RC 21837 (98265) 15 September 2000 **ComputerScience**

IBM Research Report

Multidimensional Interval Trees

Ulrich Finkler
IBM T. J. Watson Research Center
P. O. Box 218
Yorktown Heights, NY 10598

Irena Pashchenko
Smith College
Northampton, Massachusetts

Research Division
Almaden - Austin - Beijing - Haifa - T. J. Watson - Tokyo - Zurich

Multidimensional Interval Trees

Ulrich Finkler* Irena Pashchenko†

Abstract

A multidimensional interval tree supports orthogonal range searching on d-dimensional intervals (i.e., isothetic hyper-parallelepipeds). The problem is also known as 'rectangle intersection searching' or 'orthogonal intersection searching'. In two dimensions it requires $O(n)$ space and $O(n\log(n))$ construction time. Finding all k rectangles in a set of n rectangles that intersect a given query rectangle takes $O(\sqrt{n}+k\log(n))$ time.

In d dimensions a multidimensional interval tree requires $O(d^2n)$ space and $O((d+\log(n))dn)$ time for construction. An orthogonal range search finding k d-dimensional intervals intersecting the d-dimensional query-interval takes $O(d^2 4^d n^{1-1/d} + d(d+log(n))k)$ time.

1 Introduction

Let S be a set of d-boxes $([a_0,b_0]\times[a_1,b_1]\times\ldots\times[a_{d-1},b_{d-1}])$ in a d-dimensional universe $U_0\times U_1\ldots\times U_{d-1}$ (we use 'd-box' as a shortcut for *d-dimensional isothetic hyper-parallelepiped*). Given a query d-box $q\in U_0\times\ldots U_{d-1}$, the problem of finding a set $A=\{v\in S\mid v\cap q\neq\{\}\}$ of elements that intersect q is called *orthogonal range searching* or *windowing*. This type of problem arises in computational geometry as well as in databases, where U_i denotes a set of properties with a linear order, e.g. last names and dates of birth.

In VLSI design the problem of orthogonal range searching on sets of rectangles is an essential component of computations regarding the lithographic masks of microchips. Layouts of recent microprocessors contain 10^8 shapes or more, even in a hierarchical representation, and require an efficient solution for range searching using linear space.

In the early days of computational geometry *orthogonal range searching* was one of the most important topics. This lead to a large number of results, but most of the reported data structures target special cases. Either the elements of S are reduced to points (range searching on sets of points) or the query is reduced to a point (stabbing queries). For an overview of these structures with results and analysis refer to [12],[13],[11].

One of the first data structures dealing with multiple dimensions was the *quad-tree*, developed by Finkel and Bentley in 1974 [1]. Every node in a quad-tree corresponds to a rectangle, the children of a node correspond to the equal quadrants of this rectangle. Already for *orthogonal range searching on sets of points* quad trees may become quite unbalanced. The depth of a quad tree on a set of N points is $O(N)$ in the worst case.

Kd-trees, first described by Bentley [2], solve the problem of balancing by partitioning the points into sets of equal size with respect to alternating directions. A 2d-tree answers an orthogonal range query in the plane on a set of N points returning k 'hits' in $O(\sqrt{N}+k)$ time. The *range tree* reduces the query time to $O(\log^2(N)+k)$ using $O(N\log(N))$ space [4], the query time was reduced to $O(\log(N)+k)$ by *fractional cascading* [5]. All these structures have extensions to higher dimensions. Chazelle described a structure for the two-dimensional case that reduces the storage to $O(N\log(N)/\log(\log(N)))$ maintaining the query time of $O(\log(N)+k)$ and showed that this is optimal [6][7].

Edelsbrunner [9] and McCreight [8] introduced the *interval tree*. An interval tree operates on one-dimensional intervals, requires $O(N)$ space and allows to find all intervals in a set S that intersect a query interval q in $O(\log(N)+k)$ time. There are combinations of priority trees and intervals trees etc. that solve the problem for axis-parallel line segments [12]. An extension of the segment tree [3] solves orthogonal range searching on a set of disjoint line segments in the plane with $O(N\log(N))$ storage in $O(\log^2(N)+k)$ time [12]. There is no higher dimensional version of an interval tree described in the literature [12]. Edelsbrunner and Maurer reported on how to combine range, segment and interval trees to solve the *rectangle intersection search problem* in $O(\log^{d-1}(N)+k)$ time and $O(N\log^d(N))$ space [10]. There is no data structure described in the literature that solves the orthogonal range query for sets of d-boxes using linear space and less than $O(N)$ time.

In this paper we introduce *multidimensional interval trees*. Two-dimensional interval trees solve the problem of orthogonal range searching in two dimensions in $O(\sqrt{N}+k\log(N))$ query time and $O(N)$ space. Construction requires $O(N\log(N))$ time. Additionally, the

*IBM T.J. Watson Research Center, Yorktown Heights, New York.

†Smith College, Northampton, Massachusetts.

individual universes U_i do not require arithmetic, only a comparison defining a linear order on the elements is necessary. Many descriptions of interval trees for example require the computation of the center of an interval for the determination of separators. Compared to these, our definition of multidimensional interval trees is feasible for database applications with very general data types.

The extension of two-dimensional interval trees to higher dimensions is analogous to the extension of 2d-trees to Kd-trees. In d dimensions a multidimensional interval tree requires $O(d^2N)$ space and $O((d + \log(N))dN)$ time for construction. An orthogonal range search takes $O(d^2 4^d N^{1-1/d} + d(d + log(N))k)$ time.

Since orthogonal range searching on d-boxes is decomposable – the solution of a query on $S = A \cup B$ is the union of the solutions to identical queries on A and B – dynamization is possible and analogous to the dynamization of Kd-trees.

In section 2, 3 and 4 we define the multidimensional interval tree and analyze the orthogonal range search for $d = 2$ dimensions. Section 5 covers the generalization into d dimensions.

2 The data structure

The two-dimensional case covers all crucial concepts of multidimensional interval trees. In the following descriptions we map the universes U_0 and U_1 to orthogonal coordinates in the plane, if necessary by introducing an artificial metric, to allow for easier visualization.

A 'two-dimensional interval tree' consists of three different types of trees, in the following called *primary*, *secondary* and *tertiary* trees. The primary tree partitions the plane in alternating directions analogously to a 2d-tree. Each *primary* node contains a *secondary tree*, which stores rectangles intersecting its 'separation line' similar to an interval tree. Each *secondary* node contains two tertiary trees analogous to the ordered extrema lists of an interval tree.

