2.5.3 Mutual Pruning Strategien

Idee:

- Ausschluss von Seiten/Objekten mittels Distanzabschätzungen (ohne Vorberechnung, zur Laufzeit der Anfrage ermittelt)
- Für mindestens *k* Objekte ist die Distanz zu Objekt o kleiner als die Distanz zwischen Anfrageobjekt q und o
  => Objekt o $\notin$ RkNN(q)
- Distanzabschätzungen über Indexseitenregionen

Vorteile:

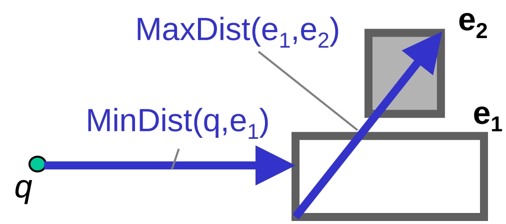
- Keine Vorberechnung notwendig
  → keine Update-Problematik mehr
- Parameter *k* zur Anfragezeit frei wählbar

– Min/Max-Distanz-Ansatz

[Achtert, Kröger, Kriegel, Renz, Züfle, EDBT, 2009]

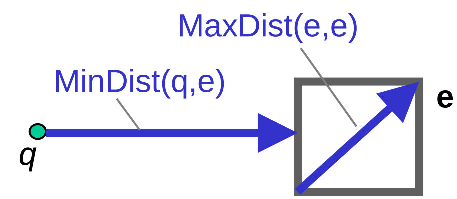
– Vorraussetzung: Indexseite speichert Anzahl der Objekte |e| in Seite e

Ausschluß von Seite e_1 durch Seite e_2 ($k \geq 1$):



MaxDist(e_1,e_2) < MinDist(q,e_1)
$\Rightarrow$ e_1 kann keine RkNN(q)-Ergebnisse enthalten wenn $|e_2| \geq k$

Seite e schließt sich selbst aus ($k \geq 1$):



MaxDist(e,e) < MinDist(q,e)
$\Rightarrow$ e kann keine RkNN(q)-Ergebnisse enthalten wenn $|e|-1 \geq k$

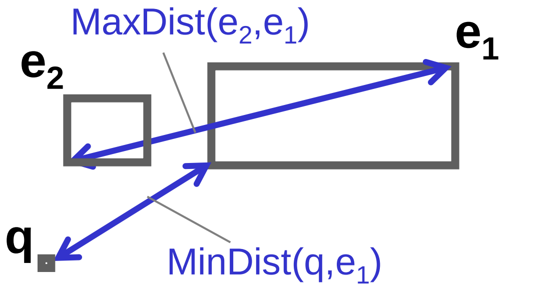
MinDist(e_1,e_2): Distanz zwischen den beiden voneinander am nächsten liegenden Punkten $p_1 \in e_1$ und $p_2 \in e_2$

MaxDist(e_1,e_2): Distanz zwischen den beiden voneinander entferntesten Punkten $p_1 \in e_1$ und $p_2 \in e_2$

– Vorteile:
  - » Keine Vorberechnung mehr (keine Update-Problematik)
  - » Parameter k zur Anfragezeit frei wählbar
  - » Pruningkonzept auf allg. metrische Daten anwendbar

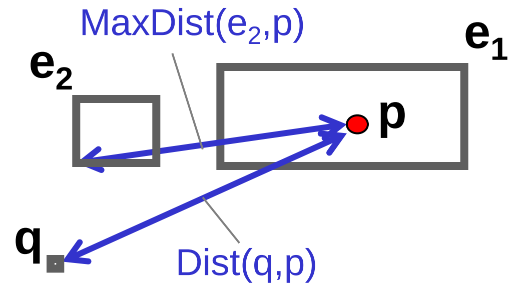
– Problem:
  » Abhängigkeiten zwischen Distanzen unberücksichtigt

Min/Max-Dist Pruning:



$\text{MinDist}(q,e_1) < \text{MaxDist}(e_2,e_1)$

„Optimales“ Pruning:



$\forall p \in e_1: \text{Dist}(q,p) > \text{MaxDist}(e_2,p)$

» Die Distanz MinDist(q,e_1) schätzt die Distanz dist(q,p) zwischen q und einem Objekt p in Seitenregion e_1 (untere Distanzabschätzung)

» Die Distanz MaxDist(e_2,e_1) schätzt die Distanz dist(o,p) zwischen einem Objekt o in Seitenregion e_2 und einem Objekt p in Seitenregion e_1 (obere Distanzabschätzung)

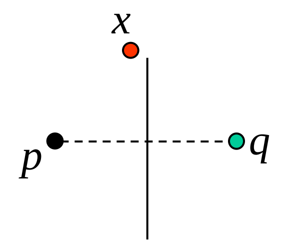
» Beide Distanzen dist(q,p) und dist(o,p) hängen von der Lage des Objektes p in Region e_1 ab => dist(q,p) abhängig von dist(o,p)

» Diese (Abhängigkeits-) Information geht bei der Verwendung von MaxDist(e_2,e_1) und MinDist(q,e_1) verloren

→ geringeres Pruningpotential

– Geometrische RNN-Suche (Filter/Verfeinerung)

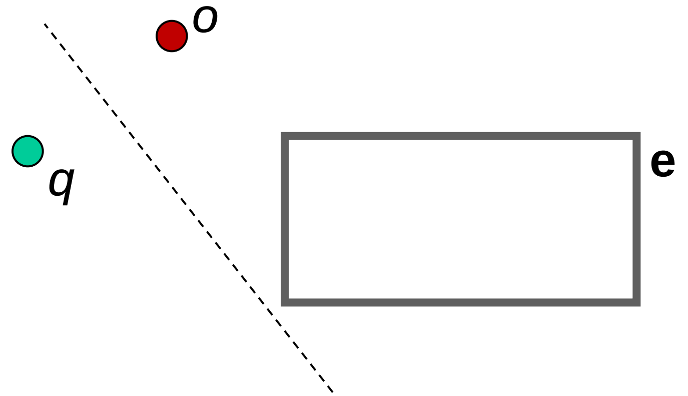
[Tao, Papadias, Lian. Int. Conf. Very Large Databases (VLDB), 2004]



