Chapter 4

Extensions of the CUBE METHOD

We present several generalizations of the CUBE METHOD. In section 4.1 we analyze the query algorithm GROWING-CUBE. The difference to the CUBE METHOD occurs in the case when the cube $C_{q,\alpha}$ is empty : instead of calling the brute-force method, the GROWING-CUBE enlarges the cube with center q until it contains data points. In Section 4.2 we extend the algorithm GROWING-CUBE to compute the k nearest neighbors to some query point q from the data set P. Next we generalize in Section 4.3 the expected runtime analysis of our algorithms to other "well-behaved" probability distributions. In Section 4.4 we develop extensions of our algorithms which work efficiently in the external-memory model of computation introduced in [2]. Section 4.5 shows an experimental comparison of implementations of the presented variants of the CUBE METHOD. We emphasize the factors by which the algorithms are faster than the brute-force method.

A part of the results presented in this chapter has been published in [39].

4.1 The GROWING-CUBE variant

If the cube $C_{q,\alpha}$ considered by the CUBE METHOD is empty, the alternative to determining the nearest neighbor by brute-force is to consider a larger cube with center q and determine the points contained in it. The size of the current cube should be increased as long as it contains no points of P. This searching algorithm is called the GROWING-CUBE method.

The first cube C_{q,α_1} around q contains an expected number φ_1 of points. φ_t is the expected number of points which are contained in the cube C_{q,α_t}, considered in the t-th iteration, i.e. $\mathrm{E}\,[|C_{q,\alpha_t} \cap P|] = \varphi_t$. The side length α_t of this cube C_{q,α_t} is determined by the procedure SIDE_LENGTH, as described in Section 2.1.2. SIDE_LENGTH gets a sequence of parameters $1 \leq \varphi = \varphi_1^* < \varphi_2^* < \ldots < \varphi_t^* < \ldots < \varphi_{t_{\max}}^*$ such that

$$\min\{\varphi_{t-1}^* + 1, n\} \;\leq\; \varphi_t^* \;\leq\; \max\{t\varphi, n\}\,. \tag{4.1}$$

A more exact setting will be specified later. The side lengths α_t are determined such that

$$\varphi_t^* \;\leq\; \mathrm{E}\,[\,|C_{q,\alpha_t} \cap P|\,] = \varphi_t \;< \varphi_t^* + 1\,. \tag{4.2}$$

The sequences $\{\varphi_t^*\}_{t\geq 1}$ and $\{\varphi_t\}_{t\geq 1}$ are strictly monotone increasing. The termination of the algorithm is ensured and we define $t_{\max} = \min\{t \mid \varphi_t^* = n\}$. The following gives a schematic description of the algorithm.

GROWING_CUBE $(\mathbf{q}, \mathbf{P})$

$t := 0$;
repeat
t:=t+1;
if ($\varphi_t^* < n$) then
$(\mathcal{D}, \alpha_t)$:= SIDE_LENGTH(φ_t^*, q);
P_{α_t} := SEARCH_CUBE($\alpha_t, q, P, \mathcal{D}$);
else $P_{\alpha_t} := P$;
until ($|P_{\alpha_t}| > 0$);
$\hat{p}$:= BRUTE_FORCE(P_{α_t});
return $\hat{p}$;

SEARCH_CUBE stands for one of the orthogonal range searching procedures SCAN, REJECT and SCAN_REJECT. Procedure SIDE_LENGTH was introduced in Section 2.1.2. It computes the side lengths of the cubes C_{q,α_t}. The order $\mathcal{D}$ of dimensions corresponds to the increasing order of the side lengths of a box $C_{q,\alpha_t} \cap [0,1]^d$. This order depends only on the query point and is also computed by the SIDE_LENGTH procedure, but just only once in the first iteration in $O(d \log d)$ time.

We measure the running time T_{grow} of the GROWING-CUBE method by the number of performed comparisons and arithmetic operations.

For the expected runtime analysis we introduce the discrete random variable X^t that represents the number of points of P contained in the cube C_{q,α_t}. To enable compactness of formulas, we define $X^0 = \alpha_0 = \varphi_0 = 0$. Let S^t be the discrete random variable representing the number of tests required by an independent call of SEARCH_CUBE(α_t, q, P) to determine the points of P contained in the cube C_{q,α_t}.

The expected runtime of the GROWING-CUBE method is given by:

$$\mathrm{E}[\, T_{grow} \,] \;=\; \mathrm{E}[\, T_{side} \,] \;+\; \mathrm{E}[\, T_{search} \,] \;+\; \mathrm{E}[\, T_{brute} \,]$$

where $\mathrm{E}[\, T_{search} \,]$ is the expected runtime of the SEARCH_CUBE procedure:

$$\mathrm{E}[\, T_{search} \,] \;=\; \sum_{t=1}^{t_{\max}-1} \Pr\left(X^{t-1} = 0\right) \cdot c_S \cdot \mathrm{E}[\, S^t \mid X^{t-1} = 0 \,]\,, \tag{4.3}$$

where c_S is the corresponding constant of the SEARCH_CUBE procedure.

$\mathrm{E}[\, T_{brute} \,]$ is the expected runtime of the brute-force part of the GROWING-CUBE method:

$$\mathrm{E}[\, T_{brute} \,] \;=\; \sum_{t=1}^{t_{\max}} \Pr\left(\, X^{t-1} = 0 \,\cap\, X^t > 0 \,\right) \cdot d \cdot c_B \cdot \mathrm{E}[\, X^t \mid X^{t-1} = 0, X^t > 0 \,]\,, \tag{4.4}$$

where c_B is the corresponding constant of the brute-force method.

The expected runtime $\mathrm{E}[\, T_{side} \,]$, required to compute the side lengths of the cubes C_{q,α_t} and the order $\mathcal{D}$ of dimensions, is given by:

$$\mathrm{E}[\, T_{side} \,] = \sum_{t=1}^{t_{\max}-1} \Pr\left(X^{t-1} = 0\right) \cdot c_L \cdot d \cdot \log(d\varphi_t) \tag{4.5}$$

where c_L is the constant of the SIDE_LENGTH procedure.

In the following we first give bounds on E[T_{side}] and E[T_{brute}]. Next we present the expected runtime analysis of the SEARCH_CUBE part for each of the 3 searching strategies : SCAN, REJECT and SCAN_REJECT.

The analysis uses a bound on the probability that the cube C_{q,α_t} is empty:

$$\Pr\left(X^t = 0\right) \;=\; \Pr(\; C_{q,\alpha_t} \text{ is empty }) \;=\; \left(1 - \frac{\varphi_t}{n}\right)^n \leq\; e^{-\varphi_t} \;. \tag{4.6}$$

We used $\Pr(\; p \in C_{q,\alpha_t} \;) = \frac{\varphi_t}{n}$ for some random point $p \in [0,1]^d$.

The expected runtime to determine the side lengths of the cubes can be bounded by using (4.6), $\varphi_t \leq t\varphi + 1$ for $1 \leq t < t_{\max}$, $\log(t\varphi + 1) \leq \log\varphi + t$ for $t \geq 1$ and (4.5):

$$\begin{aligned}
\mathrm{E}[\; T_{side} \;] \;&\leq\; c_L \cdot d \cdot \log(\varphi d) \cdot \sum_{t=0}^{t_{\max}-2} \left(1 - \frac{\varphi_t}{n}\right)^n \;+\; c_L \cdot d \cdot \sum_{t=0}^{t_{\max}-2} (t+1) \cdot \left(1 - \frac{\varphi_t}{n}\right)^n \\
&\leq\; c_L \cdot d \cdot \log(\varphi d) \cdot \sum_{t\geq 0} e^{-\varphi_t} \;+\; c_L \cdot d \cdot \sum_{t\geq 0} (t+1) \cdot e^{-\varphi_t} \;.
\end{aligned} \tag{4.7}$$

Let $X_i^t \in \{0,1\}$ with $i = 1, \ldots, n$ be discrete random variables such that $(X_i^t = 1) \iff$ (point $p^i \in C_{q,\alpha_t}$). We use $X^t = \sum_{i=1}^n X_i^t$ and $\Pr\left(X_i^t = 1\right) = \frac{\varphi_t}{n}$. We bound the expected runtime of the brute-force part as follows:

$$\begin{aligned}
\mathrm{E}[\; T_{brute} \;] \;&=\; c_B \cdot \sum_{t=1}^{t_{\max}} \Pr\left(\; X^{t-1} = 0 \;\cap\; X^t > 0 \;\right) \cdot d \cdot \sum_{i=1}^{n} \mathrm{E}[\; X_i^t \mid X^{t-1} = 0, X^t > 0 \;] \\
&=\; c_B \cdot d \cdot \sum_{t=1}^{t_{\max}} \left(\sum_{i=1}^{n} \Pr\left(\; X_i^t = 1 \;\cap\; X^{t-1} = 0 \;\cap\; X^t > 0 \;\right) \right) \\
&=\; c_B \cdot d \cdot \sum_{t=1}^{t_{\max}} \left(\sum_{i=1}^{n} \Pr\left(\; X_i^t = 1 \;\cap\; X^{t-1} = 0 \;\right) \right) \\
&=\; c_B \cdot d \cdot \sum_{t=1}^{t_{\max}} \left(\sum_{i=1}^{n} \left(\Pr\left(\; X_i^t = 1 \;\cap\; X_i^{t-1} = 0 \;\right) \cdot \prod_{j\neq i} \Pr\left(X_j^{t-1} = 0\right) \right) \right) \\
&=\; c_B \cdot d \cdot \sum_{t=1}^{t_{\max}} n \cdot \left(\frac{\varphi_t}{n} - \frac{\varphi_{t-1}}{n}\right) \cdot \left(1 - \frac{\varphi_{t-1}}{n}\right)^{n-1}
\end{aligned} \tag{4.8}$$

We used $\Pr\left(\; X_i^t = 1 \;\cap\; X_i^{t-1} = 0 \;\right) \;=\; \Pr(\; p^i \in C_{q,\alpha_t} \setminus C_{q,\alpha_{t-1}} \;) \;=\; \frac{\varphi_t}{n} - \frac{\varphi_{t-1}}{n}$.

Expected runtime of GROWING_CUBE when searching with SCAN

We choose the sequence $\{\varphi_t^*\}_{t\geq 1}$ such that

$$\varphi_1^* = \varphi \geq 1 \;,\;\; \varphi_2^* = 2\varphi \;,\;\; \ldots \;,\; \varphi_t^* = t\varphi \;,\; \ldots \;,\; \varphi_{t_{\max}}^* = n \tag{4.9}$$

We specify the value φ later.

By (4.7) and by $\varphi_t \geq \varphi_t^* \geq t\varphi$, $1 \leq t < t_{\max}$, we obtain:

$$\mathrm{E}[\; T_{side} \;] \;=\; O\left(\; d\log(d\varphi) \;\right) \;. \tag{4.10}$$

Since $e^{-\frac{n-1}{n}} \leq \frac{1}{2}$ for $n \geq 4$ and $\varphi_t \geq t\varphi$, we obtain, by (4.8) and (4.2), the following:

$$\begin{aligned} \mathrm{E}[T_{brute}] &< c_B \cdot d \cdot \sum_{t=1}^{t_{\max}} (\varphi+1)\left(1-\frac{\varphi_{t-1}}{n}\right)^{n-1} \leq c_B \cdot (\varphi+1) \cdot d \cdot \sum_{t\geq 0} e^{-\varphi_t \frac{n-1}{n}} \\ &\leq c_B \cdot (\varphi+1) d \cdot \sum_{t\geq 0} \frac{1}{2^{t\varphi}} \leq 2 \cdot c_B \cdot (\varphi+1) d \end{aligned} \tag{4.11}$$

To estimate the scanning part we recall the analysis of the SCAN procedure, which we presented in Section 2.1.2. We focus on the number T_{scan} of performed tests of coordinates.

