Chapter 3

Nearest-Neighbor Search with preprocessing

In this chapter we present two strategies to speed up the orthogonal range searching procedure SCAN of the CUBE METHOD introduced in Chapter 2.

In Section 3.1 we make use of a simple preprocessed data structure of linear size, where the coordinates of the data points of P are stored sorted in each of the dimensions $\{1, \ldots, d\}$. The query algorithm has an expected runtime of $O\left(n \ln(\frac{d}{\ln n} + 1) + n + d \ln d\right)$, assuming that the data set is drawn randomly under uniform distribution. A summary of the results presented in this section has appeared in [39].

The query algorithm that we present in Section 3.2 is based on a preprocessed partition of the data set, such that each subset allows an efficient orthogonal range searching. The partition consists of subsequences that are monotone with respect to some of the dimensions. The query algorithm works well for dimensions $d \leq \left(\frac{\ln n}{\ln \ln n}\right)^2$, when it has an expected running time of $O\left(\sqrt{d} \cdot n^{1-\frac{1}{\sqrt{d}}} \cdot \ln n\right)$. For higher dimensions $d \geq \left(\frac{\ln n}{\ln \ln n}\right)^2$ its expected runtime is $O\left(d \cdot n^{1-\frac{1}{\sqrt{d}}} \cdot \ln \ln n\right)$.

3.1 Speeding up the CUBE METHOD by rejecting points

The contribution of this section is the improvement of the CUBE METHOD for high dimensions $d > \ln n$. The method requires $O(nd)$ storage and $O(nd \ln n)$ preprocessing time. Its average runtime assuming that the set P is drawn randomly under uniform distribution is $O\left(n \ln(\frac{d}{\ln n} + 1) + n + d \ln d\right)$, thereby improving the brute-force method by a factor of $O(\frac{\ln(1+d/ln\ n)}{d})$.

Drawback of the searching procedure SCAN

Depending on whether $C_{q,\alpha} \subseteq [0,1]^d$ or $C_{q,\alpha} \not\subseteq [0,1]^d$ either the side length α or the geometric mean of the side lengths of the box $C_{q,\alpha} \cap [0,1]^d$ equals $\sqrt[d]{\varphi/n}$, where φ is the expected number of points contained in the cube $C_{q,\alpha}$. The value $\sqrt[d]{\varphi/n}$ is very large (larger than 0.9) for the case of high dimension d, in the range of some hundreds, and reasonable values n, in the range of several

thousands. Figure 3.1 considers the case $C_{q,\alpha} \subseteq [0,1]^d$ with $\varphi = 1$ and shows values of the side length $\alpha = \sqrt[d]{1/n}$ for high dimension d.

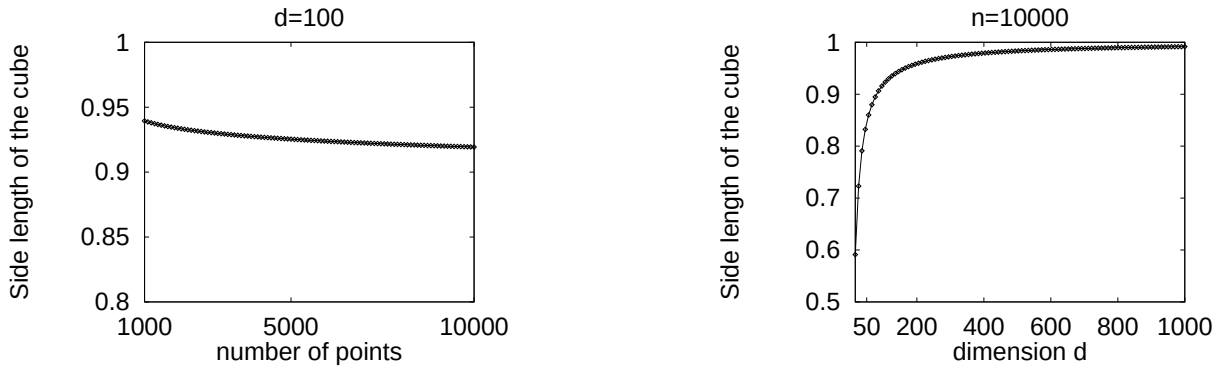


Figure 3.1: Side length $\alpha = \sqrt[d]{1/n}$ is large for high dimension d

Large side lengths imply a large number of points that *do not* fail the test in step labeled by (TESTS) of procedure SCAN. This means that in each of the dimensions the expected number of points whose coordinates are *outside* of the appropriate interval is a small number. This observation leads to the idea to look at the points which are not in the cube with respect to each dimension independently and reject these points in order to determine the points which are in the cube[1].

3.1.1 Searching the cube by rejecting

During the *preprocessing* the coordinates of the points are sorted in each of the dimensions $j \in \{1, \ldots, d\}$. This takes $O(nd \ln n)$ time. For a dimension $j \in \{1, \ldots, d\}$ an array T_j with n items is stored. Item $T_j[i]$ contains the i-th largest coordinate $T_j[i].coord$ in dimension j and the index $T_j[i].index = l$ of the point p^l corresponding to that coordinate. The data structure consists of the indexed data set P and the arrays T_j, $j = 1, \ldots, d$.

The *query algorithm* is the CUBE-METHOD with an orthogonal range searching procedure which works as follows: for each dimension $j \in \{1, \ldots, d\}$ those points are marked as *rejected* in a bit array whose coordinates in dimension j are outside of the interval $[q_j - \frac{\alpha}{2}, q_j + \frac{\alpha}{2}]$. At the end the bit array is scanned and all non-rejected points are returned. This orthogonal searching procedure is called REJECT.

The points outside the interval $I_j := [q_j - \frac{\alpha}{2}, q_j + \frac{\alpha}{2}]$ could be determined by binary search for each dimension j. We do not analyze this variant since for each coordinate outside I_j we perform a marking operation anyway.

Figure 3.2 illustrates the *rejecting* process and the following gives a schematic description of procedure REJECT. $\mathcal{D}$ is here the set of dimensions $\{1, 2, \ldots, d\}$.

REJECT$(\boldsymbol{\alpha}, \boldsymbol{q}, \boldsymbol{T}, \boldsymbol{P}, \boldsymbol{\mathcal{D}})$

[1]The author would like to thank Piotr Indyk for this hint.

forall $j \in \mathcal{D}$
  $i := 1$;
  while ($i \leq n$ and $T_j[i].coord < q_j - \frac{\alpha}{2}$) (TESTS)
  do mark point with index $T_j[i].index$ as rejected; $i := i + 1$;

  $i := n$;
  while ($i \geq 1$ and $T_j[i].coord > q_j + \frac{\alpha}{2}$) (TESTS)
  do mark point with index $T_j[i].index$ as rejected; $i := i - 1$;

$P_\alpha := \emptyset$;
for $i := 1$ to n do
  if (p^i not rejected) then $P_\alpha := P_\alpha \cup \{p^i\}$; (BITARRAY)
return P_α ;

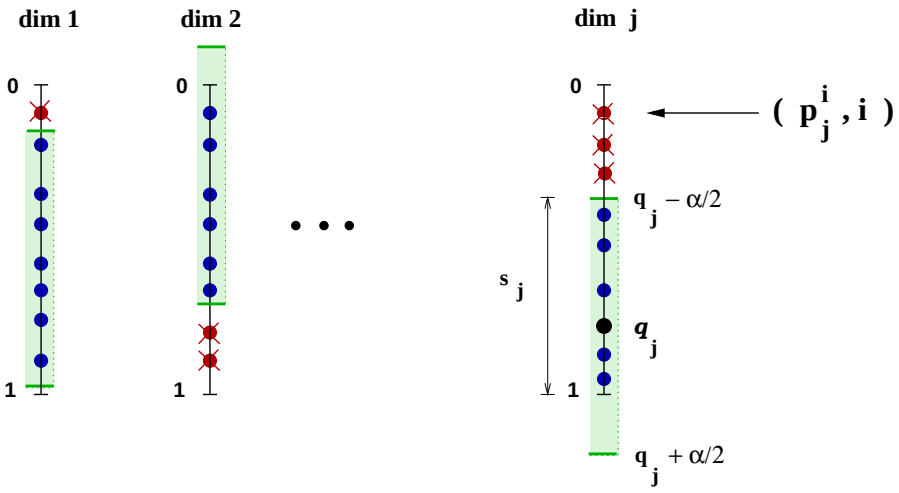


Figure 3.2: The REJECT procedure

Expected runtime of the REJECT procedure

We measure the runtime T_{reject} of the REJECT procedure by the number of comparisons of coordinates and of bit array entries performed in steps labeled by (TESTS) and (BITARRAY), respectively.

Clearly, the total number of comparisons of bit array entries performed in step (BITARRAY) is n. The expected total number of comparisons of coordinates performed in steps (TESTS) is given by:

$$\sum_{j=1}^{d} \left(2 + E\left[\text{ number of points outside of } [q_j - \alpha/2, q_j + \alpha/2] \right] \right) \;=\; 2d + \sum_{j=1}^{d} n \cdot (1 - s_j) \,,$$

where $s_j = s_j(\alpha)$ is the side length of the box $C_{q,\alpha} \cap [0,1]^d$ in dimension j. Let $I_j := [q_j - \frac{\alpha}{2}, q_j + \frac{\alpha}{2}]$ for $j \in \{1, \ldots, d\}$. By using the inequality between the arithmetic and geometric mean we obtain:

$$\sum_{j=1}^{d} E\,[\,\#\text{points outside } I_j\,] \;=\; \sum_{j=1}^{d} n \cdot (1 - s_j) \;\leq\; n(\,d - d\sqrt[d]{s_1 \cdot \ldots \cdot s_d}\,). \tag{3.1}$$

The product $\prod_{j=1}^{d} s_j$ is the volume of the box $C_{q,\alpha} \cap [0,1]^d$ and equals $\frac{\varphi}{n}$, where φ is the expected number of points in $C_{q,\alpha} \cap P$. By Lemma 2.3, we obtain:

$$\sum_{j=1}^{d} E\,[\,\#\text{points outside } I_j\,] \;\leq\; nd(1 - \sqrt[d]{\varphi/n}) \;\leq\; n \ln(n/\varphi). \tag{3.2}$$

Thus,

$$\mathrm{E}[T_{reject}] \;=\; n + 2d + \sum_{j=1}^{d} n \cdot (1 - s_j) \;\leq\; n + 2d + nd(1 - \sqrt[d]{\varphi/n}) \;\leq\; n + 2d + n \ln(n/\varphi)\,. \tag{3.3}$$

If special case $C_{q,\alpha} \subseteq [0,1]^d$ occurs then the above upper bound on the expected number of tests is asymptotically tight. In this special case we have

$$\mathrm{E}[T_{reject}] \;=\; n + 2d + nd(1 - \sqrt[d]{\varphi/n})\,. \tag{3.4}$$

Thus, by Lemma 2.2, and by using $\left(\sqrt[d]{\varphi/n} < \frac{1}{e} \iff d < \ln(n/\varphi)\right)$ we obtain:

$$\mathrm{E}[T_{reject}] \;\geq\; \begin{cases} \frac{n \ln(n/\varphi)}{2} + n + 2d & \text{if } d \geq \ln(n/\varphi) \\ (1 - \frac{1}{e})nd + n + 2d & \text{if } d < \ln(n/\varphi) \end{cases}\,. \tag{3.5}$$

The total expected runtime

The probability that the cube $C_{q,\alpha}$ does not contain any point of P is $(1 - \frac{\varphi}{n})^n$. In that case, the brute-force method is called, having a runtime of $\Theta(nd)$. With probability $1 - (1 - \frac{\varphi}{n})^n$ there is at least one point in $C_{q,\alpha}$ and in this case the brute-force method will be called with the set $C_{q,\alpha} \cap P$ having the expected runtime $\Theta(\varphi d)$.

The expected runtime $\mathrm{E}[T_{cube}]$ of the CUBE METHOD with the REJECT searching procedure is proportional to the number of performed comparisons and arithmetic operations. By (2.1), we have:

$$\begin{aligned} \mathrm{E}[T_{cube}] \;&=\; \Theta(\mathrm{E}[T_{reject}]) + (1 - \tfrac{\varphi}{n})^n \Theta(nd) + \left(1 - (1 - \tfrac{\varphi}{n})^n\right) \cdot \Theta(\varphi d) + \Theta(d \ln(d\varphi)) \\ &=\; O\left(\, n + n \ln(n/\varphi) + e^{-\varphi} nd + \varphi d + d \ln d \,\right) \end{aligned} \tag{3.6}$$

The choice of the parameter φ^*

Procedure SIDE_LENGTH(φ^*, q) computes side length α such that the expected number φ of points in $C_{q,\alpha} \cap P$ lies in $[\varphi^*, \varphi^* + 1)$. A suitable parameter φ^* is to be determined such that the expected runtime is minimized asymptotically.

For fixed $n, d \in \mathbb{N}$, $n \geq 2$ and $d \geq 2$, and a real $\gamma \in [1, n)$ we introduce the following notation:

$$U(n, d, \gamma) \;=\; n \ln(n/\gamma) + e^{-\gamma} nd + \gamma d \tag{3.7}$$

We obtain in (3.6) an upper bound on $\mathrm{E}[T_{cube}]$ which is $\Theta\left(U(n, d, \varphi^*) + n + d \ln d\right)$, where the constant in the Θ-notation is close to 1. Note that the computation of the side length α does not effect the total asymptotic expected runtime. Let φ_{min} be the value which minimizes $U(n, d, \gamma)$ for fixed numbers n and d over all possible settings of $\gamma \in [1, n)$. Note that any value different from φ_{min} cannot improve asymptotically the expected runtime we obtain with $\varphi^* = \varphi_{min}$. We have

$$\frac{\partial U(n, d, \gamma)}{\partial \gamma} = 0 \iff 1 = \frac{n}{e^\gamma} + \frac{n}{\gamma d} \tag{3.8}$$

and, obviously, $\gamma = \varphi_{min}$ is the unique value which satisfies (3.8).

By (3.8), we obtain

$$\max\left\{\frac{n}{e^\gamma}, \frac{n}{\gamma d}\right\} \leq 1 \leq 2 \max\left\{\frac{n}{e^\gamma}, \frac{n}{\gamma d}\right\}, \quad \text{if } \gamma = \varphi_{min}. \tag{3.9}$$

We have $\left(\frac{n}{e^\gamma} \leq 1 \;\Leftrightarrow\; \ln n \leq \gamma\right)$ and $\left(\frac{n}{\gamma d} \leq 1 \;\Leftrightarrow\; \frac{n}{d} \leq \gamma\right)$. By (3.9), these observations imply $\max\left\{\frac{n}{d}, \ln n\right\} \;\leq\; \varphi_{min}$.

We obtain the upper bound $\max\left\{\frac{2n}{d}, \ln(2n)\right\}$ on φ_{min} by using $\left(1 \leq \frac{2n}{e^\gamma} \Leftrightarrow \gamma \leq \ln(2n)\right)$ and $\left(1 \leq \frac{2n}{\gamma d} \Leftrightarrow \gamma \leq \frac{2n}{d}\right)$. Altogether, we get:

$$\max\left\{\frac{n}{d}, \ln n\right\} \;\leq\; \varphi_{min} \;\leq\; \max\left\{\frac{2n}{d}, \ln(2n)\right\} \tag{3.10}$$

The following shows that $\max\left\{\frac{n}{d}, \ln n\right\}$ is a good approximation of φ_{min}.