Voronoi-Ebene $E(p,q)$ zwischen p und q

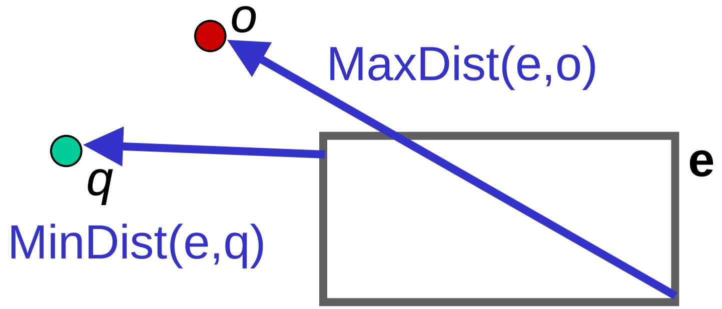
- Idee
  - » Gegeben: Voronoi-Ebene zwischen *q* und beliebigen Punkt *p*
  - » Liegt ein Punkt *x* auf der Seite von *p* dieser Voronoi-Ebene, kann *q* nicht NN von *x* sein und damit $x \notin$ RNN(*q*)
  - » Voronoi-Ebene *E* (*p*,*q*): für alle Punkte $e \in E$ gilt: dist(*q*,*e*) = dist(*p*,*e*)
- „Geometrisches“ Pruning ist optimal bzgl. der Pruning-Stärke

geometrisches Pruning:



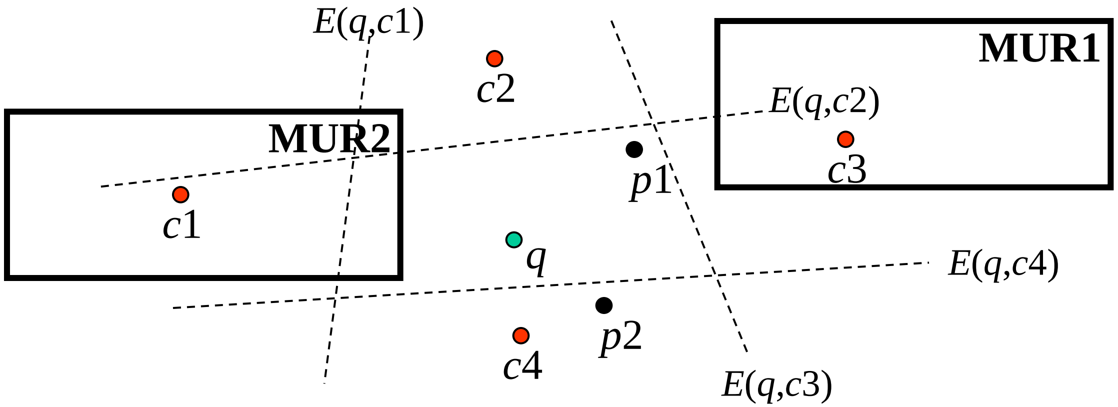
e hinter der Hyperebene
=> prune e

Min/Max-dist Pruning:



MaxDist(e,o) > MinDist(e,q)
=> prune e nicht!

- Algorithmus: Filter-Schritt (Skizze)
  - Berechne ein NN-Ranking der DB
  - Solange noch Objekte im Ranking sind:
    - » Rufe getNext() auf
    - » Wenn aktueller Punkt *p* nicht „hinter“ einer Voronoi-Ebene liegt, konstruiere neue Voronoi-Ebene *E*(*p*,*q*); *p* wird zur Kandidatenmenge hinzugefügt
    - » Punkte/Directoryseiten, die „hinter“ einer der Voronoi-Ebenen liegen (außer der eigenen), können aus dem Ranking/Kandidatenmenge gelöscht werden
  - Punkte, die die Ebenen bestimmen, müssen verfeinert werden, d.h. für diese Punkte muss jeweils eine NN-Anfrage berechnet werden
- Beispiel



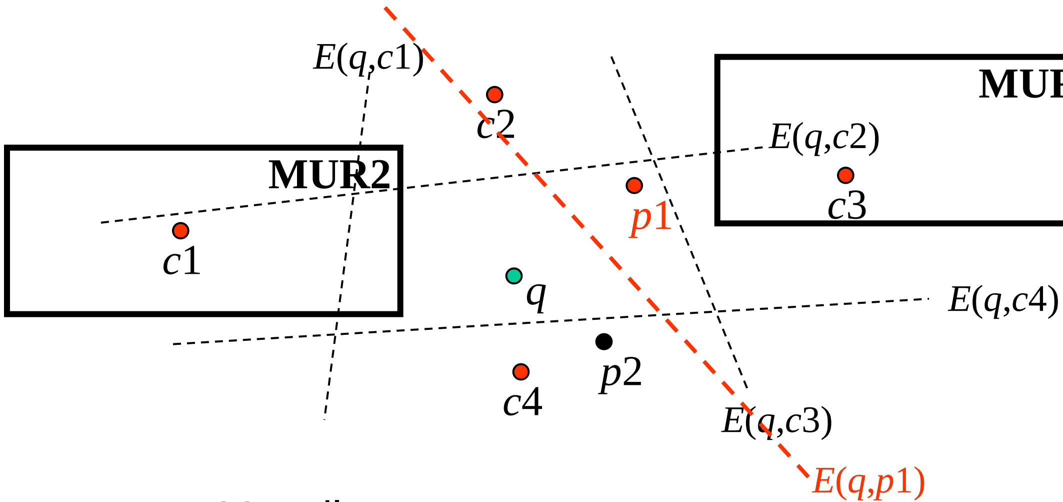
Bisherige Kandidaten:

{*c*1, *c*2, *c*3, *c*4}

Inhalt des Rankings (ungeordnet):

MUR1: nicht verfeinern
MUR2: verfeinern
*p*1: verfeinern
*p*2: nicht verfeinern

- Verfeinerung von *p*1
  - » Streiche *c*2 und *c*3 aus Kandidatenliste (liegt nun hinter *E*(*q*,*p*1))
  - » MUR2 muss weiterhin verfeinert werden