DEFINITION 2.1. *A* binary tree with orthogonal extensions B_0 *on a set*

$$S = \{\tilde{v} = (x_v, [y_v, z_v]) \,|\, x_v \in U_0 \ \wedge \ [y_v, z_v] \subseteq U_1\}$$

with

$$x_v \neq x_w \ \forall \ \tilde{v} \neq \tilde{w},\ \tilde{v}, \tilde{w} \in S$$

and

$$[y_v, z_v] \cap [y_w, z_w] \neq \{\} \ \forall \ \tilde{v}, \tilde{w} \in S$$

consists of a root node t with

- *an element $\tilde{t} = (x_t, [y_t, z_t]) \in S$*
- *a left child l, which is the root of a binary tree with orthogonal extensions $B_{0<}$ on the set*

$$S_< = \{\tilde{v} \in S \mid x_v < x_t\}$$

- *a right child r, which is the root of a $B_{0>}$ on*

$$S_> = \{\tilde{v} \in S \mid x_v > x_t\}$$

- *an* orthogonal extension

$$[\alpha_t, \beta_t] = [y_t, z_t] \cup [\alpha_l, \beta_l] \cup [\alpha_r, \beta_r]$$

such that $[\alpha_l, \beta_l]$ and $[\alpha_r, \beta_r]$ are the orthogonal extensions of l and r, respectively.

An empty binary tree with extensions consists out of the node NIL. *The extension of an empty tree (or* NIL*) is the empty set. B_0 is called ordered in U_0 with extensions in U_1. B_1 is defined analogously.*

DEFINITION 2.2. *An* interval tree with orthogonal extensions I_0 *on a set*

$$S = \{\vec{v} \,|\, [p_v, q_v] \subseteq U_0 \ \wedge \ [y_v, z_v] \subseteq U_1\}$$

$$\vec{v} = ([p_v, q_v] \times [y_v, z_v])$$

with

$$[y_v, z_v] \cap [y_w, z_w] \neq \{\} \ \forall \ \vec{v}, \vec{w} \in S$$

consists of a root node t with

- *a separator $\delta_t \in U_0$*
- *a left child l, which is the root of an interval tree with orthogonal extensions $I_{0<}$ on the set*

$$S_< = \{\vec{v} \in S \mid q_v < \delta_t\}$$

- *a right child r, which is the root of $I_{0>}$ on*

$$S_> = \{\vec{v} \in S \mid p_v > \delta_t\}$$

- *two binary trees with orthogonal extension B_{0L} and B_{0R}, called extrema trees, such that*

$$S_= = \{\vec{v} \in S \mid p_v \leq \delta_t \leq q_v\}$$

$$S_{=p} = \{(p_v, [y_v, z_v]) \mid \vec{v} \in S_=\}$$

$$S_{=q} = \{(q_v, [y_v, z_v]) \mid \vec{v} \in S_=\}$$

  - *B_{0L} is a binary tree with extensions on $S_{=p}$ ordered in U_0*
  - *B_{0R} is a binary tree with extensions on $S_{=q}$ ordered in U_0*

such that each node holds a reference to the two-dimensional interval that defined its key.

- *the orthogonal extension*

$$[\alpha_t, \beta_t] = [\alpha_l, \beta_l] \cup [\alpha_r, \beta_r] \cup \bigcup_{v \in S_=} [y_v, z_v]$$

such that $[\alpha_l, \beta_l]$ and $[\alpha_r, \beta_r]$ are the orthogonal extensions of l and r, respectively.

An empty interval tree with orthogonal extensions consists of the node NIL. *The extension of an empty tree (or* NIL*) is the empty set. I_0 is called ordered in U_0 with extensions in U_1. The interval tree with orthogonal extension I_1 is defined analogously.*

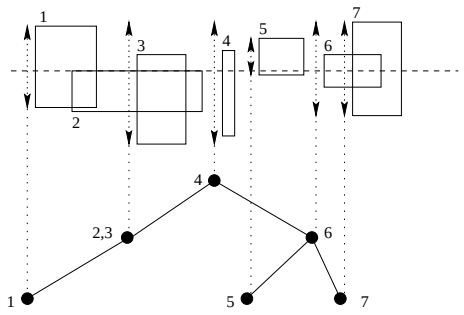

Figure 1: An interval tree with orthogonal extension on a set of rectangles.

The definition of the interval tree with orthogonal extensions I_0 requires that the elements of the set of left extrema are disjoint and that the elements of the set of right extrema are disjoint for each extrema tree. This is achieved by enumeration of the elements of each tertiary tree B with integers $\phi(\tilde{a}) > 0 \; \forall \, \tilde{a} \in B$, such that $\phi(\tilde{a}) \neq \phi(\tilde{b}) \; \forall \; \tilde{a}, \tilde{b} \in B$. If the extrema of a pair $\tilde{a}, \tilde{b}$ of entries in a tertiary tree B_{0L} are equal, the comparison places them in *descending* order with respect to their numbers $\phi(\tilde{a})$ and $\phi(\tilde{b})$, i.e., if $p_a = p_b$ and $\phi(\tilde{a}) > \phi(\tilde{b})$ then the comparisons results in $\tilde{a} < \tilde{b}$. If extrema of a pair of entries in a tertiary tree B_{0R} are equal, the comparison places them in *ascending* order with respect to their number. Entries in tertiary trees B_{1L} and B_{1R} are treated analogously.

Note that any set D of separators such that for each $[p, q] \in S$ there exists a $\delta \in D$ such that $\delta \in [p, q]$ is feasible for an *interval tree with orthogonal extensions.* The choice of D as a subset of the set of left extrema $\{p_v \mid ([p_v, q_v] \times [y_v, z_v]) \in S\}$ allows the data structure to operate based on comparisons within U_0 and U_1, no arithmetic is required.

Figure 1 shows an example for an *interval tree with orthogonal extensions.* The set of rectangles is enumerated and the nodes in the corresponding tree carry the numbers of the rectangles stored in their extrema trees. The separators are a subset of the left boundaries of the rectangles, the bidirectional arrows visualize the orthogonal extension for each node in the tree.

DEFINITION 2.3. *A* two-dimensional interval tree T_0 *on a set*

$$S = \{\vec{v} \,|\, [p_v, q_v] \subseteq U_0 \; \wedge \; [y_v, z_v] \subseteq U_1\}$$
$$\vec{v} = ([p_v, q_v] \times [y_v, z_v])$$

consists of a root node t with

- *a separator $\delta_t \in U_0$*
- *a left child l, which is root of $T_{1<}$ on*

$$S_< = \{\vec{v} \in S \mid q_v < \delta_t\}$$

- *a right child r, which is root of $T_{1>}$ on*

$$S_> = \{\vec{v} \in S \mid p_v > \delta_t\}$$

- *an interval tree with orthogonal extension I_1 on*

$$S_= = \{\vec{v} \in S \mid p_v \leq \delta_t \leq q_v\}$$

An empty two-dimensional interval tree consists of the node NIL. *T_1 is defined analogously.*

Thus, the tertiary nodes are nodes in the *binary trees with extensions* and the secondary nodes are the separator nodes in the *interval trees with extensions.* Primary nodes correspond to the separation lines.

Figure 2 shows a set of rectangles in the plane with separations lines and the corresponding two-dimensional interval tree. Rectangles not intersecting separation lines are denoted with integers, separation lines with letters. The secondary trees IT_a, IT_b and IT_c contain the rectangles intersecting the separation lines A, B and C, respectively.