Proposition 3.1. *Consider the function $u : \mathbb{R}_+ \to \mathbb{R}$, $u(\gamma) := U(n, d, \gamma)$ defined in (3.7) for fixed $n, d \in \mathbb{N}$ with $n \geq 4$ and $d \geq 3$. Let φ_{min} be such that $\frac{\partial u}{\partial \gamma}(\varphi_{min}) = 0$. Then the following holds for all positive $n, d \in \mathbb{N}$:*

$$\frac{1}{2} U\left(n, d, \max\left\{\tfrac{n}{d}, \ln n\right\}\right) \;\leq\; U(n, d, \varphi_{min}) \;\leq\; 2U\left(n, d, \max\left\{\tfrac{n}{d}, \ln n\right\}\right)$$

Proof. If $\max\left\{\frac{n}{d}, \ln n\right\} = \frac{n}{d}$ then $\ln n \leq \frac{n}{d} \leq \varphi_{min} \leq \frac{2n}{d}$. Since $d \leq \frac{n}{\ln n}$ we obtain $d \ln d \leq n$, which implies $d \leq e^{n/d}$, which is equivalent to $\frac{nd}{2e^{n/d}} \leq \frac{n}{2}$. Thus, by (3.10):

$$\begin{aligned}
\frac{1}{2} u\left(\max\left\{\tfrac{n}{d}, \ln n\right\}\right) \;&=\; \frac{1}{2} n \ln d + \frac{1}{2}\frac{nd}{e^{n/d}} + \frac{n}{2} \;\leq\; n \ln(d/2) + n \;<\; n \ln(d/2) + \frac{nd}{e^{2n/d}} + n \\
&\leq\; n \ln(n/\varphi_{min}) + \frac{nd}{e^{\varphi_{min}}} + \varphi_{min} d \;=\; u(\varphi_{min}) \\
&\leq\; n \ln d + \frac{nd}{e^{n/d}} + 2n \;\leq\; 2u(n/d) \;=\; 2u\left(\max\left\{\tfrac{n}{d}, \ln n\right\}\right)
\end{aligned}$$

If $\max\left\{\frac{n}{d}, \ln n\right\} = \ln n$ then, by (3.10), $\frac{n}{d} \leq \ln n \leq \varphi_{min}$, and either $\ln(2n) \geq \frac{2n}{d}$, thus, $\ln n \leq \varphi_{min} \leq \ln(2n)$ and

$$\begin{aligned}\frac{1}{2}u\left(\max\left\{\tfrac{n}{d}, \ln n\right\}\right) &= \frac{n}{2}\ln\left(\frac{n}{\ln n}\right) + \frac{d}{2} + \frac{d}{2}\ln n \leq n\ln\left(\frac{n}{\ln(2n)}\right) + \frac{d}{2} + d\ln n \\ &\leq u(\varphi_{min}) \leq n\ln\left(\frac{n}{\ln n}\right) + d + d\ln(2n) \leq 2u\left(\max\left\{\tfrac{n}{d}, \ln n\right\}\right)\end{aligned}$$

or $\ln(2n) < \frac{2n}{d}$, thus, $\frac{n}{d} \leq \ln n \leq \varphi_{min} \leq \frac{2n}{d}$ and

$$\begin{aligned}\frac{1}{2}u(\varphi^*) &= \frac{n}{2}\ln\left(\frac{n}{\ln n}\right) + \frac{d}{2} + \frac{d}{2}\ln n \leq n\ln(d/2) + \frac{nd}{e^{2n/d}} + d\ln n \\ &\leq u(\varphi_{min}) \leq n\ln\left(\frac{n}{\ln n}\right) + d + 2n \leq 2u(\varphi^*)\end{aligned}$$

□

We choose parameter φ^* to equal $\max\left\{\frac{n}{d}, \ln n\right\}$. As described in Section 2.1.3, procedure SIDE_LENGTH computes for the parameter $\varphi^* \in [1, n-1]$ the side length α of a cube $C_{q,\alpha}$ with center q such that the expected number of points in $P \cap C_{q,\alpha}$ is $\varphi \in [\varphi^*, \varphi^* + 1)$. We obtain by (3.6):

$$\mathrm{E}[T_{cube}] = \begin{cases} O(n\ln d) & \text{if } d \leq \frac{n}{\ln n} \\ O\left(n\ln\left(\frac{n}{\ln n}\right) + d\ln d\right) & \text{if } d > \frac{n}{\ln n} \end{cases} \tag{3.11}$$

Thus, $\mathrm{E}[T_{cube}] = O(n\ln d + d\ln d)$.

If $C_{q,\alpha} \subseteq [0,1]^d$ we have by (3.5) and because of $\ln(n/\varphi) \leq \ln(n/\varphi^*) \leq \ln d < d$:

$$\mathrm{E}[T_{cube}] = \begin{cases} \Theta(n\ln d) & \text{if } d \leq \frac{n}{\ln n} \\ \Theta\left(n\ln\left(\frac{n}{\ln n}\right) + d\ln d\right) & \text{if } d > \frac{n}{\ln n} \end{cases} \tag{3.12}$$

Observe that if $d > \frac{n}{\ln n}$ then $\Theta\left(n\ln\left(\frac{n}{\ln n}\right) + d\ln d\right) = \Theta(n\ln d + d\ln d)$. Thus, by (3.11) and (3.12), we have $\mathrm{E}[T_{cube}] = \Theta(n\ln d + d\ln d)$ if the special case $C_{q,\alpha} \subseteq [0,1]^d$ occurs.

Remark 3.1. *By the choice of φ_{min} and by Proposition 3.1 there is no value of parameter φ^* such that the upper bound in (3.6) that we obtain for $\varphi^* = \max\left\{\frac{n}{d}, \ln n\right\}$ can be improved asymptotically.*

Theorem 3.1. *Let P be a set of n points from $[0,1]^d$. The* CUBE_METHOD *with the* REJECT *searching procedure finds the nearest neighbor from P to the query point $q \in [0,1]^d$ with an expected asymptotic runtime of $O(n\ln d + d\ln d)$ if the points of P are drawn independently at random from $[0,1]^d$ under uniform distribution.*

3.1.2 Searching the cube by rejecting and scanning

The drawback of the REJECT procedure is that points might be rejected more than once, as opposed to SCAN, which considers each point only once. The contribution of this section is to combine the procedures REJECT and SCAN in order to determine the points of P contained in the cube $C_{q,\alpha}$.

The *preprocessing* is done as described in Section 3.1.1. The data structure consists of the indexed data set P and the arrays T_j, $j = 1, \ldots, d$ as introduced in Section 3.1.1. The *query algorithm* is the CUBE METHOD with an orthogonal range searching algorithm, which works as follows. The algorithm determines a partition $\mathcal{D}_1 \cup \mathcal{D}_2$ of the set of dimensions $\mathcal{D}$. Procedure REJECT is called with the set $\mathcal{D}_1$ of dimensions to determine the set $P_\alpha^{\mathcal{D}_1}$ of the points p^i such that $p_j^i \in [q_j - \frac{\alpha}{2}, q_j + \frac{\alpha}{2}]$, for all $j \in \mathcal{D}_1$. In the second phase, procedure SCAN is called with the set $\mathcal{D}_2$ of dimensions and the set $P_\alpha^{\mathcal{D}_1}$ of points. In the following we give a schematic description of this searching algorithm called REJECT_SCAN and fill in the details later.

REJECT_SCAN$(\boldsymbol{\alpha, q, T, P, \mathcal{D}})$

$(\mathcal{D}_1, \mathcal{D}_2)$:= PARTITION$(\mathcal{D})$;
$P_\alpha^{\mathcal{D}_1}$:= REJECT$(\alpha, q, T, P, \mathcal{D}_1)$;
P_α := SCAN$(\alpha, q, P_\alpha^{\mathcal{D}_1}, \mathcal{D}_2)$;
return P_α ;

We prove that for a suitable partition $\mathcal{D}_1 \cup \mathcal{D}_2$ of the set of dimensions $\mathcal{D}$ and suitable parameter φ^*, the CUBE METHOD with REJECT_SCAN has an expected runtime of $O(n \ln(\frac{d}{\ln n} + 1) + n + d \ln d)$. The result is summarized in Theorem 3.2 at the end of this section.

3.1.2.1 Details of the algorithm and analysis of the expected runtime

To analyze the expected runtime of the combined parts REJECT and SCAN we determine the total number of comparisons of coordinates and bit array entries performed in steps labeled by (TESTS) and (BITARRAY) of procedure REJECT.

We will compute a suitable partition $\mathcal{D} = \mathcal{D}_1 \cup \mathcal{D}_2$ with $O(d)$ comparisons and multiplications.

We denote by λ the geometric mean of the side lengths of the box $C_{q,\alpha} \cap [0,1]^d$, which contains an expected number φ of points from P. By uniform distribution, the volume of $C_{q,\alpha} \cap [0,1]^d$ equals $\frac{\varphi}{n}$, thus, $\lambda = \sqrt[d]{\frac{\varphi}{n}}$.

First we analyze the special case $C_{q,\alpha} \subseteq [0,1]^d$.

A) Special case $\mathbf{C_{q,\alpha} \subseteq [0,1]^d}$: All side lengths equal $\alpha = \lambda = \sqrt[d]{\varphi/n}$.

The partition $\mathcal{D}_1 \cup \mathcal{D}_2$ is chosen such that

$$\mathcal{D}_1 = \{1, \ldots, p\} \qquad \mathcal{D}_2 = \{p+1, \ldots, d\}$$

where p is a parameter of the analysis, which we specify later. We express the expected runtime of REJECT_SCAN as a function of p and compute the optimal value p_{min} that minimizes it.

By (3.4), the expected total number of tests performed in the REJECT-part is

$$n + 2p + p(1-\lambda)n \ .$$

The expected total number of tests performed in the SCAN-part is given by:

$$\mathrm{E}[\ |P_\alpha^{\mathcal{D}_1}|\ ] \cdot (1 + \lambda + \ldots + \lambda^{d-p-1}) \ ,$$

which holds since events with respect to different dimensions of the points are independent. For reasons of simplicity, we ignore here the computation of parameter p, which can be done in this

case in constant time (see below). Thus, the expected total number $\mathrm{E}[T_{rej\_scan}]$ of tests performed by REJECT_SCAN for $p \geq 1$ is given by:

$$\begin{aligned} \mathrm{E}[T_{rej\_scan}] &= n + 2p + p(1-\lambda)n + n \cdot \lambda^p \cdot (1 + \lambda + \ldots + \lambda^{d-p-1}) \\ &= n + 2p + p(1-\lambda)n + \frac{n\lambda^p - \varphi}{1-\lambda} \,. \end{aligned} \tag{3.13}$$

The following lemma computes p such that $f : \mathbb{R} \to \mathbb{R}$ with $f(p) := 2p + p(1-\lambda)n + \frac{n\lambda^p}{1-\lambda}$ is minimized.

Lemma 3.1. *Consider the function $f : \mathbb{R} \to \mathbb{R}$ with $f(p) = 2p + p(1-\lambda)n + \frac{n\lambda^p}{1-\lambda}$, where $\lambda = \sqrt[d]{\frac{\varphi}{n}} \in (0,1)$, $n > 4$ and $d \geq 2$. Function f takes its minimum in $(0,\infty)$ at*

$$p_{min} = \frac{1}{\ln(1/\lambda)} \cdot \ln\left(\frac{\ln(1/\lambda)}{(1-\lambda)} \cdot \frac{1}{1-\lambda+\frac{2}{n}}\right) \,. \tag{3.14}$$

Proof. We have $f'(p) = 2 + (1-\lambda)n + n \cdot \frac{\lambda^p \ln\lambda}{1-\lambda}$. Since $\lambda \in (0,1)$ we obtain:

$$\begin{aligned} f'(p) = 0 \quad &\Longleftrightarrow \quad n \cdot \frac{\lambda^p \ln(1/\lambda)}{1-\lambda} = 2 + (1-\lambda)n \\ &\Longleftrightarrow \quad \lambda^p = \frac{(1-\lambda)^2}{\ln(1/\lambda)} + \frac{2}{n} \cdot \frac{1-\lambda}{\ln(1/\lambda)} \;=\; \frac{(1-\lambda)(1-\lambda+\frac{2}{n})}{\ln(1/\lambda)} \,. \end{aligned} \tag{3.15}$$

Since $\frac{(1-\lambda)(1-\lambda+\frac{2}{n})}{\ln(1/\lambda)} > 0$ for $\lambda \in (0,1)$, we get:

$$f'(p) = 0 \;\Longleftrightarrow\; p = \frac{1}{\ln(1/\lambda)} \cdot \ln\left(\frac{\ln(1/\lambda)}{(1-\lambda)} \cdot \frac{1}{1-\lambda+\frac{2}{n}}\right) \,.$$

We have $\lim\limits_{p\to\infty} \lambda^p = 0$ since $0 < \lambda < 1$. Thus, $\lim\limits_{p\to\infty} f'(p) = 2 + (1-\lambda)n > 0$. On the other hand, by Lemma 2.3, we get:

$$f'(0) \;=\; 2 + n(1-\lambda) - n \cdot \frac{\ln(1/\lambda)}{1-\lambda} \;\leq\; 2 + n(1-\lambda) - n \cdot \frac{1-\lambda}{1-\lambda} \;\leq\; 2 - n\lambda \;<\; 0 \,.$$

We used $n\lambda - 2 = n^{1-1/d} \cdot \sqrt[d]{\varphi} - 2 \geq n^{1-1/d} - 2 > 0$ for $n > 4$ and $d \geq 2$. Therefore, function f takes its minimum in $(0,\infty)$ at

$$p_{min} = \frac{1}{\ln(1/\lambda)} \cdot \ln\left(\frac{\ln(1/\lambda)}{(1-\lambda)} \cdot \frac{1}{1-\lambda+\frac{2}{n}}\right) \,.$$

We claim that $p_{min} > 0$. By Lemma 2.3, we obtain $\frac{\ln(1/\lambda)}{1-\lambda} \geq 1$ for $\lambda \in (0,1)$. For $n > 4$ we have $n\lambda = n \cdot \sqrt[d]{\frac{\varphi}{n}} \;>\; 2$, which is equivalent to $\left(1 - \lambda + \frac{2}{n}\right) < 1$. These imply $\frac{\ln(1/\lambda)}{(1-\lambda)} \cdot \frac{1}{1-\lambda+\frac{2}{n}} > 1$ and $p_{min} > 0$ follows.

□

We set parameter p to equal the value $p^* = \min\{d, \lceil p_{min} \rceil\} \geq 1$, where p_{min} is defined in (3.14).

Lemma 3.2. *If $C_{q,\alpha} \subseteq [0,1]^d$ then the expected asymptotic runtime of* REJECT_SCAN *with parameter $p = \min\{d, \lceil p_{min} \rceil\}$ is $O\left(n \ln(\frac{d}{\ln(n/\varphi)} + 1) + n \right)$.*

Proof.

CASE $d < \ln(n/\varphi)$ We claim $p_{min} < 1$. This is equivalent to $\lambda^{p_{min}} > \lambda$, since $\lambda \in (0,1)$. By (3.15), $\lambda^{p_{min}} = \frac{(1-\lambda)(1-\lambda+\frac{2}{n})}{\ln(1/\lambda)} > \frac{(1-\lambda)^2}{\ln(1/\lambda)}$. Thus, it suffices to prove $\frac{(1-\lambda)^2}{\ln(1/\lambda)} > \lambda$. Since $\lambda \in (0,1)$, we have

$$\frac{(1-\lambda)^2}{\ln(1/\lambda)} > \lambda \iff \frac{(1-\lambda)^2}{\lambda} > \ln(1/\lambda) \iff \frac{1}{\lambda} - 2 + \lambda > \ln(1/\lambda)\,. \tag{3.16}$$

In this case, we have

$$d < \ln(n/\varphi) \iff \lambda = \sqrt[d]{\varphi/n} < \frac{1}{e}\,. \tag{3.17}$$

Consider the function $g : (0, \frac{1}{e}] \to \mathbb{R}$, $g(\lambda) = \frac{1}{\lambda} - 2 + \lambda - \ln(1/\lambda)$. The derivative of this function is $g'(\lambda) = -\frac{1}{\lambda^2} + 1 + \frac{1}{\lambda} = \frac{\lambda^2+\lambda-1}{\lambda^2} < 0$ for $\lambda < \frac{1}{e}$. Since $g(\frac{1}{e}) = e + \frac{1}{e} - 3 = \frac{e^2-3e+1}{e} > 0$, we obtain $g(\lambda) > 0$ for $\lambda \in (0, \frac{1}{e}]$. By (3.16), we get $\lambda^{p_{min}} > \lambda$, thus $p_{min} < 1$ and $p^* = 1$. So the SCAN part dominates the searching algorithm. By (3.13), we obtain:

$$\mathrm{E}[T_{rej\_scan}] \;=\; n + 2 + (1-\lambda)n + \frac{n\lambda^2 - \varphi}{1-\lambda} \;\in\; (2 + n(1-1/e), 3n+2)\,.$$

CASE $d \geq \ln(n/\varphi)$ We have

$$\mathrm{E}[T_{rej\_scan}] \;=\; n + 2p^* + p^*(1-\lambda)n + \frac{n\lambda^{p^*} - \varphi}{1-\lambda}, \tag{3.18}$$

where $p^* = \min\{\lceil p_{min} \rceil, d\}$.