Bisherige Kandidaten:

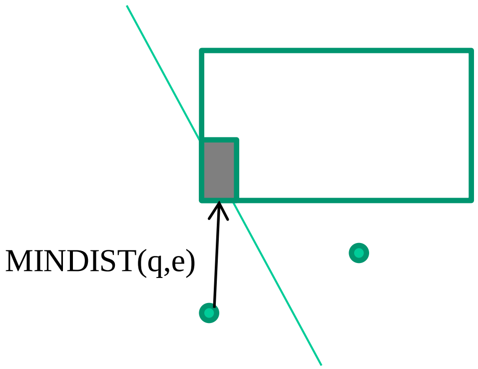
{*c*1, *c*4, *p*1}

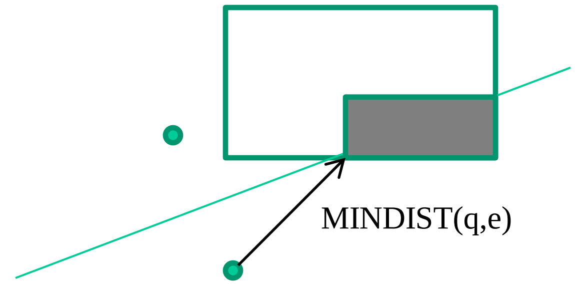
Inhalt des Rankings (ungeordnet):

MUR2: verfeinern

- Vorteil
  - » *k* kann beliebig sein (Ausschlusskriterium: Objekt/Seite muss hinter *k* Ebenen liegen
  - » Keine vorberechneten Distanzen, daher keine Update-Problematik und bessere Speicherkomplexität
- Nachteil
  - » Nur für Vektordaten
  - » Teurer Verfeinerungsschritt (eine NN-Anfrage pro Kandidat)
  - » Teilweise komplexe Ebenenverwaltung

- Trimmen
  - Partielles abschneiden (trimmen) von Seitenregionen (Rechtecken) bzgl. einer Pruningebene





  - Anpassung der MINDIST(q,e) nach dem Trimmen

    ➔ führt eventuell zur Erhöhung der MINDIST(q,e)

    ➔ erhöht die Chance, dass Seitenregion e früher ausgefiltert (geprunt) werden kann.

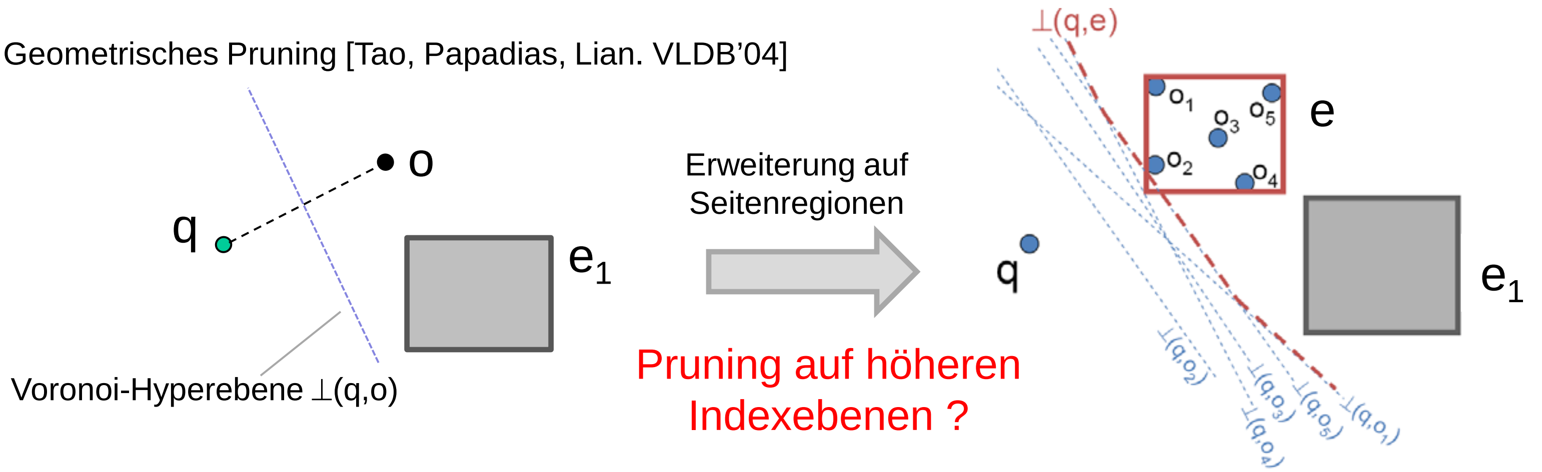
- Algorithmus (Filter):

```
Algorithm TPL-filter(q) /* q is the query point */
initialize a min-heap H accepting entries of the form (e, key)
initialize sets S_cnd=∅, S_rfn=∅
insert (R-tree root, 0) to H
while H is not empty
    (e, key)=de-heap H
    if (trim(q, S_cnd, e)=∞) then S_rfn=S_rfn∪{e}
    else // entry may be or contain a candidate
        if e is data point p
            S_cnd=S_cnd∪{p}
        else if e points to a leaf node N
            for each point p in N (sorted on dist(p,q))
                if (trim(q,S_cand,p)≠∞) then insert (p,dist(p,q)) in H
                else S_rfn=S_rfn∪{p}
        else // e points to an intermediate node N
            for each entry N_i in N
                mindist(N_i^resM, q)=trim(q, S_cnd, N_i)
                if (mindist(N_i^resM, q)=∞) then S_rfn=S_rfn∪{N_i}
                else insert (N_i, mindist(N_i^resM, q)) in H
End TPL-filter
```

falls e hinter einer bestehenden Pruningebene

Quelle: [Tao, Papadias, Lian. Int. Conf. Very Large Databases (VLDB), 2004]

- Weitere Eigenschaft der Geometrischen RNN-Suche
  - Ausschluß (Pruning) von Punkten/Seiten nur aufgrund anderer Punkte
  - Ausschlußkriterium könnte auch vollständig auf Directory-Ebene angewandt werden, aber wie?
- Erweiterung des geometrischen Pruning-Konzepts auf Basis von Voronoi-Hyperebenen auf Seitenregionen