DEFINITION 2.4. *A two-dimensional interval tree is called* ideal, *if the following conditions are satisfied:*

- *Let t be the root of a two-dimensional interval tree with N elements $\vec{v} \in S$. Then the left subtree of t contains at most $N/2$ elements and the right subtree of t contains at most $N/2$ elements.*
- *Each secondary tree I is balanced, i.e., if I contains N secondary nodes, then its depth is bounded by $O(\log(N))$.*
- *Each tertiary tree B is balanced, i.e., if B contains N tertiary nodes, then its depth is bounded by $O(\log(N))$.*

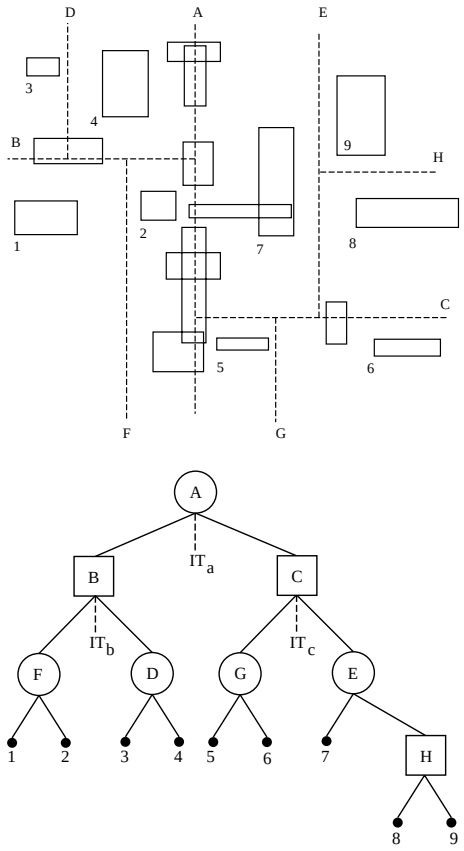

Figure 2: Example configuration of rectangles and separation lines and the corresponding two-dimensional interval tree. Circular nodes represent vertical separation lines, rectangular nodes represent horizontal separation lines and dots represent leaves corresponding to interval trees with one element.

3 Construction

To construct a two-dimensional interval tree on a set S the median γ in S with respect to the 'left', i.e., lower, extremum in U_i, $i \in \{0,1\}$ is determined. The point set

$$\{(x,y) \mid x = \gamma \ \wedge \ x \in U_i \wedge y \in U_{i+1 \bmod 2}\}$$

defines a 'separation line'. An interval tree $I_{i+1 \bmod 2}$ with extensions is constructed for the elements intersecting the separation line. Two-dimensional interval trees with respect to $U_{i+1 \bmod 2}$ are constructed for the elements completely to the 'left' and completely to the 'right' of the separation line. Note that there are at most $N/2$ elements completely to the left and at most $N/2$ elements completely to the right of the separation line. Since the median out of the set of left extrema defines the 'separation line', there is at least one element that intersects the separation line.

3.1 Secondary and Tertiary Trees

The construction of an interval tree with extensions requires the computation of a set of separators for a set $S = \{[p,q] \mid p,q \in U_i\}$ of intervals. The set of left extrema of all intervals is a feasible set of separators, but in most cases a subset of this set of separators suffices. The following procedure results in an *interval tree with orthogonal extensions* in which every node contains at least one interval:

1. Start with an empty dynamic balanced tree.
2. For each interval $[p,q]$
   - Search for a separator δ in the tree such that $\delta \in [p,q]$. This is achieved by searching for p and searching for q. The two search paths are identical until a node with $\delta \in [p,q]$ is found.
   - If no such node is present, insert the left extremum p as a separator into the tree.

   This first pass on the intervals produces the secondary tree.
3. For each interval $[p,q]$, find the first node a on the common search paths to p and q with $\delta_a \in [p,q]$ and store $[p,q]$ in its tertiary trees.
4. Compute the orthogonal extensions by depth first search.

Note that only comparison of extrema of intervals is necessary, no arithmetic is required. Note also that the union of any set of extensions in an *interval tree with orthogonal extensions* or a *binary tree with orthogonal extensions* is always a single interval since all extensions have at least one point in common.

A rotation in a tertiary tree changes the extension for exactly the two nodes a and b adjacent to the rotated edge. Since the new extensions of a and b depend only on the values of these two nodes and their direct children, a rotation in a binary tree with orthogonal extension takes $O(1)$ time. Thus, any type of dynamically balanced binary tree based on rotations is a feasible choice for the tertiary trees.

3.2 Complexity

Since a tertiary tree is a balanced binary tree in which rotation is possible in constant time, the construction of a tertiary tree with N nodes takes $O(N \log(N))$ time and it requires $O(N)$ space.

The computation of the secondary tree and its separator values for N elements as described in section

3.1 takes $O(N\log(N))$ time. It will contain $M \leq N$ separator nodes, such that we need $2M$ extrema trees, which contain altogether $N_1 + \ldots N_M = N$ nodes when we fill in the intervals with their extensions. Because of

$$\sum_{j=1}^{M} N_j \log(N_j) \leq \log(N) \sum_{j=1}^{M} N_j$$

the construction of an interval tree with orthogonal extensions for N entries takes $O(N\log(N))$ time and $2N + M$ secondary and tertiary nodes and hence $O(N)$ space.

For the construction of the primary tree for N elements we determine at most $\log(N)$ times the medians for at most $N/2$ disjoint sets of elements that contain together at most N elements. A median can be determined in linear time [11], such that computing m medians on m sets with together N elements takes $O(N)$ time. Thus, the primary tree is constructed in $O(N\log(N))$ time. It will contain $P \leq N$ secondary trees, which hold at most N elements altogether.

The observations in this subsection lead to the following theorem:

THEOREM 3.1. *For any set*

$$S = \{\vec{v} \,|\, [p_v, q_v] \subseteq U_0 \ \wedge \ [y_v, z_v] \subseteq U_1\}$$

$$\vec{v} = ([p_v, q_v] \times [y_v, z_v])$$

with $|S| = N$ a two-dimensional interval tree can be constructed in $O(N\log(N))$ time and $O(N)$ space, if comparison between elements of U_i takes $O(1)$ time for all U_i.

4 Orthogonal Range Search ...

Let

$$\vec{f} = ([p_f, q_f] \times [y_f, z_f]) \mid [p_f, q_f] \subseteq U_0 \ \wedge \ [y_f, z_f] \subseteq U_1$$

be a rectangle in $U_0 \times U_1$. Orthogonal range searching in two dimensions on a set of two-dimensional intervals is the problem of finding the set

$$A = \{\vec{v} \in S \mid \vec{v} \cap \vec{f} \neq \{\}\}$$

of elements that intersect $\vec{f}$. In the following we use the convention, that if a child $l(a)$ or $r(a)$ of a node a does not exist, $l(a)$ or $r(a)$ returns the value NIL, respectively. If a is set to be the root of an empty tree, primary, secondary or tertiary, a obtains the value NIL.

4.1 ... In a Two-dimensional Interval Tree

W.l.o.g. we choose the two-dimensional interval tree T_0 to start with a separator in U_0.

