell]|$
3  for $k = 1$ to $n$ do
4      for $p = 1$ to $|X_k|$ do
5          Let $z \leftarrow X_k[p]$
6          Let $top \leftarrow X_k[p]$
7          while $X_k[p+1] == z + 1$ do
8              Increase $p{+}{+}$
9              Increase $z{+}{+}$
10         for $\ell = top$ to $z$ do
11             Insert $z - \ell + 1$ to the next empty cell of row $\mathcal{S}'[\ell]$
12             Increase $\ell{+}{+}$
13         Increase $p{+}{+}$
```

$\sum_{k=1}^{n} |X_k| = N$. Thus the total space requirement is linear.

Answering queries: The query process is similar to the previous section. Since $\mathcal{S}'$ is not a matrix in this case, we use binary searches before querying for the range maximum value. For query indices p, q, i and j, we query $\mathcal{S}'[p]$ using the interval $[i, j]$ to find the range $x_{i'j'} = x_{i'}, \ldots, x_{j'}$ with the smallest index $i' \geq i$ and the largest index $j' \leq j$. We query the range maximum data structure RM_p for the maximum value in the range $x_{i'j'}$ and follow the same process. The query time is dominated by the binary searches that require $O(\log N)$ time.

Theorem 7. *Let $\mathcal{S}$ be a matrix of size $m \times n$, where m and n are positive integers. Suppose for $p = 1, \ldots, m$ and $i = 1, \ldots, n$, each element of $\mathcal{S}[p, i]$ is a number $\alpha_{pi} \in \{0, 1\}$ and $\tau = 1$. Let N be the total number of elements in $\mathcal{S}$ that have value 1. $\mathcal{S}$ can be preprocessed into data structures of size $O(N)$ so that given query indices i, j, p and q, with $1 \leq i < j \leq n$ and $1 \leq p < q \leq m$, it can answer the MLCE queries in $O(\log N)$ time and report these common elements in $O(\log N + w)$, where w is the size of the output.*

4 Conclusion

We present data structures to answer two types of window queries for stochastic input sequences. The first problem is a c-colored most likely maximal point problem (MLMP) where given an instance of

MLMP$(P_1, \alpha_1; P_2, \alpha_2; \ldots; P_c, \alpha_c)$, with $P = P_1 \cup \ldots \cup P_c$, and a pair of query indices our data structures can report the most likely maximal point within a query interval in P in $O(1)$ time using $O(n \log n)$ space, where n is the number of points in the input sequence. Secondly, we solve the most likely common element (MLCE) problem, where a matrix of size $m \times n$, with all entries being real numbers in $[0, 1]$ and a fixed real number τ in $[0, 1]$ are given. For query indices p, q, i and j, our data structure can decide if there exists some integer k for which $i \leq k \leq j$ and $\prod_{r=p}^{q} \alpha_{rk} \geq \tau$ in $O(1)$ time using $O(mn)$ space. The reporting version takes $O(\log n + w)$ time, where w is the size of the output.

The following are natural problems that require further explorations.

Problem 1. Given an instance of $(B, \alpha_1; R, \alpha_2)$, construct a data structure that can report the most likely maximal point in a given query interval in logarithmic query time using linear space.

Problem 2.
Given a sequence $P = (p_1, \ldots, p_n)$ of n stochastic points in $\mathbb{R}^d$, with $d \geq 1$, such that for $k = 1 \ldots n$, each point p_k exists with probability $\gamma_k \in [0, 1]$, report the most likely maximal point in a given query interval.

Problem 3.
Given a sequence of random subsets $S_1, S_1, \ldots, S_m$ of $\mathcal{U} = \{1, \ldots, n\}$ and query indices p, q, i and j, with $1 \leq p \leq q \leq m$ and $1 \leq i \leq j \leq n$, find the element $k \in \{i, \ldots, j\}$ that is most likely to be in $S_p \cap S_{p+1} \cap \ldots, \cap S_q$.