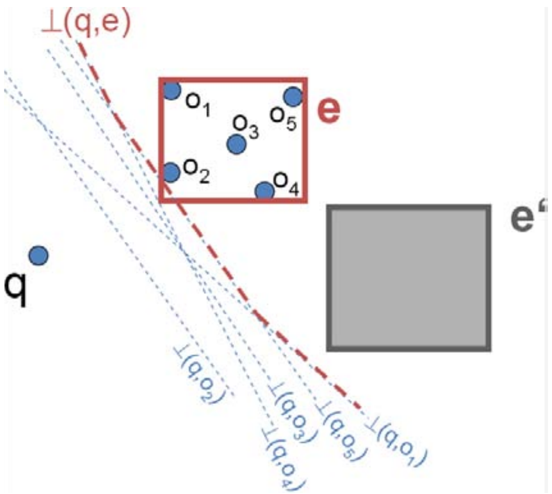


- Frage: Effiziente Repräsentation von Voronoi-Hyperebenen zwischen einem Punkt und einer Seitenregion ?
- Idee: Konservative Approximation der Hyperebenen

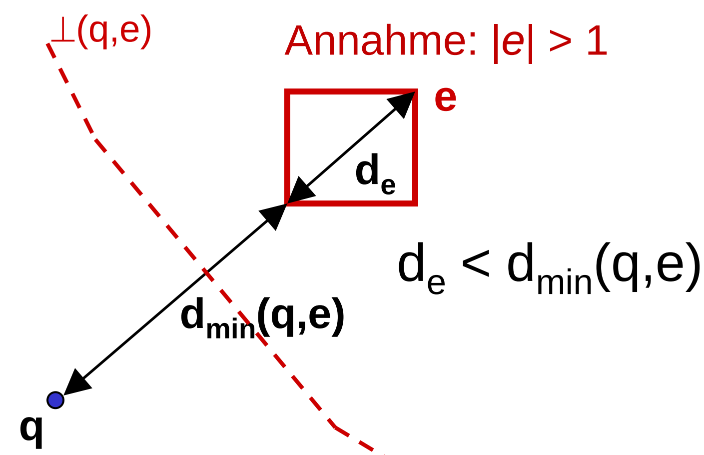
  Eigenschaft: $\forall p \in \perp(q,e)$: dist(q,p) = MaxDist(e,p)

– Erweiterung der Geometrischen RNN-Suche (gilt nur für Vektordaten!!!) [Kriegel, Kröger, Renz, Züfle: SSDBM, 2009]
  - Ziel: Pruning auf höheren Indexebenen
  - Definition von Hyper-Ebenen zwischen Anfragepunkt und Seitenregion e
  - Bilde konservative Approximation $\mathcal{H}$ aller Hyperebenen bzgl. aller Punkte innerhalb der Seitenregion e
  - Anzahl der konservativ approximierten Hyperebenen kann zum Ausschluß von Seitenregionen (Self/Mutual Pruning) verwendet werden (insbesondere auch für RkNN-Suche mit k>1)

Seite e schließt Seite e‘ aus (k=1):



Seite e schließt sich selbst aus (k=1):

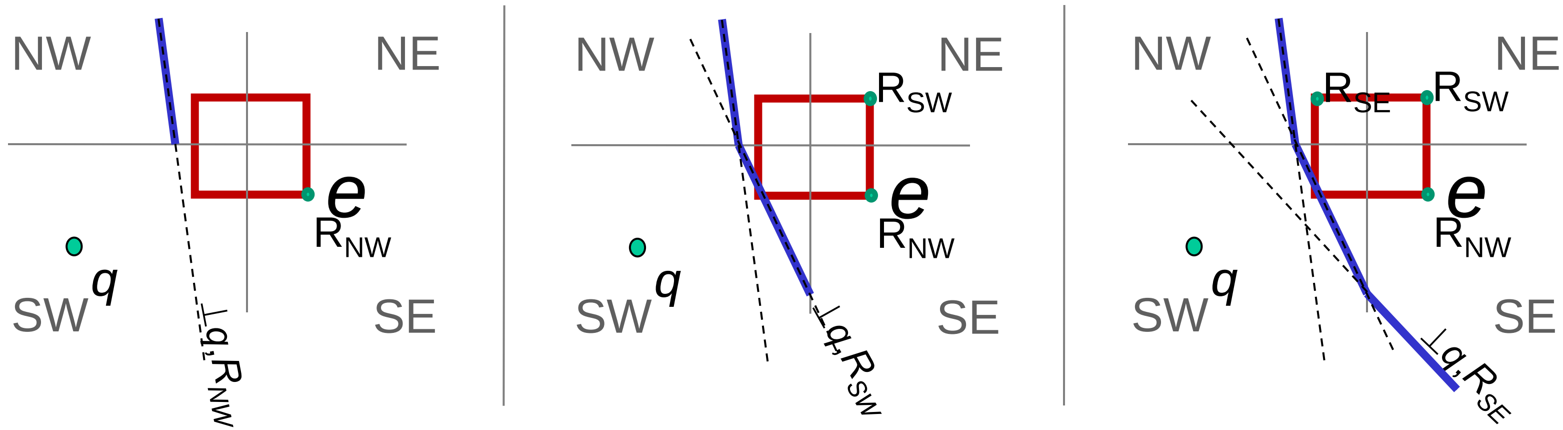


$\forall o \in e'$: $o \notin RNN(q)$, da $\exists o_i \in e$: e’ hinter Hyperebene $\perp(q,o_i)$

- Berechnung der konservativen Approximation aller Hyperebenen zwischen Anfragepunkt q und Seitenregion e
  - Aufsplittung des Datenraums in 2^d Partitionen orientiert am Mittelpunkt der Seitenregion E (z.B. 4 Partitionen NW, NE, SE und SW in 2D Raum)
  - Für jede Partition P: Wähle Referenzpunkt $R \in e$, sodaß gilt:

    $$\forall p \in P, \forall e \in E: d(p,e) \leq d(p,R)$$

    Bemerkung: Referenzpunkt eindeutig durch die gewählte Partitionierung
  - Für jede Partition P bilde Hyperebene zwischen q und Referenzpunkt zu P