1. Set $a = t$, i.e., set actual position to the root of the primary tree T_0
2. If $a =$ NIL return.
3. If $q_f < \delta_a$:
   - Range search in $I_1(a)$.
   - Range search in $T_{1<}$ rooted in $l(a)$
4. If $p_f > \delta_a$:
   - Range search in $I_1(a)$.
   - Range search in $T_{1>}$ rooted in $r(a)$
5. If $p_f \leq \delta_a \leq q_f$:
   - Range search in $I_1(a)$
   - If $q_f > \delta_a$ range search in $T_{1>}$ rooted in $r(a)$
   - If $p_f < \delta_a$ range search in $T_{1<}$ rooted in $l(a)$

Note that we do not descend into proper subtrees if the query is a subset of a separation plane.

4.2 ... In an Interval Tree with Extensions

In this section we describe the orthogonal range search in an interval tree with orthogonal extensions, i.e., given

$$\vec{f} = ([p_f, q_f] \times [y_f, z_f]) \mid [p_f, q_f] \subseteq U_0 \ \wedge \ [y_f, z_f] \subseteq U_1$$

find

$$A = \{\vec{v} \in S \mid \vec{v} \cap \vec{f} \neq \{\}\}$$

W.l.o.g. I_0 is chosen to be an interval tree with orthogonal extensions ordered in U_0 with extensions in U_1.

- Set $a = t$, i.e., set actual position to root of I_0
- If $a =$ NIL return.
- If $[\alpha_a, \beta_a] \cap [y_f, z_f] = \{\}$ return
- If $q_f < \delta_a$:
  - Left range search in $B_{0L}(a)$ with $(q_f, [y_f, z_f])$
  - Range search in $I_{0<}$ rooted in $l(a)$
- if $p_f > \delta_a$:
  - Right range search in $B_{0R}(a)$ with $(p_f, [y_f, z_f])$
  - Range search in $I_{0>}$ rooted in $r(a)$
- If $p_f \leq \delta_a \leq q_f$:
  - Range DFS in $B_{0R}(a)$
  - If $q_f > \delta_a$ Range search in $I_{0>}$ rooted in $r(a)$
  - If $p_f < \delta_a$ Range search in $I_{0<}$ rooted in $l(a)$

4.3 ...In a Binary Tree with Extensions

W.l.o.g. we describe the left range search in a *binary tree with orthogonal extensions* B_{0L}, i.e., given

$$S = \{\tilde{v} = (x_v, [y_v, z_v]) \,|\, x_v \in U_0 \ \wedge \ [y_v, z_v] \subseteq U_1\}$$

and

$$\tilde{f} = (x_f, [y_f, z_f]) \mid x_f \in U_0 \ \wedge \ [y_f, z_f] \subseteq U_1$$

find

$$A = \{\tilde{v} \in S \mid x_v \leq x_f \ \wedge \ [y_f, z_f] \cap [y_v, z_v] \neq \{\}\}$$

The search walks down the path in the tree defined by a search for x_f and traverses all subtrees of left children along this path, using the orthogonal extensions as an additional stop criterion.

- Set $a = t$, start with root of tree
- If $a = \mathsf{NIL}$, return.
- If $[y_f, z_f] \cap [\alpha_a, \beta_a] = \{\}$ return
- If $x_f = x_a$
  - If $[y_f, z_f] \cap [y_a, z_a] \neq \{\}$ add a to A
  - Range DFS on $l(a)$
- If $x_a < x_f$
  - If $[y_f, z_f] \cap [y_a, z_a] \neq \{\}$ add a to A
  - Range DFS on $l(a)$
  - Left range search with $(x_f, [y_f, z_f])$ on subtree rooted in $r(a)$
- If $x_a > x_f$
  - Left range search with $(x_f, [y_f, z_f])$ on subtree rooted in $l(a)$

To deal with sets of extrema whose elements are not disjoint, the elements are enumerated and the key is extended to $(\phi(a), x_a, [y_a, z_a])$. As mentioned in section 2, the appropriate comparison function is used for left and right extrema trees, respectively. The search key is extended to $(0, x_f, [y_f, z_f])$.

4.4 Range DFS in Binary Tree with Extensions

W.l.o.g. we describe the *range DFS* in a binary tree ordered with respect to U_0. The problem is to find all nodes $\tilde{v}$ in the binary tree with $[y_f, z_f] \cap [y_v, z_v] \neq \{\}$. The *range DFS* performs a *depth first search* without entering subtrees with an extension that does not intersect with $[y_f, z_f]$.

- Set $a = t$, i.e., start with root if $a = \mathsf{NIL}$ return.
- If $[y_f, z_f] \cap [\alpha_a, \beta_a] = \{\}$ return.
- Range DFS on subtree rooted in $l(a)$.
- Range DFS on subtree rooted in $r(a)$.

4.5 Correctness

LEMMA 4.1. *The range search in a two-dimensional interval tree T visits every primary node $a \in T$ that contains an element $\vec{v} \in A$. A primary node is called visited if a range search in its interval tree with extensions is performed.*

Proof. The recursive range search in a two-dimensional interval tree visits every node v except nodes in subtrees $T(u)$ rooted in nodes u such that w.l.o.g. $T(a)$ is ordered on U_0 and one of the following cases holds:

- $u = r(a) \ \wedge \ q_f < \delta_a$
  $\Rightarrow p_v > \delta_a \geq q_f \ \forall\, v \in T(u)$
  $\Rightarrow \vec{v} \cap \vec{f} = \{\} \ \forall\, v \in T(u)$
- $u = l(a) \ \wedge \ p_f > \delta_a$
  $\Rightarrow q_v < \delta_a \leq p_f \ \forall\, v \in T(u)$
  $\Rightarrow \vec{v} \cap \vec{f} = \{\} \ \forall\, v \in T(u)$

Any primary node that is not visited cannot contain an element intersecting the query $\vec{f}$. q.e.d.

LEMMA 4.2. *The range search in an interval tree with orthogonal extensions I visits every secondary node $a \in I$, which contains an element $\vec{v} \in A$. A secondary node is called visited if a range search or range DFS is performed in at least one of its tertiary trees.*

Proof. W.l.o.g. we choose I to be ordered on U_0. The range search in I visits all secondary nodes except nodes in subtrees $I(u)$ rooted in nodes u such that one of the three following cases holds:

- $[\alpha_u, \beta_u] \cap [y_f, z_f] = \{\}$
  $\Rightarrow [y_v, z_v] \cap [y_f, z_f] = \{\} \ \forall\, v \in I(u)$
  $\Rightarrow \vec{v} \cap \vec{f} = \{\}$
- $u = r(a) \ \wedge \ q_f < \delta_a$
  $\Rightarrow p_v > \delta_a \geq q_f \ \forall\, v \in I(u)$
  $\Rightarrow \vec{v} \cap \vec{f} = \{\}$
- $u = l(a) \ \wedge \ p_f > \delta_a$
  $\Rightarrow q_v < \delta_a \leq p_f \ \forall\, v \in I(u)$
  $\Rightarrow \vec{v} \cap \vec{f} = \{\}$

q.e.d.