In Section 2.1.2 we denoted by C^j_{q,α_t} the box obtained by relaxing the side lengths of the cube C_{q,α_t} to unit intervals with respect to those dimensions different from the first j ones in the list $\mathcal{D}$. For $1 \leq t < t_{\max}$ we consider:

$$\mathrm{E}[\, T_{scan}(\varphi_t) \,|\, X^{t-1} = 0\,] \;=\; n \cdot (\, 1 + \sum_{j=1}^{d-1} \Pr\left[\, p^i \in C^j_{q,\alpha_t} \,|\, X^{t-1} = 0\right]\,)\,, \tag{4.12}$$

where $T_{scan}(\varphi_t)$ is the number of comparisons required by an independent call of SCAN(α_t, P). Since the event "$p^i \in C^j_{q,\alpha_t}$" and the event "$X^{t-1}_l = 0$" are independent for all $l \neq i, 1 \leq l \leq n$ we obtain

$$\Pr\left[\, p^i \in C^j_{q,\alpha_t} \,|\, X^{t-1} = 0\right] = \Pr\left[\, p^i \in C^j_{q,\alpha_t} \,|\, X^{t-1}_i = 0\right] = \Pr\left[\, p^i \in C^j_{q,\alpha_t} \,\middle|\, p^i \notin C_{q,\alpha_{t-1}}\right]$$

Because of $C_{q,\alpha_{t-1}} \subseteq C^j_{q,\alpha_t}$ the following holds:

$$\Pr\left[\, p^i \in C^j_{q,\alpha_t} \,\middle|\, p^i \notin C_{q,\alpha_{t-1}}\right] = \frac{\Pr[\, p^i \in C^j_{q,\alpha_t}\,] - \Pr[\, p^i \in C_{q,\alpha_{t-1}}\,]}{1 - \Pr[\, p^i \in C_{q,\alpha_{t-1}}\,]} \leq \Pr[\, p^i \in C^j_{q,\alpha_t}\,] \tag{4.13}$$

which implies:

$$\mathrm{E}[\, T_{scan}(\varphi_t) \,|\, X^{t-1} = 0\,] \;\leq\; \mathrm{E}[\, T_{scan}(\varphi_t)\,]\,. \tag{4.14}$$

By (4.6) and (4.14), we obtain:

$$\begin{aligned} \mathrm{E}[\, T_{scan}\,] &= \sum_{t=1}^{t_{\max}-1} \Pr\left(X^{t-1} = 0\right) \cdot \mathrm{E}[\, T_{scan}(\varphi_t) \,|\, X^{t-1} = 0\,] \\ &\leq \sum_{t=1}^{t_{\max}-1} \left(1-\frac{\varphi_{t-1}}{n}\right)^n \cdot \mathrm{E}[\, T_{scan}(\varphi_t)\,]\,, \end{aligned}$$

where $\mathrm{E}[\, T_{scan}(\varphi_t)\,] \;\leq\; n + \frac{nd}{\ln(n/\varphi_t)}$, which we obtain by (2.7). Note that $\mathrm{E}[\, T_{scan}(\varphi_t)\,]$ is also bounded by nd.

If $\varphi_t < \sqrt{n\varphi}$ then $\frac{n}{\ln(n/\varphi_t)} < \frac{2nd}{\ln(n/\varphi)}$. Let t^* be the smallest t such that $\varphi_t \geq \sqrt{n\varphi}$. Then, by (4.2) and (4.9), we obtain:

$$\begin{aligned} \mathrm{E}[\, T_{scan}\,] &\leq \sum_{t=1}^{t^*-1} \left(1-\frac{\varphi_{t-1}}{n}\right)^n \cdot \left(n + \frac{2nd}{\ln(n/\varphi)}\right) + \sum_{t=t^*}^{t_{\max}-1} \left(1-\frac{\varphi_{t-1}}{n}\right)^n \cdot nd \\ &\leq \left(n + \frac{2nd}{\ln(n/\varphi)}\right) \cdot \sum_{t\geq 0} e^{-t\varphi} \;+\; nd \cdot \sum_{t=t^*}^{t_{\max}-1} e^{-\varphi_{t-1}} \qquad (4.15) \\ &\leq \left(n + \frac{2nd}{\ln(n/\varphi)}\right) \cdot \sum_{t\geq 0} e^{-t\varphi} \;+\; nd \cdot e^{-\varphi_{t^*-1}} \cdot \sum_{t\geq 0} e^{-t\varphi} \qquad (4.16) \\ &\leq \left(1 + \frac{1}{e^{\varphi}-1}\right) \cdot \left(\frac{2nd}{\ln(n/\varphi)} + n + nd \cdot e^{-\varphi_{t^*-1}}\right)\,. \qquad (4.17) \end{aligned}$$

Now, by (4.2) and (4.9), we get $\varphi_t < \varphi_t^* + 1 = \varphi_{t-1}^* + \varphi + 1 < \varphi_{t-1} + \varphi + 1$ for $1 < t \leq t_{\max}$, which implies $e^{-\varphi_{t^*-1}} \leq e^{-\sqrt{n\varphi}+\varphi+1}$, since $\varphi_{t^*} \geq \sqrt{n\varphi}$. For φ such that $\sqrt{n} - \sqrt{\varphi} \geq 1$ we have:

$$\ln n \;\leq\; \sqrt{n} - 1 \;=\; \sqrt{\varphi} - 1 + (\sqrt{n} - \sqrt{\varphi}) \;\leq\; \sqrt{\varphi} \cdot (\sqrt{n} - \sqrt{\varphi}) \;=\; \sqrt{n\varphi} - \varphi \;.$$

for $n \geq 4$. Thus, the inequality $e^{-\sqrt{n\varphi}+\varphi} \leq \frac{1}{n}$ holds, and, by (4.17), we get:

$$\mathrm{E}[\, T_{scan}\,] \;\leq\; \left(1 + \frac{1}{e^{\varphi} - 1}\right) \cdot \left(\frac{2nd}{\ln(n/\varphi)} + n + e \cdot d\right) \;. \tag{4.18}$$

By (4.10), (4.11) and (4.18), we obtain an upper bound on the expected runtime of the GROWING_CUBE method:

$$\mathrm{E}[\, T_{grow}\,] \;=\; O\left(\frac{nd}{\ln(n/\varphi)} + n + \varphi d + d\ln d\right) \;.$$

The function $f(\varphi) := \frac{nd}{\ln(n/\varphi)} + \varphi d$ is monotone increasing in φ on $[1, n)$. Thus, we choose $\varphi = 1$ and obtain $\mathrm{E}[\, T_{grow}\,] = O(\frac{nd}{\ln n} + n + d\ln d)$.

Expected runtime of GROWING_CUBE when searching with REJECT

Obviously, $\mathrm{E}[T_{grow}] \;=\; \Omega\left(\mathrm{E}[T_{reject}(\varphi)] + \varphi d + d\ln(d\varphi)\right)$, where $T_{reject}(\varphi)$ is the number of comparisons required by REJECT for a cube $C_{q,\alpha}$ expected to contain φ points. By (3.5), we have $\mathrm{E}[T_{reject}] = O\left(n\ln(n/\varphi) + n + d\right)$. The aim is to prove that $\mathrm{E}[T_{grow}] = O\left(n\ln(n/\varphi) + n + \varphi d + d\ln d\right)$. The function $b\colon \mathbb{R}_+ \to \mathbb{R}$, $b(\varphi) = n\ln(n/\varphi) + \varphi d$ takes its minimum at $\varphi^* = \max\left\{\frac{n}{d}, 1\right\}$. We set $\varphi = \frac{n}{d}$. Because of the small probability of an empty first cube, we choose here the sequence $\{\varphi_t^*\}_{t\geq 1}$ to grow more slowly than in the case of SCAN:

$$\varphi_1^* = \varphi = \max\left\{\tfrac{n}{d}, 1\right\},\ ,\ \ \varphi_2^* = \varphi + 1\,,\ \ \ldots\ ,\ \varphi_t^* = \varphi + (t-1)\,,\ \ldots\ ,\ \varphi_{t_{\max}}^* = n\,.$$

By (4.7), we obtain:

$$\mathrm{E}[\, T_{side}\,] \;=\; O\left(\, d\log(d\varphi)\,\right) \;. \tag{4.19}$$

In order to bound $\mathrm{E}[T_{brute}]$ we use the following inequalities, which we obtain by (4.2):

$$\begin{aligned} &\varphi_1 < \varphi_1^* + 1 = \varphi + 1 \\ &\varphi_t < \varphi_t^* + 1 = \varphi_{t-1}^* + 1 < \varphi_{t-1} + 2 \quad , \qquad \text{where } 2 \leq t \leq t_{\max} \end{aligned}$$

Now, by (4.8) we obtain the following bound on $\mathrm{E}[T_{brute}]$:

$$\begin{aligned} \mathrm{E}[T_{brute}] \;&\leq\; c_B \cdot d \cdot \varphi_1 + c_B \cdot d \cdot \sum_{t=2}^{t_{\max}} (\varphi_t - \varphi_{t-1}) \cdot \left(1 - \frac{\varphi_{t-1}}{n}\right)^{n-1} \\ &\leq\; c_B \cdot d \cdot (\varphi + 1) + c_B \cdot d \cdot \sum_{t=1}^{t_{\max}} 2\left(1 - \frac{\varphi_t}{n}\right)^{n-1} \\ &\leq\; c_B \cdot (\varphi + 1) \cdot d + 2 \cdot c_B \cdot d \cdot \sum_{t=1}^{t_{\max}} e^{-\varphi_t \frac{n-1}{n}} \end{aligned} \tag{4.20}$$

By $e^{-\frac{n-1}{n}} \leq \frac{1}{2}$ for $n \geq 4$, we have :

$$\begin{aligned}\sum_{t=1}^{t_{\max}} e^{-\varphi_t \frac{n-1}{n}} &\leq \sum_{t=1}^{t_{\max}} \left(\frac{1}{2}\right)^{\varphi_t} \leq \sum_{t=1}^{t_{\max}} \left(\frac{1}{2}\right)^{\varphi+(t-1)} \\ &\leq \frac{1}{2^\varphi} \sum_{t\geq 1} \frac{1}{2^{t-1}} \leq \frac{2}{2^\varphi} \leq 1 . \end{aligned} \tag{4.21}$$

This together with (4.20) implies

$$\mathrm{E}[T_{brute}] \leq c_B \cdot (\varphi+1) \cdot d + c_B \cdot \frac{4d}{2^\varphi} . \tag{4.22}$$

For the runtime analysis of the procedure REJECT we focus on the number of tests of coordinates and tests of bit array entries. In the following we bound $\mathrm{E}[\, T_{reject}(\varphi_t) \mid X^{t-1} = 0\,]$ for $1 \leq t < t_{\max}$. Let $X_{i,j}^t \in \{0,1\}$ be discrete random variables such that $(X_{i,j}^t = 1) \iff (\, p_j^i \in I_j^t := [q_j - \frac{\alpha_t}{2}, q_j + \frac{\alpha_t}{2}]\,)$, $1 \leq i \leq n$, $1 \leq j \leq d$. We obtain:

$$\begin{aligned}\mathrm{E}[\, T_{reject}(\varphi_t) \mid X^{t-1} = 0\,] &= n + \sum_{j=1}^{d} \big(\, 2 + E\,[\ \#\text{points outside } I_j^t \mid X^{t-1} = 0\,]\,\big) \\ &= (n+2d) + \sum_{j=1}^{d}\sum_{i=1}^{n} \Pr(\, X_{i,j}^t = 0 \mid X^{t-1} = 0\,) . \end{aligned} \tag{4.23}$$

Since "$X_{i,j}^t = 0$" and "$X_l^{t-1} = 0$" are independent events for all $l \neq i$, $1 \leq l \leq n$ we have

$$\Pr(X_{i,j}^t = 0 \mid X^{t-1} = 0) = \Pr(X_{i,j}^t = 0 \mid X_i^{t-1} = 0) \leq \frac{\Pr(X_{i,j}^t = 0)}{\Pr(X_i^{t-1} = 0)} = \frac{1 - s_j^t}{1 - \frac{\varphi_{t-1}}{n}} . \tag{4.24}$$

To enable compactness of formulas we set $X_i^0 = 0$ for all $i \in \{1, \ldots, n\}$ and $\varphi_0 = 0$. Note that $\Pr(X_i^{t-1} = 0) = 1 - \frac{\varphi_{t-1}}{n} > 0$ for $1 \leq t \leq t_{\max}$. Furthermore, the side length s_j of the first cube is smaller than the side length s_j^t of the t-th cube. Thus, by (4.24):

$$E\,[\ \#\text{points outside } I_j^t \mid X^{t-1} = 0\,] = \sum_{i=1}^{n} \Pr(\, X_{i,j}^t = 0 \mid X^{t-1} = 0\,) \tag{4.25}$$

$$\leq n \cdot \frac{1 - s_j^t}{1 - \frac{\varphi_{t-1}}{n}} \leq n \cdot \frac{1 - s_j}{1 - \frac{\varphi_{t-1}}{n}} \tag{4.26}$$

By (3.1) and (3.2), we have

$$\sum_{j=1}^{d} n \cdot (1 - s_j) \leq n \ln(n/\varphi) . \tag{4.27}$$

Thus,

$$\sum_{j=1}^{d} E\,[\ \#\text{points outside } I_j^t \mid X^{t-1} = 0\,] \leq \frac{n \ln(n/\varphi)}{1 - \frac{\varphi_{t-1}}{n}} \tag{4.28}$$

Thus, by (4.23) and (4.28), we obtain for $1 \leq t < t_{\max}$:

$$\mathrm{E}[\, T_{reject}(\varphi_t) \mid X^{t-1} = 0\,] \leq n + 2d + \frac{n \ln(n/\varphi)}{1 - \frac{\varphi_{t-1}}{n}} \tag{4.29}$$

$$< \frac{n + 2d + n \ln(n/\varphi)}{1 - \frac{\varphi_{t-1}}{n}} \leq \frac{n + 2d + n \ln(n/\varphi)}{1 - \frac{\varphi_{t-1}}{n}} . \tag{4.30}$$

This together with (4.3) and (4.21) implies:

$$\begin{aligned}
\mathrm{E}[T_{search}] &\leq c_S \cdot (n\ln(n/\varphi) + n + 2d) \cdot \left(1 + \sum_{t\geq 1} \left(1 - \frac{\varphi_t}{n}\right)^{n-1}\right) \\
&\leq c_S \cdot (n\ln(n/\varphi) + n + 2d) \cdot \left(1 + \sum_{t\geq 1} e^{-\varphi_t \frac{n-1}{n}}\right) \\
&\leq 2 \cdot c_S \cdot (n\ln(n/\varphi) + n + 2d)
\end{aligned}$$

Summarizing, $\mathrm{E}[T_{grow}] = O\left(n\ln(n/\varphi) + \varphi d + d\ln d\right)$ where $\varphi = \frac{n}{d}$, thus, $\mathrm{E}[T_{grow}] = O\left(n\ln d + d\ln d\right)$.