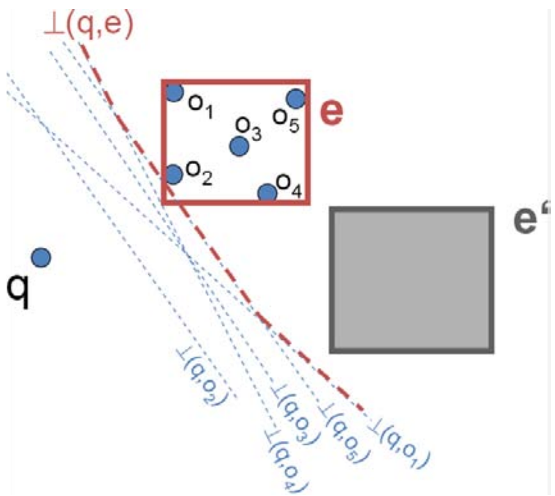


  - Die konservative Hyperebenen-Approximation wird aus 2^d-1 Hyperebenen gebildet

- Erweiterung der Geometrischen R*k*NN-Suche für k > 1
  - Verwende aR-Baum anstatt R-Baum, d.h. R-Baum mit zusätzlicher Angabe der Anzahl der Punkt-Objekte |e| die innerhalb einer Seitenregion e organisiert werden.
  - Menge der Punkt-Objekte auf die sich eine konservative Approximation $\mathcal{H}$ bezieht ist zur Anfragezeit bekannt.

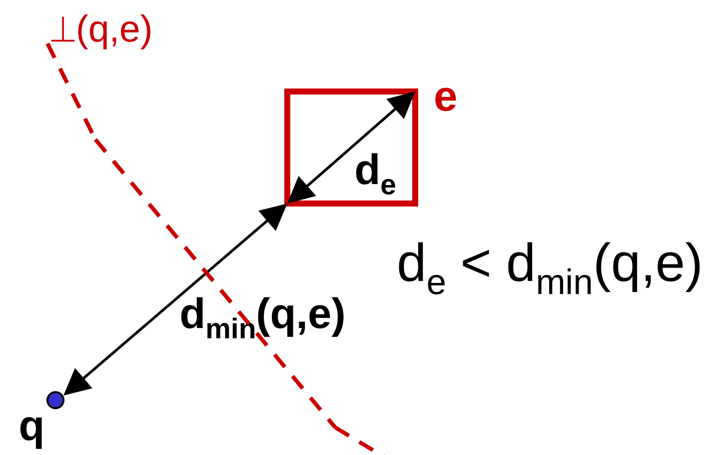
    => Anzahl N der Objekte, die näher an einem Objekt o bzw. einer Seitenregion e' sind als der Anfragepunkt q lässt sich einfach ermitteln
  - Prune Seitenregion e' (bzw. e bei self pruning) falls N > k (bzw. N-1 > k)

Seite e schließt Seite e' aus (k>1):



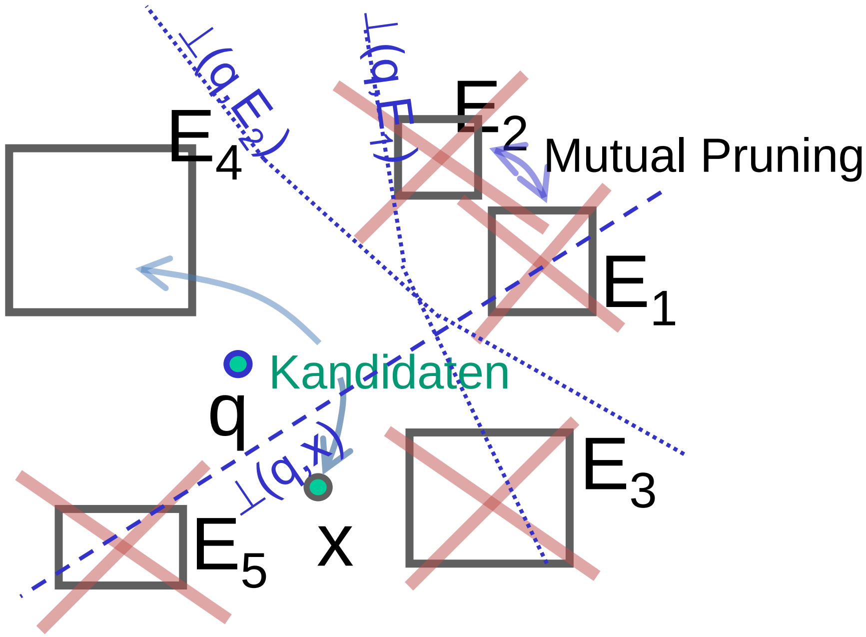
$\forall o \in e'$: $o \notin \mathrm{R}k\mathrm{NN}(q)$, where $k \leq |e|$

Seite e schließt sich selbst aus (k>1):



$\forall o \in e$: $o \notin \mathrm{R}k\mathrm{NN}(q)$, where $k \leq |e\text{-}1|$

- Beispiel:



- Bemerkung zum „Optimalen“ Pruning:
  - Die Suche mittels Mutual-Pruning ohne Vorberechnung ist weniger selektiv als mit vorberechneten NN-Distanzen