LEMMA 4.3. *The range DFS in a binary tree with extensions B visits every tertiary node $a \in B$, which contains an element $\vec{v} \in A$.*

Proof. W.l.o.g. we choose B to be ordered on U_0. The range DFS in B visits all tertiary nodes in B except nodes in subtrees $B(u)$ rooted in nodes $u \in B$ such that

- $[y_f, z_f] \cap [\alpha_u, \beta_u] = \{\}$
  $\Rightarrow [y_v, z_v] \cap [y_f, z_f] = \{\}\ \forall\, v \in B(u)$
  $\Rightarrow \vec{v} \cap \vec{f} = \{\}$

Note that no assumption about the disjointness of extrema in U_0 is necessary. q.e.d.

LEMMA 4.4. *The range search in a binary tree with orthogonal extensions B visits every tertiary node $a \in B$, which contains an element $\vec{v} \in A$. A tertiary node is called visited if the test for adding it to A is performed.*

Proof. W.l.o.g. we choose B to be ordered on U_0 and a left range search.

Using Lemma 4.3, the range search in B visits all tertiary nodes in B except nodes in subtrees $B(u)$ rooted in nodes $u \in B$ such that

- $[y_f, z_f] \cap [\alpha_u, \beta_u] = \{\}$
  $\Rightarrow [y_v, z_v] \cap [y_f, z_f] = \{\}\ \forall\, v \in B(u)$
  $\Rightarrow \vec{v} \cap \vec{f} = \{\}$
- $u = r(a) \wedge x_a \geq x_f$
  $\Rightarrow x_v > x_a \geq x_f\ \forall\, v \in B(u)$
  $\Rightarrow \vec{v} \cap \vec{f} = \{\}$

Note that the prefixing of the keys in the tree with unique positive integers and the prefixing of the search key with zero ensures that all elements in the tree with the same extremum as the search key are smaller than the search key with respect to the comparison on U_0 for a left extrema tree because of the descending order with respect to the enumeration.
q.e.d.

During the search in a secondary tree, we perform a left range search in $B_{0L}(a)$ of the actual position a if $q_f < \delta_a$, i.e., the interval of $\vec{f}$ in U_0 is completely to the left of the separator of a. Analogously, we perform a right range search in $B_{0R}(a)$ if $p_f > \delta_a$, i.e., the interval of $\vec{f}$ in U_0 is completely to the right of the separator of a. This is sufficient since all intervals stored in a that intersect $[p_f, q_f] \ni \delta_a$ have a left extremum $p_v \leq q_f$, if $q_f < \delta_a$ or a right extremum $q_v \geq p_f$, if $p_f > \delta_a$.

Thus, the lemmas 4.1 to 4.4 ensure that every tertiary node that may contribute to the solution is visited. For nodes v that are not visited it holds $\vec{v} \cap \vec{f} = \{\}$. This leads to the following theorem:

THEOREM 4.1. *Given*

$$\vec{f} = ([p_f, q_f] \times [y_f, z_f]) \mid [p_f, q_f] \subseteq U_0 \ \wedge\ [y_f, z_f] \subseteq U_1$$

a range search for $\vec{f}$ in a two-dimensional interval tree T on the set

$$S = \{\vec{v} \mid [p_v, q_v] \subseteq U_0 \ \wedge\ [y_v, z_v] \subseteq U_1\}$$

will return

$$A = \{\vec{v} \in S \mid \vec{v} \cap \vec{f} \neq \{\}\}$$

4.6 Complexity

The analysis is based on the analysis for Kd-trees in [11]. The number of operations of a range search is proportional to the number of primary, secondary and tertiary nodes visited. Let k be the number of 'hits' returned by the search. Each hit is reached through a path with at most $3\log(N)$ nodes (there is one subpath in the primary tree, one subpath in a secondary tree and one subpath in a tertiary tree). Thus, the number of nodes visited on paths to hits is $O(k\log(N))$ (already a binary tree with extensions requires this time for the successful part of its range search). To complete the analysis we determine the number of visited nodes that are not on a path to a hit.

DEFINITION 4.1. *Let T be the set of primary nodes of a two-dimensional interval tree. The* region *$Reg(v)$ of a node $v \in T$ is a rectangle in $U_0 \times U_1$ such that*

- *If $v \in T$ is the root, then*
  $Reg(v) = U_0 \times U_1$.
- *If $v \in T$ is left child of $w \in T$ and w.l.o.g. $\delta_w \in U_0$ is the separator of w, then*

  $$Reg(v) = Reg(w) \cap \{(x, y) \in U_0 \times U_1 \mid x < \delta_w\}$$

- *If $v \in T$ is right child of $w \in T$ and w.l.o.g. $\delta_w \in U_0$ is the separator of w, then*

  $$Reg(v) = Reg(w) \cap \{(x, y) \in U_0 \times U_1 \mid x > \delta_w\}$$

For primary nodes with a separator in U_1 the partitioning is analogous.

A search for a query rectangle that is degenerated to a point $x \in Reg(v)$ goes through $Reg(v)$. Let R be a query rectangle. For nodes $v \in T$ there are the following cases:

- $Reg(v) \cap R = \{\}$
  $\Rightarrow v$ is not visited during the range search. There exists a separator line such that R is completely on the opposite side of the separator line than $Reg(v)$. Thus, no rectangle stored in the subtree rooted in v may intersect with R.

- $Reg(v) \subseteq R$
  $\Rightarrow$ Everything in the subtree rooted in v is a hit since all rectangles in that region are subsets of R. All the elements in subtrees rooted in nodes v with $Reg(v) \subseteq R$ are covered by the paths to hits.

  Note that traversing a tertiary tree with n nodes takes $O(n)$ time. A secondary tree whose tertiary trees hold together n nodes has at most n secondary nodes, hence its traversing takes $O(n)$ time. Finally, a primary tree that contains n tertiary nodes has at most $O(n)$ primary nodes. Altogether, the traversing of a two-dimensional interval tree with n nodes takes $O(n)$ time. Thus, collecting all $|A_v|$ rectangles in subtrees with $Reg(v) \subseteq R$ by iterating through the three levels of subtrees takes $O(|A_v|)$ time.

- $Reg(v) \cap R \neq \{\} \ \wedge \ Reg(v) \not\subseteq R$
  $\Rightarrow$ The query region R intersects $Reg(v)$ partially. The node v is visited during the range search, but it is not necessarily accounted for through the paths to hits.

Partial intersection between R and $Reg(v)$ means that at least one of the borders of R intersects with $Reg(v)$. Let P_h be the number of regions at depth h in the tree formed by the primary nodes v such that $Reg(v)$ intersects a axis-parallel line L. Then it holds:

$$P_0 \leq P_1 \leq 2$$

$$P_{h+2} \leq 2P_h$$

This recursion resolves to

$$P_h \leq 2^{\frac{h}{2}+1}$$

For an *ideal* two-dimensional interval tree the depth of the primary tree is limited by $\log(N)$. Thus, the number of primary nodes v that were visited and for which $Reg(v)$ intersects the border of the query rectangle is

$$M \leq 4 \sum_{h=0}^{\log(N)} 2^{\frac{h}{2}+1} \tag{4.1}$$

since there are four lines separating the query rectangle. The sum covers all possible depths of the primary tree. The subtree rooted in a primary node v at depth h carries at most $\frac{N}{2^h}$ elements. For each of these nodes, a range search is performed in its secondary tree.