Expected runtime of GROWING_CUBE when searching with REJECT_SCAN

We recall the analysis of the REJECT_SCAN procedure presented in Section 3.1.2. If the expected number φ of points in $C_{q,\alpha} \cap P$ fulfills $d < \ln(n/\varphi)$, then REJECT_SCAN(q,α) consists essentially of the SCAN-part, since rejecting is done only with respect to one of the dimensions. In this case the expected running time of REJECT_SCAN is $O(n)$, which is also the running time of SCAN(q,α) if $d < \ln(n/\varphi)$. Thus, for reasons of simplicity, if $d < \ln(n/\varphi)$ then we consider the GROWING-CUBE method to call the searching algorithm SCAN. In this case the expected runtime of the method is $O(n)$.

Assume $d \geq \ln(n/\varphi)$. We prove that $\mathrm{E}[T_{grow}] = O\left(n\ln\left(\frac{d}{\ln(n/\varphi)}\right) + n + \varphi d + d\ln d\right)$ when searching is done by REJECT_SCAN. Observe that the function $b : [1,\infty) \to \mathbb{R}$, $b(\varphi) = n\ln\left(\frac{d}{\ln(n/\varphi)}\right) + \varphi d$ is monotone increasing in φ. We choose the sequence $\{\varphi_t^*\}_{t\geq 1}$ such that

$$\varphi_1^* = \varphi = 1\,,\ \varphi_2^* = 2\varphi\,,\ \ldots\,,\ \varphi_t^* = \min\{t\varphi, n\}\,,\ \ldots\,,\ \varphi_{t_{\max}}^* = n\,.$$

The expected running times $\mathrm{E}[\,T_{side}\,]$ and $\mathrm{E}[\,T_{brute}\,]$ can be bounded as shown in (4.19) and (4.22), respectively. To estimate the expected running time required to search the cubes for points, we use (4.3) and prove a bound on $E\left[\,T_{rej\_scan}(\varphi_t) \;\middle|\; X^{t-1} = 0\,\right]$, where $1 \leq t < t_{\max}$. $T_{rej\_scan}(\varphi_t)$ is the number of comparisons required by an independent call of REJECT_SCAN(α_t, P) and is composed of the number of performed comparisons to partition the dimensions and the number of tests of coordinates and tests of bit array entries required in the reject-part and scanning-part.

Let $\mathcal{D} = \mathcal{D}_1^t \cup \mathcal{D}_2^t$ be the partition of variables done in the t-th iteration by REJECT_SCAN. It is computed to fulfill (3.27) and (3.30), and it can be determined with at most d comparisons. Let $P^{\mathcal{D}_1^t}$ be the points which were not rejected in the t-th iteration. Thus, $P^{\mathcal{D}_1^t}$ contains all points p^i such that $p_j^i \in [q_j - \frac{\alpha}{2}, q_j + \frac{\alpha}{2}]$ for all dimensions $j \in \mathcal{D}_1^t$. Only these points are scanned with respect to the dimensions of $\mathcal{D}_2^t$. We have:

$$\begin{aligned}
&E\left[\,T_{rej_scan}(\varphi_t) \;\middle|\; X^{t-1} = 0\,\right] \\
&\quad\leq E\left[\,T_{reject}(\varphi_t, \mathcal{D}_1^t) \;\middle|\; X^{t-1} = 0\,\right] + E\left[\,T_{scan}(\varphi_t, P^{\mathcal{D}_1^t}, \mathcal{D}_2^t) \;\middle|\; X^{t-1} = 0\,\right] + d\,, \quad (4.31)
\end{aligned}$$

where $T_{reject}(\varphi_t, \mathcal{D}_1^t)$ and $T_{scan}(\varphi_t, P^{\mathcal{D}_1^t}, \mathcal{D}_2^t)$ are the numbers of comparisons required by an independent call of REJECT$(\alpha_t, \mathcal{D}_1^t)$ and by an independent call of REJECT$(\alpha_t, P^{\mathcal{D}_1^t}, \mathcal{D}_2^t)$, respectively.

The reject part is given by

$$\begin{aligned}
&E\left[\,T_{reject}(\varphi_t, \mathcal{D}_1^t) \;\middle|\; X^{t-1} = 0\,\right] \\
&\qquad = 2|\mathcal{D}_1^t| + \sum_{j\in\mathcal{D}_1^t} E\left[\,\#\text{points outside } I_j^t \;\middle|\; X^{t-1} = 0\,\right] + n\,. \qquad (4.32)
\end{aligned}$$

We recall the arguments used in the analysis of the expected runtime of GROWING_CUBE with the searching algorithm REJECT. By (4.23) and (4.30), we obtain:

$$E\left[\ \#\text{points outside } I_j^t \ \middle|\ X^{t-1}=0\ \right] \ \le\ n\frac{1-s_j}{1-\frac{\varphi_{t-1}}{n}}\ . \tag{4.33}$$

By (4.32), (3.36) and (4.33), we obtain:

$$E\left[\ T_{reject}(\varphi_t,\mathcal{D}_1^t)\ \middle|\ X^{t-1}=0\ \right] \ \le\ 2d+n+\frac{n\ln\left(\frac{d}{\ln(n/\varphi)}\right)+2n}{1-\frac{\varphi_{t-1}}{n}}\ . \tag{4.34}$$

Let $S^t(p^i,\mathcal{D}_2^t)$ be the number of comparisons required to scan some data point p^i with respect to its dimensions in $\mathcal{D}_2^t$. Then the scan part of the expected runtime is given by

$$E\left[\ T_{scan}(\varphi_t,P^{\mathcal{D}_1^t},\mathcal{D}_2^t)\ \middle|\ X^{t-1}=0\ \right] \ =\ E\left[\ |P^{\mathcal{D}_1^t}|\ \middle|\ X^{t-1}=0\ \right]\cdot E\left[\ S^t(p^i,\mathcal{D}_2^t)\ \middle|\ X^{t-1}=0\ \right]\ ,$$

which holds since events with respect to different dimensions of the points are independent. By arguments similar to those in (4.12) and (4.13), we get $E\left[S^t(p^i,\mathcal{D}_2^t)\ \middle|\ X^{t-1}=0\right] \le E\left[S^t(p^i,\mathcal{D}_2^t)\right]$. Based on the analysis of SCAN, property (3.27) of partition $\mathcal{D}=\mathcal{D}_1^t\cup\mathcal{D}_2^t$ and arguments of (2.5), we obtain for $1\le t<t_{\max}$:

$$E\left[S^t(p^i,\mathcal{D}_2^t)\ \middle|\ X^{t-1}=0\right] \ \le\ E\left[S^t(p^i,\mathcal{D}_2^t)\right] \ \le\ \frac{1-\prod_{j\in\mathcal{D}_2^t}s_j^t}{1-\lambda_t}\,, \tag{4.35}$$

where $\lambda_t=\sqrt[d]{\frac{\varphi_t}{n}}<1$ is the geometric mean of the side lengths s_j^t of the box $C_{q,\alpha_t}\cap[0,1]^d$.

$P^{\mathcal{D}_1^t}$ are the points contained in the box $C_{q,\alpha_t}(\mathcal{D}_1^t)$, which is obtained by relaxing the side lengths of the cube C_{q,α_t} to unit intervals, except those with respect to the dimensions in $\mathcal{D}_1^t$. In analogy to (4.13), the following inequalities are easily seen to hold:

$$\begin{aligned} E\left[\ |P^{\mathcal{D}_1^t}|\ \middle|\ X^{t-1}=0\ \right] \ &=\ n\cdot\Pr\left[\,p^i\in C_{q,\alpha_t}(\mathcal{D}_1^t)\ \middle|\ p^i\notin C_{q,\alpha_{t-1}}\right] \\ &\le\ n\cdot\Pr\left[\,p^i\in C_{q,\alpha_t}(\mathcal{D}_1^t)\,\right] \ =\ E\left[\ |P^{\mathcal{D}_1^t}|\ \right]\ , \end{aligned} \tag{4.36}$$

where p^i is some data point. Following the arguments of (3.29) and (3.34) in Section 3.1.2, we obtain for $1\le t<t_{\max}$:

$$E\left[\ |P^{\mathcal{D}_1^t}|\ \right]\cdot E\left[\ S^t(p,\mathcal{D}_2^t)\ \right] \ \le\ \frac{n\prod\limits_{j\in\mathcal{D}_1^t}s_j^t\ -\ \varphi_t}{1-\lambda_t} \ \le\ n\ .$$

This combined with (4.34), (4.31) and (4.3) implies

$$\begin{aligned} \mathrm{E}[T_{search}] \ &<\ c_S\cdot\left(\ n\ln\left(\frac{d}{\ln(n/\varphi)}\right)+4n+2d\ \right)\cdot\sum_{t=1}^{t_{\max}}\left(1-\frac{\varphi_{t-1}}{n}\right)^{n-1} \\ &\le\ c_S\cdot\left(\ n\ln\left(\frac{d}{\ln(n/\varphi)}\right)+4n+2d\ \right)\cdot\sum_{t\ge0}2^{-t\varphi}\ \ , \end{aligned}$$

by using $\left(1-\frac{\varphi_t}{n}\right)^{n-1}\ \le\ e^{-\varphi_t\frac{n-1}{n}}\ \le\ 2^{-t\varphi}$. Since we chose $\varphi=1$, we obtain $\mathrm{E}[T_{grow}]=O\left(n\ln\left(\frac{d}{\ln n}\right)+n+d\ln d\right)$.

We summarize the results in the following theorem.

Theorem 4.1. *The* GROWING_CUBE *method finds the nearest neighbor from the set P of data points to some query point q and has the expected asymptotic runtime of the* CUBE METHOD.

In practice, GROWING_CUBE may perform better than the CUBE METHOD. This consideration is confirmed by the experiments in Section 4.5 and it is based on the fact that the GROWING-CUBE starts with a smaller cube. Furthermore, the CUBE METHOD calls the brute-force method if the considered cube does not contain any points of P.

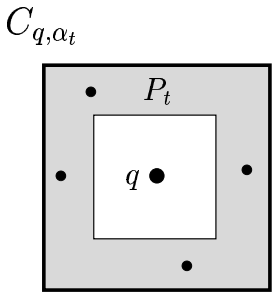
4.2 k-Nearest-Neighbor Search

4.2.1 The algorithm

The GROWING-CUBE method can be extended to compute the k nearest neighbors of the query point q from the point set P. The extended GROWING-CUBE method works as follows : the points in the current cube are determined; if still less than k points have been found so far, then a larger cube is considered. We use the notations introduced in Section 4.1. In the t-th iteration the procedure SIDE_LENGTH is called with the parameter φ_t^* and computes side lengths α_t such that $\varphi_t^* \leq E[\,|C_{q,\alpha_t} \cap P|\,] = \varphi_t < \varphi_t^* + 1$. For the first iteration we choose $\varphi_1^* \geq k$. The following gives a schematic description of the algorithm.

GROWING_CUBE ($\mathbf{k}, \mathbf{q}, \mathbf{P}$)
$t := 0$; $L_0 := \emptyset$; $O_0 := P$;
repeat
  $t := t + 1$;
  if ($\varphi_t < n$) then
    $(\mathcal{D}, \alpha_t)$:= SIDE_LENGTH(φ_t, q);
    (P_t, O_t) := SEARCH_CUBE($\alpha_t, q, O_{t-1}, \mathcal{D}$);
    $L_t := L_{t-1} \cup P_t$;
  else $L_t := P$; $P_t := O_{t-1}$; (SET_UPDATE)
until ($|L_t| \geq k$);
if ($|L_t| > k$)
  then $L := L_{t-1} \cup$ SELECT($k - |L_{t-1}|, q, P_t$); (SELECT)
else $L := L_t$;
return L;

Set L contains the k nearest neighbors to the query point q. L_t is the set of points found by the end of the t-th iteration. O_t represents the set of points outside the cube C_{q,α_t}, which is together with $P_t = O_{t-1} \setminus O_t$ the result of a slightly modified SEARCH_CUBE. The iteration terminates if $|L_t| \geq k$. If $|L_t| > k$ the additional $k - |L_{t-1}|$ nearest neighbors, which we still need, are extracted from the set P_t by the SELECT-procedure. The procedure SELECT(k', q, P') computes the distances of all points of P' to the query q, selects the k' smallest distances and returns as result the corresponding points from P'. Using a linear time selection algorithm, the running time of SELECT(k', q, P') is $c_S \cdot d \cdot |P'|$ for some appropriate constant $c_S > 0$.