If $p_{min} > d$ then $p^* = d$ and $\mathrm{E}[T_{rej\_scan}] = \Theta(n \ln(n/\varphi) + n + d)$, which is the special case when REJECT_SCAN consists only of the REJECT part. Since

$$\lambda = \sqrt[d]{\frac{\varphi}{n}} \iff \ln(1/\lambda) = \frac{\ln(n/\varphi)}{d} \iff \ln(1/\lambda) \cdot d = \ln(n/\varphi)\,,$$

we have by (3.14):

$$(p_{min} > d) \iff \frac{1}{\ln(1/\lambda)} \cdot \ln\left(\frac{\ln(1/\lambda)}{(1-\lambda)\cdot(1-\lambda+\frac{2}{n})}\right) > d \iff \frac{\ln(1/\lambda)}{(1-\lambda)\cdot(1-\lambda+\frac{2}{n})} > \frac{n}{\varphi}.$$

Thus, by Lemma 2.2, the following holds in this case: $\frac{n}{\varphi} < \frac{\ln(1/\lambda)}{(1-\lambda)^2} \leq \ln(1/\lambda) \cdot \left(1 + \frac{1}{\ln(1/\lambda)}\right)^2 = \frac{d}{\ln(n/\varphi)} + \frac{\ln(n/\varphi)}{d} + 2$. Therefore, $\Theta(n\ln(n/\varphi) + n + d) = O\left(n \ln(\frac{d}{\ln(n/\varphi)}) + n \right)$ in this case.

If $p_{min} \leq d$ then $p^* = \lceil p_{min} \rceil$ and we need an estimation of p_{min}, which ist defined in (3.14). We use the following bounds implied by Lemma 2.3:

$$\frac{\ln(1/\lambda)}{(1-\lambda)} \cdot \frac{1}{1-\lambda+\frac{2}{n}} \;\leq\; \frac{1}{\ln(1/\lambda)} \cdot \left(\frac{\ln(1/\lambda)}{1-\lambda}\right)^2\,. \tag{3.19}$$

To get an upper bound on this expresion, we observe that $0 < \frac{\ln(1/\lambda)}{1-\lambda}$ is decreasing in λ, and, by (3.17), $\lambda = \sqrt[d]{\frac{\varphi}{n}} \geq \frac{1}{e}$ because of $d \geq \ln(n/\varphi)$. By (3.14), (3.19), by $\ln(1/\lambda) = \frac{\ln(n/\varphi)}{d}$, and since $\left(\frac{e}{e-1}\right)^2 < e$, we get the upper bound

$$p_{min} \leq \frac{d}{\ln(n/\varphi)} \cdot \ln\left(\frac{ed}{\ln(n/\varphi)}\right) ,$$

which implies:

$$p^* \leq \frac{d}{\ln(n/\varphi)} \cdot \ln\left(\frac{ed}{\ln(n/\varphi)}\right) + 1 . \tag{3.20}$$

By Lemma 2.3, we have $p^*(1-\lambda)n \leq n \cdot p^* \cdot \frac{\ln(n/\varphi)}{d}$, thus, we get:

$$p^*(1-\lambda)n \;\leq\; n \cdot \ln\left(\frac{ed}{\ln(n/\varphi)}\right) + \frac{n\ln(n/\varphi)}{d} . \tag{3.21}$$

By (3.15) and by Lemma 2.3, we have:

$$\frac{n\lambda^{p_{min}} - \varphi}{1-\lambda} = n \cdot \frac{1-\lambda+\frac{2}{n}}{\ln(1/\lambda)} - \frac{\varphi}{1-\lambda} \;\leq\; n - \frac{d(\varphi-2)}{\ln(n/\varphi)} . \tag{3.22}$$

This positive upper bound is also an upper bound for $\frac{n\lambda^{p^*}-\varphi}{1-\lambda}$, which either equals 0 (if $p^* = d$) or is less than $\frac{n\lambda^{p_{min}}-\varphi}{1-\lambda}$. Thus, $0 \leq \frac{n\lambda^{p^*}-\varphi}{1-\lambda} \leq n$.

This together with (3.18), (3.20) and (3.21) implies :

$$\begin{aligned} \mathrm{E}[T_{rej\_scan}] &\leq 2n + n \cdot \ln\left(\frac{ed}{\ln(n/\varphi)}\right) + \frac{n\ln(n/\varphi)}{d} + \frac{2d}{\ln(n/\varphi)} \cdot \ln\left(\frac{ed}{\ln(n/\varphi)}\right) + 2 \\ &= O\left(n\ln\left(\frac{d}{\ln(n/\varphi)}\right) + n\right) . \end{aligned}$$

□

B) General case: The ordered list $\mathcal{D} = (\mathcal{D}(1), \ldots, \mathcal{D}(d))$ of dimensions corresponds to the increasing order of the side lengths of the box $C_{q,\alpha} \cap [0,1]^d$:

$$s_{\mathcal{D}(1)} \leq s_{\mathcal{D}(2)} \leq \ldots \leq s_{\mathcal{D}(d)} . \tag{3.23}$$

$\mathcal{D}$ is computed once at the beginning, by procedure SIDE_LENGTH.

$\mathcal{D}_1 \cup \mathcal{D}_2$ is a partition of $\{1, \ldots, d\}$, where sets $\mathcal{D}_1$ and $\mathcal{D}_2$ are considered to be ordered. The size $p = |\mathcal{D}_1|$ is a parameter.

By (3.3), the expected total number of tests performed in the REJECT-part is

$$n + 2p + n \cdot \sum_{j \in \mathcal{D}_1} (1 - s_j) .$$

The expected total number of tests performed in the SCAN-part is given by:

$$\mathrm{E}[\,|P_\alpha^{\mathcal{D}_1}|\,] \cdot (1 + s_{j_1} + \ldots + \prod_{k=1}^{d-p-1} s_{j_k}) \;=\; n \cdot (\prod_{j \in \mathcal{D}_1} s_j\,) \cdot (1 + s_{j_1} + \ldots + \prod_{k=1}^{d-p-1} s_{j_k})\,, \quad (3.24)$$

where $(j_1, \ldots, j_{d-p}) = \mathcal{D}_2$. This holds since events with respect to different dimensions of the points are independent.

Let $T_{rs}\,(\mathcal{D}_1, \mathcal{D}_2)$ be the expected total number of tests performed in the parts REJECT and SCAN:

$$T_{rs}\,(\mathcal{D}_1, \mathcal{D}_2) \;=\; n + 2p + n \cdot \sum_{j \in \mathcal{D}_1} (1 - s_j) + n \cdot (\prod_{j \in \mathcal{D}_1} s_j\,) \cdot (1 + s_{j_1} + \ldots + \prod_{k=1}^{d-p-1} s_{j_k})\,, \quad (3.25)$$

where $(j_1, \ldots, j_{d-p}) = \mathcal{D}_2$.

CASE $d < \ln(n/\varphi)$: We prove that in this case the optimal partition of variables is:

$$\mathcal{D}_1 = (\,\mathcal{D}(1)\,) \qquad \mathcal{D}_2 = (\,\mathcal{D}(2), \ldots, \mathcal{D}(d-p+1)\,)$$

This proof is based on two observations. The first one uses the increasing order of the side lengths for partitions where parameter $p = 1$. Let $(l_1, l_2, \ldots, l_d)$ be a permutation different from $(\mathcal{D}(1), \ldots, \mathcal{D}(d))$. By (3.23), we have

$$\begin{aligned} T_{rs}\,(\mathcal{D}_1, \mathcal{D}_2) \;&=\; n + 2 + n(1 - s_{\mathcal{D}(1)}) + n \cdot (\,s_{\mathcal{D}(1)} + s_{\mathcal{D}(1)} \cdot s_{\mathcal{D}(2)} + \ldots + \prod_{k=1}^{d-p} s_{\mathcal{D}(k)}\,) \\ &\leq\; n + 2 + n(1 - s_{l_1}) + n \cdot (\,s_{l_1} + s_{l_1} \cdot s_{l_2} + \ldots + \prod_{k=1}^{d-p} s_{l_k}\,)\,. \end{aligned}$$

This proves that if $p = 1$ then $(\mathcal{D}_1, \mathcal{D}_2)$ is the optimal partition.

The second observation refers to partitions with parameter $p > 1$. Let $(l_1, l_2, \ldots, l_d)$ be some permutation and $\lambda = \sqrt[d]{\prod_{i=1}^{d} s_i} = \sqrt[d]{\frac{\varphi}{n}}$ is the geometric mean of all side lengths. We have:

$$\begin{aligned} T_{rs}\,(\mathcal{D}_1, \mathcal{D}_2) \;&=\; n + 2 + n(1 - s_{\mathcal{D}(1)}) + n \cdot s_{\mathcal{D}(1)} + n \cdot (\,s_{\mathcal{D}(1)} \cdot s_{\mathcal{D}(2)} + \ldots + \prod_{k=1}^{d-p} s_{\mathcal{D}(k)}\,) \\ &=\; 2 + 2n + n \cdot (\,s_{\mathcal{D}(1)} \cdot s_{\mathcal{D}(2)} + \ldots + \prod_{k=1}^{d-p} s_{\mathcal{D}(k)}\,) \;\leq\; 2 + 2n + n \cdot (\lambda^2 + \ldots + \lambda^{d-p}) \\ &\leq\; n + 2 + n + n \cdot \lambda^2 \cdot \frac{1}{1 - \lambda} \;\leq\; n + 2 + n \cdot (2 - 4\lambda) \qquad (3.26) \\ &<\; n + 2p + n \cdot \sum_{i=1}^{p} (1 - s_{l_i}) + n \cdot \prod_{i=1}^{p} s_{l_i} \cdot (1 + \; s_{l_{p+1}} + \ldots + \prod_{k=p+1}^{d-1} s_{l_k}\,)\,. \end{aligned}$$

We used $s_i \leq 2\lambda\ \forall i$, $p \geq 2$ and $1 + \frac{\lambda^2}{1-\lambda} \leq 2 - 4\lambda$ for $\lambda < \frac{1}{e}$, which holds by (3.17). This complete the proof that $(\mathcal{D}_1, \mathcal{D}_2)$ is the optimal partition in this case.

Since $\lambda < \frac{1}{e}$ and by (3.26), we get $T_{rs}\left(\mathcal{D}_1, \mathcal{D}_2\right) < 2 + n + n \cdot (1 + \frac{\lambda^2}{1-\lambda}) < \frac{5}{2}n + 2$.

CASE $d \geq \ln(n/\varphi)$: It is difficult to efficiently find the best partition $\mathcal{D}_1 \cup \mathcal{D}_2$ of $\mathcal{D}$ such that the expected runtime of REJECT_SCAN is minimized. We can prove good bounds for the partition strategy which uses the REJECT procedure for the dimensions corresponding to the largest side lengths. The intuitive motivation for this choice is the observation that SCAN works better with respect to dimensions corresponding to smaller side lengths of the box $C_{q,\alpha} \cap [0,1]^d$. We refer for illustration to Figure 3.3.

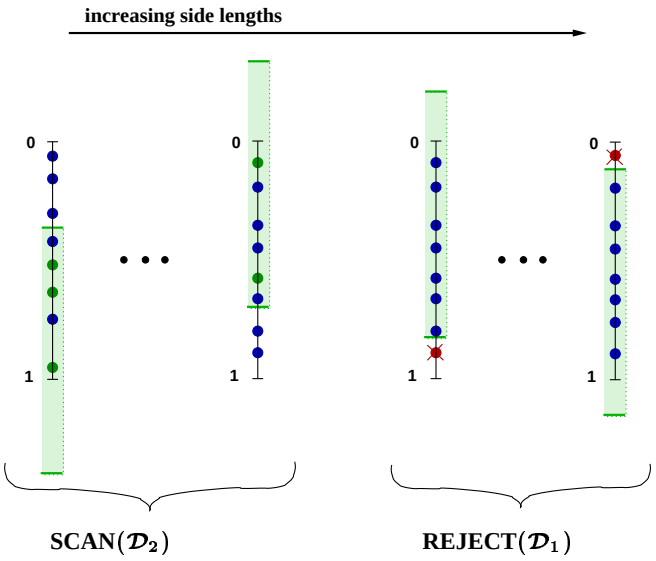


Figure 3.3: The REJECT_SCAN procedure

Procedure PARTITION gets the ordered set $\mathcal{D}$ with property (3.23) as input. The partition $\mathcal{D}_1 \cup \mathcal{D}_2$ of $\{1, \ldots, d\}$ is chosen as follows

$$\mathcal{D}_1 = (\ \mathcal{D}(d-p+1), \ldots, \mathcal{D}(d)\ ) \qquad \mathcal{D}_2 = (\ \mathcal{D}(1), \ldots, \mathcal{D}(d-p)\ )\ . \tag{3.27}$$

It remains to determine a suitable value of the parameter p.

The geometric mean of all side lengths is $\lambda = \sqrt[d]{\prod_{i=1}^{d} s_i} = \sqrt[d]{\frac{\varphi}{n}}$. Additionally, we introduce the following geometric means of side lengths:

$$\gamma(p) = \sqrt[d-p]{\prod_{j \in \mathcal{D}_2} s_j} \qquad \beta(p) = \sqrt[p]{\prod_{j \in \mathcal{D}_1} s_j} \tag{3.28}$$

We have $\gamma(p) \leq \lambda \leq \beta(p)$ for $1 \leq p \leq d$. $T_{rs}\left(\mathcal{D}_1, \mathcal{D}_2\right)$ is given by:

$$T_{rs}\left(\mathcal{D}_1, \mathcal{D}_2\right) = n + 2p + n \cdot \sum_{j \in \mathcal{D}_1} (1 - s_j) + n \cdot (\beta(p))^p \cdot (\ 1 + s_{\mathcal{D}(1)} + \ldots + \prod_{j=1}^{d-p-1} s_{\mathcal{D}(j)}\ )$$

Since $s_{\mathcal{D}(1)} \leq s_{\mathcal{D}(2)} \leq \ldots \leq s_{\mathcal{D}(d)}$ we have

$$1 + s_{\mathcal{D}(1)} + \ldots + \prod_{j=1}^{d-p-1} s_{\mathcal{D}(j)} \;\leq\; 1 + \gamma(p) + \ldots + (\gamma(p))^{d-p-1} \;=\; \frac{1-(\gamma(p))^{d-p}}{1-\gamma(p)} \,,$$

which can be shown in analogy to (2.5) (see the proof of Lemma 2.1). We obtain:

$$T_{rs}\,(\mathcal{D}_1, \mathcal{D}_2) \;\leq\; n + 2p + n \cdot \sum_{j \in \mathcal{D}_1} (1 - s_j) + \frac{n \cdot (\beta(p))^p - \varphi}{1 - \gamma(p)} \tag{3.29}$$

Parameter p is set to equal p^* defined as follows:

$$p^* \;=\; \begin{cases} \min\{\, p \mid \prod_{j \in \mathcal{D}_1} s_j \;\leq\; \lambda^{p_{min}} \text{ and } 0 \leq p = |\mathcal{D}_1| \leq d \,\} & \text{if } p_{min} \leq d \\ 1 & \text{otherwise} \end{cases} \tag{3.30}$$

where p_{min} is defined in (3.14). $(\mathcal{D}_1|_{p=p^*}, \mathcal{D}_2|_{p=p^*})$ is here an approximation for the optimal solution.

The value p^* can be determined with p^* comparisons and p^* multiplications. By (3.14), $\lambda^{p_{min}} = \frac{(1-\lambda)(1-\lambda+\frac{2}{n})}{\ln(1/\lambda)}$ is a constant. The value p^* is well defined: obviously, the function $(\beta(p))^p = \prod_{j \in \mathcal{D}_1} s_j$ is monotone decreasing in p. In addition $\beta(0) := 1 > \lambda^{p_{min}}$. Thus, $p^* \geq 1$.

