School of
Information Systems

SMU
SINGAPORE MANAGEMENT UNIVERSITY

Geometric Approaches for Top-k Queries

[Tutorial]

Kyriakos Mouratidis

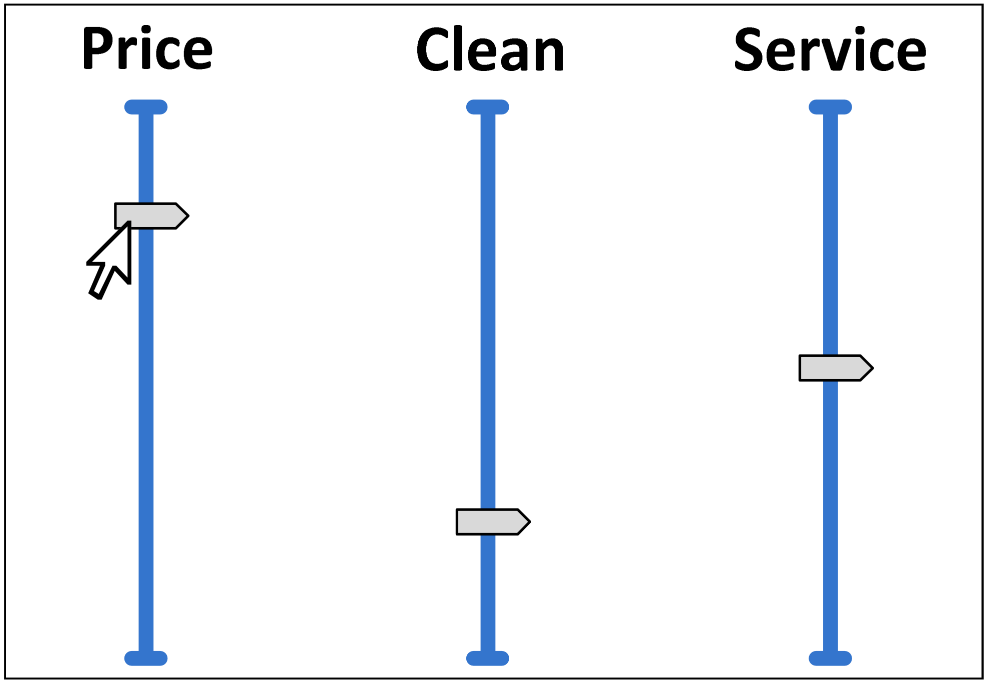
Singapore Management University

VLDB 2017

Introduction

- Top-*k* query: shortlists **top options** from a set of alternatives
- E.g. tripadvisor.com
  - rate (and browse) hotels according to price, cleanliness, location, service, etc.
- A user's criteria: ***price***, ***cleanliness*** and ***service***, with different ***weights***

Weights could be captured by slide-bars:

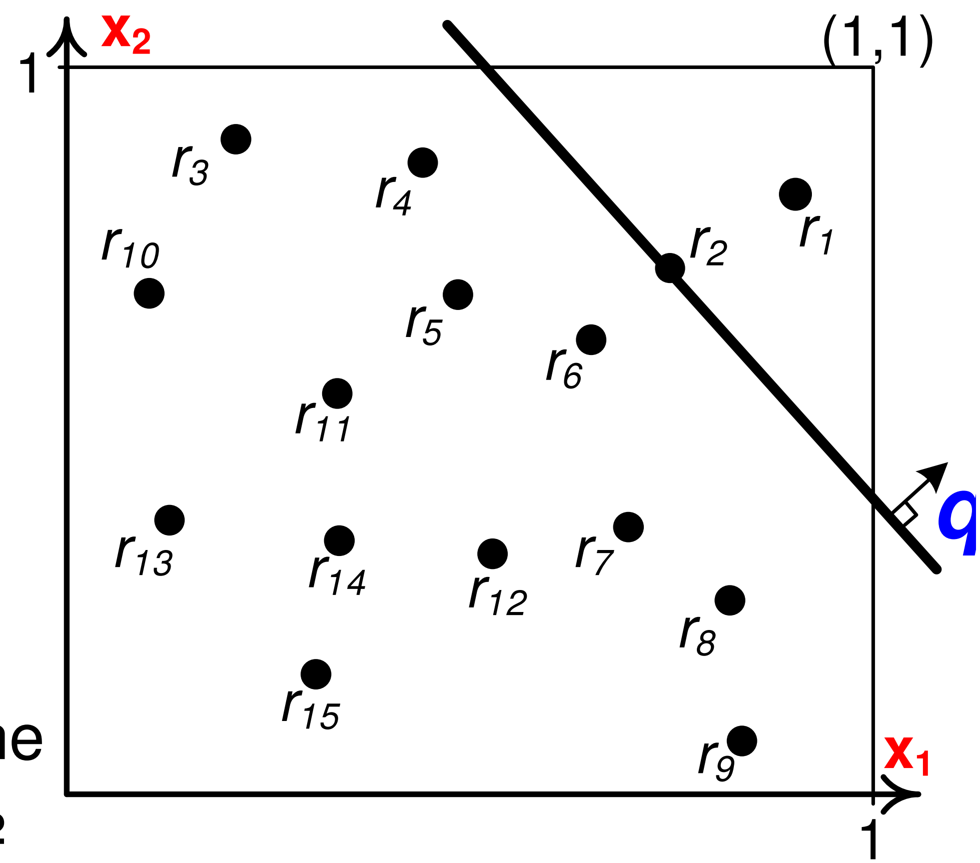


Introduction

- Slide-bar locations → **numerical weights**
- We call **q** = <0.8, 0.3, 0.5> the ***query vector***
  - and its domain the **query domain** or **query space**
- Linear function ranks hotels (i.e. **records**)
  - *score* = **0.8**·*price* + **0.3**·*clean* + **0.5**·*service*
  - if record **r** is seen as vector, *score* = dot product **r·q**
- Top-k returned (e.g. the top-10)
- Top-k processing is well-studied
  - E.g. [Fagin01,Tao07] for processing w/o & w/ index
  - Excellent survey [Ilyas08]

Top-k as sweeping the data space [Tsaparas03]

- Assume all **query weights** are **positive**
- …and each **record attribute** is in range [0,1]
- Example for d = 2 (showing: **data space**)
- **Sweeping line** normal to vector **q**
- Sweeps from top-corner (1,1) towards origin
- Order a rec. is met ↔ order in ranking!
  - E.g. top-2 = { r_1, r_2 }
- At current position:
  - ∀ rec. above (below) the line higher (lower) score than $\mathbf{r_2}$