    ==> schlechteres Pruning-Verhalten als bei (reinen) Self-Pruning-Methoden

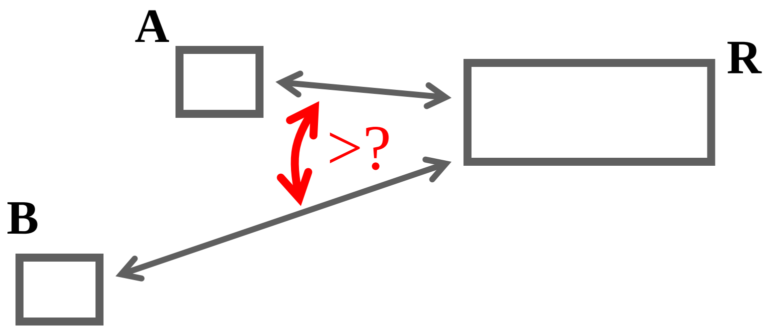
- Eigenschaften der Geometrischen R*k*NN-Suche
  - Vorteile:
    - » Vollständige Flexibilität bzgl. *k*
    - » Keine Zusätzliche Kosten für Änderungen im Index (Update-Kosten)
    - » Pruning-Filter selektiver als bei Min/Max-Dist-Ansatz
  - Nachteile:
    - » 2^d Hyperebenen pro Seitenregion müssen materialisiert werden => Overhead der Ebenenverwaltung (schlechte Performanz in höher-dimensionalen Räumen)
    - » Test ob Seitenregion geprunt werden kann ist teuer (2^d Ebenen-Pruning-Tests)
- Fazit:
  - Min/Max-Dist-Ansatz hat geringeres Pruningpotential als Geometrisches Pruning
  - Geometrisches Pruning eignet sich nur für niedrig-dimensionale Räume
- Gewünscht:
  - Verfahren mit „optimalen" Pruningpotential, das auch für hoch-dimensionale Räume effizient funktioniert

  → Welche Beziehung gilt zwischen dem Min/Max-Dist-Basierten Ansatz und dem Geometrischen Pruning Ansatz?

– Generelle Problemstellung:

Gegeben:

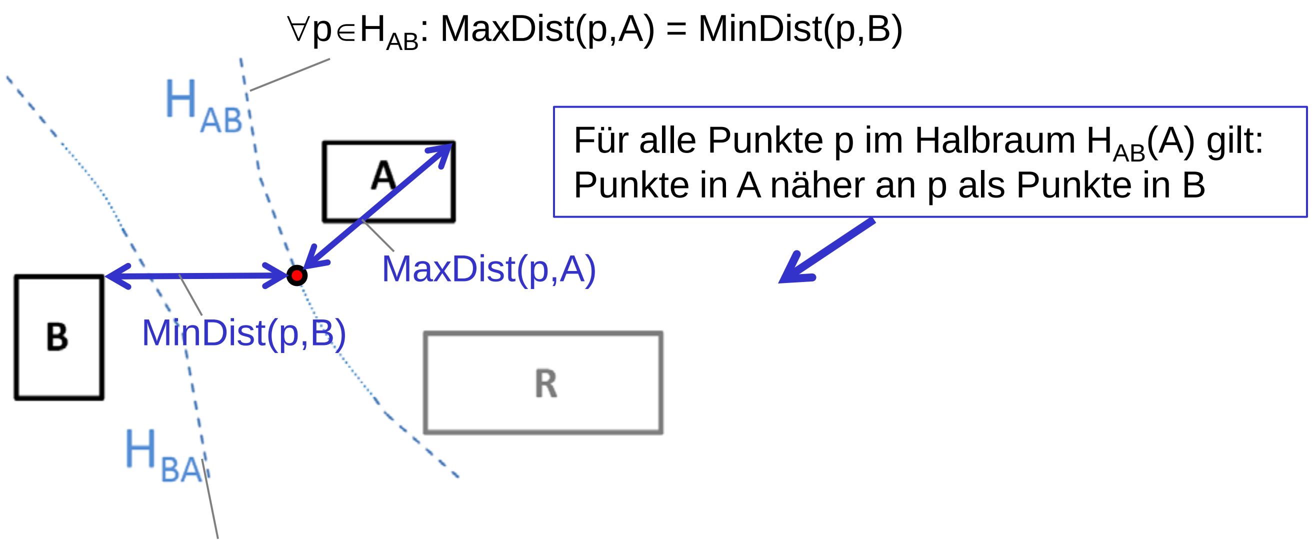
Drei Punktobjekt-Approximationen A, B und R gegeben als achsenparellele Rechtecke



Frage:

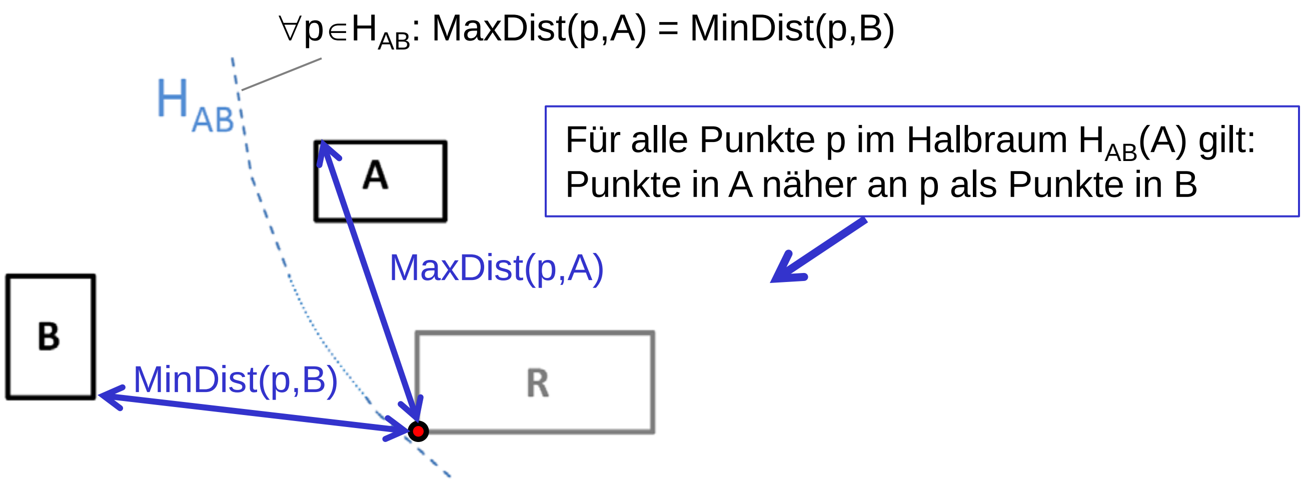
Wie kann man effizient bestimmen ob die Objekte in R näher an den Objekten in A oder näher an den Objekten in B liegen