If the special case $C_{q,\alpha} \subseteq [0,1]$ occurs, all side lengths equal $\lambda = \sqrt[d]{\varphi/n}$, and p^* defined above in (3.30) takes the same value as discussed in the special case analysis, that is $p^* = \min\{d, \lceil p_{min} \rceil\}$.

The partition $\mathcal{D}_1 \cup \mathcal{D}_2$ is determined as given in (3.27), where parameter p is set to equal the value p^* defined in (3.30). To estimate the expected runtime of REJECT_SCAN, it is sufficient to focus on the number T_{rej_scan} of comparisons performed:

$$\mathrm{E}[T_{rej\_scan}] = T_{rs}(\mathcal{D}_1, \mathcal{D}_2) + p^* \leq T_{rs}(\mathcal{D}_1, \mathcal{D}_2) + d \,. \tag{3.31}$$

Lemma 3.3. *If $d \geq \ln(n/\varphi)$ then the total expected number of tests performed by* REJECT_SCAN *is bounded as follows:*

$$E[T_{rej\_scan}] \;<\; n \ln\left(\frac{d}{\ln(n/\varphi)}\right) + 4n + 3d$$

Proof. From the definition of p^* in (3.30) and using the notations from (3.28), we have:

$$\prod_{j=d-p^*+2}^{d} s_{\mathcal{D}(j)} \;=\; (\beta(p^*-1))^{p^*-1} \;>\; \lambda^{p_{min}} \tag{3.32}$$

$$\prod_{j \in \mathcal{D}_1} s_j \;=\; \prod_{j=d-p^*+1}^{d} s_{\mathcal{D}(j)} \;=\; (\beta(p^*))^{p^*} \;\leq\; \lambda^{p_{min}} \,, \quad \text{if } p_{min} \leq d \tag{3.33}$$

If $p_{min} > d$ then $(\beta(p^*))^{p^*} = \lambda^d$.

We first estimate the term $\frac{n\cdot(\beta(p^*))^{p^*}-\varphi}{1-\lambda}$ of the bound in (3.29), which is either 0 or can be bounded in the case $p_{min} \leq d$ by using (3.33) and (3.22):

$$\frac{n\cdot(\beta(p^*))^{p^*}-\varphi}{1-\lambda} \;\leq\; \frac{n\cdot\lambda^{p_{min}}-\varphi}{1-\lambda} \;\leq\; n-\frac{d(\varphi-2)}{\ln(n/\varphi)} \tag{3.34}$$

The term $n\cdot\sum_{j\in\mathcal{D}_1}(1-s_j)$ of the sum in (3.29) can be bounded by using the inequality between the arithmetic and geometric mean. Additionally we use inequality (3.32), Lemma 2.3 and the fact $s_j \geq \frac{\alpha}{2} \geq \frac{\lambda}{2}$ for any $j \in \{1,\ldots,d\}$, which follows by (2.3).

$$\begin{aligned}
\sum_{i=d-p^*+1}^{d}(1-s_{\mathcal{D}(i)}) &< \sum_{i=d-p^*+2}^{d}(1-s_{\mathcal{D}(i)})+\left(1-s_{\mathcal{D}(d-p^*+1)}\right)\\
&\leq (p^*-1)\cdot(\,1-\beta(p^*-1)\,)+1-\lambda/2\\
&\leq (p^*-1)\cdot\ln\left(\frac{1}{(\beta(p^*-1))}\right)+\frac{1}{2}+\left(\frac{1}{2}-\frac{\lambda}{2}\right)\\
&< \ln\left(\frac{1}{\lambda^{p_{min}}}\right)+\frac{1}{2}+\frac{1}{2}\cdot\frac{\ln(n/\varphi)}{d}\\
&\leq \ln\left(\frac{\ln(1/\lambda)}{(1-\lambda)(1-\lambda+\frac{2}{n})}\right)+1
\end{aligned} \tag{3.35}$$

By (3.19), we get $\ln\left(\frac{\ln(1/\lambda)}{(1-\lambda)((1-\lambda+\frac{2}{n}))}\right) \leq \ln\left(\frac{ed}{\ln(n/\varphi)}\right)$, thus:

$$n\cdot\sum_{i=d-p^*+1}^{d}(1-s_{\mathcal{D}(j)}) \;<\; n\cdot\ln\left(\frac{d}{\ln(n/\varphi)}\right)+n\cdot\ln(e)+n\,. \tag{3.36}$$

By (3.29), (3.31), (3.36) and (3.35), we obtain:

$$\begin{aligned}
\mathrm{E}[T_{rej_scan}] &\leq n+2p+n\cdot\ln\left(\frac{d}{\ln(n/\varphi)}\right)+2n+n-\frac{d(\varphi-2)}{\ln(n/\varphi)}+d\\
&< n\ln\left(\frac{d}{\ln(n/\varphi)}\right)+4n+3d\,.
\end{aligned}$$

□

This bound on $\mathrm{E}[T_{rej\_scan}]$ is not tight for very large values of d, e.g. $d=\Omega\left(\frac{n}{\varphi}\cdot\ln(n/\varphi)\right)$. By using $\sum_{j\in\mathcal{D}_1}(1-s_j) \leq \sum_{j=1}^{d}(1-s_j) \leq n\ln(n/\varphi)$ (see (3.1) and (3.2)), we get $\mathrm{E}[T_{rej\_scan}] = O(n\ln(n/\varphi)+n+d)$, which is a better bound in this case.

The total expected runtime

The expected runtime $\mathrm{E}[T_{cube}]$ of the CUBE METHOD with the REJECT_SCAN searching procedure is proportional to the number of performed comparisons and arithmetic operations. By (2.1), we have:

$$\mathrm{E}[T_{cube}] \;=\; \Theta(\mathrm{E}[T_{rej\_scan}])+(1-\tfrac{\varphi}{n})^n\Theta(nd)+\left(1-(1-\tfrac{\varphi}{n})^n\right)\cdot\Theta(\varphi d)+\Theta(d\ln(d\varphi))\,.$$

Thus,

$$\begin{aligned}
&\text{If } d < \ln(n/\varphi) \quad \text{then} \quad \mathrm{E}[T_{cube}] \;=\; O\left(n + e^{-\varphi}nd + \varphi d + d\ln d\right) \\
&\text{If } d \geq \ln(n/\varphi) \quad \text{then} \quad \mathrm{E}[T_{cube}] \;=\; O\left(n\ln\left(\tfrac{d}{\ln(n/\varphi)}\right) + n + e^{-\varphi}nd + \varphi d + d\ln d\right)
\end{aligned} \tag{3.37}$$

To obtain a total asymptotic runtime of $O\left(n\ln\left(\frac{d}{\ln n}+1\right)+n+d\ln d\right)$, it is sufficient to guarantee that parameter φ fulfills the following inequality:

$$e^{-\varphi}nd \;\leq\; \max\left\{n\ln\left(\tfrac{d}{\ln(n/\varphi)}+1\right), n\right\}.$$

We have $e^{-\varphi}nd \leq n \iff \varphi \geq \ln d$ and

$$e^{-\varphi}nd \leq n\ln\left(\tfrac{d}{\ln(n/\varphi)}+1\right) \iff \varphi \geq \ln d - \ln\left(\tfrac{d}{\ln(n/\varphi)}+1\right)$$

We choose parameter φ^* of procedure SIDE_LENGTH to equal $\ln d$. Since $\varphi \in [\varphi^*, \varphi^*+1)$ we obtain:

$$\mathrm{E}[T_{cube}] \;=\; \begin{cases} O(n) & \text{if } d < \ln n \\ O\left(n\ln\left(\frac{d}{\ln n}\right)+n+d\ln d\right) & \text{if } \ln n \leq d < n \\ O(d\ln d) & \text{if } d \geq n \end{cases} \tag{3.38}$$

Altogether, we have proven the following theorem.

Theorem 3.2. *The* CUBE METHOD *with the procedure* REJECT_SCAN *finds the nearest neighbor from P to a query point with an expected asymptotic runtime of $O\left(n\ln(\frac{d}{\ln n}+1)+n+d\ln d\right)$ if the n points of P are drawn independently at random under uniform distribution. The method requires $O(nd)$ storage and $O(nd\ln n)$ preprocessing time.*

3.2 Speeding up the CUBE METHOD by using monotone sequences

In this section we present a query algorithm that is based on a preprocessed partition of the data set P, such that each subset of the partition allows an efficient orthogonal range searching. The partition consists of sequences of points that are monotone in $\mathbb{R}^d$ with respect to some of the dimensions. For dimensions $d \leq \left(\frac{\ln n}{\ln \ln n}\right)^2$ the query algorithm has an expected runtime of $O\left(\sqrt{d} \cdot n^{1-\frac{1}{\sqrt{d}}} \cdot \ln n\right)$. For higher dimensions $d > \left(\frac{\ln n}{\ln \ln n}\right)^2$ its expected runtime is $O\left(d \cdot n^{1-\frac{1}{\sqrt{d}}} \cdot \ln \ln n\right)$.

3.2.1 Preliminaries

We call a d-dimensional sequence to be monotone, if it is monotone in each of its d dimensions. Since in higher dimensions a partition into monotone sequences might be very large (see Section 3.2.3 and Section 3.2.4), we consider a preprocessed partition of the point set into sequences which are monotone only with respect to a subset $\mathcal{D}_1 \subset \{1, \ldots, d\}$ of the dimensions. The query algorithm is the CUBE METHOD, which considers for a given query point q the cube $C_{q,\alpha}$. To determine the point set $P \cap C_{q,\alpha}$ we proceed as follows. For each sequence of the preprocessed partition we perform a logarithmic orthogonal range searching restricted to the dimensions of $\mathcal{D}_1$. We obtain the set $P_\alpha^{\mathcal{D}_1}$ of points p^i such that $p_j^i \in [q_j - \frac{\alpha}{2}, q_j + \frac{\alpha}{2}]$, for all $j \in \mathcal{D}_1$. These points are tested by the SCAN algorithm with respect to the rest of the dimensions $\{1, \ldots, d\} \setminus \mathcal{D}_1$, to finally obtain the points contained in the cube. This searching method will be described in more detail in Section 3.2.5.

In Section 3.2.2 we mention methods to compute a suitable partition of the set P into monotone sequences. Section 3.2.3 and Section 3.2.4 deal with the worst-case complexity and expected complexity of monotone partitions, respectively. In Section 3.2.5 we finally present the searching method and its analysis, and summarize the results in Theorem 3.7.

In the following we define monotone sequences in $\mathbb{R}^d$ and motivate their use in geometric searching. We show that monotone sequences in $\mathbb{R}^d$ allow efficient orthogonal range searching and efficient nearest-neighbor search.

Monotone sequences in two dimensions

Erdős and Szekeres [25] proved that any sequence $\{a_j\}$ of n real numbers has a monotone (increasing or decreasing) subsequence of length $\lceil \sqrt{n} \rceil$. Now considering a set $S = \{(a_i, b_i) \mid i \in \{1, \ldots, n\}\}$ of n distinct points in the plane we can easily find a monotone subsequence of length $\lceil \sqrt{n} \rceil$. The elements of S are sorted with respect to the increasing order of the first coordinate of the points; w.l.o.g. let this order be $a_1 \leq a_2 \leq \ldots \leq a_n$. The sequence $\{b_i\}$ has a monotone subsequence $\{b_{i_j}\}$ of length $\lceil \sqrt{n} \rceil$. Thus, the subsequence $\{(a_{i_j}, b_{i_j})\}$ of S of length $\lceil \sqrt{n} \rceil$ is monotone with respect to some order $o \in \{ (\leq, \leq); (\leq, \geq) \}$.

A consequence of the Erdős-Szekeres result is the existence of a partition of a set of n points in the plane into $O(\sqrt{n})$ monotone subsequences. This partition can be computed in time $O(n^{3/2})$ [14]. A longest monotone increasing subsequence of a sequence of n real numbers can be computed in time $O(n \log n)$.

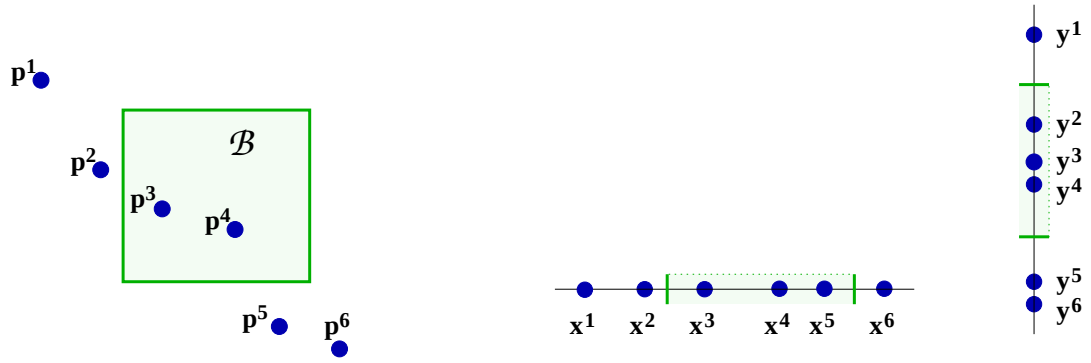
Partitioning into monotone subsequences is a useful tool for various applications in the plane. Matoušek and Welzl developed an algorithm for the halfspace range-counting problem in the plane, using the Erdős-Szekeres result [52]. This technique has also been applied to solve some other geometric-searching problems, including ray shooting and intersection searching [13].

In the following we show that monotone sequences in $\mathbb{R}^d$ allow efficient orthogonal range searching and efficient nearest-neighbor search.

Monotone sequences in higher dimensions

A sequence of points in $\mathbb{R}^d$ is called monotone in $\mathbb{R}^d$ if it is monotone with respect to some order from $\{\leq, \geq\}^d$, in other words if it is monotone in each dimension $i \in \{1, \ldots, d\}$. We define the set $\mathcal{R}_d = \{\leq, \geq\}^d$ of *reflexive partial orders* on $\mathbb{R}^d$. Let $o \in \mathcal{R}_d$, $o = (o(1), \ldots, o(d))$ with $o(i) \in \{\leq, \geq\}$, $i \in \{1, \ldots, d\}$. Consider two points in $\mathbb{R}^d$ $a = (a_1, \ldots, a_d)$ and $b = (b_1, \ldots, b_d)$. where $a_i, b_i \in \mathbb{R}$. We write as usual $a \, o \, b$ to mean that (a, b) is in the order o, which is defined as follows: $a \, o \, b \iff a_i \; o(i) \; b_i \;\; \forall \, i \in \{1, \ldots, d\}$.

Definition 3.1. A sequence $\mathcal{S} = [\, p_1, p_2, \ldots, p_r \,]$ of distinct points from $\mathbb{R}^d$ is *monotone* in $\mathbb{R}^d$ if and only if there is some order $o \in \mathcal{R}_d$ such that $p_1 \, o \, p_2 \, o \, \ldots \, o \, p_r$ holds. We call $\mathcal{S}$ to be *monotone with respect to* o.



$S = \{p^1, p^2, \ldots, p^6\}$

$b = \max\{2, 3\} \qquad t = \min\{5, 4\}$

$\mathcal{B} \cap S = \{p^b, \ldots, p^t\} = \{p^3, p^4\}$

Figure 3.4: Orthogonal range searching for a monotone sequence

For any monotone sequence orthogonal range searching can be done in time $O(d \log r)$. Given are a monotone sequence $S = [\, p^1, p^2, \ldots, p^r \,]$ and an orthogonal range $\mathcal{B} = [l_1, r_1] \times \ldots \times [l_d, r_d]$. To determine $\mathcal{B} \cap S$ we find by binary search in time $O(\log r)$ for each dimension $j \in \{1, \ldots, d\}$ the indices b_j and t_j of the point whose coordinate is leftmost in $[l_j, r_j]$ and of the point whose coordinate is rightmost in $[l_j, r_j]$, respectively. We can determine $b = \max\{b_1, \ldots, b_d\}$ and $t =$

$\min\{t_1, \ldots, t_d\}$ in time $O(d)$. Obviously, $\mathcal{B} \cap S = \{p^b, p^{b+1}, \ldots, p^t\}$. Figure 3.4 illustrates this orthogonal range searching.