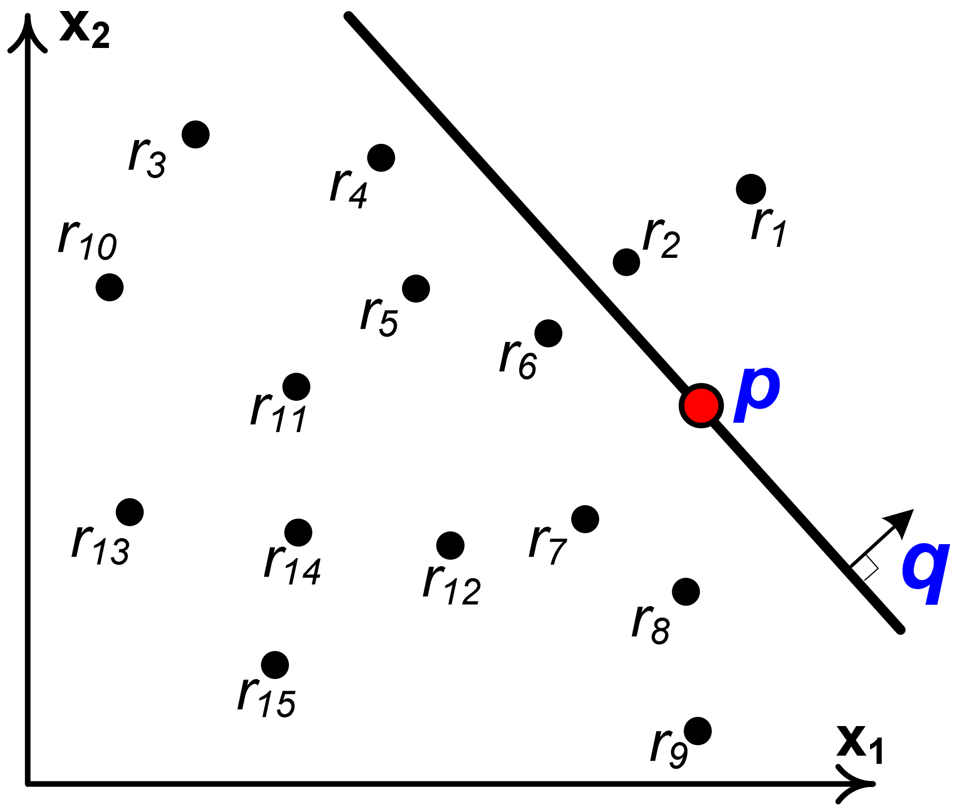


Notes on dim/nality of query domain

- Ranking of recs. depends only on **orientation** of sweeping line (or hyper-plane, in higher dim.)
  - query vector <0.8,0.3,0.5> same effect as <8,3,5>
- $\Rightarrow$ we can normalize **q** so that sum of weights is 1 (without affecting at all the top-k semantics)
  - e.g. in 2-D we can rewrite scoring function as $S(r) = \boldsymbol{\alpha} \cdot x_1 + (1-\boldsymbol{\alpha}) \cdot x_2$
- This reduces dim/nality of query domain by 1
  - Geom. operations in query domain become faster
- We'll ignore this in the following for simplicity