References

1. Abdullah, A., Daruki, S., Phillips, J.M.: Range counting coresets for uncertain data. In: Symposuim on Computational Geometry 2013, SoCG '13, Rio de Janeiro, Brazil, June 17-20, 2013. pp. 223–232. ACM (2013), https://doi.org/10.1145/2462356.2462388
2. Afshani, P., Agarwal, P.K., Arge, L., Larsen, K.G., Phillips, J.M.: (approximate) uncertain skylines. Theory Comput. Syst. 52(3), 342–366 (2013), https://doi.org/10.1007/s00224-012-9382-7
3. Agarwal, P.K., Cheng, S., Tao, Y., Yi, K.: Indexing uncertain data. In: Proceedings of the Twenty-Eigth ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems, PODS 2009, June 19 - July 1, 2009, Providence, Rhode Island, USA. pp. 137–146 (2009), https://doi.org/10.1145/1559795.1559816
4. Agarwal, P.K., Har-Peled, S., Suri, S., Yildiz, H., Zhang, W.: Convex hulls under uncertainty. Algorithmica 79(2), 340–367 (2017), https://doi.org/10.1007/s00453-016-0195-y
5. Agarwal, P.K., Kumar, N., Sintos, S., Suri, S.: Range-max queries on uncertain data. J. Comput. Syst. Sci. 94, 118–134 (2018), https://doi.org/10.1016/j.jcss.2017.09.006
6. Aggarwal, C.C. (ed.): Managing and Mining Sensor Data. Springer (2013), https://doi.org/10.1007/978-1-4614-6309-2
7. Agrawal, A., Li, Y., Xue, J., Janardan, R.: The most-likely skyline problem for stochastic points. In: Proceedings of the 29th Canadian Conference on Computational Geometry, CCCG 2017, July 26-28, 2017, Carleton University, Ottawa, Ontario, Canada. pp. 78–83 (2017)
8. Bannister, M.J., Devanny, W.E., Goodrich, M.T., Simons, J.A., Trott, L.: Windows into geometric events: Data structures for time-windowed querying of temporal point sets. In: Proceedings of the 26th Canadian Conference on Computational Geometry, CCCG 2014, Halifax, Nova Scotia, Canada, 2014 (2014), http://www.cccg.ca/proceedings/2014/papers/paper02.pdf
9. Bannister, M.J., DuBois, C., Eppstein, D., Smyth, P.: Windows into relational events: Data structures for contiguous subsequences of edges. In: Proceedings of the Twenty-Fourth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2013, New Orleans, Louisiana, USA, January 6-8, 2013. pp. 856–864 (2013), https://doi.org/10.1137/1.9781611973105.61
10. Bender, M.A., Farach-Colton, M.: The LCA problem revisited. In: LATIN 2000: Theoretical Informatics, 4th Latin American Symposium, Punta del Este, Uruguay, April 10-14, 2000, Proceedings. pp. 88–94 (2000), https://doi.org/10.1007/10719839_9
11. de Berg, M., Cheong, O., van Kreveld, M.J., Overmars, M.H.: Computational geometry: algorithms and applications, 3rd Edition. Springer (2008), http://www.worldcat.org/oclc/227584184
12. Bokal, D., Cabello, S., Eppstein, D.: Finding all maximal subsequences with hereditary properties. In: 31st International Symposium on Computational Geometry, SoCG 2015, June 22-25, 2015, Eindhoven, The Netherlands. LIPIcs, vol. 34, pp. 240–254. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik (2015), https://doi.org/10.4230/LIPIcs.SOCG.2015.240

13. Chan, T.M., Pratt, S.: Two approaches to building time-windowed geometric data structures. In: 32nd International Symposium on Computational Geometry, SoCG 2016, June 14-18, 2016, Boston, MA, USA. LIPIcs, vol. 51, pp. 28:1–28:15. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik (2016), https://doi.org/10.4230/LIPIcs.SoCG.2016.28
14. Chanchary, F., Maheshwari, A.: Counting subgraphs in relational event graphs. In: WALCOM: Algorithms and Computation - 10th International Workshop, WALCOM 2016, Kathmandu, Nepal, March 29-31, 2016, Proceedings. Lecture Notes in Computer Science, vol. 9627, pp. 194–206. Springer (2016), https://doi.org/10.1007/978-3-319-30139-6_16
15. Chanchary, F., Maheshwari, A.: Time windowed data structures for graphs. Journal of Graph Algorithms and Applications 23(2), 191–226 (2019)