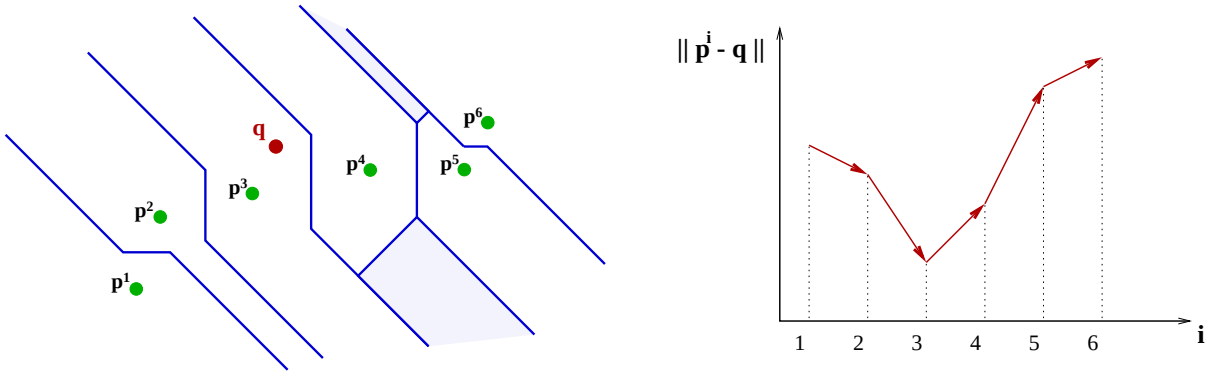


Figure 3.5: Voronoi diagram of the monotone sequence $[\ p^1, p^2, \ldots, p^r\ ]$ and the corresponding distance sequence $\{\ ||p^i - q||_\infty\ \}$

Another advantageous property of a monotone sequence is that nearest-neighbor search can be done by binary search in time $O(d \log r)$.

Lemma 3.4. *Consider a monotone sequence* $S = [\ p^1, p^2, \ldots, p^r\ ]$ *and a query point* $q \in \mathbb{R}^d$. *Let* $d_i = ||p^i - q||_\infty$, $i \in \{1, \ldots, r\}$. *Then there exists no index* $i \in \{2, \ldots, r-1\}$ *such that* $d_{i-1} < d_i > d_{i+1}$.

Proof. Assume $d_{i-1} < d_i > d_{i+1}$ for some $i \in \{2, \ldots, r-1\}$.

Let l be the dimension such that $||p^i - q||_\infty = |p^i_l - q_l|$. For the dimension l the sequence $S_l = [\ p^1_l, p^2_l, \ldots, p^r_l\ ]$ is monotone. By symmetry, it is sufficient to consider the case when S_l is monotone increasing. In this case we have:

$$p^{i-1}_l \ \leq\ p^i_l \ \leq\ p^{i+1}_l\ .$$

If $q_l \geq p^i_l$ then $d_i = |p^i_l - q_l| \leq |p^{i-1}_l - q_l| \leq d_{i-1}$, which is a contradiction to our assumption. If $q_l \leq p^i_l$ then $d_i = |p^i_l - q_l| \leq |p^{i+1}_l - q_l| \leq d_{i+1}$, which is a contradiction to our assumption.

□

Lemma 3.4 is illustrated in Figure 3.5 where the Voronoi diagram of a monotone sequence in 2 dimensions is also shown.

Lemma 3.4 implies that the minimal distance $d^* = ||p^* - q||_\infty$ can be found by binary search in $O(\log r)$ steps. In each step a distance $||p^j - q||_\infty$ is computed. Thus, the nearest neighbor to q from S can be found in time $O(d \log r)$.

3.2.2 Computing a partition into monotone subsequences

Consider a partially ordered set (poset) $P = (P, \prec)$ to be a pair of a ground set P and a reflexive, antisymmetric and transitive binary relation $\prec$ on P. A *chain* of P is a set of pairwise comparable elements and an *antichain* of P is a set of pairwise incomparable elements, where two elements $x \neq y \in P$ are incomparable if neither $x \prec y$ nor $y \prec x$. The *height* of P is the size of a maximum chain and the *width* of P is the size of a maximum antichain in P.

A w*-cover* of $P = (P, \prec)$ where $w \in \mathbb{N}$ is a set of w disjoint chains such that every element of $\mathcal{P}$ is in some chain. A fundamental theorem of partial orders is the following.

Theorem 3.3 (Dilworth's Theorem). *Every finite poset of width w has a w-cover and w is the minimum number of chains needed to cover the elements of P.*

We consider partial orders from $\mathcal{R}_d$. Let us first restrict the partition of P into monotone sequences in one of these partial orders, $\prec \in \mathcal{R}_d$. The corresponding poset graph $G_{(P,\prec)}$ can be determined in time $O(dn^2)$. $G_{(P,\prec)} = (V, E)$ has as vertex set $V = \{1, \ldots, n\}$ the indices of the points of P and the edge set is defined $E = \{\, (i, j) \,:\, p^i \prec p^j \,\}$. Figure 3.6 shows a 2-dimensional poset $(\, P, (\leq, \leq)\,)$ and its corresponding embedded poset graph.

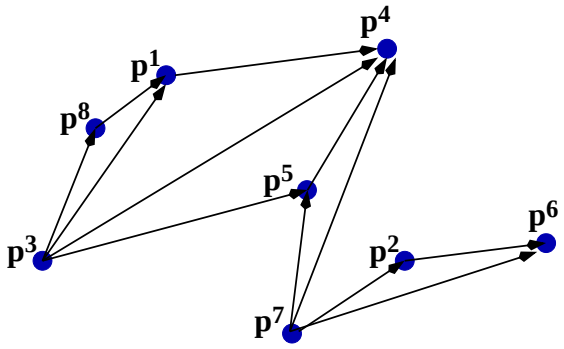


Figure 3.6: Poset $(\, P, (\leq, \leq)\,)$ and its corresponding embedded poset graph

We will compute a minimum chain cover of $(P, \prec)$. The width of an n element poset with the corresponding antichain and chain cover can be obtained using a max-flow computation on a bipartite network with unit capacities in $O\left(n^{\frac{5}{2}}/\sqrt{\log n}\right)$ time. If the width of $(P, \prec)$ is at most k, then a minimum chain cover can be computed in $O(kn^2)$ time [28].

The algorithms are based on the Fulkerson's proof of Dilworth's Theorem by reducing it to the König-Egervary duality theorem for bipartite graphs, which states that in a bipartite graph the size of a minimum vertex cover equals the size of a maximum matching. Fulkerson's proof uses the *split* $S(P)$ of $(P, \prec)$, i.e., the bipartite graph having as vertices two copies P', P'' of P and an edge (x', y'') whenever $x \prec y$, $x \neq y$ in P. A matching M in $S(P)$ corresponds to a partition of P into $|P| - |M|$ chains : begin with the partition of P into 1-element chains, then for each edge $(x', y'') \in M$ hook the tail of the chain ending with x to the beginning of the chain with y thus reducing the number of chains by one. Let U be a vertex cover of $S(P)$ and associate the antichain $A_U = \{x \in P \,:\, x', x'' \notin U\}$ with it. Because $S(P)$ comes from a transitive order relation it

can be shown that $|A_U| = |P| - |U|$ when U is a minimal vertex cover. Then Dilworth's theorem follows from $\max |M| = \min |U|$, the König-Egervary duality theorem.

By the above considerations the algorithm for bipartite matching of Hopcroft and Karp [41] allows to compute the width of P in $O(n^{\frac{5}{2}})$ time. An improvement to $O\left(n^{\frac{5}{2}}/\sqrt{\log n}\right)$ has been obtained by Alt et al. [6].

To find a minimum chain cover in $O\left(n^{\frac{5}{2}}/\sqrt{\log n}\right)$ time, let M be the maximum matching obtained by the algorithm, view P as a directed graph and consider the subgraph obtained by restricting to the edges $\{(x, y) : (x', y'') \in M\}$. As in Fulkerson's proof, it is easy to see that this subgraph yields the desired minimum chain cover.

For orders of width at most k the minimum chain cover can be computed in $O(kn^2)$ time as shown in [28]. They compute the max-flow on the bipartite network in two phases. In the first phase a greedy chain decomposition is computed by the recursive extraction of chains of maximum length from P and is used to obtain a feasible flow which is at most $k \log n$ units less than the maximum. In the second phase the true max-flow is computed using augmenting paths.

Corollary 3.1. [41, 6] *For a given order $(P, \prec)$, where P is a set of n points in $\mathbb{R}^d$ and $\prec \in \mathcal{R}_d$, a minimum chain partition can be found in $O\left(n^{\frac{5}{2}}/\sqrt{\log n}\right)$ time.*

To compute a partition of the point set P into monotone sequences we do not have to restrict the monotonicity to only one of the orders in $\mathcal{R}_d$. We can proceed as follows.

We consider a set $\mathcal{O} = \{\prec_1, \ldots, \prec_r\} \subseteq \mathcal{R}_d$ of orders and compute a decomposition of P into sequences that are monotone with respect to one of the orders from $\mathcal{O}$. Let $k_1, \ldots, k_r$ be the widths of the posets $(P, \prec_1), \ldots, (P, \prec_r)$, respectively. We define the *$\mathcal{O}$-minimum width* of P to be $k = \min\{k_1, \ldots, k_r\}$. We compute a *greedy $\mathcal{O}$-monotone chain decomposition* of the point set P, that is obtained by the recursive extraction from P of chains of maximum length over the orders $\mathcal{O} = \{\prec_1, \ldots, \prec_r\}$ of P. We define $P_0 = P$ and recursively define chain C_i to be the longest chain of all maximum chains of $(P^{i-1}, \prec_1), \ldots, (P^{i-1}, \prec_r)$, respectively. $(P^i, \prec_l)$ are the orders induced on the set $P^i = P^{i-1} \setminus C_i$. A maximum chain of an induced suborder of $(P, \prec_l)$ can be found by a modified depth-first-search for finding a longest path in the corresponding poset graph, which is a directed acyclic graph. In analogy to [28], we observe the following: the longest maximum chain in $(P^i, \prec_1), \ldots, (P^i, \prec_r)$ has size at least P^i/k. Therefore, $|P^i| \leq n(1 - \frac{1}{k})^i \leq ne^{-\frac{i}{k}}$, which is less than one for $i > k \ln n$. Thus, the greedy monotone chain decomposition with respect to $\{\prec_1, \ldots, \prec_r\}$ consists of at most $k \ln n$ chains.

The above upper bound on the size of the greedy decomposition is not tight. Intuitively, we could get a smaller decomposition into monotone sequences if we do not restrict ourselves to only one of the orders in $\mathcal{R}_d$. For our theoretical analysis we restrict the decomposition to contain subsequences which are monotone with respect to only one of the orders in $\mathcal{R}_d$. The expected runtime analysis of the query algorithm based on monotone decompositions is presented in Section 3.2.5.

We conclude the sections preparing this runtime analysis with a survey on the length of a longest monotone sequence in $\mathbb{R}^d$ and the size of a monotone decomposition. In high dimensions the longest monotone subsequence of a point set can be very small, thus, any monotone decomposition is large. The following section shows that there exists a set $P^* \subset \mathbb{R}^d$ of n points which has

no monotone subsequence of length larger then $\lceil n^{\frac{1}{2^{d-1}}} \rceil$, even though the subsequences of P are allowed to be monotone with respect to any of the $\mathcal{R}_d$-orders.

More positive results are known on the expected length of the longest chain of an order $(P, \prec)$, $\prec \in \mathcal{R}_d$ if the points of P are chosen uniformly independently at random from $[0,1]^d$. These are mentioned in Section 3.2.4.

3.2.3 A set with longest monotone subsequences of length $\lceil n^{\frac{1}{2^{d-1}}} \rceil$

A monotone subsequence of length $\lceil n^{\frac{1}{2^{d-1}}} \rceil$ can be determined easily : by the repeated application of a "folklore" algorithm for finding a monotone subsequence of length $\lceil \sqrt{n} \rceil$ from a sequence of n reals in time $O(n \log n)$ (see e.g. [31], [52]). It is also known that $\lceil n^{\frac{1}{2^{d-1}}} \rceil$ is the worst-case longest length of a monotone subsequence one can guarantee in $\mathbb{R}^d$ (this is mentioned e.g. in [48] but without a proof). In two dimensions this is shown by Erdős and Szekeres [25].

We show here an own construction of a set $P^* \subset \mathbb{R}^d$ of n points which has no monotone subsequence of length larger then $\lceil n^{\frac{1}{2^{d-1}}} \rceil$. We described this construction of the set P^* also in [37].

Construction of the set P^*

We build P^* such that

$$P^* = \left\{ \, p^i = (\sigma_1(i), \ldots, \sigma_d(i)) \mid i = 1, \ldots, n \, \right\} \, , \tag{3.39}$$

where $\sigma_j : \{1, \ldots, n\} \to \{1, \ldots, n\}$, $1 \leq j \leq d$, are appropriate permutations. $\sigma_j(i)$ is the j-th coordinate of the point p^i of P^*. Permutation σ_j defines the coordinates $p_j^1, \ldots, p_j^n$ in dimension j of the points $p^1, \ldots, p^n$ of P^*. We have $\{p_j^1, \ldots, p_j^n\} = \{1, \ldots, n\}$.

Let $\mathcal{O}_d = \{<, >\}^d$. Note that, by (3.39), any 2 points of P^* will be related with respect to some order in $\mathcal{O}_d$. Let $L_d = [o_1, o_2, \ldots, o_{2^{d-1}}]$ be a list of the 2^{d-1} orders $\{o \mid o \in \mathcal{O}_d \text{ and } o(1) = \, ' <'\}$, where the monotonicity with respect to the first coordinate is kept fixed to $' <'$. The rest of the orders in $\mathcal{O}_d$ can be obtained by inverting those in L_d. As an example the list L_3 is defined as $L_3 = [\, (<,<,<); (<,<,>); (<,>,<); (<,>,>) \,]$. Figure 3.7 shows the L_d-orders as the rows of the table T.

Note that it suffices to construct P^* such that it has the property that there cannot be more than $\lceil n^{\frac{1}{2^{d-1}}} \rceil$ points in any o_i-monotone subsequence of P^* for any $o_i \in L_d$, $i \in \{1, \ldots, 2^{d-1}\}$.

We consider first the case $n^{\frac{1}{2^{d-1}}} \in \mathbb{N}$. To define the above permutations σ_l, we consider the 2^{d-1}-dimensional grid-cube $G = \{1, \ldots, n^{\frac{1}{2^{d-1}}}\}^{2^{d-1}}$ of side length $n^{\frac{1}{2^{d-1}}}$:

$$G \; = \; \left\{ \, X = [x_1, x_2, \ldots, x_{2^{d-1}}] \; \middle| \; x_i \in \{1, \ldots, n^{\frac{1}{2^{d-1}}}\} \, \right\} \, . \tag{3.40}$$

Note that the grid cube G is a set of n elements. We will define d total orders $<_j$ ($1 \leq j \leq d$) on G. The order $<_j$ on G will define the permutation σ_j, where $1 \leq j \leq d$.

The first order $<_1$ is defined to be the lexicographic order of the elements of G. Let $X = [x_1, x_2, \ldots, x_{2^{d-1}}]$ and $Y = [y_1, y_2, \ldots, y_{2^{d-1}}]$ be two different grid-points of G, then:

$$X \; <_1 \; Y \iff \left(\, x_1 = y_1, \ldots, x_{i-1} = y_{i-1} \text{ and } x_i \; < \; y_i, \quad \text{for some } i \in \{1, \ldots, 2^{d-1}\} \, \right) \, .$$

	1 2 3 ...	**j**	**..., d**
$\mathbf{o_1}$	$(<, <, <, \ldots$	$o_1(j)$	$\ldots, <)$
$\mathbf{o_2}$	$(<, <, <, \ldots$	$o_2(j)$	$\ldots, >)$
$\vdots$		$\vdots$	
$\mathbf{o_i}$	$(<, >, <, \ldots$	$o_i(j)$	$\ldots, >)$
$\vdots$		$\vdots$	
$\mathbf{o_{2^d-1}}$	$(<, >, >, \ldots$	$o_{2^d-1}(j)$	$\ldots, >)$

Figure 3.7: Table T with the L_d-orders

Let $X_1 <_1 X_2 <_1 \ldots <_1 X_n$ be the lexicographically ordered elements of G. The order $<_1$ induces the identity permutation σ_1. Thus, in the first dimension we have:

$$p_1^1 = 1\ ,\ \ p_1^2 = 2\ ,\ \ ,\ldots,\ p_1^n = n\ ,$$

where p_1^i is the first coordinate of the point $p^i \in P^*$.