→ Nachbarschafts-Entscheidungskriterium [SIGMOD 10]



- Beobachtung:
  - Halbraum H_{AB}(A) in der A liegt ist konvex

  → alle Eckpunkte von R liegen in H_{AB}(A) => R liegt vollständig in H_{AB}(A)

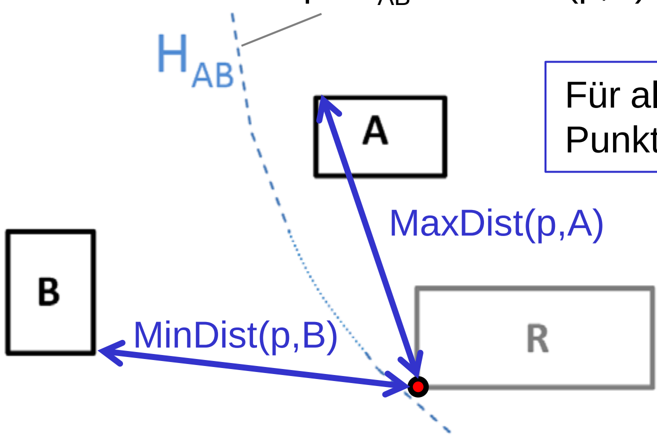


- Beobachtung:
  - Halbraum H_{AB}(A) in der A liegt ist konvex

  → alle Eckpunkte von R liegen in H_{AB}(A) => R liegt vollständig in H_{AB}(A)
- Idee:
  - Finde Eckpunkt p von R bei dem MaxDist(p,A)-MinDist(p,B) den größten Wert hat
  - Falls dieser Wert kleiner 0 (d.h. p in H_{AB}(A))

  → Alle Ecken von R in H_{AB}(A) => Region R vollständig in H_{AB}(A)

$\forall p \in H_{AB}$: MaxDist(p,A) = MinDist(p,B)



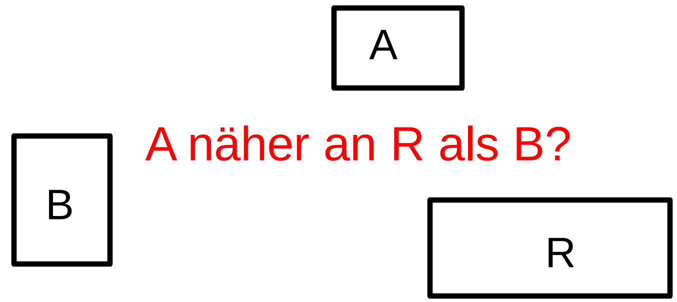
Für alle Punkte p im Halbraum H_{AB}(A) gilt:
Punkte in A näher an p als Punkte in B

Nachbarschafts-Entscheidungskriterium:
Objekte in R näher an A falls

$$\boxed{\sum_{i=1}^{d} \max_{r_i \in \{R_i^{\min}, R_i^{\max}\}} \left(MaxDist(A_i, r_i)^2 - MinDist(B_i, r_i)^2 \right) < 0}$$

- Beobachtung:
  - Halbraum H_{AB}(A) in der A liegt ist konvex
  - → alle Eckpunkte von R liegen in H_{AB}(A) => R liegt vollständig in H_{AB}(A)
- Idee:
  - Finde Eckpunkt p von R bei dem MaxDist(p,A)-MinDist(p,B) den größten Wert hat
  - Falls dieser Wert kleiner 0 (d.h. p in H_{AB}(A))
  - → Alle Ecken von R in H_{AB}(A) => Region R vollständig in H_{AB}(A)

- Herleitung
  - Idee: Umformung der Distanzrelation:



A näher an R als B?

$$\forall a \in A, b \in B, r \in R : dist(a,r) < dist(b,r)$$

$$\Leftrightarrow \forall r \in R : \boxed{MaxDist}(A,r) < \boxed{MinDist}(B,r)$$

konservative Distanzabschätzung zu A und B

$$\Leftrightarrow \forall r \in R : \sqrt[p]{\sum_{i=1}^{d} MaxDist(A_i, r_i)^p} < \sqrt[p]{\sum_{i=1}^{d} MinDist(B_i, r_i)^p}$$

$$\Leftrightarrow \forall r \in R : \sum_{i=1}^{d} MaxDist(A_i, r_i)^p - \sum_{i=1}^{d} MinDist(B_i, r_i)^p < 0$$

$$\Leftrightarrow \max_{r \in R} \left(\boxed{\sum_{i=1}^{d}} (MaxDist(A_i, r_i)^p - MinDist(B_i, r_i)^p) \right) < 0$$

$$\Leftrightarrow \sum_{i=1}^{d} \max_{\boxed{r_i \in R_i}} (MaxDist(A_i, r_i)^p - MinDist(B_i, r_i)^p) < 0$$

$$\Leftrightarrow \sum_{i=1}^{d} \max_{\boxed{r_i \in \{R_i^{\min}, R_i^{\max}\}}} (MaxDist(A_i, r_i)^p - MinDist(B_i, r_i)^p) < 0$$

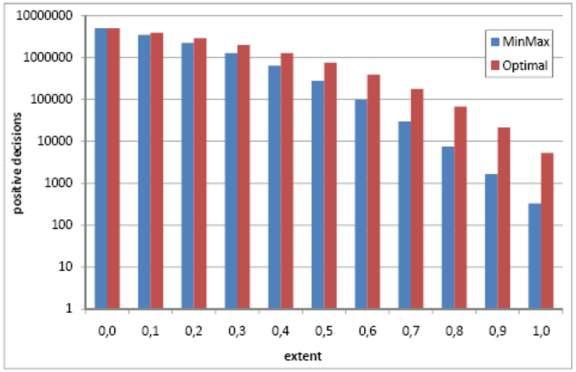
Reduzierung des Problems auf die einzelnen Dimensionen

Reduzierung des Problems auf Extremwerte in R_i (pro Dimension nur 2 Werte)