Consider the number of secondary nodes visited during a range search in an interval tree with orthogonal extension with n nodes. Analogously to the search in an interval tree, they can be partitioned into two groups, w.l.o.g. let the query be $[p_f, q_f] \in U_0$:

- The first group of secondary nodes consists of nodes on the two search paths in the secondary tree to p_f and q_f. These nodes are visited although they might not contain a hit with respect to U_0.

- The second group of secondary nodes consists of nodes with separator δ_a such that $p_f \leq \delta_a \leq q_f$ that are not element of group one. All these nodes are in subtrees, which are rooted in right children of nodes on the path to p_f or which are rooted in left children of nodes on the path to q_f. All intervals stored in these nodes are a hit with respect to U_0, since each such interval contains the separator δ_a.

All nodes in the second group are potential hits with respect to U_0 in an interval tree with extensions I_0. Thus, we only descend into a secondary subtree, if it contains at least one hit. For each of the binary trees with orthogonal extensions stored in a secondary node in the second group we have to perform a *range DFS*. We only descend into a tertiary subtree, if it is guaranteed to contain a hit. Thus every path taken through a node of group two is accounted for by the paths to hits.

The length of the two paths that define group one is at most $O(\log(n))$. For each of these nodes a range search in one binary tree with extensions is necessary. The range search in the *binary tree with orthogonal extensions* partitions the visited tertiary nodes in that tree into two groups. The first group consists of nodes on the search path to the extremum of the search, the second group consists of nodes left or right of that path for a left or right range search in the binary tree with extensions, respectively. Again all the nodes in group two are accounted for by the paths to hits and there are $O(\log(n))$ nodes on the path. Altogether an orthogonal range search in an interval tree with extensions takes $O(\log^2(n) + k\log(n))$ time.

Using equation 4.1 the number of nodes W (primary, secondary and tertiary) that is visited during an orthogonal range search due to paths that are not guaranteed to end in a hit is

$$\begin{aligned} W &\leq \sum_{k=0}^{\log(N)} 2^{\frac{k}{2}+1} \, 2 \left(\log(\frac{N}{2^k})\right)^2 \\ &= 4\sqrt{N} \sum_{k=0}^{\log(N)} 2^{\frac{k-\log(N)}{2}} \, (\log(N)-k)^2 \\ &= O(\sqrt{N}) \end{aligned}$$

since the sum converges to

$$\frac{2+\sqrt{2}}{(\sqrt{2}-1)^3}$$

for $N \to \infty$.

Altogether, the observations in this section lead to the following theorem:

THEOREM 4.2. *A range search in a two-dimensional interval tree holding N two-dimensional intervals $i \in U_0 \times U_1$ that returns k elements intersecting the query interval takes $O(\sqrt{N} + k\log(N))$ time.*

5 Higher Dimensions

In order to perform an orthogonal range search on a set of d-dimensional d-boxes such that the query itself is a d-dimensional d-box, the primary tree is extended analogously to the extension of a 2d-tree to a Kd-tree.

DEFINITION 5.1. *Let $x_0, \ldots x_\beta$ be the orthogonal axes in a $(\beta+1)$-dimensional universe $U_0 \times \ldots \times U_\beta$. A* multidimensional interval tree *$T_h(x_0, \ldots x_{d-1}; x_d, \ldots, x_\beta)$ of dimension d with orthogonal extensions in $U_d \times \ldots \times U_\beta$ on a set*

$$S = \{([p_v^{[0]}, q_v^{[0]}] \times \ldots \times [p_v^{[\beta]}, q_v^{[\beta]}]) \mid [p_v^{[i]}, q_v^{[i]}] \subseteq U_i\}$$

$$\vec{v} = ([p_v^{[0]}, q_v^{[0]}] \times \ldots \times [p_v^{[\beta]}, q_v^{[\beta]}])$$

of β-boxes consists of a root node t with

- *A separator $\delta_t \in U_h$ with $h \in \{0, \ldots, d-1\}$.*
- *A left child l, which is root of $T^{<}_{h+1 \bmod d}(x_0, \ldots x_{d-1}; x_d, \ldots x_\beta)$ on*

$$S_< = \{\vec{v} \in S \mid q_v^{[h]} < \delta_t\}$$

- *A right child r, which is root of $T^{>}_{h+1 \bmod d}(x_0, \ldots x_{d-1}; x_d, \ldots x_\beta)$ on*

$$S_> = \{\vec{v} \in S \mid p_v^{[h]} > \delta_t\}$$

- *An* improper *child m. If $d = 2$ then the node m is the root of an interval tree with orthogonal extensions $I_{h+1 \bmod d}$. If $d > 2$ then m is the root of a $(d-1)$-dimensional interval tree $T_{h+1 \bmod d}(x_0, .., x_{h-1}, x_{h+1}, .., x_{d-1}; x_h, x_d, .., x_\beta)$ In both cases the orthogonal extensions are in $U_h \times U_d \ldots \times U_\beta$ and the tree rooted in m contains the set*

$$S_= = \{\vec{v} \in S \mid p_v^{[h]} \leq \delta_t \leq q_v^{[h]}\}$$

of β-boxes that intersect the hyperplane defined by $x_h = \delta_t$.

A d-dimensional interval tree *$T(x_0, \ldots, x_{d-1};)$ is consequently a multidimensional interval tree with extensions in which the subspace of the orthogonal extensions has dimension zero.*

A multidimensional interval tree is called ideal, *if the size of the subproblems rooted in proper children is not larger than half of the size of the problem rooted in the parent and if the improper subtrees are ideal.*

In a d-dimensional interval tree $T(x_0, \ldots, x_{d-1};)$ each node u in the primary tree defines a partition of the set $\{x_0, x_1, \ldots, x_{d-1}\}$:

$$\begin{aligned} \mathcal{X}(u) &= \{x_{j_1}, x_{j_2}, \ldots, x_{j_{d-e}}\} \\ \mathcal{Y}(u) &= \{x_{j_{d-e+1}}, \ldots, x_{j_d}\} \\ \mathcal{X}(u) \cup \mathcal{Y}(u) &= \{x_0, x_1, \ldots, x_{d-1}\} \end{aligned}$$

such that $\mathcal{Y}(u)$ is the subset of coordinates in the orthogonal extension of u, which generates the *inactive subspace* of u. $\mathcal{X}(u)$ is the set of coordinates generating the *active subspace* of u. Consequently, the *orthogonal extension* of a node u is the projection of the smallest d-box that contains all elements stored in the subtree rooted in u into the inactive subspace.

Described in geometric terms, a separator $\delta_t \in U_h$ defines a hyperplane orthogonal to axis x_h. The proper children are the roots of trees on subproblems located 'left' and 'right' of the hyperplane. The improper child is the root of a tree on a subproblem intersecting the separation hyperplane.