Each other order $<_j$, $j = 2, \ldots, d$, will produce another unique sequence of the elements $X_1, X_2, \ldots, X_n$ of G. It remains now to specify the order $<_j$, $j \in \{2, \ldots, d\}$. We define order $<_j$ by making use of the j-th column of the table T of the L_d-orders, which we illustrated in Figure 3.7. Let $X = [x_1, x_2, \ldots, x_{2^{d-1}}]$ and $Y = [y_1, y_2, \ldots, y_{2^{d-1}}]$ be two different grid-points of G, then $<_j$ is defined such that:

$$(X\ <_j\ Y) \Longleftrightarrow \left(\ x_1 = y_1, \ldots, x_{i-1} = y_{i-1},\ \text{and},\ x_i\ o_i(j)\ y_i,\ \text{for some } i \in \{1, \ldots, 2^{d-1}\}\ \right)\ . \tag{3.41}$$

Note that for any $X = [x_1, x_2, \ldots, x_{2^{d-1}}]$ and $Y = [y_1, y_2, \ldots, y_{2^{d-1}}]$ distinct elements of G we have $x_1 = y_1, \ldots, x_{i-1} = y_{i-1}$ and $x_i \neq y_i$ for some $i \in \{1, \ldots, 2^{d-1}\}$. The definition of the order $<_j$ in (3.41) means that:

- if $(x_i < y_i)$ and $(o_i(j) = '<')$ then $X <_j Y$,
- if $(x_i < y_i)$ and $(o_i(j) = '>')$ then $Y <_j X$.

Each order $<_j$, $j = 1, \ldots, d$, produces a unique sequence of the elements $X_1, X_2, \ldots, X_n$ of G. This unique sequence induces the permutation $\sigma_j : \{1, \ldots, n\} \to \{1, \ldots, n\}$ such that the following holds:

$$X_{\sigma_j(1)}\ <_j\ X_{\sigma_j(2)}\ <_j\ \ldots\ <_j\ X_{\sigma_j(i)}\ <_j\ \ldots\ <_j\ X_{\sigma_j(n)}. \tag{3.42}$$

Thus, in the j-th dimension we have

$$p_j^1 = \sigma_j(1)\ ,\ \ p_j^2 = \sigma_j(2)\ ,\ \ ,\ldots,\ p_j^n = \sigma_j(n)\ ,$$

where p_j^i is the j-th coordinate of the point $p^i \in P^*$, and σ_j is defined in (3.42).

By the construction of P^* we can define the bijection $\Phi : G \to P^*$ such that

$$\Phi(X) = (\ \Phi_1(X), \ldots, \Phi_j(X), \ldots, \Phi_d(X)\ ) \ \in P^*, \tag{3.43}$$

where $\Phi_j : G \to \{1, \ldots, n\}$ is a bijection such that

$$\Phi_j(X) = i \iff X \text{ is the } i\text{-th smallest element with respect to the order } <_j . \tag{3.44}$$

Properties of the set P^*

We show that there cannot be more than $n^{\frac{1}{2^{d-1}}}$ points in any o_i-monotone subsequence of P^* for any $o_i \in L_d$.

Consider some order $o_i \in L_d$ ($i \in \{1, \ldots, 2^{d-1}\}$). For any points $p, q \in P^*$ with $p\ o_i\ q$ let $X = [x_1, \ldots, x_{2^{d-1}}]$ and $Y = [y_1, \ldots, y_{2^{d-1}}]$ be their preimages with respect to Φ, which means $\Phi^{-1}(p) = X$ and $\Phi^{-1}(q) = Y$. There is some $k \in \{1, \ldots, 2^{d-1}\}$ such that $x_1 = y_1, \ldots, x_{k-1} = y_{k-1}$ and $x_k \neq y_k$. Without loss of generality let us assume that $x_k < y_k$. By (3.41),(3.44) and (3.44), the coordinates p_j and q_j ($1 \leq j \leq d$) of $p = (p_1, \ldots, p_d)$ and $q = (q_1, \ldots, q_d)$ fulfill the following for all $j \in \{1, \ldots, d\}$:

$$\begin{aligned} p_j < q_j \iff X <_j Y \iff o_k(j) =' <' \\ p_j > q_j \iff Y <_j X \iff o_k(j) =' >' \end{aligned}$$

Thus, $p\ o_k\ q$, and since there exists a unique order $o \in L_d$ with $p\ o\ q$, we have $k = i$, as we assumed $p\ o_i\ q$. This means, that if $p\ o_i\ q$, $p, q \in P^*$ then their preimages $X, Y \in G$ with respect to Φ have the property $x_i \neq y_i$. By the definition of G in (3.40) we have $x_i, y_i \in \{1, \ldots, n^{\frac{1}{2^{d-1}}}\}$. Thus, there cannot be more than $n^{\frac{1}{2^{d-1}}}$ points in any o_i-monotone sequence of points from P^*.

We conclude that P^* has no monotone subsequence of length larger then $n^{\frac{1}{2^{d-1}}}$. The general case works as follows. Let $m = \lceil n^{\frac{1}{2^{d-1}}} \rceil$. Thus, $n \leq m^{2^{d-1}}$ holds. Now let P_m be a set of $m^{2^{d-1}}$ points in $\mathbb{R}^d$ with longest monotone subsequence of length at most m, and which is constructed as discussed above. Take any subset $P^* \subset P_m$ of $n \leq m^{2^{d-1}}$ points. P^* has also no monotone subsequence of length larger then $m = \lceil n^{\frac{1}{2^{d-1}}} \rceil$.

3.2.4 Expected height and width of a random order

Let $\prec$ be the *dominance order*, which is defined such that $a = (a_1, \ldots, a_d) \ \prec \ b = (b_1, \ldots, b_d)$ if and only if $a_i \leq b_i$ for each $i = 1, \ldots, d$. All results mentioned in this section hold for any other partial order of $\mathcal{R}_d$.

Let $X = \{X_i : i = 1, \ldots, n\}$ be n points independently and uniformly distributed on the unit d-cube. The random variables of interest are $L_{n,d} = L_d(X_1, \ldots, X_n)$, the length of the longest chain in the set $(X, \prec)$, which is the *height* of the poset $(X, \prec)$, and $W_{n,d} = W_d(X_1, \ldots, X_n)$ the cardinality of the largest anti-chain in $(X, \prec)$, which is the *width* of the poset $(X, \prec)$. Determining the expected height and width of $(X, \prec)$ is equivalent to determining the expected height and width of a *random d-dimensional order*. Random d-dimensional orders were introduced by Winkler [66]. There are two useful and very natural equivalent definitions of a random d-dimensional order $P_d(n)$ [66], [16], [18].

1. The d-dimensional unit cube $[0,1]^d$ is equipped with the normal product measure and the componentwise order. The random order is defined by choosing n points uniformly, independently at random from $[0,1]^d$, and taking the order induced on them.

2. Consider all the $n!$ orders on the set $\{1,\ldots,n\}$. The random order $P_d(n)$ is defined by taking d of these uniformly, independently at random and forming their intersection.

The problem to determine the expected height $\mathrm{E}[L_{n,d}]$ turned out to be surprisingly difficult. Determining the height of $P_2(n)$ is known as Ulam's problem. Hammersley [36] showed that the expected length of a maximum increasing subsequence in a random permutation of $\{1,2,\ldots,n\}$ converges to $c\sqrt{n}$ with increasing n, for some constant c. A simple proof that $c \leq 2$ is given by Pilpel [56]. A review on the length of the longest increasing subsequence of n real numbers, which covers results on random and pseudo-random sequences is given in [62].

The problem of estimating $L_{n,d}$ for general d is dealt by Winkler [66], Bollobás and Winkler [17], and, Bollobás and Brightwell [16]. Bollobás and Winkler proved in [17] that the expected height $\mathrm{E}[L_{n,d}]$ of $P_d(n)$ converges to $c_d \cdot \sqrt[d]{n}$ with increasing n, where c_d is a constant depending on d. It is known [50, 63] that $c_2 = 2$ and that $\lim_{d\to\infty} c_d = e$. The strongest result [16] shows that there is a sharp concentration about the mean for $L_{n,d}$ with $d \geq 2$:

Theorem 3.4 (Bollobás and Brightwell '92). *For each integer $d \geq 2$, there is constant C_d such that, for n sufficiently large,*

$$Pr\left(|L_{n,d} - c_d n^{1/d}| > \frac{\lambda C_d n^{1/2d}\log^{3/2} n}{\log\log n}\right) \leq 80\lambda^2 e^{-\lambda^2}$$

for every λ with $2 < \lambda < \frac{n^{1/2d}}{\log\log n}$.

A consequence of this result is that the variance of $L_{n,d}$ is at most $n^{1/d}\log^2 n$, and another is that:

Theorem 3.5 ([16]). $|\ \mathbf{E}[L_{n,d}] - c_d\sqrt[d]{n}\ | \ \leq\ \sqrt[2d]{n}\log^{3/2} n\ .$

To precisely determine c_d with $d > 2$ is a challenging problem. Bounds are given by Bollobás and Winkler [17]:

Lemma 3.5 (Bollobás and Winkler '88). *For each $d \geq 2$:*

$$\frac{d^2}{(d!)^{1/d}\Gamma(1/d)} \leq c_d < e$$

The estimation of the width $W_{n,d}$ has been much less studied. Winkler [66] proved that the probability that $W_{n,d}$ is lying between $e^{-1}n^{1-1/d}$ and $n^{1-1/d}\log n$ tends to 1 as $n \to \infty$. Brightwell [18] improved the upper and lower bound:

Theorem 3.6 (Brightwell '92). *a) For each integer $d \geq 2$ and for n sufficiently large:*

$$E[W_{n,d}]\ \leq\ 4dn^{1-1/d}.$$

b) Let C be any constant less than $(1/e)\sqrt{6/\pi} \simeq 0.508$. There is a $d_0 = d_0(C)$ and $\alpha = \alpha(C)$ such that

$$E[W_{n,d}] \geq C\sqrt{d}n^{1-1/d}$$

whenever $d \geq d_0$ and $n \geq (\alpha d)^{d/2}$.

3.2.5 Searching by using a partition into monotone sequences

As already mentioned, a partition into sequences that are monotone with respect to all dimensions $\mathcal{D} = \{1, \ldots, d\}$ might be very large. We present an orthogonal range searching algorithm that uses a preprocessed partition of the point set P into monotone sequences, such that the sequences are monotone with respect to the dimensions of a fixed subset of the dimension set $\mathcal{D} = \{1, \ldots, d\}$.

This orthogonal range searching algorithm is called by the CUBE METHOD to compute for some cube $C_{q,\alpha}$, the set $C_{q,\alpha} \cap P$ of points. The searching algorithm works as follows. It considers a *suitable* subset $\mathcal{D}_1 \subset \mathcal{D}$ of dimensions. It determines the set $P_\alpha^{\mathcal{D}_1}$ of points p^i such that $p^i_j \in [q_j - \frac{\alpha}{2}, q_j + \frac{\alpha}{2}]$, for all $j \in \mathcal{D}_1$. The computation of $P_\alpha^{\mathcal{D}_1}$ is based on a preprocessed partition of P into sequences that are monotone with respect to the dimensions of $\mathcal{D}_1$. On each sequence we apply a logarithmic orthogonal range searching as described in Section 3.2.1. In the second phase the SCAN procedure is called with the set $P_\alpha^{\mathcal{D}_1}$ of points and the set $\mathcal{D} \setminus \mathcal{D}_1$ of dimensions, to finally determine the set $C_{q,\alpha} \cap P$ of points.

Which subset $\mathcal{D}_1$ of dimensions is *suitable* depends on the query point q and the side length α of the cube $C_{q,\alpha}$ and is decided during the query algorithm. The preprocessing consists of computing partitions of P into sequences that are monotone with respect to different candidates subsets for $\mathcal{D}_1$.

In the following we present the analysis of the data structure and the query algorithm for the situation when the sequences are monotone with respect to the dominance order, which we mentioned in Section 3.2.4.

The size $p = |\mathcal{D}_1|$ is a parameter of the analysis and we specify it later.

3.2.5.1 The preprocessing

The first phase of the preprocessing is to build a set $\mathcal{S}$ of $O(d)$ subsets $S \subset \mathcal{D}$, $|S| = p$, which are the candidate sets for $\mathcal{D}_1$. $\mathcal{S}$ should have the property that for any query point q and any side length α there exists a *suitable* subset $S \in \mathcal{S}$.

The second phase is to compute and store for each $S \in \mathcal{S}$ its partition $\mathcal{P}_S$ of the point set P into sequences which are monotone with respect to the dimensions in S. The total storage requirement of the data structure $\mathcal{P} = \{(S, \mathcal{P}_S) \mid S \in \mathcal{S}\}$ is $O(nd^2)$. All partitions $\mathcal{P}_S$ are determined with respect to the dominance order and with a total runtime of $O\left(dn^{\frac{5}{2}}/\sqrt{\log n}\right)$ by the algorithm mentioned in Section 3.2.2 [41, 6].

It remains to specify what is a *suitable* subset of dimensions and how to build the set $\mathcal{S}$.

Notation 3.1. *We denote by $\lambda(M)$, where $M \subseteq \{1, \ldots d\}$, the geometric mean of the side lengths s_j of $C_{q,\alpha} \cap [0,1]^d$ with respect to the dimensions of M, that is $\lambda(M) = \sqrt[d]{\prod_{j \in M} s_j}$.*

Definition 3.2. *Given are a fixed number $1 \le p < d$, a query point $q \in [0,1]^d$ and the side length α of the cube $C_{q,\alpha}$ with center q. A subset $\mathcal{D}_1 \subset \{1, \dots d\}$ with $p = |\mathcal{D}_1|$ is* suitable *for q and α if the following holds:*

$$\lambda(\mathcal{D}_1) \;:=\; \sqrt[p]{\prod_{j \in \mathcal{D}_1} s_j} \;\le\; \sqrt[d]{\prod_{j=1}^{d} s_j} \;=:\; \lambda \tag{3.45}$$

where $s_j = \min(q_j + \alpha/2, 1) - \max(q_j - \alpha/2, 0)$ is the side length of the box $C_{q,\alpha} \cap [0,1]^d$ in dimension j.

Observation 3.1. *Let $\beta(R)$ be the geometric mean of the elements of a finite set $R \subset \mathbb{R}_+$. Let $M, M_1, M_2 \subset \mathbb{R}_+$ be finite sets such that $M = M_1 \cup M_2$ and $M_1 \cap M_2 = \emptyset$. If $\beta(M) \le A$ and $\beta(M_1) \ge A$ for some $A \in \mathbb{R}_+$ then $\beta(M_2) \le A$.*

Construction of the set $\mathcal{S}$

If $\frac{d}{p} \in \mathbb{N}$ consider a partition of the set of dimensions $\mathcal{D} = \{1, \dots, d\}$ in $\frac{d}{p}$ subsets of p elements: $S_i = \{p(i-1)+1, \dots, pi\}$, $1 \le i \le \frac{d}{p}$ and let $\mathcal{S} = \{S_1, \dots, S_{d/p}\}$. Obviously, one of the sets S_i will be suitable in this case.