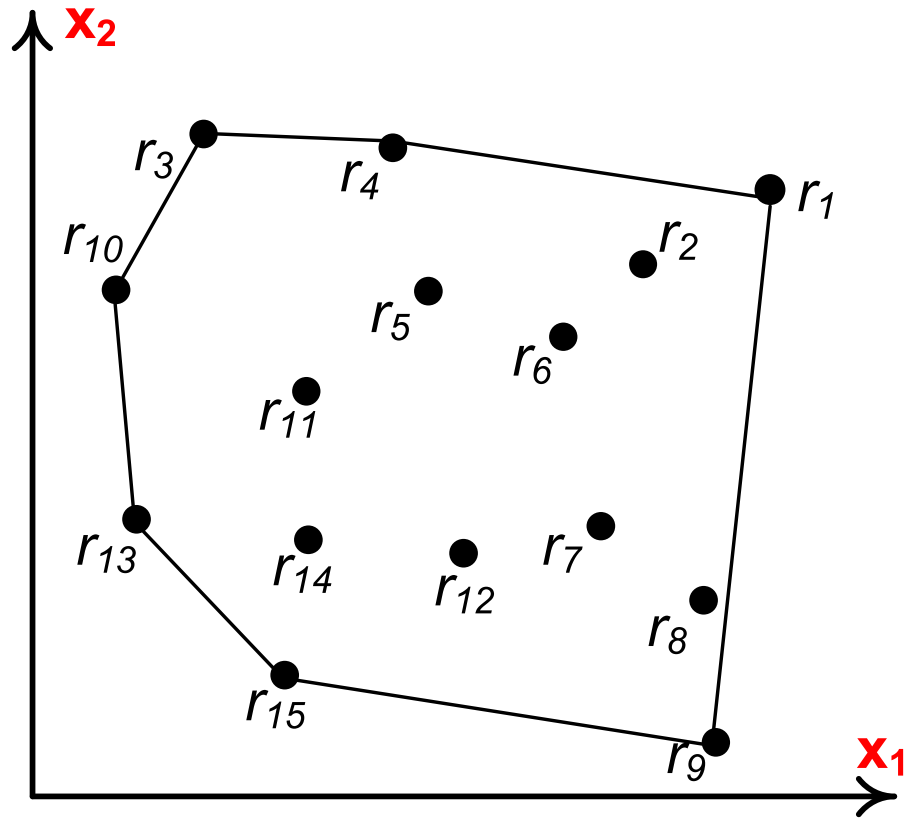
Half-space range reporting

- Half-space range (HSR) **reporting**: preprocess a set of points s.t. all points that lie above a **query hyperplane** can be reported quickly
  - Equiv: given query vector **q** and focal rec. **p**, report all recs. that score higher
- HSR **counting**: report just no. of points
  - Equiv: given **q** and **p**, report the **rank** of **p**



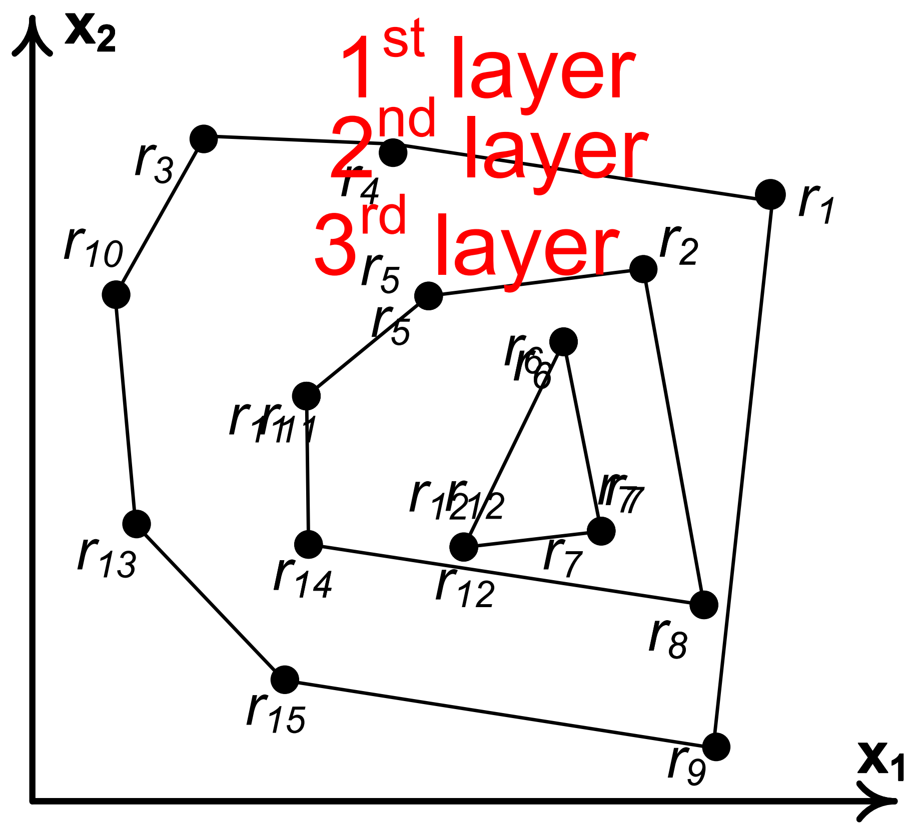
Relationship to Convex Hull

- **Convex Hull**: The smallest convex polytope that includes a set of points (records)
- Fact: The top-1 record for **any** query vector is on the hull!
  - [Dantzig63]: LP text