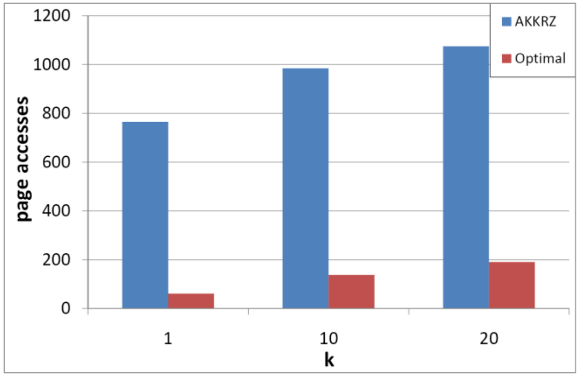
- Vorteil: Berechnungskomplexität O(d) (d = Dimensionalität)

=> geeignet auch für höherdimensionale Räume

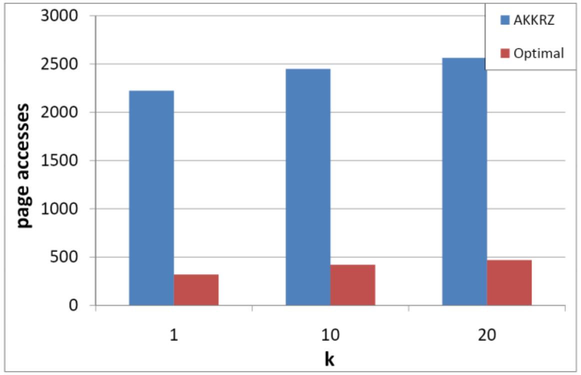
- Experimenteller Vergleich: Min/Max-Dist vs. Optimales Pruning:



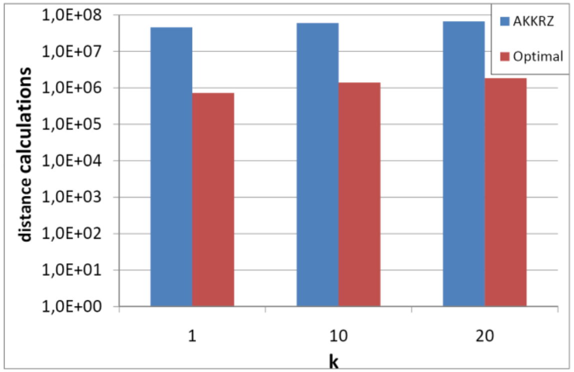
a) Anzahl der erkannten Ereignisse: Objekte in R näher an A als an B Datensatz (synthetisch): 10 Mil. Triple (A,B,R) von Rechtecken (gleichverteilt im Raum $[0,1]^2$)



b) Anzahl der Seitenzugriffe für RkNN-Anfragen mit Index Datensatz (synthetisch): 100K Punktobjekte (5D, gleichverteilt)



c) Anzahl der Seitenzugriffe für RkNN-Anfragen mit Index (I/O-Kosten) Datensatz (real): 581K Punktobjekte (10D)



d) Anzahl der Distanzberechnungen für RkNN-Anfragen mit Index (CPU-Kosten), Datensatz (real): 581K Punktobjekte (10D)

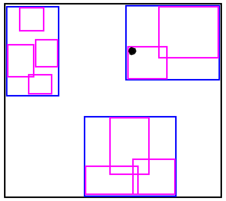
– Weiterer Anwendungsbereiche des Nachbarschaftskriteriums:

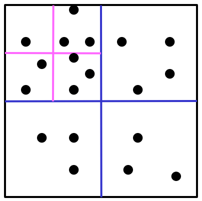
Im Prinzip lässt sich das Nachbarschaftskriterium überall dort einsetzen wo Objekte durch achsenparallele Rechtecke approximiert sind und Nachbarschaftsbeziehungen relevant sind

Beispiele:

Spatial Index:

R-tree, X-Tree, U-tree, …



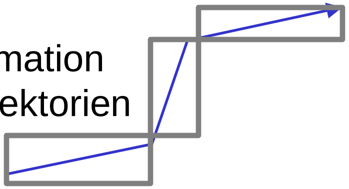


Quadtree

Anfragen:
- kNN, RkNN
- continuous kNN (RkNN)
- probabilistic kNN (RkNN)
- inverse ranking
- group Nearest Neighbor
- clustering
- usw.

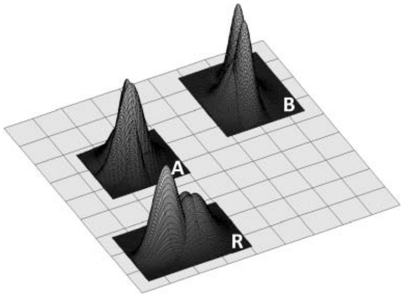
Spatio-Temporal Data:

Approximation von Trajektorien



Unsichere Daten:

Approximation von unsicheren Objekten



– Zusammenfassung

	Verfahren	Vorteile	Nachteile
self-pruning	**RNN-Tree**	Sehr gute Performanz, da keine Verfeinerung nötig	*k* fix; nur für Vektordaten; Updateproblematik; wenig selektiv bei normalen NN-Queries
	RdNN-Tree	Sehr gute Performanz, da keine Verfeinerung nötig; auf allgemein metrische Daten erweiterbar	*k* fix; Updateproblematik
	MRkNNCoP-Tree	variables *k (mit Einschränkung)*; für allgemein metrische Daten	Verfeinerung nötig, Updateproblematik
mutual-pruning	**Min/Max-Dist-basierte RkNN Suche**	variables *k*; geringe Kosten, für metrische Daten, erlaubt Pruning auf Directory-Ebene	schlechtere Filterselektivität
	Geometrische (Voronoi-basierte) RkNN Suche	variables *k*;	Nur für Vektordaten; Verfeinerung nötig
	Erweiterte geometrische RkNN Suche	variables *k*; erlaubt Pruning auf Directory-Ebene; optimale Filterselektivität	Nur für Vektordaten; teure Verwaltung der Hyperebenen
	optimal-pruning-basierte RkNN Suche	variables *k,* geringe Kosten erlaubt Pruning auf Directory-Ebene; optimale Filterselektivität;	Nur für Vektordaten;