For the case $\frac{d}{p} \notin \mathbb{N}$ consider some partition $\mathcal{D} = S_1 \cup \ldots \cup S_m \cup R$ where $|S_i| = p$, $i = 1, \dots, m = \lfloor \frac{d}{p} \rfloor$ and $|R| = d \bmod p$. The subsets S_i, $i = 1 \dots, m$ are candidates for suitable subsets. Then, we take some partition of S_m, $S_m = R_1 \cup \ldots \cup R_{\lfloor \frac{p}{|R|} \rfloor} \cup H$ with $|R_i| = |R|$, and replace R_i ($i \in \{1, \dots, \lfloor \frac{p}{|R|} \rfloor\}$) by R in S_m and get another $\lfloor \frac{p}{|R|} \rfloor$ candidates for suitable subsets. Finally, we replace H in S_m by subsets $R' \subset R$ from the set $\mathcal{R}$ of candidates for suitable subsets of size $|H|$ in R, and get the rest candidate subsets in $\mathcal{S}$. See Figure 3.8 for illustration. Procedure SUIT_CAND computes the set $\mathcal{S}$. PART(M, r) returns a partition of a set M into $\lfloor \frac{|M|}{r} \rfloor$ sets of size r and one of size $|M| \bmod r$.

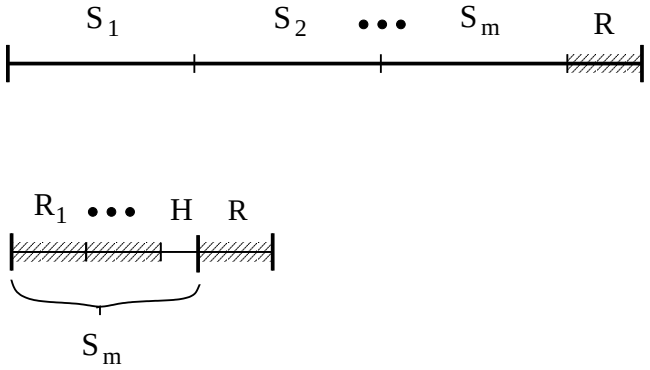


Figure 3.8: Building candidates for suitable subsets

SUIT_CAND$(\mathcal{D}, p)$

$(S_1, \ldots, S_m, R) := \text{PART}(\mathcal{D}, p);$ // $m = \lfloor \frac{d}{p} \rfloor$

$\mathcal{S} = \{S_1, \ldots, S_m\};$

if ($|R| > 0$) // $|R| = d \bmod p$

$\left(R_1, \ldots, R_{\lfloor \frac{p}{|R|} \rfloor}, H\right) := \text{PART}(S_m, |R|);$

foreach R_j ; // $j = 1, \ldots, \lfloor \frac{p}{|R|} \rfloor$

$\mathcal{S} := \mathcal{S} \cup \{(S_m \setminus R_j) \cup R\};$

if ($|H| > 0$) // $|H| = p \bmod |R|$

$\mathcal{R} := \text{SUIT\_CAND}(R, |H|);$

foreach $(R' \in \mathcal{R})$

$\mathcal{S} := \mathcal{S} \cup \{(S_m \setminus H) \cup R'\};$

return $\mathcal{S}$;

Lemma 3.6. SUIT_CAND$(\mathcal{D}, p)$ *computes a set $\mathcal{S}$, such that for each query point q and side length α, there exists a subset $S \subset \mathcal{D}$, $|S| = p$, such that S is an element of $\mathcal{S}$ and is suitable for q and α.*

Proof. Given are the query point q and the side length α. We prove by induction on p that SUIT_CAND$(\mathcal{D}^*, p)$ yields a subset $S^* \subseteq \mathcal{D}^*$ such that $\lambda(S^*) \leq \lambda(\mathcal{D}^*)$, for any set of dimensions $\mathcal{D}^* \subseteq \{1, \ldots, d\}$, $|\mathcal{D}^*| \geq p$.

For reasons of simplicity we develop the proof for the case $\mathcal{D}^* = \mathcal{D} = \{1, \ldots, d\}$. The following arguments hold if we replace $\mathcal{D}$ by some subset of the dimension set $\{1, \ldots, d\}$ of size larger or equal to p.

For $p = 1$, $\mathcal{D} = \bigcup_{i=1}^{d} S_i$ and $|S_i| = 1$. Obviously, there exists S_j such that $\lambda(S_j) \leq \lambda(\mathcal{D}) = \lambda$.

Now, consider some $p > 1$. If $\frac{d}{p} \in \mathbb{N}$ we have $\bigcup_{i=1}^{d/p} S_i = \mathcal{D}$. Thus, $\lambda(S_i) \leq \lambda$ for some i, otherwise $\lambda^d = (\lambda^p)^{\frac{d}{p}} < \prod_{i=1}^{d/p} \lambda(S_i) = \lambda^d$ which is a contradiction.

The case $\frac{d}{p} \notin \mathbb{N}$ is more involved. Suppose $\lambda(S_i) > \lambda$, $\forall i = 1, \ldots, m$, otherwise we are done. In this case, by Observation 3.1,

$$\lambda(R) \leq \lambda \text{ and } \lambda(S_m \cup R) \leq \lambda. \tag{3.46}$$

If $(S_m \setminus R_j) \cup R$ is suitable for some $j \in \{1, \ldots, \lfloor \frac{p}{|R|} \rfloor\}$ then we are done. Suppose, this is not the case. Then, by (3.46) $\lambda(R_j) \leq \lambda$, $\forall j \in \{1, \ldots, \lfloor \frac{p}{|R|} \rfloor\}$, which implies $\lambda(S_m \setminus H) \leq \lambda$. If $H = \emptyset$ then we get $\lambda(S_m) \leq \lambda$ which is a contradiction to our assumption $\lambda(S_i) > \lambda$, $\forall i = 1, \ldots, m$. If $|H| > 0$, since $|H| < p$ there exists by the induction assumption a set $R' \in \mathcal{R}$, $|R'| = |H|$ and $R' \subset R$ with $\lambda(R') \leq \lambda(R) \leq \lambda$, which is computed by SUIT_CAND$(\mathcal{R}, |H|)$ Thus, $\lambda((S_m \setminus H) \cup R') \leq \lambda$, i.e. $(S_m \setminus H) \cup R'$, which is a set of $\mathcal{S}$, is suitable.

□

We conclude the preprocessing specification with the following lemma:

Lemma 3.7. SUIT_CAND$(\mathcal{D}, p)$ *computes the set $\mathcal{S}$ of size $O(d)$ in time $O(d)$.*

Proof. Let denote this size of $\mathcal{S}$ by $A(d,p)$. Consider the following finite, strictly decreasing sequence of remainders $r_0 > r_1 > \ldots > r_{j+1}$, defined such that: $r_0 = (d \bmod p)$, $r_1 = (p \bmod r_0)$, $\ldots$, $r_{i+1} = (r_{i-1} \bmod r_i)$, where r_{j+1} is the first integer in the sequence which equals 0.

$$A(d,p) \;=\; \begin{cases} \lfloor \frac{d}{p} \rfloor & \text{if } r_0 = 0 \\ \lfloor \frac{d}{p} \rfloor + \lfloor \frac{p}{r_0} \rfloor & \text{if } r_1 = 0 \\ \lfloor \frac{d}{p} \rfloor + \lfloor \frac{p}{r_0} \rfloor + A(r_0, r_1) & \text{otherwise} \end{cases} \tag{3.47}$$

Notice that either $\left(p < \frac{d}{2} \;\Rightarrow r_0 < \frac{d}{2}\right)$ or $\left(p \geq \frac{d}{2} \;\Rightarrow\; d = p + r_0 \;\Rightarrow r_0 < \frac{d}{2}\right)$. Thus, $r_0 < \frac{d}{2}$. We denote $A(x,y) = 0$ if $xy = 0$, then by (3.47) we get $A(d,p) < d + A(r_0, r_1)$, where $r_0 < \frac{d}{2}$. Therefore, $A(d,p) < 2d$. The runtime complexity is $O(|\mathcal{S}|) = O(d)$.

□

3.2.5.2 The query algorithm

The query algorithm is the CUBE METHOD, which considers a cube $C_{q,\alpha}$ around the query point q and calls the searching procedure MONSEQ_SCAN to compute the set P_α of the points contained in the cube $C_{q,\alpha}$. Finally, the set P_α is checked by the brute-force method to determine the nearest neighbor to q.

The preprocessed data structure $\mathcal{P} = \{(S, \mathcal{P}_S) \mid S \in \mathcal{S}\}$, introduced in Section 3.2.5.1, is used by the searching procedure MONSEQ_SCAN as follows. For the given query point q and the side length α, a suitable subset $\mathcal{D}_1$ of dimensions is determined from the preprocessed set $\mathcal{S}$ of candidate subsets. The preprocessed partition $\mathcal{P}_{\mathcal{D}_1}$ of the data set P in sequences monotone with respect to the dimensions in $\mathcal{D}_1$ is used to determine the set $P_\alpha^{\mathcal{D}_1}$ of points p^i such that $p^i_j \in [q_j - \frac{\alpha}{2}, q_j + \frac{\alpha}{2}]$, for all $j \in \mathcal{D}_1$. The procedure SCAN, introduced in Section 2.1.2, is called with the dimension set $\mathcal{D} \setminus \mathcal{D}_1$ and the point set $P_\alpha^{\mathcal{D}_1}$, and it has as result the set P_α.

The following gives a schematic description of the procedure MONSEQ_SCAN.

MONSEQ_SCAN$(\alpha, q, \mathcal{P})$

$(\mathcal{D}_1, \mathcal{D}_2)$:= DIM_PARTITION$(q, \mathcal{S})$; (PARTITION)
$P_\alpha^{\mathcal{D}_1}$:= MON_SEQ$(\alpha, q, \mathcal{P}_{\mathcal{D}_1}, \mathcal{D}_1)$; (MONOTONE)
P_α := SCAN$(\alpha, q, P_\alpha^{\mathcal{D}_1}, \mathcal{D}_2)$; (SCAN)
return P_α ;

where procedure DIM_PARTITION finds a suitable subset $\mathcal{D}_1$ and procedure MON_SEQ determines the point set $P_\alpha^{\mathcal{D}_1}$. These procedures are described below.

Procedure DIM_PARTITION finds a suitable subset $\mathcal{D}_1$ by checking subsets $S \in \mathcal{S}$ for suitability. The first found suitable S is returned as the subset $\mathcal{D}_1$. Lemma 3.6 implies the existence of a suitable subset $\mathcal{D}_1 \in \mathcal{S}$. The result of DIM_PARTITION is the set $\mathcal{D}_1$ and the dimension set $\mathcal{D} \setminus \mathcal{D}_1$, sorted into a list $\mathcal{D}_2$. The order of the dimensions in the list $\mathcal{D}_2$ is chosen to correspond to the increasing order of the side lengths $\{s_j \mid j \in \mathcal{D} \setminus \mathcal{D}_1\}$ of the box $C_{q,\alpha} \cap [0,1]^d$, since the searching algorithm SCAN works best by checking the dimensions of the points in this order.

DIM_PARTITION$(q, \mathcal{S})$

for $S \in \mathcal{S}$
    if (S is suitable)
    then break for-loop;
$\mathcal{D}_1 := S$;
$\mathcal{D}_2$:=SORT_SIDE($q, \mathcal{D} \setminus \mathcal{D}_1$);
return $(\mathcal{D}_1, \mathcal{D}_2)$;

To determine the point set $P_\alpha^{\mathcal{D}_1}$, procedure MON_SEQ($\alpha, q, \mathcal{P}_{\mathcal{D}_1}, \mathcal{D}_1$) performs for each monotone sequence of $\mathcal{P}_{\mathcal{D}_1}$ a binary searching in the orthogonal range $\mathcal{I}_1^{\mathcal{D}_1} \times \ldots \times \mathcal{I}_d^{\mathcal{D}_1}$, where

$$\mathcal{I}_l^{\mathcal{D}_1} \;=\; \begin{cases} [q_l - \frac{\alpha}{2}, q_l + \frac{\alpha}{2}] & \text{if } l \in \mathcal{D}_1 \\ [0,1] & \text{otherwise} \end{cases}$$

and restricted to the dimensions of $\mathcal{D}_1$. The logarithmic orthogonal range searching is performed for each monotone sequence as described in Section 3.2.1.

3.2.5.3 Analysis of the expected runtime of the query algorithm

We measure the running time T_{mon_scan} of the procedure MONSEQ_SCAN by the number of comparisons and arithmetic operations.

The runtime of DIM_PARTITION is measured by the number of multiplications, divisions and comparisons. The parameter p, which is the size of a suitable set, will be specified at the end of the analysis.

To determine whether a set $S \in \mathcal{S}$ is suitable, DIM_PARTITION computes the geometric mean $\lambda(S)$ of the side lengths s_j of the box $W_{q,\alpha} = C_{q,\alpha} \cap [0,1]^d$ with $j \in S$. The side lengths of the box $W_{q,\alpha}$ depend only on the query point q and can be computed in $O(d)$ time. Let λ be the geometric mean of all side lengths. For each tested $S \in \mathcal{S}$ the product $(\lambda(S))^p$ is computed and compared with $\lambda^p = (\varphi/n)^{\frac{p}{d}}$, which can be computed once in $O(d)$ time. The products $(\lambda(S_i))^p$ for the sets S_i can be computed with a total number of $O(d)$ multiplications. We recall the sequence of remainders introduced in Lemma 3.7. DIM_PARTITION can be implemented such that the products of the sets $(S_m \setminus R_j) \cup R$ cost $\frac{p}{r_0} \cdot r_0 = p$ multiplications and additionally one division and one multiplication. Thus, for some small constants $c_1, c_2 > 0$ the costs $C(d,p)$ of DIM_PARTITION to determine a suitable set $\mathcal{D}_1$ can be bounded as follows:

$$C(d,p) \;\le\; c_1 \cdot (d+p) + c_2 \cdot C(r_0, r_1) \;\le\; c \cdot (d + p + r_0 + \ldots + r_j),$$

where $c = \max\{c_1, c_2\}$. Since $r_i < \frac{d}{2 \cdot 2^{\lfloor i/2 \rfloor}}$ we get $C(d,p) \le 4 \cdot c \cdot d = O(d)$.

Let $s_1, \ldots s_p, s_{p+1}, \ldots, s_d$ be the side lengths of the box $C_{q,\alpha} \cap [0,1]^d$ such that $s_{p+1} \le \ldots \le s_d$ are in increasing order. The order $s_{p+1} \le \ldots \le s_d$ can be computed in $O(d \log d)$ time. The corresponding order of dimensions is stored in $\mathcal{D}_2$.

Altogether, the total runtime of DIM_PARTITION is $O(d \log d)$.

The runtime of the MON_SEQ and SCAN parts is proportional to the number of performed comparisons. Let T_{mon} and T_{scan} be the discrete random variables for the number of comparisons performed by the MON_SEQ procedure in the *first phase* and by the SCAN procedure in the *second phase* of the procedure MONSEQ_SCAN, respectively.

We first estimate $\mathrm{E}[T_{mon}]$ and $\mathrm{E}[T_{scan}]$ in terms of the parameter p. Next we show how to choose a suitable parameter p and summarize the analysis of the query time.