We provide the set O_t of points outside C_{q,α_t}, since we do not want to test in the next iteration the points of L_t, which have been found so far. We need some small modifications of SEARCH_CUBE, which is one of the searching procedures SCAN, REJECT and REJECT_SCAN:

- Since SCAN checks each point of its input set, it can simply maintain two index lists, for the points inside and for those outside of the current cube.

- REJECT gets the set O_{t-1} as a bit array, where an entry is set to 1 if and only if this represents the index of a point outside $C_{q,\alpha_{t-1}}$. After the rejecting phase, the bit array contains 1-entries exactly for the points in $C_{q,\alpha_t} \setminus C_{q,\alpha_{t-1}}$, and represents the result P_t. We built the bit array O_t by the bitwise-AND between O_{t-1} and the bitwise-complement of P_t.

- REJECT_SCAN has the bit array O_{t-1} as part of the input. At the end of the rejecting phase we have a bit array with 1-entries exactly for the points to be scanned in this t-th iteration. The scanning phase has P_t as result and bit array O_t is built as described above.

The order $\mathcal{D}$ of dimensions corresponds to the increasing order of the side lengths of a box $C_{q,\alpha_t} \cap [0,1]^d$. This order depends only on the query point and is computed by the SIDE_LENGTH procedure only once in the first iteration in $O(d \log d)$ time.

4.2.2 The expected runtime analysis

We introduce the discrete random variable S^t to represent the total number of tests of coordinates and tests of bit array entries required by an independent call of SEARCH_CUBE(α_t, q, O_{t-1}). Let X^t be here the discrete random variable for the number $|L_t|$ of points found during the first t iterations, $t \geq 1$. We have $\mathrm{E}[X^t] = \varphi_t$ for $t \geq 1$, and we define $X^0 = 0$.

In practice the numbers φ_t^* could be chosen to depend on the number of nearest neighbors which are still to be found, thus, if $|X^{t-1}| < k$ then $\varphi_t^* = \varphi_{t-1}^* + k - |X^{t-1}|$. The analysis of the algorithm with random variables φ_t^* is involved. We consider a fixed sequence $\{\varphi_t^*\}_{t\geq 1}$, that is known at the beginning of the method, and which satisfies (4.1). The termination of the algorithm is ensured and we can define $t_{\max} = \min\{t \mid \varphi_t^* \geq n\}$.

The expected runtime $\mathrm{E}[\, T_{grow}\,]$ of the GROWING-CUBE is given by:

$$\mathrm{E}[\, T_{grow}\,] \;=\; \mathrm{E}[\, T_{side}\,] \;+\; \mathrm{E}[\, T_{search}\,] \;+\; \mathrm{E}[\, T_{sel}\,] \;,$$

where $\mathrm{E}[\, T_{search}\,]$ is the expected runtime of the SEARCH_CUBE procedure:

$$\mathrm{E}[T_{search}] \;\leq\; \sum_{t=1}^{t_{\max}-1} \Pr(X^{t-1} < k) \cdot c_S \cdot \mathrm{E}[S^t \mid X^{t-1} < k] \tag{4.37}$$

and $\mathrm{E}[T_{sel}]$ is the expected runtime of step (SELECT):

$$\begin{aligned} \mathrm{E}[T_{sel}] \;\leq\; & \sum_{t=1}^{t_{\max}} \Pr\left[\, (X^{t-1} < k) \,\cap\, (X^t > k)\,\right] \\ & \cdot c_B \cdot d \cdot \mathrm{E}\left[\, X^t - X^{t-1} | X^{t-1} < k, X^t > k\,\right] \;. \end{aligned} \tag{4.38}$$

The values c_S and c_B are the corresponding constants of the SEARCH_CUBE procedure and of the SELECT-procedure, respectively. The expected runtime $\mathrm{E}[\, T_{side}\,]$ for the computation of the side lengths of the cubes C_{q,α_t} is given by:

$$\mathrm{E}[\, T_{side}\,] \;\leq\; \sum_{t=1}^{t_{\max}-1} \Pr\left(X^{t-1} < k\right) \cdot c_L \cdot d \cdot \log(d\varphi_t) \;, \tag{4.39}$$

where c_L is the constant of the SIDE_LENGTH procedure introduced in Section 2.1.2.

In the following we first bound the probability $\Pr(X^t < k)$ and then determine the expected running time of SELECT, SIDE_LENGTH and SEARCH_CUBE.

The probability of cube enlargement

To bound the probability $\Pr(X^t < k)$ we use the following theorem (see [22]) based on the Chernoff Bound.

Theorem 4.2. [22] *Let Z be the discrete variable which counts the total number of successes by performing n independent Bernoulli trials with the success probability p of a trial. Then*

$$Pr(Z - np \geq r) \leq e^{\lambda^2\sigma^2}/e^{\lambda r}$$

for each $\lambda < \frac{1}{\max(p,1-p)}$ where $\sigma^2 = np(1-p)$ is the variance of Z.

Taking $\lambda := \frac{r}{2n(1-p)}$ in Theorem 4.2 we get the bound:

Corollary 4.1. *Let Z be the discrete variable which counts the total number of successes by performing n independent Bernoulli trials with the success probability p of a trial. Then*

$$Pr(Z - np \geq r) \leq e^{\frac{-r^2}{4n(1-p)}}$$

where $0 < r < \frac{2n(1-p)}{\max(p,1-p)}$.

Corollary 4.2. *X^t is the discrete random variable for the number of nearest neighbors found by* GROWING_CUBE *during the first t iterations, and k a constant such that $\varphi_t = E[X^t] \geq k$, where $1 \leq t \leq t_{\max}$. Then the following holds:*

$$Pr(X^t < k) \leq e^{-\varphi_t/4+(k-1)/2}. \tag{4.40}$$

Proof. Let N^t be the random variable for the number of points of P within $[0,1]^d \setminus C_{\alpha_t}$. Thus, $N^t = n - X^t$, $\Pr(X^t < k) = \Pr(N^t > n-k) = \Pr(N^t \geq n-k+1)$ and for some point $p' \in [0,1]^d$ the probability $p = \Pr(p' \in N^t)$ equals $1 - \frac{\varphi_t}{n}$. Let

$$r := n - k + 1 - np = n - k + 1 - (n - \varphi_t) = \varphi_t - k + 1$$

Because of

$$0 < r = \varphi_t - k + 1 \leq \varphi_t = n(1-p) < \frac{2n(1-p)}{\max(p, 1-p)}$$

we get by Corollary 4.1:

$$\begin{aligned} \Pr(N^t \geq n-k+1) &= \Pr(N^t - np \geq n - k + 1 - np) \\ &\leq e^{\frac{-r^2}{4n(1-p)}} = e^{\frac{-(\varphi_t - k+1)^2}{4\varphi_t}}. \end{aligned}$$

By using $\frac{-(\varphi_t-k+1)^2}{4\varphi_t} \leq -\frac{\varphi_t}{4} + \frac{k-1}{2}$ we obtain the stated inequality.

□

We choose φ_t^* to fulfill

$$\varphi_t^* \;=\; \varphi + t - 1\,, \;\; t = 1, \ldots, t_{\max} \;\;, \quad \text{where} \quad \varphi \geq 2k+1\,, \tag{4.41}$$

with $t_{\max} = \min\{t \mid \varphi_t^* \geq n\}$. We specify the exact value of φ later.

We have $\mathrm{E}[X^t] \geq k+1$. By (4.40) and $\varphi_t^* \leq \varphi_t$, we obtain:

$$\Pr(X^t < k+1) \leq e^{-t/4}. \tag{4.42}$$

The total expected runtime of the GROWING-CUBE method

We introduce some preliminary notations and observations. The event "$X^{t-1} < k$" is the union of the following 2 disjoint events for any $i \in \{1, \ldots, n\}$:

$$\{X^{t-1} < k\} = \left(\{ \sum_{j \neq i} X_j^{t-1} < k-1 \} \cap \{X_i^{t-1} = 1\} \right) \cup \left(\{ \sum_{j \neq i} X_j^{t-1} < k \} \cap \{X_i^{t-1} = 0\} \right) , \tag{4.43}$$

where $(X_i^t = 1) \iff$ (point $p^i \in C_{q,\alpha_t}$) for any $i \in \{1, \ldots, n\}$. We denote by $A_k^{t-1}(i)$ and by $B_k^{t-1}(i)$ the left event and the right event of the union in (4.43), respectively. We will use the following inequality:

$$\begin{aligned} &\Pr(X^{t-1} < k+1) \\ &\quad \geq \Pr(X_i^{t-1} = 1)\Pr(\sum_{j \neq i} X_j^{t-1} < k) + \Pr(X_i^{t-1} = 0)\Pr(\sum_{j \neq i} X_j^{t-1} < k) \\ &\quad = \Pr(\sum_{j \neq i} X_j^{t-1} < k) \end{aligned} \tag{4.44}$$

Proposition 4.1. *The expected runtime of step* (SELECT) *is* $O(\varphi d)$.

Proof. We have:

$$\begin{aligned} \mathrm{E}[T_{sel}] &= c_B \cdot d \cdot \sum_{t=1}^{t_{\max}} \Pr\left[(X^{t-1} < k) \cap (X^t > k) \right] \cdot \mathrm{E}[X^t - X^{t-1} | X^{t-1} < k, X^t > k] \\ &\leq c_B \cdot d \cdot \sum_{t=1}^{t_{\max}} \sum_{i=1}^{n} \Pr\left[(p^i \in C_{q,\alpha_t} \setminus C_{q,\alpha_{t-1}}) \cap (X^{t-1} < k) \right] \end{aligned} \tag{4.45}$$

Since variables X_i^t, $i = 1, \ldots, n$ are independent, we obtain by (4.43), (4.44) and (4.45), the following for $1 < t \leq t_{\max}$:

$$\begin{aligned} &\Pr\left[(p^i \in C_{q,\alpha_t} \setminus C_{q,\alpha_{t-1}}) \cap (X^{t-1} < k) \right] \\ &\quad \leq \left(\frac{\varphi_t}{n} - \frac{\varphi_{t-1}}{n} \right) \cdot \Pr(\sum_{j \neq i} X_j^{t-1} < k) \\ &\quad \leq \frac{2}{n} \cdot \Pr(\sum_{j \neq i} X_j^{t-1} < k) \leq \frac{2}{n} \cdot \Pr(X^{t-1} < k+1) \end{aligned}$$

We used the fact $\varphi_t - \varphi_{t-1} \leq \varphi_t^* + 1 - \varphi_{t-1}^* = 2$.

If $t = 1$ then, obviously, $\Pr\left[(p^i \in C_{q,\alpha} \cap (X^0 < k) \right] = \frac{\varphi}{n}$. Altogether, we have:

$$\begin{aligned} \mathrm{E}[T_{sel}] &\leq c_B \cdot \varphi d + c_B \cdot 2d \cdot \sum_{t \geq 0} \Pr(X^t < k+1) \\ &\leq c_B \cdot \varphi d + c_B \cdot 2d \cdot \sum_{t \geq 0} e^{-t/4} = O(\varphi d) . \end{aligned}$$

□

In a similar way, we bound the expected running time to compute the side lengths of the cubes C_{q,α_t} and the order $\mathcal{D}$. We proceed in analogy to (4.7), use $\sum_{t \geq 0} t \cdot e^{-t/4} = O(1)$ and (4.42), and we obtain:

$$\mathrm{E}[T_{side}] \leq \sum_{t \geq 1} \Pr(X^{t-1} < k) \cdot c_L \cdot d \cdot \log(d\varphi_t) = O\left(d \log(\varphi d)\right) . \tag{4.46}$$

In the following we bound the expected runtime $\mathrm{E}[\, T_{search}\,]$ for each of the considered searching procedures, SCAN, REJECT and REJECT_SCAN.