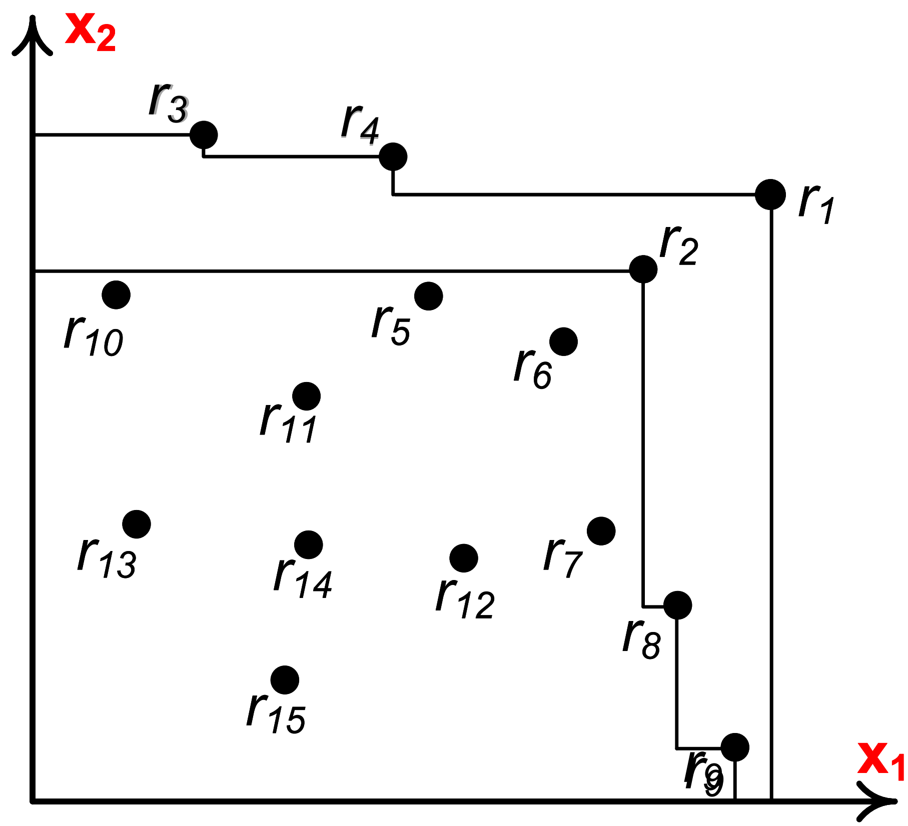
[Chang00]: Onion Technique

- **Onion**: Materialization to speed up top-k search
- 1st layer = CH
  - contains top-1 rec. ∀ **q**
- 2nd layer = CH of recs. except 1st layer
  - 1st and 2nd layer contain top-2 recs. ∀ **q**
- 3nd layer = CH of recs. except 1st and 2nd layer...
- Top-k records for any **q** are among k top layers!



[Börzsönyi01, Papadias03]: Skyline

- **Dominance:** rec. $\mathbf{r_1}$ dominates $\mathbf{r_2}$ iff it has higher values in all dimensions [ignore ties]
- $\Rightarrow S(\mathbf{r_1}) > S(\mathbf{r_2}) \ \forall \ \mathbf{q}$
- **Skyline**: all recs. that aren't dominated
- Includes top-1 $\forall$ **q**
- **k-skyband**: all recs. not dominated by k or more others
- Includes top-k $\forall$ **q**