16. Chanchary, F., Maheshwari, A., Smid, M.H.M.: Querying relational event graphs using colored range searching data structures. In: Algorithms and Discrete Applied Mathematics - Third International Conference, CALDAM 2017, Sancoale, Goa, India, February 16-18, 2017, Proceedings. Lecture Notes in Computer Science, vol. 10156, pp. 83–95. Springer (2017), https://doi.org/10.1007/978-3-319-53007-9_8
17. Chanchary, F., Maheshwari, A., Smid, M.H.M.: Window queries for problems on intersecting objects and maximal points. In: Algorithms and Discrete Applied Mathematics - 4th International Conference, CALDAM 2018, Guwahati, India, February 15-17, 2018, Proceedings. Lecture Notes in Computer Science, vol. 10743, pp. 199–213. Springer (2018), https://doi.org/10.1007/978-3-319-74180-2_17
18. De Caluwe, R.: Fuzzy and uncertain object-oriented databases: concepts and models, vol. 13. World Scientific (1997)
19. Goodchild, M.F.: Uncertainty in remote sensing and GIS. edited by giles m. foody and peter m. atkinson (chichester, UK: john wiley, 2002). International Journal of Geographical Information Science 18, 103–105 (2004), https://doi.org/10.1080/13658810310001620861
20. Held, M., Mitchell, J.S.B.: Triangulating input-constrained planar point sets. Inf. Process. Lett. 109(1), 54–56 (2008), https://doi.org/10.1016/j.ipl.2008.09.016
21. Jørgensen, A., Löffler, M., Phillips, J.M.: Geometric computations on indecisive and uncertain points. CoRR abs/1205.0273 (2012), http://arxiv.org/abs/1205.0273
22. Kamousi, P., Chan, T.M., Suri, S.: Stochastic minimum spanning trees in euclidean spaces. In: Proceedings of the 27th ACM Symposium on Computational Geometry, Paris, France, June 13-15, 2011. pp. 65–74. ACM (2011), https://doi.org/10.1145/1998196.1998206
23. Kamousi, P., Chan, T.M., Suri, S.: Closest pair and the post office problem for stochastic points. Comput. Geom. 47(2), 214–223 (2014), https://doi.org/10.1016/j.comgeo.2012.10.010
24. Li, J., Wang, H.: Range queries on uncertain data. Theor. Comput. Sci. 609, 32–48 (2016), https://doi.org/10.1016/j.tcs.2015.09.005
25. Löffler, M., Snoeyink, J.: Delaunay triangulation of imprecise points in linear time after preprocessing. Comput. Geom. 43(3), 234–242 (2010), https://doi.org/10.1016/j.comgeo.2008.12.007
26. MacKeown, P.K.: Stochastic Simulation in Physics. Springer-Verlag New York, Inc. (2001)
27. McCreight, E.M.: Priority search trees. SIAM J. Comput. 14(2), 257–276 (1985), https://doi.org/10.1137/0214021
28. Mena, R.H.: Book review - J.B. kadane, principles of uncertainty, texts in statistical science series, chapman & hall/crc, 2011. pp. xxviii+475, ISBN 978-1-4398-6161-5. J. Classification 31, 270–271 (2014), https://doi.org/10.1007/s00357-014-9158-7
29. Pei, J., Jiang, B., Lin, X., Yuan, Y.: Probabilistic skylines on uncertain data. In: Proceedings of the 33rd International Conference on Very Large Data Bases, University of Vienna, Austria, September 23-27, 2007. pp. 15–26. ACM (2007), http://www.vldb.org/conf/2007/papers/research/p15-pei.pdf
30. Suri, S., Verbeek, K., Yildiz, H.: On the most likely convex hull of uncertain points. In: Algorithms - ESA 2013 - 21st Annual European Symposium, Sophia Antipolis, France, September 2-4, 2013. Proceedings. pp. 791–802. Springer (2013), https://doi.org/10.1007/978-3-642-40450-4_67
31. Vazirgiannis, M., Halkidi, M., Gunopulos, D.: Uncertainty Handling and Quality Assessment in Data Mining. Advanced Information and Knowledge Processing, Springer (2003), https://doi.org/10.1007/978-1-4471-0031-7
32. Xue, J., Li, Y.: Colored stochastic dominance problems. CoRR abs/1612.06954 (2016), http://arxiv.org/abs/1612.06954