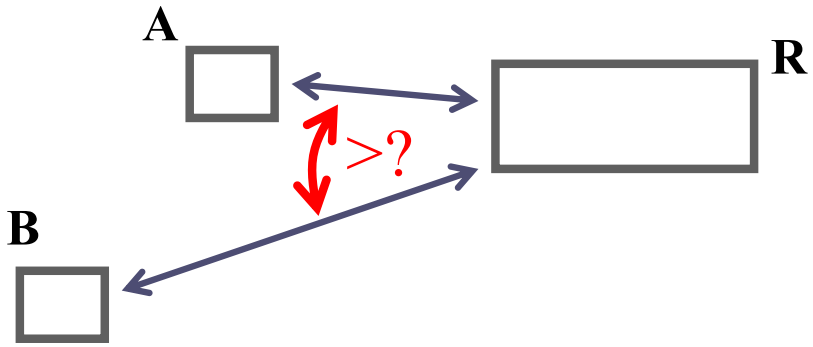
2.5 Reverse nächste Nachbarn Anfragen

- Generelle Problemstellung:

  Gegeben:

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

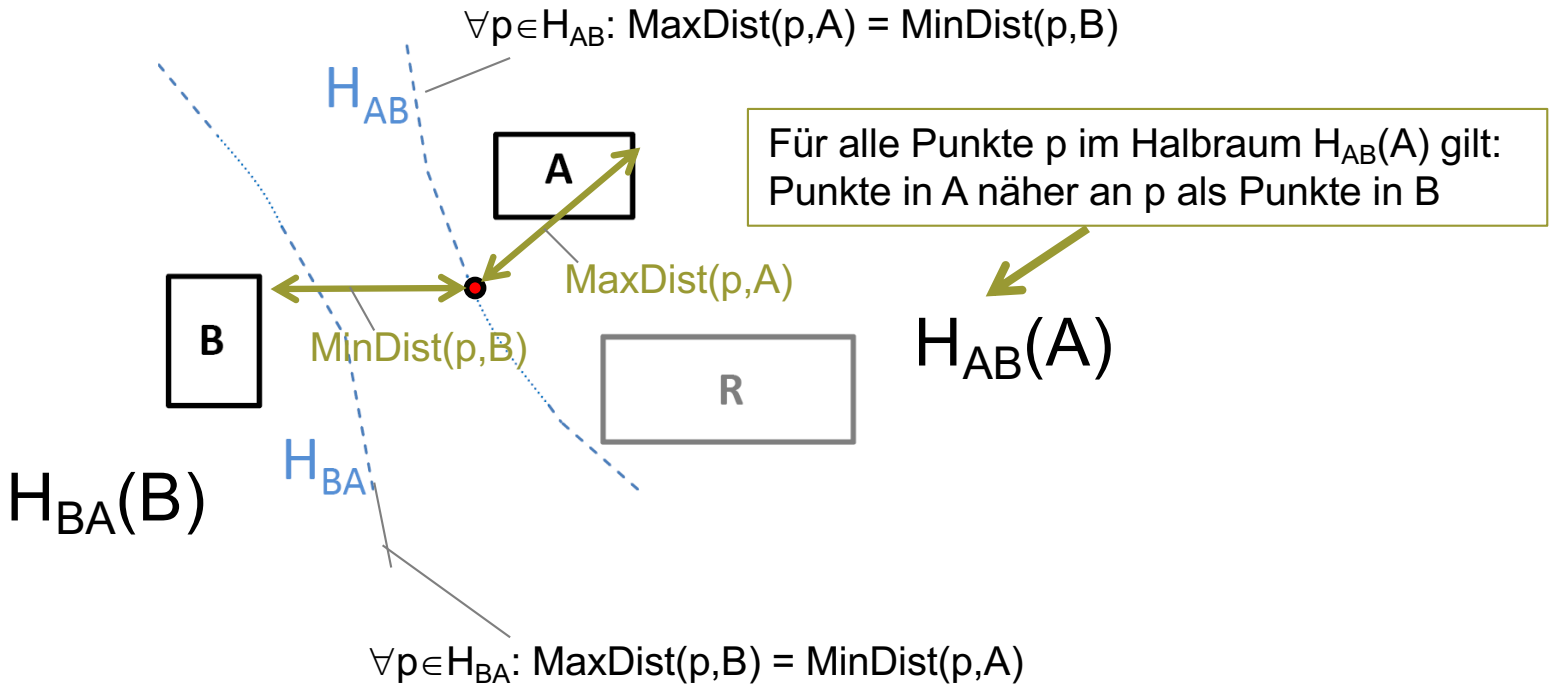


  Frage:

  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]

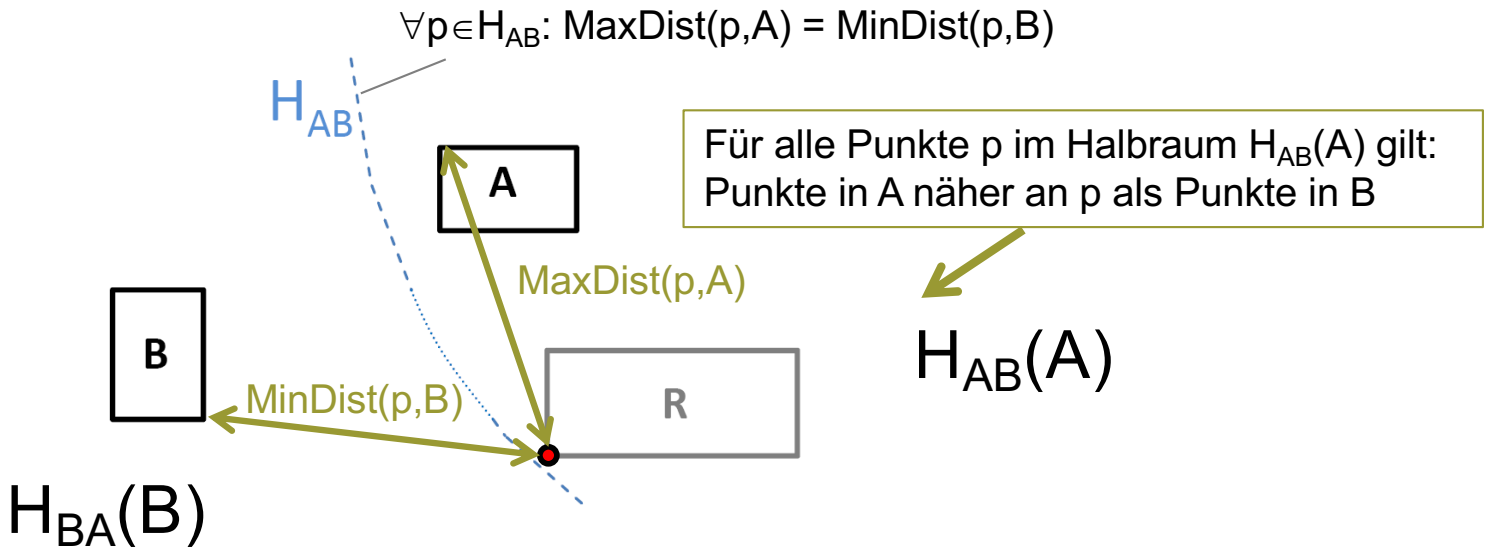
2.5 Reverse nächste Nachbarn Anfragen



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

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

2.5 Reverse nächste Nachbarn Anfragen



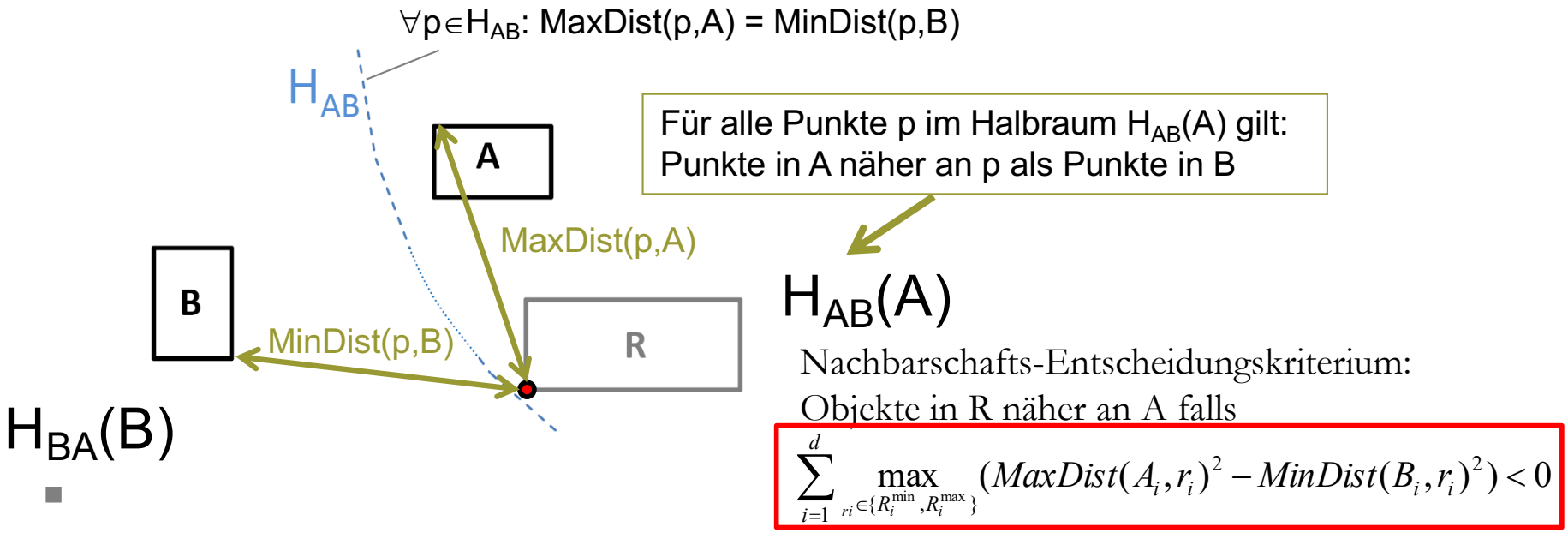
- Halbraum $H_{AB}(A)$ in der A liegt ist konvex