A) If SCAN is used, then for iteration t, $1 \leq t \leq t_{\max}$, we obtain

$$\mathrm{E}[S^t \mid X^{t-1} < k] \;=\; n \cdot \sum_{j=0}^{d-1} \Pr\left[p^i \in C^j_{q,\alpha_t} \setminus C_{q,\alpha_{t-1}} \;\middle|\; X^{t-1} < k\right] \;, \tag{4.47}$$

where $C^0_{q,\alpha_t} = [0,1]^d$ and C^j_{q,α_t}, $1 \leq j \leq d$ is, as introduced in Section 2.1.2, the box obtained by relaxing the side lengths of the cube C_{q,α_t} to unit intervals with respect to those dimensions different from the first j ones in $\mathcal{D}$. The equation (4.47) is easily verified by adapting the analysis of SCAN to the fact that only points outside $C_{q,\alpha_{t-1}}$ are scanned. By using $\left\{p^i \in C^j_{q,\alpha_t} \setminus C_{q,\alpha_{t-1}}\right\} \cap A_k^{t-1}(i) = \emptyset$, (4.43), (4.44) and the fact that the points are drawn independently at random, we obtain:

$$\begin{aligned}
&\Pr\left[(p^i \in C^j_{q,\alpha_t} \setminus C_{q,\alpha_{t-1}}) \;\cap\; (X^{t-1} < k)\right] \\
&\quad= \Pr\left[(p^i \in C^j_{q,\alpha_t} \setminus C_{q,\alpha_{t-1}}) \;\cap\; B_k^{t-1}(i)\right] \\
&\quad= \left(\Pr(\, p^i \in C^j_{q,\alpha_t}\,) - \frac{\varphi_{t-1}}{n}\right) \cdot \Pr(\, \sum_{j \neq i} X_j^{t-1} < k\,) \\
&\quad< \Pr(\, p^i \in C^j_{q,\alpha_t}\,) \cdot \Pr(\, X^{t-1} < k+1\,)
\end{aligned} \tag{4.48}$$

This implies together with (2.2) and (4.37)

$$\mathrm{E}[T_{search}] < c_S \cdot \sum_{t=1}^{t_{\max}} \Pr(X^{t-1} < k+1) \cdot \mathrm{E}[T_{scan}(\varphi_t)]\;,$$

where $T_{scan}(\varphi_t)$ is the total number of tests required by an independent call of SCAN(α_t, q, P). Clearly, $T_{scan}(\varphi_t) \leq nd$ and by (2.7) $\mathrm{E}[T_{scan}(\varphi_t)] \leq n + \frac{nd}{\ln(n/\varphi_t)}$ for $1 \leq t < t_{\max}$.

Note that the upper bound on $\mathrm{E}[T_{search}]$ is monotone increasing in φ for this case when the SCAN procedure is used. We choose the smallest possible value of φ which fulfills (4.41), that is $\varphi = 2k+1$.

If $\varphi_t \leq \sqrt{nk}$ then:

$$\frac{nd}{\ln(n/\varphi_t)} \leq \frac{nd}{\ln(n/\sqrt{nk})} = \frac{2nd}{\ln(n/k)}$$

Thus, by using (4.37), $\mathrm{E}[T_{search}]$ can be bounded as follows:

$$\mathrm{E}[T_{search}] \;<\; \sum_{\varphi_t \leq \lfloor\sqrt{nk}\rfloor} \Pr(X^{t-1} < k+1) \cdot \left(n + \frac{2nd}{\ln(n/k)}\right) + \sum_{\varphi_t > \lfloor\sqrt{nk}\rfloor} \Pr(X^{t-1} < k+1) \cdot nd$$

If $\varphi_t \geq \lfloor\sqrt{nk}\rfloor$ we have $-\frac{t}{4} = -\frac{\varphi_t}{4} + \frac{k-1}{2} \leq -\frac{\lfloor\sqrt{nk}\rfloor}{4} + \frac{k-1}{2}$. It can be shown that $\frac{\lfloor\sqrt{nk}\rfloor}{4} - \frac{k-1}{2} \geq \frac{\sqrt{n}}{4} \geq \ln n$ for n big enough and $k \leq n/8$. In this case, if $\varphi_t \geq \lfloor\sqrt{nk}\rfloor$ then $-\frac{t}{4} \leq -\ln n$. Let t_* be the smallest t such that $\varphi_t \geq \lfloor\sqrt{nk}\rfloor$. Since $-\frac{t_*}{4} \leq -\ln n$, we have by (4.42):

$$\begin{aligned}
\sum_{\varphi_t > \lfloor\sqrt{nk}\rfloor} \Pr(X^{t-1} < k+1) \;&=\; \sum_{\varphi_t \geq \lfloor\sqrt{nk}\rfloor} \Pr(X^t < k+1) \\
&\leq\; \sum_{t \geq t_*} e^{-t/4} = e^{-t_*/4} \cdot O(1) \;\leq\; \frac{1}{n} \cdot O(1)
\end{aligned}$$

By (4.42), we get: $\sum_{\varphi_t \le \lfloor \sqrt{nk} \rfloor} \Pr(X^{t-1} < k+1) \;\le\; \sum_{t \ge 0} e^{-t/4} = O(1)$. With this we obtain $\mathrm{E}[T_{search}] = O\left(\frac{nd}{\ln(n/k)} + n\right)$ for $k \le n/8$. If $k > n/8$ then the expected runtime of the searching procedure SCAN is $O(nd)$.

Thus, by (4.46) and Proposition 4.1, the expected runtime of the GROWING-CUBE method with the searching procedure SCAN is $O\left(\frac{nd}{\ln(n/k)} + n + kd + d\ln d\right)$ for $k < n$.

Notice that our method is only of interest if k is significantly smaller then n, since $\Theta(kd)$ tests are needed in any case and the brute-force method cannot be beaten asymptotically by the GROWING-CUBE method if $k = \Theta(n)$.

B) We consider the procedure REJECT to determine the point sets $C_{q,\alpha_t} \cap P$. By (4.37), we obtain:

$$\mathrm{E}[T_{search}] \;\le\; \sum_{t=1}^{t_{\max}-1} \Pr(X^{t-1} < k) \cdot c_S \cdot \mathrm{E}[\, T_{reject}(\varphi_t) \mid X^{t-1} < k] \,,$$

where $T_{reject}(\varphi)$ is the number of comparisons required by REJECT for a cube $C_{q,\alpha}$ expected to contain φ points. For the runtime analysis of the procedure REJECT we focus on the number of tests of coordinates and tests of bit array entries. We will count also the $2n$ additional bit operations to build bit array O_t.

The discrete random variables $X^t_{i,j} \in \{0,1\}$ are defined as in Section 4.1 such that $(X^t_{i,j} = 1) \iff (\, p^i_j \in I^t_j := [q_j - \frac{\alpha_t}{2}, q_j + \frac{\alpha_t}{2}]\,)$, $1 \le i \le n$, $1 \le j \le d$. We obtain:

$$\begin{aligned}
\mathrm{E}[\, T_{reject}(\varphi_t) \mid X^{t-1} < k\,] &= 3n + \sum_{j=1}^{d} \left(\, 2 + E\left[\, \#\text{points outside } I^t_j \mid X^{t-1} < k \,\right]\,\right) \\
&= (3n + 2d) + \sum_{j=1}^{d} \sum_{i=1}^{n} \Pr(\, X^t_{i,j} = 0 \mid X^{t-1} < k\,) \,. \qquad (4.49)
\end{aligned}$$

Thus,

$$\begin{aligned}
\mathrm{E}[\, T_{search}\,] &\le c_S \cdot (3n + 2d) \cdot \sum_{t=1}^{t_{\max}} \Pr(X^{t-1} < k) \\
&\quad + c_S \cdot \sum_{t=1}^{t_{\max}} \sum_{j=1}^{d} \sum_{i=1}^{n} \Pr[\, (X^t_{i,j} = 0) \,\cap\, (X^{t-1} < k)\,] \qquad (4.50)
\end{aligned}$$

The essential step is to bound $\Pr[(X^t_{i,j} = 0) \,\cap\, (X^{t-1} < k)]$ for $1 \le t \le t_{\max}$, where "$X^t_{i,j} = 0$" is the event stating that dimension j of point p^i lies outside the interval $I^t_j = [q_j - \frac{\alpha_t}{2}, q_j + \frac{\alpha_t}{2}]$. Observing that $\{X^t_{i,j} = 0\} \cap A^{t-1}_k(i) = \emptyset$, we obtain:

$$\begin{aligned}
\Pr\left[(X^t_{i,j} = 0) \,\cap\, (X^{t-1} < k)\right] &= \Pr\left[(X^t_{i,j} = 0) \,\cap\, B^{t-1}_k(i)\right] \\
&\le \Pr[\, (X^t_{i,j} = 0) \,\cap\, (X^{t-1}_i = 0) \,\cap\, (\sum_{j \ne i} X^{t-1}_j < k)\,] \\
&\le \Pr(\, X^t_{i,j} = 0\,) \cdot \Pr(\, X^{t-1} < k+1\,) \\
&\le \Pr(\, X^1_{i,j} = 0\,) \cdot \Pr(\, X^{t-1} < k+1\,) \\
&\le (1 - s_j) \cdot \Pr(\, X^{t-1} < k+1\,) \,, \qquad (4.51)
\end{aligned}$$

where s_j is the side length of the first cube. We also used (4.44), the fact that the points are drawn independently at random, and that in each iteration the side length increases. Obviously, since $\Pr(X^{t-1} < k) \leq \Pr(X^{t-1} < k+1)$, the inequalities (4.50), (4.51) and (4.27) imply:

$$\begin{aligned} \mathrm{E}[T_{search}] &\leq c_S \cdot (n \ln(n/\varphi) + 3n + 2d) \cdot \sum_{t \geq 0} \Pr\left(X^t < k+1\right) \\ &\leq c_S \cdot (n \ln(n/\varphi) + 3n + 2d) \cdot \sum_{t \geq 0} e^{-t/4} \\ &= O\left(n \ln(n/\varphi) + n + d\right) \end{aligned}$$

Thus, the expected runtime $\mathrm{E}[T_{grow}]$ of the GROWING-CUBE method with the searching procedure REJECT is $O\left(n \ln(n/\varphi) + \varphi d + d \ln d\right)$, where $\varphi \geq 2k+1$. The function $b\colon \mathbb{R}_+ \to \mathbb{R}$ with $b(\varphi) = n \ln(n/\varphi) + \varphi d$ takes its minimum at $\frac{n}{d}$ and is monotone increasing on $[\frac{n}{d}, \infty)$. We choose $\varphi = \max\{\frac{n}{d}, 2k+1\}$. If $2k+1 \geq \frac{n}{d}$ then $\mathrm{E}[T_{grow}] = O\left(n \ln(n/k) + kd + d \ln d\right)$, otherwise $\mathrm{E}[T_{grow}] = O\left(n \ln d + d \ln d\right)$.

C) We consider the procedure REJECT_SCAN to determine the point sets $C_{q,\alpha_t} \cap P$. The analysis is based on the results obtained in Section 3.1.2 and Section 4.1.

If $d < \ln(n/\varphi)$ then the GROWING-CUBE method calls the searching algorithm SCAN, as an extremal case of the procedure REJECT_SCAN. In this case the expected runtime of the method is $O(n + \varphi d + d \ln d)$.

Now assume $d \geq \ln(n/\varphi)$. To compute partition $\mathcal{D} = \mathcal{D}_1^t \cup \mathcal{D}_2^t$ such that (3.27) and (3.30) are fulfilled we need at most d comparisons. To bound the rejecting part in iteration t, $1 \leq t < t_{\max}$, we investigate $E\left[\, T_{reject}(\varphi_t, \mathcal{D}_1^t) \;\middle|\; X^{t-1} < k \,\right]$, where $T_{reject}(\varphi_t, \mathcal{D}_1^t)$ is the number of comparisons required by an independent call of REJECT_SCAN$(\alpha_t, P, \mathcal{D}_1^t)$. In analogy to (4.32) and (4.49) and by using (4.51), we have

$$\begin{aligned} E\left[\, T_{reject}(\varphi_t, \mathcal{D}_1^t) \;\middle|\; X^{t-1} < k \,\right] &= 3n + \sum_{j \in \mathcal{D}_1^t} \left(\, 2 + E\left[\, \#\text{points outside } I_j^t \mid X^{t-1} < k \,\right] \right) \\ &\leq (3n + 2d) + \sum_{j \in \mathcal{D}_1^t} \sum_{i=1}^{n} \Pr(\, X_{i,j}^t = 0 \mid X^{t-1} < k \,) \\ &\leq (3n + 2d) + \sum_{j \in \mathcal{D}_1^t} \sum_{i=1}^{n} \frac{\Pr(X^{t-1} < k+1)}{\Pr(X^{t-1} < k)} \cdot (1 - s_j)\,. \end{aligned}$$

By using the properties of the partition $\mathcal{D} = \mathcal{D}_1^t \cup \mathcal{D}_2^t$ and (3.36), we get:

$$E\left[\, T_{reject}(\varphi_t, \mathcal{D}_1^t) \;\middle|\; X^{t-1} < k \,\right] \leq 3n + 2d + \frac{\Pr(X^{t-1} < k+1)}{\Pr(X^{t-1} < k)} \cdot \left(n \ln\left(\frac{d}{\ln(n/\varphi)}\right) + 2n \right)$$

The scanning part in iteration t, $1 \leq t < t_{\max}$, is given by:

$$\sum_{j=0}^{d-1} n \cdot \Pr\left[p^i \in C^{\mathcal{D}_1^t, j} \setminus C_{q,\alpha_{t-1}} \;\middle|\; X^{t-1} < k \right]\,,$$

where $C^{\mathcal{D}_1^t, j}$ is the box obtained by relaxing the side lengths of the cube C_{q,α_t} to unit intervals except those with respect to the dimensions in $\mathcal{D}_1^t$ and the first j dimensions in the list $\mathcal{D}_2^t$. By using the

same arguments as in (4.48), we obtain the following bound for the scanning part:

$$\sum_{t=1}^{t_{\max}-1} \Pr(X^{t-1} < k+1) \cdot \sum_{j=0}^{d-1} n \cdot \Pr\left[p^i \in C^{\mathcal{D}_1^t, j}\right] ,$$

In analogy to the analysis presented in Section 4.1, we obtain, by the properties of the partition $\mathcal{D} = \mathcal{D}_1^t \cup \mathcal{D}_2^t$, the following:

$$\sum_{j=0}^{d-1} n \cdot \Pr\left[p^i \in C^{\mathcal{D}_1^t, j}\right] \;=\; \frac{n\lambda(\mathcal{D}_1^t) - \varphi_t}{1 - \lambda(\mathcal{D}^t)} \;\leq\; n \,,$$

where $\lambda(\mathcal{D}_1^t)$ is the geometric mean of those side lengths of C_{q,α_t} which correspond to the dimensions in $\mathcal{D}_1^t$. Since $\sum_{t=1}^{t_{\max}} \Pr(X^{t-1} < k+1) = O(1)$ for $\varphi \geq 2k+1$, we obtain $\mathrm{E}[T_{search}] = O\left(n \ln\left(\frac{d}{\ln(n/\varphi)}\right) + n + d\right)$. Thus, $\mathrm{E}[T_{grow}] = O\left(n \ln\left(\frac{d}{\ln(n/\varphi)}\right) + n + \varphi d + d \ln d\right)$, where $\varphi \geq 2k+1$.