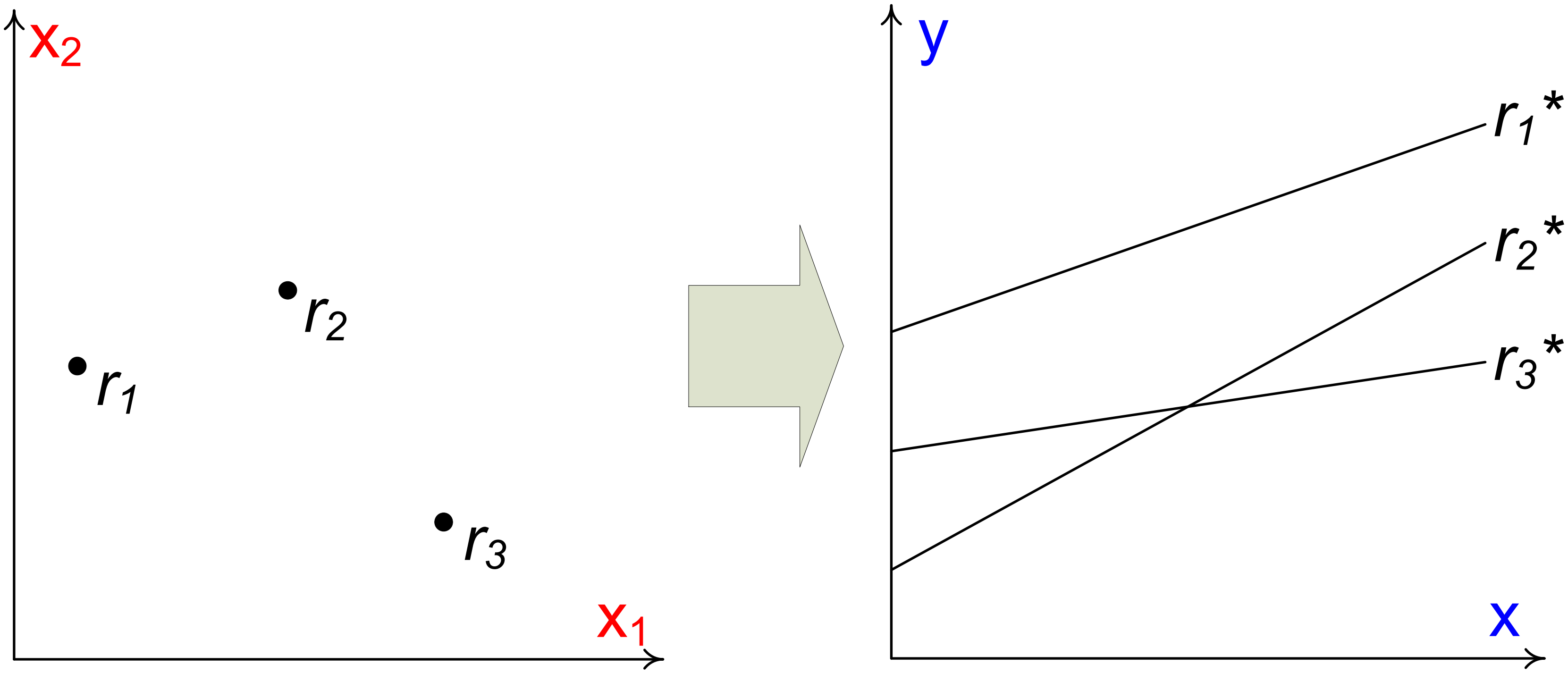
[Das07]: Duality, 2D

- Overview: **dual transformation** used to process **ad-hoc** top-k queries on a dynamic buffer (e.g. sliding window)

- Insertions and deletions made to the buffer
- One-off (snapshot) top-k queries posed
- Objective: to maintain a **subset** of records in buffer, guaranteed to include the top-k result of ***any*** ad-hoc query