Expected runtime of the first phase

Procedure MON_SEQ performs a logarithmic orthogonal range searching for each monotone sequence of the partition $\mathcal{P}_{\mathcal{D}_1}$ of the point set P into sequences that are monotone with respect to the dimensions of $\mathcal{D}_1$. Let M be the discrete random variable for the size of $\mathcal{P}_{\mathcal{D}_1}$. By construction, the partition $\mathcal{P}_{\mathcal{D}_1}$ is a minimal chain decomposition of $(P, \prec_{\mathcal{D}_1})$, where the order $\prec_{\mathcal{D}_1}$ denotes the dominance order on P with respect to the dimension set $\mathcal{D}_1$. We have

$$\mathrm{E}[T_{mon}] = \sum_m \mathrm{E}[T_{mon} | M = m] \cdot \Pr[M = m] \, .$$

The expected value $\mathrm{E}[T_{mon} | M = m]$ is bounded by

$$\mathrm{E}[T_{mon} | M = m] \leq c_O \cdot E \left[\, p \cdot \log X_1 + p \cdot \log X_2 + \ldots + p \cdot \log X_M \mid M = m \, \right] \, ,$$

where c_O is a constant and X_i, $i = 1, \ldots, M$ are the lengths of the sequences which build the partition $\mathcal{P}_{\mathcal{D}_1}$ of size M. We have $X_1 + X_2 + \ldots + X_M = n$. By concavity of the logarithm-function we obtain

$$\log X_1 + \log X_2 + \ldots + \log X_M \;\leq\; M \cdot \log \left(\frac{X_1 + \ldots + X_M}{M} \right) = M \cdot \log n - M \cdot \log M \, ,$$

which implies:

$$\begin{aligned} \mathrm{E}[T_{mon}] \;&\leq\; c_O \cdot \sum_m \left(\, pm \log n - pm \log m \, \right) \cdot \Pr[M = m] \\ &=\; c_O \cdot \left(\, p \cdot \log n \cdot \mathrm{E}[M] - p \cdot \mathrm{E}[M \log M] \, \right) . \end{aligned}$$

The function $u(x) := x \log x$ is convex for $x > 0$. Thus, we obtain $\mathrm{E}[M \log M] = \mathrm{E}[u(M)] \geq u\left(\mathrm{E}[M]\right) = \mathrm{E}[M] \cdot \log\left(\mathrm{E}[M]\right)$, by Jensen's inequality (see [34], page 161). This implies an upper bound on $\mathrm{E}[T_{mon}]$:

$$\mathrm{E}[T_{mon}] \leq c_O \cdot p \cdot \mathrm{E}[M] \cdot \left(\, \log n - \log\left(\mathrm{E}[M]\right) \, \right) . \tag{3.48}$$

The following lemma provides a lower bound on $\mathrm{E}[M]$.

Lemma 3.8. *We have*

$$E[M] \;\geq\; \frac{n}{e \sqrt[p]{n} + \sqrt[2p]{n} \log^{3/2} n} \, ,$$

where M is the size of the partition $\mathcal{P}_{\mathcal{D}_1}$.

Proof. The size M of the partition $\mathcal{P}_{\mathcal{D}_1}$ equals the width of the poset $(P, \prec_{\mathcal{D}_1})$, where $\prec_{\mathcal{D}_1}$ is the dominance order on P with respect to the dimension set $\mathcal{D}_1$. Let H be the discrete random variable for the height of $(P, \prec_{\mathcal{D}_1})$.

Since, in any partial order, the height times the width is at least the number of elements, we have $M \cdot H \geq n$. Since $M > 0$ and $H > 0$, we can apply the Cauchy-Schwarz inequality and obtain:

$$\mathrm{E}[M] \cdot \mathrm{E}[H] \;\geq\; \left(\mathrm{E}[\, \sqrt{MH} \,] \right)^2 \;\geq\; n \, .$$

Since the points of P are drawn independently at random, Theorem 3.5 and Lemma 3.5 imply

$$\mathrm{E}[M] \;\geq\; \frac{n}{\mathrm{E}[H]} \;\geq\; \frac{n}{e\sqrt[p]{n} + \sqrt[2p]{n}\log^{3/2} n}\,.$$

□

We are now prepared to bound the expected runtime of the MON_SEQ procedure.

Lemma 3.9. *For n sufficiently large:*

$$E[T_{mon}] = O\left(pn^{1-1/p}\log n + p^2 n^{1-1/p}\log\log n\right)\,.$$

Proof. Lemma 3.8 provides

$$\begin{aligned}\log\left(\mathrm{E}[M]\right) &\geq \log n - \log\left(e\sqrt[p]{n}\right) - \log\left(\sqrt[2p]{n}\log^{3/2} n\right)\\ &= \log n - 1 - \frac{3}{2}\cdot\frac{\log n}{p} - \frac{3}{2}\cdot\log\log n\,.\end{aligned}$$

This implies together with (3.48):

$$\mathrm{E}[T_{mon}] = O\left(\,\mathrm{E}[M]\log n + p\mathrm{E}[M]\log\log n\,\right)\,.$$

By Theorem 3.6, we obtain:

$$\mathrm{E}[T_{mon}] \;\leq\; O\left(\,pn^{1-1/p}\log n + p^2 n^{1-1/p}\log\log n\,\right)\,,$$

since the points of P are drawn independently at random and since $\mathcal{P}_{\mathcal{D}_1}$ is a partition of P into sequences which are monotone with respect to p of the dimensions $\{1,\ldots,d\}$. □

Expected runtime of the second phase

We measure the running time T_{scan} of the scanning part of the algorithm by the number of comparisons of coordinates.

Lemma 3.10. *For n sufficiently large and $\varphi \leq \frac{n}{e}$ we obtain*

$$E[T_{scan}] \;\leq\; n^{1-\frac{p}{d}}\cdot\varphi^{\frac{p}{d}}\cdot\left(\frac{d}{\ln(n/\varphi)}+1\right)$$

Proof. $\mathrm{E}[T_{scan}]$ is given by:

$$\mathrm{E}[T_{scan}] \;=\; n\cdot\prod_{i=1}^{p} s_i\cdot\left(\,1 + s_{p+1} + s_{p+1}\cdot s_{p+2} + \ldots + \prod_{i=p+1}^{d-1} s_i\,\right)\,,$$

where $s_1,\ldots,s_p$ are the side lengths of the box $C_{q,\alpha}\cap[0,1]^d$ with respect to the dimensions of $\mathcal{D}_1$ and $s_{p+1}\leq\ldots\leq s_d$ are the side lengths corresponding to the list $\mathcal{D}_2$ of dimensions. The set $\mathcal{D}_1$ was chosen such that $\beta := \lambda(\mathcal{D}_1) \leq \lambda \leq \lambda(\mathcal{D}_2) := \gamma$ where:

$$\lambda = \sqrt[d]{\prod_{i=1}^{d} s_i} \;=\; \sqrt[d]{\frac{\varphi}{n}}\,,\qquad \beta := \lambda(\mathcal{D}_1) = \sqrt[p]{\prod_{i=1}^{p} s_i}\,,\qquad \gamma := \lambda(\mathcal{D}_2) = \sqrt[d-p]{\prod_{i=p+1}^{d} s_i}$$

Making use of the increasing order of the side lengths $s_{p+1} \leq \ldots \leq s_d$ we get:

$$\begin{aligned} \mathrm{E}[T_{scan}] &\leq n \cdot \beta^p \cdot \left(1 + \gamma + \gamma^2 + \ldots + \gamma^{d-p-1}\right) \\ &\leq n \cdot \beta^p \cdot \min\left\{d-p, \frac{1}{1-\gamma}\right\} \leq n \cdot \frac{\lambda^d}{\gamma^{d-p}(1-\gamma)} \end{aligned} \tag{3.49}$$

It would be convenient if $\frac{1}{\gamma^{d-p}(1-\gamma)} \leq \frac{1}{\lambda^{d-p}(1-\lambda)}$. Consider the function $h\colon (0,1) \to \mathbb{R}$, $h(x) = \frac{1}{x^{d-p}(1-x)}$. Obviously, h is monotone decreasing on $(0, 1 - \frac{1}{d-p+1}]$ and monotone increasing on $[1 - \frac{1}{d-p+1}, 1)$. Since $\lambda \leq \gamma$ it remains to compare γ with $1 - \frac{1}{d-p+1}$.

Case $\gamma^{d-p} \leq \frac{1}{e}$. We have $\left(1 + \frac{1}{d-p}\right)^{d-p} \leq e \leq \frac{1}{\gamma^{d-p}}$ which implies $\gamma \leq 1 - \frac{1}{d-p+1}$. Thus,

$$\begin{aligned} \mathrm{E}[T_{scan}] &\leq n \cdot \frac{\lambda^d}{\gamma^{d-p}(1-\gamma)} \leq n \cdot \frac{\lambda^d}{\lambda^{d-p}(1-\lambda)} \\ &= n \cdot \frac{\lambda^p}{1-\lambda} \leq n \cdot \left(\frac{\varphi}{n}\right)^{\frac{p}{d}} \cdot \left(\frac{d}{\ln(n/\varphi)} + 1\right) \end{aligned} \tag{3.50}$$

For the last inequality we used Lemma 2.2.

Case $\gamma^{d-p} > \frac{1}{e}$. Since $\gamma^{d-p} \cdot \beta^p = \lambda^d$ we get $\beta^p < e\lambda^d$. By (3.49):

$$\mathrm{E}[T_{scan}] \leq n \cdot \beta^p \cdot (d-p) < n \cdot e \cdot \lambda^d \cdot (d-p) = e \cdot \varphi \cdot (d-p) \tag{3.51}$$

The following holds:

$$e \cdot \varphi \cdot (d-p) \leq n^{1-\frac{p}{d}} \cdot \varphi^{\frac{p}{d}} \cdot \frac{d}{\ln(n/\varphi)} \iff e \cdot \left(1 - \frac{p}{d}\right) \cdot \ln(n/\varphi) \leq \left(\frac{n}{\varphi}\right)^{1-\frac{p}{d}} \tag{3.52}$$

The last inequality holds since $\frac{n}{\varphi} \geq e$ and since $e \cdot \ln x \leq x$ for all $x \geq e$.

Summarizing, we get by (3.50), (3.51) and (3.52):

$$\mathrm{E}[T_{scan}] \leq n^{1-\frac{p}{d}} \cdot \varphi^{\frac{p}{d}} \cdot \left(\frac{d}{\ln(n/\varphi)} + 1\right)$$

□

The choice of the parameter p

It remains to choose the parameter p of the data structure $\mathcal{P} = \{(S, \mathcal{P}_S) \mid S \in \mathcal{S}\}$ introduced at the beginning of Section 3.2.5.

For n sufficiently large the expected runtime of procedure MONSEQ_SCAN is bounded by:

$$\mathrm{E}[T_{mon\_scan}] \leq C_1 \cdot f_1(p) + C_2 \cdot f_2(p)$$

for constants C_1 and C_2 and functions $f_1(p)$ and $f_2(p)$ defined as follows

$$\begin{aligned} f_1(p) &= p n^{1-\frac{1}{p}} \log n + p^2 n^{1-\frac{1}{p}} \log\log n \\ f_2(p) &= n^{1-\frac{p}{d}} \cdot \varphi^{p/d} \cdot \left(\frac{d}{\ln(n/\varphi)} + 1\right) \end{aligned}$$

$f_1(p)$ is monotone increasing in p and $f_2(p)$ is monotone decreasing in p. Additionally, $f_1(1) < f_2(1)$ and $f_1(d) > f_2(d)$. A suitable p for the asymptotic expected runtime $\mathrm{E}[T_{mon\_scan}]$ is the value p^* such that $f_1(p^*) = f_2(p^*)$. We do not determine p^* exactly. In the following we give a lower and an upper bound on p^* as functions of n, d and φ. The following holds:

$$f_1(p) = f_2(p) \iff \log\left(f_1(p)\right) = \log\left(f_2(p)\right) \tag{3.53}$$
$$\iff p^2 \log(n/\varphi) + p \cdot A(p) - d \log n = 0\,, \tag{3.54}$$

where $A(p) = d \cdot \left(\log(p \log n) + \log\left(p^2 \log(\log n)\right) - \log\left(\frac{d}{\ln(n/\varphi)} + 1\right)\right)$.

(3.54) implies:

$$p^* \;=\; \frac{-A(p^*) + \sqrt{A^2(p^*) + 4d \log n \log(n/\varphi)}}{2\log(n/\varphi)} \;<\; \sqrt{d} \cdot \sqrt{\frac{\log n}{\log(n/\varphi)}} \tag{3.55}$$

It can be verified that for the case $d < \log(n/\varphi)$ the lower bound for p^* holds: $\frac{\sqrt{d}}{2} \cdot \sqrt{\frac{\log n}{\log(n/\varphi)}} < p^*$.

We choose $p = \sqrt{d}$ which is, by above observations, for $\varphi = o(n)$ a good approximation of p^*. We consider the following cases:

CASE $d \leq \left(\frac{\ln n}{\ln \ln n}\right)^2$:

We have in this case

$$\mathrm{E}[T_{mon\_scan}] \;\leq\; 2C_1 \cdot \sqrt{d} n^{1-\frac{1}{\sqrt{d}}} \log n + C_2 \cdot n^{1-\frac{1}{\sqrt{d}}} \cdot \varphi^{\frac{1}{\sqrt{d}}} \cdot \left(\frac{d}{\ln(n/\varphi)} + 1\right)$$

The expected runtime of the CUBE METHOD with the searching procedure MONSEQ_SCAN is given by

$$\mathrm{E}[T_{cube}] \leq c_M \cdot \mathrm{E}[T_{mon\_scan}] + e^{-\varphi} \cdot c_B \cdot nd + c_B \cdot \varphi d + c_L \cdot d \ln d\;, \tag{3.56}$$

where c_M, c_B, c_L are appropriate constants. The expected number φ of points in the cube $C_{q,\alpha}$ is contained in the interval $[\varphi^*, \varphi^* + 1)$, where φ^* is the parameter of the procedure SIDE_LENGTH. It is sufficient to guarantee that φ fulfills

$$e^{-\varphi} nd \leq \sqrt{d} n^{1-\frac{1}{\sqrt{d}}} \log n$$

which is satisfied if we choose $\varphi^* \geq \frac{\ln n}{\sqrt{d}}$, since $d \leq \ln^2 n$ in this case. Since the upper bound on $\mathrm{E}[T_{mon\_scan}]$ and φd are monotone increasing in φ, we choose $\varphi^* = \frac{\ln n}{\sqrt{d}}$.

Altogether, we obtain $\mathrm{E}[T_{cube}] = O\left(\sqrt{d} \cdot n^{1-\frac{1}{\sqrt{d}}} \cdot \ln n\right)$ in this case.

CASE $d > \left(\frac{\ln n}{\ln \ln n}\right)^2$:

We obtain

$$\mathrm{E}[T_{mon\_scan}] \;\leq\; 2C_1 \cdot d n^{1-\frac{1}{\sqrt{d}}} \log\log n + 2C_2 \cdot n^{1-\frac{1}{\sqrt{d}}} \cdot \varphi^{\frac{1}{\sqrt{d}}} \cdot \frac{d}{\ln(n/\varphi)}\;.$$

The expected runtime of the CUBE METHOD with the searching procedure MONSEQ_SCAN is given in (3.56). We guarantee

$$e^{-\varphi} nd \;\leq\; dn^{1-\frac{1}{\sqrt{d}}} \log\log n + n^{1-\frac{1}{\sqrt{d}}} \cdot \varphi^{\frac{1}{\sqrt{d}}} \cdot \frac{d}{\ln(n/\varphi)}$$

which is satisfied if $\varphi \geq \frac{\ln n}{\sqrt{d}} + \ln\ln n$. Argumenting as in the previous case, a suitable value of φ^* is $\frac{\ln n}{\sqrt{d}} + \ln\ln n$. Thus, we obtain $\mathrm{E}[T_{cube}] = O\left(d \cdot n^{1-\frac{1}{\sqrt{d}}} \cdot \ln\ln n\right)$.

Theorem 3.7. *Let P be a set of n points. The* CUBE METHOD *with the searching procedure* MONSEQ_SCAN *finds the nearest neighbor from P to the query point q*

A) if $d \leq \left(\frac{\ln n}{\ln\ln n}\right)^2$ with an expected asymptotic runtime of $O\left(\sqrt{d} \cdot n^{1-\frac{1}{\sqrt{d}}} \cdot \ln n\right)$

B) if $d > \left(\frac{\ln n}{\ln\ln n}\right)^2$ with an expected asymptotic runtime of $O\left(d \cdot n^{1-\frac{1}{\sqrt{d}}} \cdot \ln\ln n\right)$

if the points of P are drawn independently at random from $[0,1]^d$ under uniform distribution. The search procedure uses a data structure of size $O(nd^2)$, which is build during the preprocessing in $O\left(dn^{\frac{5}{2}}/\sqrt{\log n}\right)$ time.

Remark 3.2. The MONSEQ_SCAN searching procedure is competing with the REJECT_SCAN searching procedure. If $d \leq \ln n$ the MONSEQ_SCAN procedure dominates: the speedup factor by which MONSEQ_SCAN is faster than REJECT_SCAN, is $\Omega\left(\frac{n^{1/\sqrt{d}}}{\sqrt{d}\cdot\ln n}\right)$. Note that if $d \leq \ln n$ then $\lim_{n\to\infty} \frac{n^{1/\sqrt{d}}}{\sqrt{d}\cdot\ln n} = \infty$.