Since in both cases the upper bound on the expected runtime of the GROWING-CUBE method is monotone increasing in φ, we choose $\varphi = 2k+1$.

Thus, the expected runtime of the GROWING-CUBE method with the searching procedure REJECT_SCAN is $O\left(n \ln\left(\frac{d}{\ln(n/k)} + 1\right) + n + dk + d \ln d\right)$.

We summarize the results obtained for the GROWING-CUBE method with the searching procedure REJECT_SCAN:

Theorem 4.3. *The* GROWING-CUBE *method with the searching procedure* REJECT_SCAN *finds the k nearest neighbors from the set P of n points to some query point q with an expected asymptotic runtime of $O\left(n \ln\left(\frac{d}{\ln(n/k)} + 1\right) + n + dk + d \ln d\right)$, if $k < n$ and 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.*

4.3 Other distributions

In the following we present a generalization of the runtime analysis of the discussed methods to other "well-behaved" distributions. For reasons of clarity, we focus here only on the analysis of the CUBE METHOD with the orthogonal range searching procedures SCAN, REJECT and SCAN_REJECT.

Let $K = (K_1, \ldots, K_d)$ be the random vector that represents the coordinates of a point $p \in P$ and let $F_K : \mathbb{R}^d \to [0,1]$ be the joint distribution function of the random vector K. We consider the points of P to be drawn independently at random under the following distributions:

a) Independent distribution:
The random variables $K_1, \ldots, K_d$ are independent. We have $F_K(x_1, \ldots, x_d) = \prod_{j=1}^{d} F_{K_j}(x_j)$, where F_{K_j} is the distribution function of the random variable K_j.

b) Positively-bounded distribution(see [15]):
We say that $K = (K_1, \ldots, K_d)$ is positively-bounded over the bounded, convex open region $\mathcal{G}$ if there exists constants $0 < C_1 \leq C_2$ such that $C_1 \leq f_K(x) \leq C_2$ for all $x \in \mathcal{G} \setminus A$ where A has Lebesgue measure $m(A) = 0$ and $f_K(x) = 0$ for $\mathbb{R}^d \setminus \mathcal{G}$. The distribution function has the following property: $C_1 \cdot \boldsymbol{V}(S) \leq \Pr[\, x \in S \,] \leq C_2 \cdot \boldsymbol{V}(S)$ where $S \subseteq \mathcal{G}$ is a region of volume (measure) $\boldsymbol{V}$.

Expected runtime analysis under independent distribution

We assume that the n points of P are drawn independently at random under independent distribution.

For a given query point q we have:

$$V_q(\alpha) \;:=\; \Pr[\, p^i \in C_{q,\alpha}\,] \;=\; \prod_{j=1}^{d} \left(F_{K_j}(q_j + \alpha/2) - F_{K_j}(q_j - \alpha/2) \right) .$$

We introduce the functions $s_j : \mathbb{R}_+ \to [0,1]$ with

$$s_j(\alpha) = F_{K_j}(q_j + \alpha/2) - F_{K_j}(q_j - \alpha/2) \;\; , j = 1, \ldots, d\,. \tag{4.52}$$

Since the functions s_j are continuous and monotone increasing, the function $V_q(\alpha) = \prod_{j=1}^{d} s_j(\alpha)$ is continuous and monotone increasing as well. Thus, we can determine the side length α of the cube $C_{q,\alpha}$ such that

$$\frac{\varphi^*}{n} \leq \Pr\left[\, p^i \in C_{q,\alpha}\,\right] = \prod_{j=1}^{d} s_j(\alpha) < \frac{\varphi^* + 1}{n} \leq 1$$

by binary search, where $\varphi^* \in [1, n-1]$ is the parameter of the algorithm which determines the side length α of the cube $C_{q,\alpha}$.

The permutation $\mathcal{D} = (\mathcal{D}(1), \ldots, \mathcal{D}(d))$ of dimensions is chosen such that

$$s_{\mathcal{D}(1)}(\alpha) \leq \ldots \leq s_{\mathcal{D}(d)}(\alpha)\,.$$

We measure the running time of the CUBE METHOD by the number of comparisons of coordinates and bit array entries. We omit here the computation of the side length α and of the order $\mathcal{D}$ of dimensions, which depends on the complexity of computing the distribution functions F_{K_j}.

Since the points are drawn independently at random, the expected number φ of points in the cube $C_{q,\alpha}$ equals $n \cdot \Pr[\, p^i \in C_{q,\alpha}\,] = n \cdot V_q(\alpha) \in [\varphi^*, \varphi^* + 1)$. Let λ be defined such that

$$\lambda \;=\; \sqrt[d]{\frac{\varphi}{n}} \;=\; \sqrt[d]{V_q(\alpha)}\,.$$

First, we consider the SCAN procedure. The expected number $\mathrm{E}[T_{scan}]$ of tests of coordinates of the SCAN procedure is given by:

$$\mathrm{E}[T_{scan}] \;= n \cdot \left(1 + \sum_{j=1}^{d-1} \Pr\left[\, p^i \in C_{q,\alpha}^{j}\,\right] \right) \;,$$

where $C_{q,\alpha}^{j}$ are obtained by relaxing the side lengths of the cube to unit intervals with respect to those dimensions that are different from the first j ones in the list $\mathcal{D}$. The probability $\Pr\left[\, p^i \in C_{q,\alpha}^{j}\,\right]$ equals $\prod_{l=1}^{j} s_{\mathcal{D}(l)}(\alpha)$. By (2.5) and by Lemma 2.2, we obtain the same results as in Section 2.1.2:

$$\mathrm{E}[T_{scan}] \;\leq\; \frac{n}{1-\lambda} \;=\; \frac{n}{1 - \sqrt[d]{\frac{\varphi}{n}}} \;\leq\; \frac{nd}{\ln(n/\varphi)} + n\,.$$

We proceed in analogy to the analysis in Section 2.1.4 and we obtain that the CUBE METHOD with the searching procedure SCAN performs an expected number $O(\frac{nd}{\ln n} + n)$ of comparisons of coordinates.

The expected number $\mathrm{E}[T_{reject}]$ of comparisons of coordinates and bit array entries of the procedure REJECT is given by

$$\begin{aligned} n + 2d + n \cdot \sum_{j=1}^{d} (1 - s_j(\alpha)) &\leq n + 2d + nd(\ 1 - \sqrt[d]{s_1(\alpha) \cdot \ldots \cdot s_d(\alpha)}\ ) \\ &= n + 2d + nd(\ 1 - \lambda\ )\ , \end{aligned}$$

Since $\lambda = \sqrt[d]{\frac{\varphi}{n}}$, we obtain, by Lemma 2.3, $\mathrm{E}[T_{reject}] \leq n + 2d + n\ln(n/\varphi)$. The rest of the analysis of the CUBE METHOD with the searching procedure REJECT is done as discussed in Section 3.1.

We skip the details of the running time analysis of the procedure REJECT_SCAN. Its expected number $\mathrm{E}[T_{rej\_scan}]$ of comparisons of coordinates and bit array entries is given in (3.25), where the functions $s_j(\alpha)$ are defined in (4.52), instead of representing the side lengths of the box $C_{q,\alpha} \cap [0,1]^d$. The crucial observation is that the upper bound on $\mathrm{E}[T_{rej\_scan}]$ obtained in Section 3.1.2 can be expressed as function of $\lambda = \sqrt[d]{\frac{\varphi}{n}}$ (see (3.34) and (3.35)), thus, it holds also in the case when we assume independent distribution of the points. Thus, the CUBE METHOD with the searching procedure REJECT_SCAN performs an expected number of $O\left(n\ln(\frac{d}{\ln n} + 1) + n\right)$ comparisons of coordinates and bit array entries.

Expected runtime analysis under positively-bounded distribution

We assume w.l.o.g. $P \subseteq [0,1]^d$ and $\mathcal{G} = (0,1)^d$. We consider the n points of P to be drawn independently at random under positively-bounded distribution. In the following we use the notations $C^j_{q,\alpha}$, T_{scan}, T_{reject} and T_{rej_scan} as introduced above.

We consider the procedure REJECT to determine the point set $C_{q,\alpha} \cap P$. For each point $p^i \in P$ we obtain

$$\Pr\left[\, p^i \in C^j_{q,\alpha}\,\right] = \Pr\left[\, p^i \in W^j_{q,\alpha}\,\right]\ ,$$

where $W^j_{q,\alpha}$ is the box $C^j_{q,\alpha} \cap [0,1]^d$. Therefore,

$$C_1 \cdot \boldsymbol{V}(W^j_{q,\alpha}) \leq \Pr\left[\, p^i \in C^j_{q,\alpha}\,\right] \leq C_2 \cdot \boldsymbol{V}(W^j_{q,\alpha}) \tag{4.53}$$

and

$$C_1 \cdot \prod_{l=1}^{j} s_l(\alpha) \leq \Pr[p^i \in C^j_{q,\alpha}\,] \leq C_2 \cdot \prod_{l=1}^{j} s_l(\alpha)\ ,$$

where the function $s_l(\alpha)$ represents the side length of $W_{q,\alpha} = C_{q,\alpha} \cap [0,1]^d$ in dimension $l \in \{1,\ldots,d\}$ and is given by (2.3). The side length α is determined by the procedure SIDE_LENGTH, as described in Section 2.1.3, such that

$$\frac{\varphi^*}{n} \leq \boldsymbol{V}(\,W_{q,\alpha}\,) = \prod_{j=1}^{d} s_j(\alpha) < \frac{\varphi^* + 1}{n} \leq 1\ , \tag{4.54}$$

where φ^* is a real parameter in $[1, n-1]$ of SIDE_LENGTH. By (4.53) and (4.54) we have for $i \in \{1,\ldots,n\}$:

$$C_1 \cdot \frac{\varphi^*}{n} \leq \Pr\left[\, p^i \in C_{q,\alpha}\,\right] < C_2 \cdot \frac{\varphi^* + 1}{n}\ .$$

We consider the order $\mathcal{D}$ of dimensions to be defined as mentioned in Section 2.1.2 , to correspond to the increasing order of the side lengths of the box $W_{q,\alpha}$.

By (2.2) and Lemma 2.2, we obtain:

$$\begin{aligned} \mathrm{E}[T_{scan}] &\leq \sum_{j=0}^{d-1} \Pr\left[\, p^i \in C^j_{q,\alpha} \,\right] \leq C_2 \cdot \sum_{j=0}^{d-1} \lambda^j \\ &\leq C_2 \cdot \left(\frac{nd}{\ln\left(n/(\varphi^*+1)\right)} + n \right) \end{aligned}$$

where $\lambda = \sqrt[d]{\boldsymbol{V}(W_{q,\alpha})} < \sqrt[d]{\frac{\varphi^*+1}{n}}$. For the probability of an empty cube $C_{q,\alpha}$ we have $\Pr(P \cap C_{q,\alpha} = \emptyset) \leq (1 - C_1 \frac{\varphi^*}{n})^n \leq e^{-C_1 \varphi^*}$. Note that , by (4.53), we have $C_1 \frac{\varphi^*}{n} \leq 1$. To obtain a total asymptotic runtime of $O\left(\frac{nd}{\ln(n/\varphi^*)} + n + d \ln d\right)$, we guarantee

$$e^{-C_1 \varphi^*} \cdot nd \leq C_2 \cdot \frac{nd}{\ln\left(n/(\varphi^*+1)\right)} \tag{4.55}$$

which is satisfied for $\varphi^* \geq \frac{1}{C_1} \cdot \ln(\frac{1}{C_2} \ln n)$. We choose $\varphi^* = \frac{1}{C_1} \cdot \ln(\frac{1}{C_2} \ln n)$, and we obtain that the expected asymptotic runtime of the CUBE METHOD with SCAN is $O(\frac{nd}{\ln n} + n + d \ln d)$ if the points are drawn independently at random from $[0,1]^d$ under positively-bounded distribution.