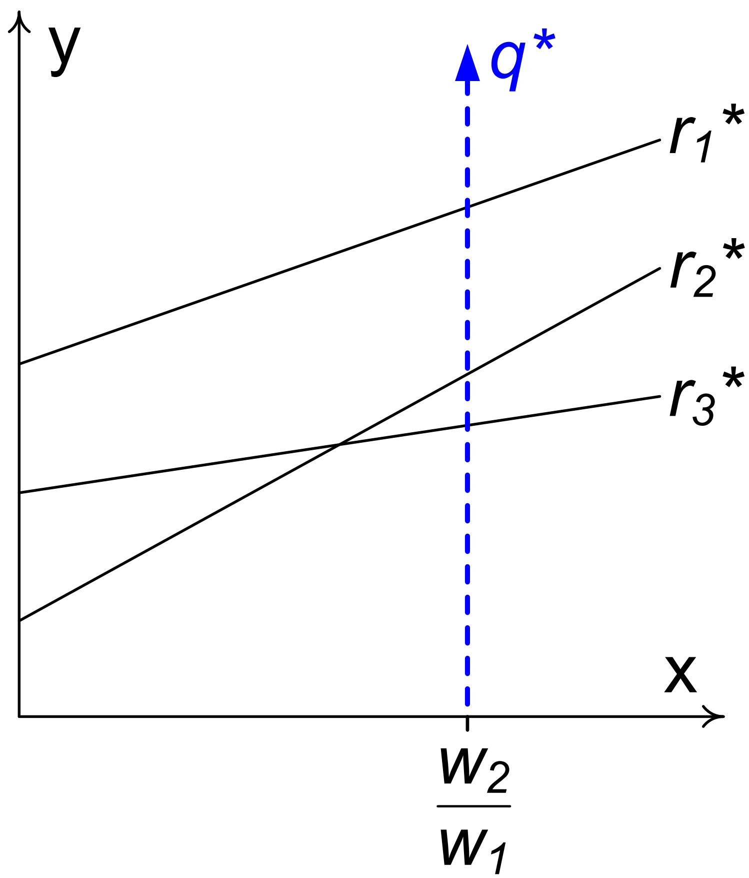
[Das07]: Duality, 2D

- **Dual transformation**: Points mapped to lines
  - rec. (x_1,x_2) mapped to line $\mathbf{y} = (1 - x_2)\mathbf{x} + (1 - x_1)$
  - Observe: all lines have positive slope



[Das07]: Duality, 2D

- **Dual transformation**: Queries to vertical rays
  - q = (w_1,w_2) mapped to ray from point (w_2/w_1,0)



<u>**Order**</u> ray **q*** hits line **r*** ⇔ <u>**Rank**</u> of **r** in the result of **q**

I.e. top-2 result = {r_3,r_2}

[Das07]: Duality, 2D

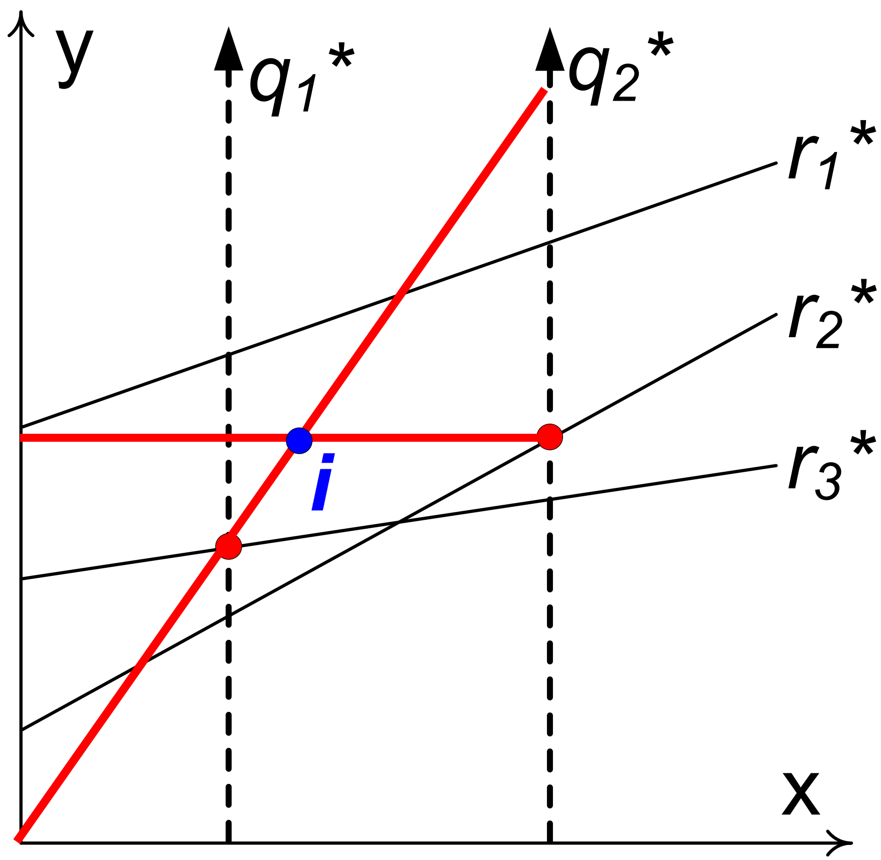
- Idea 1: Maintain arrangement of lines induced by all records in the buffer