In the definition the partitioning through separation planes is applied recursively until the problem has dimension $d = 1$, which is solved through an interval tree with orthogonal extensions. Note that a problem of size $N = 1$ and dimension d' takes $O(d')$ time with this convention. Note also that not all nodes at depth t in the primary tree have separator hyperplanes orthogonal to the same axis, this is only the case if they are in a subtree formed by 'proper' edges, since the 'cycle length' is different in subtrees with different sizes of $\mathcal{X}(u)$.

5.1 Construction

A multidimensional interval tree will contain $M \leq N$ interval trees with extensions that contain at most N elements altogether. The complete set of these interval trees with extensions can be constructed in $O(dN\log(N))$ time, since the evaluation of an orthogonal extension takes $O(d)$ time. Altogether they require $O(dN)$ storage since an orthogonal extension requires $O(d)$ space.

The construction of the primary tree is a subset of the construction of a regular Kd-tree, which takes $O((d+\log(N))N)$ time [11]. The depth of the primary tree in d dimensions is bounded by $d + log(N)$, an edge either reduces the dimension of the subproblem by one or reduces the size of the subproblem by a factor of $1/2$. For each level of the tree – i.e., nodes at depth i – medians (with respect to one coordinate)

are determined for m' sets of elements that altogether contain at most N elements. The computation of the orthogonal extensions takes $O(d)$ time per node. Thus, the construction of a multidimensional interval tree on a set of N d-boxes takes $O((d + \log(N))Nd)$ time. A multidimensional interval tree requires $O(d^2N)$ storage since its primary tree contains $O(dN)$ nodes each of which holds an orthogonal extension.

5.2 Orthogonal Range Search

The orthogonal range search in d dimensions is analogous to the range search in 2 dimensions, except that the improper child may be a multidimensional tree itself and that the coordinates for separators cycle through larger sets than just two coordinates.

Correctness is shown analogously to the two-dimensional case. We only avoid descending into a subtree, if it is guaranteed that there exists a hyperplane between the elements in the subtree and the query Q (this hyperplane may touch Q, but is disjoint from the region of the pruned subtree). Such a hyperplane is either given through a separation hyperplane of the tree or the existence is guaranteed since an orthogonal extension of a subtree does not intersect the corresponding projection of Q into the same subspace the extension is defined in.

The region $Reg(v)$ of a node v is defined analogously to the two-dimensional case. It is a d'-box in the active subspace of v, which may be partially unbounded. The projections of all d-boxes stored in the subtree $T(v)$ rooted in v into the active subspace defined through $\mathcal{X}(v)$ are inside of $Reg(v)$. The boundaries of $Reg(v)$ are separation hyperplanes on the path to v. Note that for nodes in an improper subtree the regions are contained in the separation hyperplane of the node with the improper child.

The combination of $Reg(v)$ with the orthogonal extension of v forms a d-box. The range search enters the subtree rooted in v only if this 'combination-d-box' intersects the query d-box Q due to the check of the orthogonal extensions. If everything in a subtree is a hit with respect to the active subspace, then each path entering such a subtree must end in a hit.

The length of the path to any hit is $O(d + \log(N))$. Therefore gathering k hits takes $O(k(d+\log(N))d)$ time. To complete the analysis it is necessary to establish an upper bound for the number of nodes that are visited, but not on a path to a hit.

5.2.1 Some Properties

LEMMA 5.1. *Let T be an ideal multidimensional interval tree of dimension d. The subtree rooted in a node at depth t, the root being at depth zero, contains at most $N/2^{(t-d)}$ nodes.*

Proof. On each path are at most d improper edges, for which it is not guaranteed that the child contains at most half of the nodes of the parent.
q.e.d.

LEMMA 5.2. *Let B be a tree with two children per node except that in each downward path defined by a set of edges starting in nodes $v_q, \ldots v_{q+d-1}$ there exists at least one v_i in $v_q, \ldots v_{q+d-1}$ with at most one child. Then for the number P_h of nodes at depth h is holds*

$$P_h \leq 2(2^{d-1})^{h/d}$$

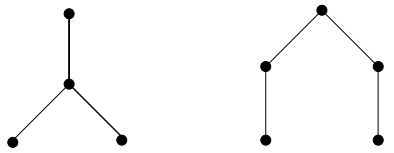

Figure 3: Tree transformation that leaves the number of leaves constant.

Proof. Consider a binary tree with depths $0, \ldots, d$ with two children per node, such that in each path from the root to a leaf there is a set of nodes such that there is one node with exactly one child. With the transformation shown in figure 3 any tree with the given properties can be transformed into a tree with the same number of leaves in which all the nodes in depth $d-1$ have exactly one child. Obviously all trees of depth d with the given property have at most 2^{d-1} leaves.

Consider a 'slice' of depth $h, \ldots, h + d$ in a binary tree in which the paths satisfy the criterion given in the lemma. The slice is composed of a set of n binary trees with depth $0, \ldots, d$ each of which has at most 2^{d-1} leaves. Thus it holds

$$\begin{aligned} P_i &\leq 2^i \ \forall\, i \in \{0 \ldots d-1\} \\ P_h &\leq 2^{d-1} P_{h-d} \end{aligned}$$

which resolves to

$$P_h \leq 2(2^{d-1})^{h/d}$$

q.e.d.

5.2.2 The Bound

Let T be a multidimensional interval tree of dimension d. An orthogonal range search visits a subtree T' of this tree. We compute an upper bound for the size of

the subtree $T'' \subseteq T'$, which consists of all visited nodes that are no hits and paths from the root to these nodes. A node is considered as 'visited' if the extension-check was performed on its children. Performing the extension check for all the children of a node v takes $O(d)$ time and is considered to be part of the visit of v. An upper bound for the number of nodes in T'' will be an upper bound for the number of investigated nodes that are not covered by a path to a hit.

T consists of a set of subtrees T_i of proper edges, which are connected through improper edges. The trees T_i are called proper subtrees and $T_i'' = T_i \cap T''$ are the subtrees of proper subtrees that were 'investigated unsuccessfully'. Let the index of T_i denote the dimension of its orthogonal extension. Then T_0 is the proper subtree including the root node of T and $X(T_0) = \{x_0, \ldots, x_{d-1}\}$ and $Y(T_0) = \{\}$. Let d_i be the dimension of the space defined by $X(T_i)$.

LEMMA 5.3. *Let T_i'' be a proper subtree of a d-dimensional interval tree as defined above with dimension $d_i > 1$. Then for the number of nodes $P_h(T_i'')$ at depth h of T_i'' it holds*

$$P_h(T_i'') \leq 2d_i 2\left(2^{d_i-1}\right)^{\frac{h}{d_i}}$$

Proof. Let Q be the d-box of the query. The nodes in T_0 can be partitioned into three groups:

- $Reg(v) \subseteq Q$: It is guaranteed, that every node in the subtree rooted in v is a hit, thus the subtree does not belong to T_0''.
- $Reg(v) \cap Q = \{\}$: Such nodes are not visited since there exists a separator Δ_0 such that v is located in a subtree that was 'pruned' due to Δ_0. Δ_0 is a hyperplane in the space defined by $X(T_0)$.
- $Reg(v) \cap Q \neq \{\} \ \wedge \ Reg(v) \not\subseteq Q$: Such nodes are visited but do not necessarily belong to a path to a hit. Additionally, parts of the subtrees rooted in their improper children may be visited unsuccessfully.