We consider the procedure REJECT to determine the point set $C_{q,\alpha} \cap P$. The expected number $\mathrm{E}[T_{reject}]$ of comparisons of coordinates and bit array entries of the procedure REJECT is given by

$$\begin{aligned} \mathrm{E}[T_{reject}] &= n + 2d + n \cdot \sum_{j=1}^{d} \Pr[\, p^i_j \notin [q_j - \alpha/2, q_j + \alpha/2] \,] \\ &\leq n + 2d + \sum_{j=1}^{d} n \cdot C_2 \cdot (1 - s_j(\alpha)) \\ &\leq n + 2d + C_2 \cdot nd(\, 1 - \sqrt[d]{s_1(\alpha) \cdot \ldots \cdot s_d(\alpha)} \,) \\ &= n + 2d + C_2 \cdot n \cdot \ln(n/\varphi^*) \; . \end{aligned}$$

We used (4.54) and Lemma 2.2.

As mentioned above we have the upper bound $\Pr(P \cap C_{q,\alpha} = \emptyset) \leq e^{-C_1 \varphi^*}$. Thus, the expected runtime of the CUBE METHOD with REJECT is $O\left(n + n \ln(n/\varphi^*) + e^{-C_1 \varphi^*} nd + \varphi^* d + d \ln d \right)$. A suitable value of φ^* is $\max\left\{\frac{n}{d}, \ln(C_1 n)/C_1\right\}$. The expected asymptotic runtime of the CUBE METHOD with REJECT is $O\left(n \ln d + d \ln d\right)$ if the points are drawn independently at random from $[0,1]^d$ under positively-bounded distribution.

The expected runtimes of the procedures SCAN and REJECT can be bounded in terms of the volume of the box $W_{q,\alpha}$, and the constants C_1 and C_2. This can be achieved also in the analysis of the procedure REJECT_SCAN by combining the above results. We obtain in a straightforward manner that the procedure REJECT_SCAN has an expected runtime of $O\left(n \ln(\frac{d}{\ln n} + 1) + n + d \ln d\right)$, assuming that the points are drawn independently at random from $[0,1]^d$ under positively-bounded distribution.

Theorem 4.4. *The* CUBE METHOD *with one of the orthogonal range searching procedures* SCAN, REJECT *and* SCAN_REJECT *has the expected asymptotic runtime given in Theorem 2.2, Theorem 3.1 and Theorem 3.2, respectively, if the n points of P are drawn independently at random under independent distribution or positively-bounded distribution.*

4.4 External-Memory Nearest-Neighbor Search

Many database applications involve massive datasets, which do not fit in the main memory of modern machines. In such cases the data structures need to be stored on external storage devices such as disks. A major performance bottleneck is the Input/Output (I/O) communication between fast internal-memory and slow external storage. Minimizing the I/O communication is the essential problem considered in the design of external-memory data structures and algorithms.

First we define the I/O model of computation and then present an external-memory variant of the CUBE METHOD and of the CELL METHOD.

4.4.1 The External-Memory Model of Computation

Modeling accurately memory and disks systems is a complex task which has been broadly surveyed. We shall work in the external-memory model of computation introduced by Aggarwal and Vitter [2]. This standard two-level disk model focuses on the fact that typical disks read or write large blocks of contiguous data at once, in order to amortize the access time over a large amount of data. The model has the following parameters [2, 65, 47]:

$$\begin{aligned} N &= \text{number of objects in the problem instance} \\ T &= \text{number of objects in the problem solution} \\ M &= \text{number of objects that can fit into internal memory} \\ B &= \text{number of objects per disk block;} \end{aligned}$$

where $B < M < N$. An I/O operation is the reading or writing a block from or into disk, respectively (see Figure 4.1). Computation is only performed on objects in internal memory. The measures of performance in this model are the number of I/Os used to solve a problem, as well as the amount of space (disk blocks) used and the internal memory computation time.

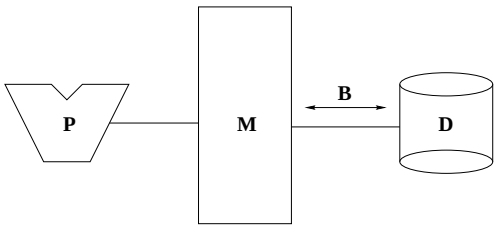


Figure 4.1: An I/O moves B contiguous elements between disk and main memory

More complex multi-level memory models have been considered in the literature, where several disks are used in parallel to increase the performance of I/O systems. We refer to a recent survey by Vitter [64].

We primarily want to model the extremely long disk access time relative to the internal running time. Therefore, we ignore the internal computation time in our algorithmic performance analysis.

4.4.2 The External-Memory Data Structure

We investigate the L_∞-nearest-neighbor problem in the external-model of computation. Here an object or item is a coordinate p^i_j ($1 \leq j \leq d$) of some data point $p^i \in P = \{p^1, \ldots, p^n\}$ or an index $i \in \{1, \ldots, n\}$ of some data point p^i. The number of objects of the problem instance is $N = nd$.

We consider the GROWING-CUBE method introduced in Section 4.1 and we compute the expected number of I/O operations performed by the external variant of this method to determine the nearest neighbor to a query point q from the data set P. We assume that the data points $p^1, \ldots, p^n$ are drawn independently at random under uniform distribution.

We compare the external GROWING-CUBE with the external brute-force method, which stores the points with their coordinates successively in contiguous locations of the external memory. The external brute-force method loads the data from the disk and determines the nearest neighbor to the query with $\Theta(nd/B)$ I/O operations.

The GROWING-CUBE method considers cubes C_{q,α_t} of increasing side lengths α_t around the query point q until a cube containing points of P is found. In order to determine the points of a cube $C_{q,\alpha}$ expected to contain φ points, the GROWING-CUBE method calls the orthogonal searching algorithm REJECT. This algorithm is the only one among the presented searching procedures having an efficient extension in the external-memory model of computation. In the case of the REJECT procedure, the objects to be tested in the main memory are stored in contiguous locations in disk blocks. More precisely, the external data structure consists mainly of the arrays T_j ($j = 1, \ldots, d$), introduced in Section 3.1.1. For a dimension $j \in \{1, \ldots, d\}$ the entries $T_j[i]$ ($i = 1, \ldots, n$) are stored successively in contiguous locations on the disk. The i-th entry is $T_j[i] = (p_j^l, l)$, where p_j^l is the i-th largest coordinate in the dimension j and l is the index of the corresponding point p^l. Additionally, the external data structure contains all points p^i of P such that all coordinates p_j^i for some point p^i are stored successively in contiguous locations of the external memory. The external data structure is stored in $O(nd/B)$ blocks on the disk.

First we analyze the expected I/O complexity of the REJECT procedure, which is required to determine the set $P \cap C_{q,\alpha}$ of the points contained in a cube $C_{q,\alpha}$, itself expected to contain φ data points.

The expected I/O complexity of testing coordinates

The external REJECT procedure consists essentially of reading blocks containing objects $T_j[i] = (p_j^l, l)$ for each dimension j and testing in the main memory whether the corresponding coordinates p_j^l are still out of the interval $I_j = [q_j - \frac{\alpha}{2}, q_j + \frac{\alpha}{2}]$. The objects are read exactly in the same order as the order considered by the internal REJECT procedure described in Section 3.1.1 : $T_j[1], \ldots, T_j[l]$ for some index $1 \leq l \leq n$ and $T_j[u], \ldots, T_j[n]$ for some index $1 \leq u \leq n$ (see Figure 4.2). Thus, for a given dimension j, almost all data blocks which are read from the disk contain exclusively relevant data, which needs to be tested in the main memory. In fact, for each dimension j, there might be at most two read blocks that contain also coordinates which are not of interest.

The number of read operations required by REJECT to test the coordinates of points is bounded by

$$\sum_{j=1}^{d} \left(2 + \frac{2 \cdot [\,(\#\text{points outside } I_j) \;+\; 2\,]}{B}\right) \;\leq\; 2d + 2 \cdot \sum_{j=1}^{d} \frac{\#\text{points outside } I_j \;+\; 2}{B}$$

The additional multiplicative factor 2 comes from the fact that an array entry $T_j[i]$ contains 2 objects. Let s_i be the side lengths of the intervals I_j ($1 \leq j \leq d$). By (3.1) and (3.2), we get $E\,[\,\#\text{points outside } I_j\,] \leq n \ln(n/\varphi)$. By using the linearity of the expectation, we obtain that the expected number of I/O transfers performed by REJECT in order to make the necessary test of coordinates is bounded by

$$2d + 2\frac{n \ln(n/\varphi) + 2d}{B}\,, \tag{4.56}$$

where φ is the expected number of points of P contained in the cube $C_{q,\alpha}$.

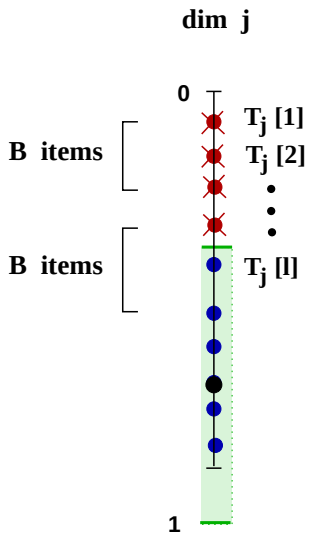


Figure 4.2: Almost all blocks contain exclusively relevant data.

The expected I/O complexity of marking rejected points

The internal searching algorithm REJECT uses a bit array to mark the rejected points, in order to determine the points inside the cube $C_{q,\alpha}$. A realistic assumption is that this bit array fits in the main memory. Under this hypothesis the total I/O complexity of the REJECT procedure to determine the points of P contained in the cube $C_{q,\alpha}$ is $O\left(d + \frac{n \ln(n/\varphi)}{B}\right)$.

In case the bit array has to be stored in external memory, we proceed as follows to mark the rejected points. While testing coordinates, the algorithm fills up a part of the main memory with indices of points which have to be marked as rejected. This part of the memory consists of contiguous locations and has size $\Theta(M)$. After the fill-up phase we determine the blocks of the bit array we need for the marking process and load them into a part of the main memory of size $\Theta(M)$, consisting of free contiguous locations. The bit entries of interest are set to 1 in the blocks loaded in the main memory. If necessary we repeatedly replace those blocks that are not relevant in the current phase anymore. These fill-up and marking phases are repeated until all coordinates were tested. Blocks of the bit array that are already in the memory due to a previous phase can be reused. The worst case situation is that the expected number of I/O operations required to mark the rejected points is linear in the number of rejecting steps, that is $O(n \ln(n/\varphi) + d)$. This is based on the fact that loaded blocks of the bit array may contain only one entry of interest.

We propose an alternative procedure to determine the unrejected points. Instead of using a bit array, we gather the indices of the rejected points during the process of testing coordinates. This is done by repeatedly filling up the internal memory, sorting the indices such that each index appears only once, and writing those indices in contiguous locations of the external memory. We end up having $O\left(\frac{n \ln(n/\varphi)+d}{B}\right)$ blocks, each of them containing sorted indices. By using the external merge sort algorithm introduced by Aggarwal and Vitter [2], we obtain a list R with the indices of the rejected points in increasing order in $O\left(\frac{n \ln(n/\varphi)+d}{B} \cdot \log_{M/B}\left(\frac{n \ln(n/\varphi)+d}{B}\right)\right)$ I/Os. This list is stored in $O\left(\frac{n}{B}\right)$ blocks in the external memory. The list R is scanned and we determine in $O\left(\frac{n}{B}\right)$ I/Os the missing indices $i \in \{1, \ldots, n\}$ from the list R. The missing indices are stored in $O\left(\frac{n}{B}\right)$ blocks in the external memory. They represent the points contained in the cube $C_{q,\alpha}$ and are the result of the external REJECT procedure.

We take a closer look at the above bound $O\left(\frac{n\ln(n/\varphi)+d}{B}\cdot\log_{M/B}\left(\frac{n\ln(n/\varphi)+d}{B}\right)\right)$. For typical values of B and M the base M/B of the logarithm is large, in the order of 10^4. Thus, $\log_{M/B}\left(\frac{n\ln(n/\varphi)+d}{B}\right)$ is less than 4 for all realistic values of n, d and M/B. Thus, in the above bound the important term is not the log-term, but the B-term in the denominator of $\frac{n\ln(n/\varphi)+d}{B}$. This observation represents an important feature of the external sorting.

The total I/O complexity

We consider in the following the realistic assumption that the bit array fits in the main memory. In this case, the REJECT procedure performs expected at most $2d+2\frac{n\ln(n/\varphi)+2d}{B}$ I/Os to determine the points of P contained in some cube $C_{q,\alpha}$, where $\varphi = \mathrm{E}[\,|P\cap C_{q,\alpha}|\,]$.