  → 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 $H_{AB}(A)$

2.5 Reverse nächste Nachbarn Anfragen



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

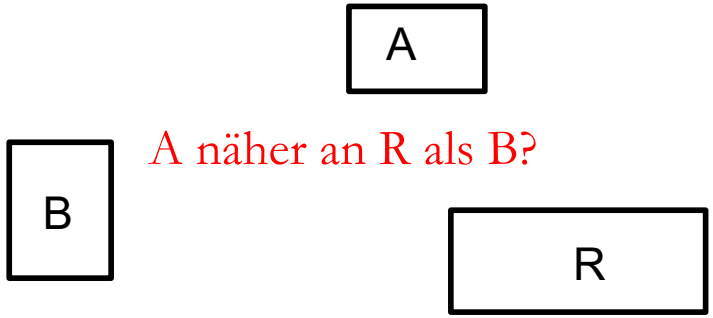
  → alle Eckpunkte von R liegen in $H_{AB}(A)$ => R liegt vollständig in $H_{AB}(A)$

- 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 $H_{AB}(A)$

2.5 Reverse nächste Nachbarn Anfragen

- Herleitung
  - Idee: Umformung der Distanzrelation:



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

$$\Leftrightarrow \forall r \in R \; MaxDist(A,r) < MinDist(B,r)$$ ← konservative Distanzabschätzung zu A und B

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

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

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

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

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

Reduzierung des Problems auf die einzelnen Dimensionen

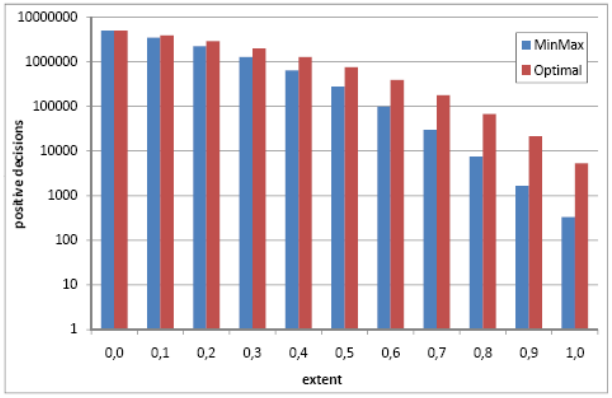
Reduzierung des Problems auf Extremwerte in R_i (pro Dimension nur 2 Werte)

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

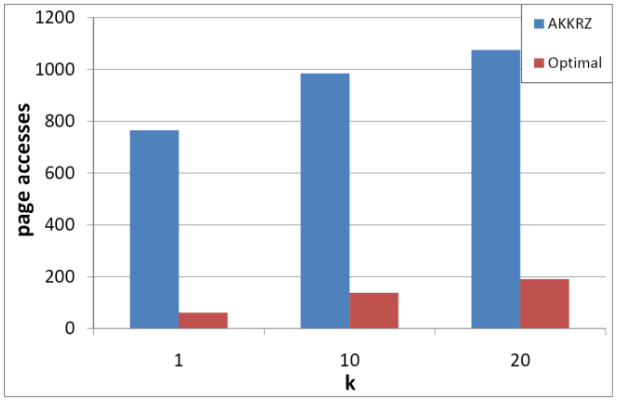
    => geeignet auch für höherdimensionale Räume