  Thus, all nodes for which it holds

$$Reg(v) \cap Q \neq \{\} \ \wedge \ Reg(v) \not\subseteq Q$$

may be part of T_0'', but no others.

For a proper subtree T_1, for which $Y(T_1) = \{x_h\}$ with $0 \leq h < d$ the situation is similar. All the regions in T_1 are contained in a separator hyperplane Δ_0 in the space defined by $X(T_0) = \{x_0, \ldots x_{d-1}\}$ that is defined by fixing one coordinate to a given value, i.e. $x_h = \delta_0$. Let $[Q \to X(T_1)]$ be the projection of the query d-box Q into the hyperplane defined by $x_h = \delta_0$. Again the nodes are partitioned into three groups.

- $Reg(v) \subseteq [Q \to X(T_1)]$: All nodes in the subtree rooted in v (not only the proper subtree!) are hits, if their orthogonal extension intersects with the corresponding projection of Q. Thus, every path into such a subtree is covered by a path to a hit and hence not in T_1''.
- $Reg(v) \cap [Q \to X(T_1)] = \{\}$: These nodes are not visited because of a separator hyperplane Δ_1 in the space defined by $X(T_1)$.
- $Reg(v) \cap [Q \to X(T_1)] \neq \{\} \ \wedge \ Reg(v) \not\subseteq [Q \to X(T_1)]$: These are again the nodes that may be in T_1''.

Note that for proper subtrees $T_2, T_3, \ldots T_{d-1}$ rooted in improper children of proper subtrees with a higher dimension the same partitioning applies, only the dimension d_f of the space defined by $X(T_f)$ is $d - f$.

Thus, in a proper subtree T_i only nodes v such that $Reg(v)$ intersects the surface of $[Q \to X(T_i)]$ without being a complete subset of the projection of Q may be part of T_i''. There are $2d_i$ surface planes in the projection of Q.

Consider a hyperplane F_h in the space defined by $X(T_i)$ that contains a surface plane of $[Q \to X(T_i)]$ orthogonal to axis $x_h \in X(T_i)$. Then in a path $v_q \ldots v_{q+d_i-1}$ there exists at least one node v_i with a separator that fixes the coordinate x_h. In this node the separator plane is orthogonal to x_h and thus F_h intersects with the region of at most one proper child. Thus $P_h \leq 2^{d_i-1} P_{h-d_i}$ and using lemma 5.2 concludes the proof.
q.e.d.

The depth of T is bounded by $d + \log(N)$. A proper subtree T_i'' of unsuccessfully visited nodes starts at a certain depth g but may not reach the full depth of $d + \log(N) - g$. We assume that it is extended such that lemma 5.3 still holds and such that it reaches $d + \log(N) - g$. This may only increase $|T''|$. Consider the proper subtree T_0''. Adding improper subtrees T_i'' such that on any path from the root to a leaf there are at most d improper edges increases the size of T_0'' at most by a factor of 2^d, if all T_i'' are extended to the maximum depth.

For each visited primary node there is at most one improper child with an interval tree with extensions. The number of nodes in this interval tree is limited by lemma 5.1. Thus it holds

$$|T''| \ \leq \ 2d2^d \sum_{h=0}^{d+\log(N)} 2\left(2^{d-1}\right)^{h/d} \log^2\left(\frac{N}{2^{h-d}}\right)$$

$$
\begin{aligned}
&= 2d2^d \sum_{h=0}^{d-1} 2\left(2^{d-1}\right)^{h/d} \log^2\left(\frac{N}{2^{h-d}}\right) \\
&\quad +2d4^d \sum_{h=0}^{\log(N)} 2^{(1-1/d)h} \left(\log(N)-h\right)^2 \\
&\leq O(d4^d \log^2(N)) \\
&\quad +2d4^d N^{1-\frac{1}{d}} \sum_{h=0}^{\log(N)} 2^{\frac{1}{2}(h-\log(N))} \left(\log(N)-h\right)^2
\end{aligned}
$$

and consequently it holds

$$|T''| = O(d4^d N^{1-1/d})$$

since the sum in the last expression is a constant in the limit $N \to \infty$.

Since a visit to a node costs $O(d)$ operations in the worst case to evaluate the orthogonal extension, an orthogonal range search takes

$$O((d^2 4^d N^{1-1/d} + d(d + log(N))k)$$

in a multidimensional interval tree of dimension d.

6 Conclusion

Multidimensional interval trees solve the problem of orthogonal range searching on sets of d-boxes using linear space. In comparison to Kd-trees, which solve orthogonal range searching on sets of points in linear space and $O(d4^d n^{1-1/d} + dk)$ time, multidimensional interval trees require $O((d^2 4^d N^{1-1/d} + d(d + log(N))k))$ time.

In practice the additional factor of $\log(N)$ per hit is rather pessimistic. For larger numbers of hits the paths to the hits are not disjoint. Additionally, for 'fat' query rectangles there is a significant portion of the hits that is computed in $O(dk)$ time through the subtrees rooted in nodes v such that $Reg(v)$ is a subset of the query.

Multidimensional interval trees are the first data structure solving orthogonal range searching on d-boxes in linear space and less than $O(N)$ query time in the worst case.

References

[1] R.A. Finkel, J.L. Bentley. *Quad Trees: A data structure for retrieval on composite keys.* Acta Inform., 4:1-9, 1974.

[2] J.L. Bentley. *Multidimensional binary search trees used for associative searching.* Commun. ACM, 18:509-517, 1975.

[3] J.L. Bentley. *Solutions to Klee's rectangle problems.* Technical report, Carnegie Mellon Univ., Pittsburg,PA, 1977.

[4] J.L. Bentley. *Decomposable searching problems.* Inform. Process. Lett., 8:244-251, 1979.

[5] G.S. Lueker. *A data structure for orthogonal range queries.* Proc. 19th Annu. IEEE Sympos. Found. Comput. Sci., 28-34,1978.

[6] B. Chazelle. *Filtering search: A new approach to query answering.* SIAM J. Comput., 15:703-724, 1986.

[7] B. Chazelle. *Lower bounds for orthogonal range searching* J.ACM, 37:200-212 and 439-463, 1990.

[8] E.M. McCreight. *Efficient algorithms fir enumerating intersecting intervals and rectangles.* Report CSL-80-9, Xerox Palo Alto Res. Center, 1980.

[9] H. Edelsbrunner. *Dynamic data structures for orthogonal intersection queries.* Report F59, Inst. Informationsverarb., Tech. Univ. Graz, Austria, 1980.

[10] H. Edelsbrunner, H.A. Maurer. *On The Intersection of Orthogonal Objects.* Information Processing Letters, 13:177-181,1981.

[11] K. Mehlhorn. *Multi-dimensional Searching and Computational Geometry* Springer,1984.

[12] M. de Berg, M. van Krefeld, M. Overmars, O. Schwarzkopf. *Computational Geometry.* Springer,1997.

[13] J.E. Goodman, J. O'Rourke (editors). *Handbook of Discrete and Computational Geometry.* CRC Press LLC, 1997.