The GROWING-CUBE method works as described in Section 4.1 and we choose the sequence $\{\varphi_t^*\}_{t\geq 1}$ such that $\varphi_t^* = \min\{\varphi+(t-1), n\}$. We specify the value of the parameter φ later. The expected number φ_t of points in the cube C_{q,α_t} fulfills $\varphi_t^* \leq \varphi_t < \varphi_t^*+1$, where $1\leq t\leq t_{\max}$ and $t_{\max}=\min\{t\mid \varphi_t^*\geq n\}$.

Let $I_j^t = [q_j-\frac{\alpha_t}{2}, q_j+\frac{\alpha_t}{2}]$. By using (4.28) and (4.21), we obtain the following upper bound on the expected number of I/Os performed by GROWING-CUBE during the REJECT phase:

$$
\begin{aligned}
&\sum_{t=1}^{t_{\max}-1}\Pr\left(X^{t-1}=0\right)\cdot\left(2d+2\cdot\sum_{j=1}^{d}\frac{\mathrm{E}\left[\#\text{points outside } I_j^t \mid X^{t-1}=0\right]+2}{B}\right)\\
&\quad\leq\sum_{t=1}^{t_{\max}-1}\left(1-\frac{\varphi_{t-1}}{n}\right)^n\cdot\left(2d+\frac{2}{B}\cdot\left(\frac{n\ln(n/\varphi)}{1-\frac{\varphi_{t-1}}{n}}+2\right)\right)\\
&\quad\leq\sum_{t=1}^{t_{\max}-1}\left(1-\frac{\varphi_{t-1}}{n}\right)^{n-1}\cdot\left(2d+2\cdot\frac{n\ln(n/\varphi)+2d}{B}\right)\\
&\quad=\left(2d+2\cdot\frac{n\ln(n/\varphi)+2d}{B}\right)\cdot\left(1+\sum_{t=1}^{t_{\max}}\left(1-\frac{\varphi_t}{n}\right)^{n-1}\right)\\
&\quad\leq\left(2d+2\cdot\frac{n\ln(n/\varphi)+2d}{B}\right)\cdot\left(1+\sum_{t=1}^{t_{\max}}e^{-\varphi_t\frac{n-1}{n}}\right)\;\leq\;c_1\cdot\left(d+\frac{n\ln(n/\varphi)+2d}{B}\right)\;,
\end{aligned}
$$

where $2<c_1<3$.

Next we look at the expected number of I/O operations performed in the brute-force part of the external GROWING-CUBE method. If the current cube C_{q,α_t} is the first cube not to be empty, then the brute-force method is called, which loads the coordinates of the data points $p^i\in C_{q,\alpha_t}$ and determines the minimum distance of these points to the query. Since the data points contained in C_{q,α_t} are not necessarily stored successively in the external memory, the brute-force method loads each point separately and requires $\lceil d/B\rceil$ I/O operations per loaded point in the worst case. We proceed in analogy to (4.8) and (4.22), and obtain that the expected number of I/Os performed in the brute-force part of the GROWING-CUBE is bounded by

$$c_2\cdot\varphi\cdot\lceil d/B\rceil\;\leq\;c_2\cdot\varphi\cdot\frac{d}{B}+c_2\cdot\varphi$$

for an appropriate constant $1<c_2<2$.

A realistic assumption is that a data point or a query point fits in the main memory. In this case the computation of the side lengths α_t can be done entirely in the main memory since they depend only on the

query point and the values φ_t^*. If $d > B$ then by (4.19) the computation of the side lengths α_t requires expected $O(d\ln(\varphi d)/B + \ln(\varphi d))$ I/O operations. This is based on the fact that as seen in Section 2.1.3 the computation of α takes $O(\ln(\varphi d))$ steps. Each step involves an evaluation of the actual volume which takes $O(\lceil d/B \rceil)$ I/O operations.

Altogether, we obtain the following upper bound on the number of I/Os required by the external GROWING-CUBE to compute the nearest neighbor to the query point q:

$$O\left(\frac{n\ln(n/\varphi)}{B} + \varphi \cdot \frac{d}{B} + \frac{d\ln d}{B} + \varphi + d\right)$$

Since the function $f(\varphi) := \frac{n\ln(n/\varphi)}{B} + \varphi \cdot \frac{d}{B} + \varphi$ takes its minimum at $\frac{n}{d+B}$, we set the parameter φ to equal $\max\{\frac{n}{d+B}, 1\}$.

Summarizing, we obtain that the external GROWING-CUBE method finds the nearest neighbor from the data set P to the query point q in $O\left(\frac{n\ln(d+B)}{B} + \frac{d\ln d}{B} + d\right)$ I/O operations.

The external GROWING-CUBE method compares favorably with the external brute-force method if $n > B$ and $d > \ln(d+B)$, which is a realistic assumption in the external-memory model, where the data points do not fit in the main memory.

4.5 Experiments

We present an experimental comparison of implementations of the presented variants of the CUBE METHOD and GROWING-CUBE method. We emphasize the factors by which the algorithms are faster than the brute-force method.

We test our algorithms on random test data. We generate data sets in $[0,1]^d$, each of size n, according to the uniform distribution. For each data set a query point is chosen randomly in $[0,1]^d$. For each of the considered settings of n and d we generate 20 pairs, each consisting of a random data set and a random query point. Based on those 20 pairs we compute for each considered setting of n and d the average speedup of our algorithms with respect to the brute-force method.

In our first experiments we consider query algorithms that do not need preprocessing. Figure 4.3 shows an experimental comparison for dimension $d = 100$ and number of points n between 1000 and 10000. The value on the vertical scale is the factor by which each algorithm is faster then the brute-force method. The speedup factors of the CUBE METHOD and the GROWING-CUBE method using the SCAN searching procedure are represented by the curves *cube* and *grow*, respectively. The curves *adaptcube* and *adaptgrow* illustrate the speedup factors of the methods when the range searching procedure ADAPTIVE_SCAN, introduced in Section 2.2, is used. The speedup factors of the algorithms measured in practice show a similar behavior to the speedup factor $\Theta(\ln n)$ presented in Chapter 2. As expected the ADAPTIVE_SCAN improves the query algorithms. The observed improvement of the GROWING-CUBE method over the CUBE METHOD is based on the fact that the GROWING-CUBE method starts with a smaller cube.

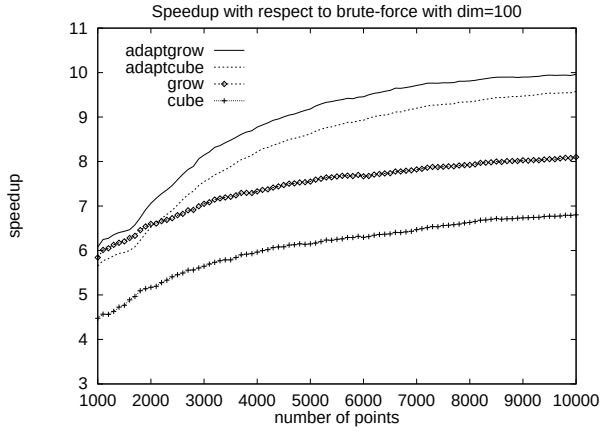


Figure 4.3: Comparison of the CUBE method and GROWING-CUBE method with the brute-force method

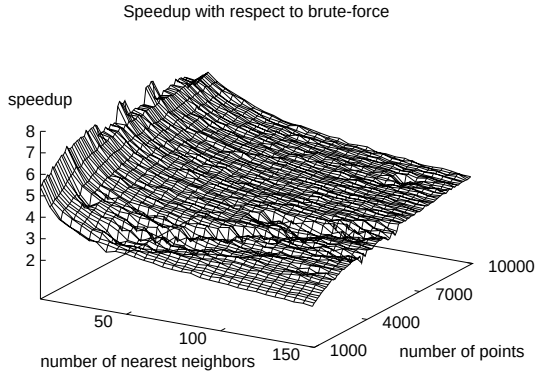


Figure 4.4: Comparison for k nearest-neighbor search of the GROWING-CUBE method using SCAN with the brute-force method

Figure 4.4 shows a comparison of the GROWING-CUBE method for k-nearest-neighbor search with the brute-force method, which computes all distances of the points of P to the query point and selects the k smallest distances among them. The GROWING-CUBE method performs the orthogonal range searching of the considered cubes by the searching algorithm SCAN. The experiments have been carried out for dimension $d = 100$, number of points n between 1000 and 10000, and number of nearest neighbors k between 5 and 150. The experimental comparison reflects the speedup factor $\Theta(\ln(n/k))$ proven in Section 4.2.2.

Next, we consider the GROWING-CUBE method with the searching procedures REJECT and REJECT_SCAN, and compare it with the brute-force method and the GROWING-CUBE method when using the procedure ADAPTIVE_SCAN.

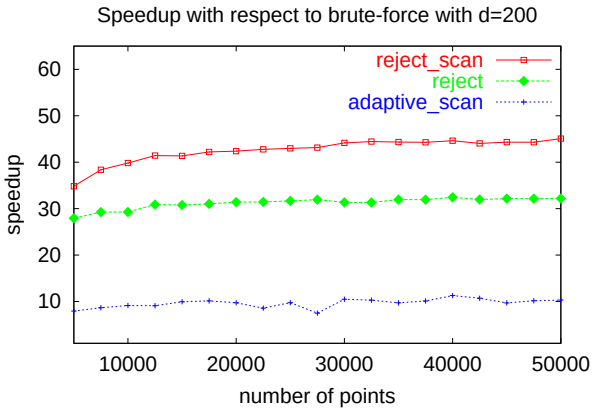


Figure 4.5: Speedup factors for fixed dimension d

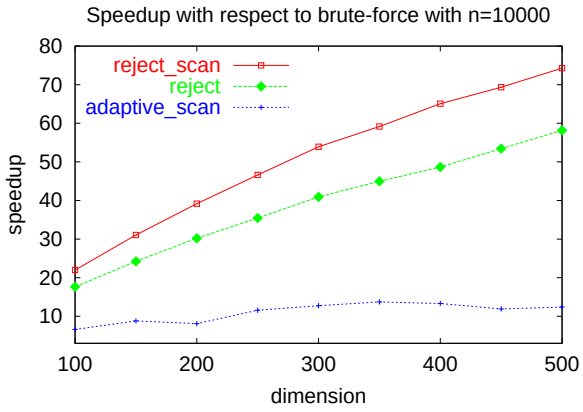


Figure 4.6: Speedup factors for fixed number of points n

Figure 4.5 shows an experimental comparison of the searching algorithms for dimension $d = 200$ and sizes n of the data set between 10000 and 50000. Figure 4.6 shows the experimental comparison of the searching algorithms for $n = 10000$ and dimension d varying between 100 and 500. The value on the vertical scale is the factor by which each algorithm is faster then the brute-force method. Both figures illustrate that for high dimensions the methods using preprocessing dominate clearly the GROWING-CUBE method with the ADAPTIVE_SCAN searching procedure. Furthermore, the REJECT_SCAN procedure is better than the procedure REJECT as expected from the runtime analysis presented in Section 3.1. The speedup factor of the GROWING-CUBE method with the REJECT_SCAN procedure is $O(\frac{d}{\ln(d/\ln n+1)})$ with respect to the brute-force method. Figure 4.6 points out the dependency of the speedup on the dimension d for the methods using preprocessing. A speedup factor above 30 is a considerable improvement in practice.

We conclude with the experimental comparison for k nearest-neighbor search. In Figure 4.7 the GROWING-CUBE method using REJECT_SCAN is compared with the brute-force method. The experiments have been carried out for number of points $n = 10000$, dimension d between 100 and 500, and number of nearest neighbors k between 5 and 150. A growth of the speedup factor with the dimension d can be seen. Even for big values of the number k of nearest neighbors the speedup factor of the GROWING-CUBE with respect to the brute-force method is between 10 and 30.

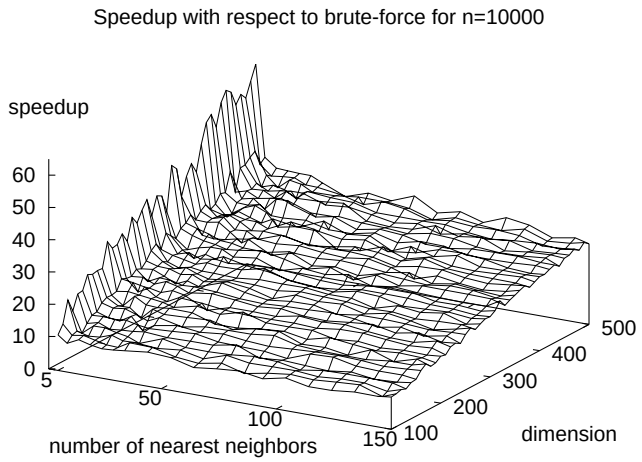


Figure 4.7: Comparison for k nearest-neighbor search of the GROWING-CUBE method using REJECT_SCAN with the brute-force method