2.5 Reverse nächste Nachbarn Anfragen

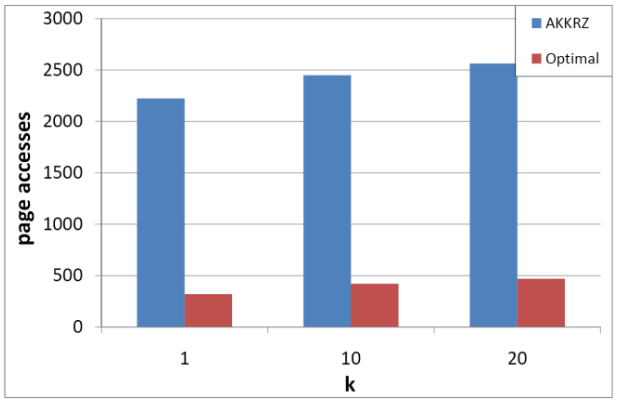
- Experimenteller Vergleich: Min/Max-Dist vs. Optimales Pruning:



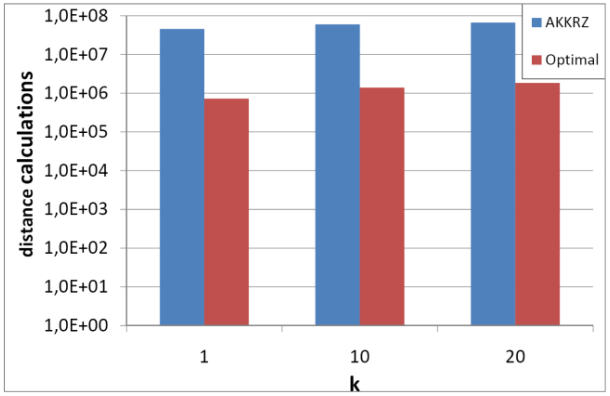
a) Anzahl der erkannten Ereignisse: Objekte in R näher an A als an B
Datensatz (synthetisch): 10 Mil. Triple (A,B,R) von Rechtecken (gleichverteilt im Raum $[0,1]^2$)



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)

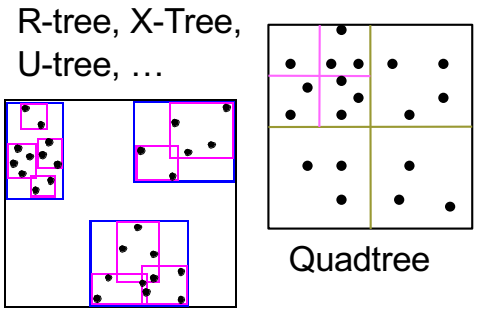
2.5 Reverse nächste Nachbarn Anfragen

- Weiterer Anwendungsbereiche des Nachbarschaftskriteriums:

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

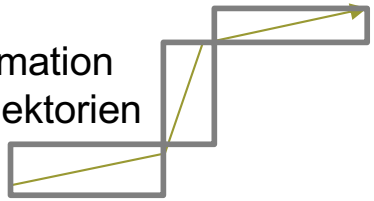
Beispiele:

Spatial Index:

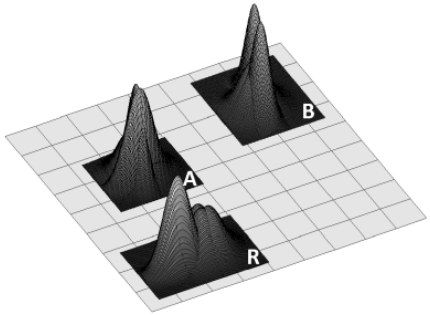


Anfragen:
- kNN, RkNN
- continuous kNN (RkNN)
- probabilistic kNN (RkNN)
- inverse ranking
- group Nearest Neighbor
- clustering
- usw.

Spatio-Temporal Data:



Unsichere Daten:



2.5 Reverse nächste Nachbarn Anfragen

- 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
self-pruning	**RdNN-Tree**	Sehr gute Performanz, da keine Verfeinerung nötig; auf allgemein metrische Daten erweiterbar	*k* fix; Updateproblematik
self-pruning	**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
mutual-pruning	**Geometrische (Voronoi-basierte) RkNN Suche**	variables *k*;	Nur für Vektordaten; Verfeinerung nötig
mutual-pruning	**Erweiterte geometrische RkNN Suche**	variables *k*; erlaubt Pruning auf Directory-Ebene; optimale Filterselektivität	Nur für Vektordaten; teure Verwaltung der Hyperebenen
mutual-pruning	**optimal-pruning-basierte RkNN Suche (Entscheidungskriterium)**	variables *k,* geringe Kosten erlaubt Pruning auf Directory-Ebene; optimale Filterselektivität;	Nur für Vektordaten;