- Issue: arrangement costly to compute/update!
  - Arrangement computation in 2-D: $O(n^2)$

- Idea 2: keep only lines that could appear among the k lowest lines in the arrangement

[Das07]: Duality, 2D

- Consider 2 queries, and their top-k points
- They define two pruning lines

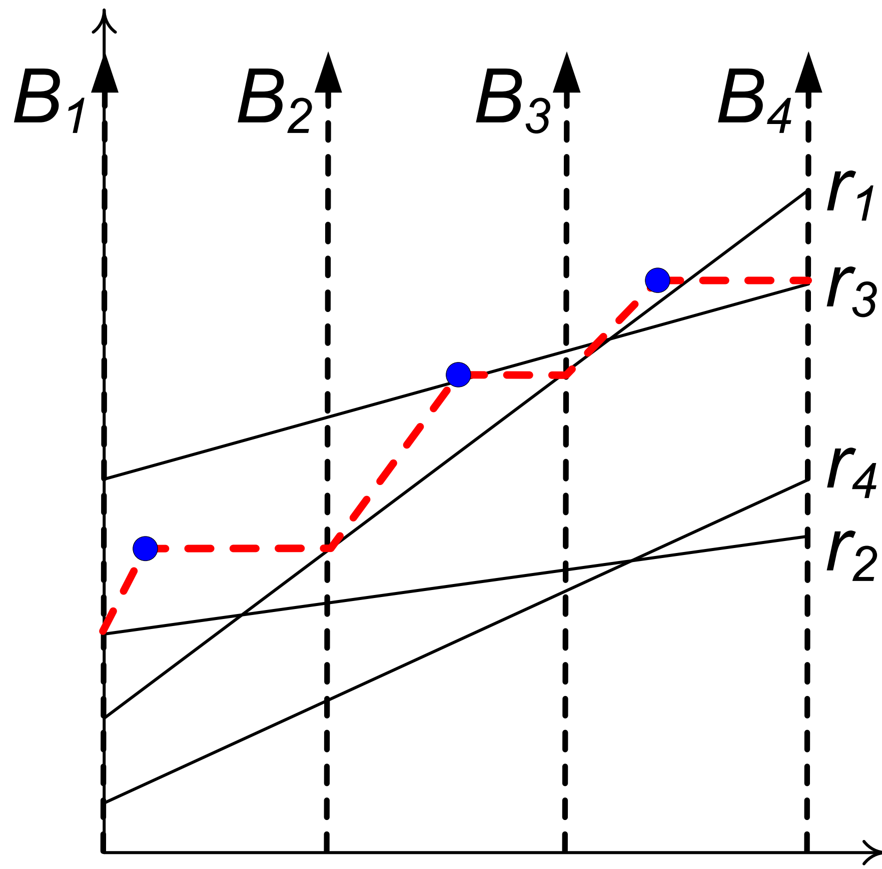


Their intersection = **pruning point** i

If a line **r*** is above **i** then **r** cannot be in the result of **any** query between q_1 and q_2

[Das07]: Duality, 2D

- Use **border queries** (like q_1, q_2) to partition the arrangement into **strips**
- Maintain <u>top-k points</u> of border queries and a <u>pruning point in each strip</u>
- In each strip, maintain a **local arrangement**, excluding lines above the pruning point
- Ad-hoc query posed: identify its strip, look for k first lines its ray hits in the local arrangement



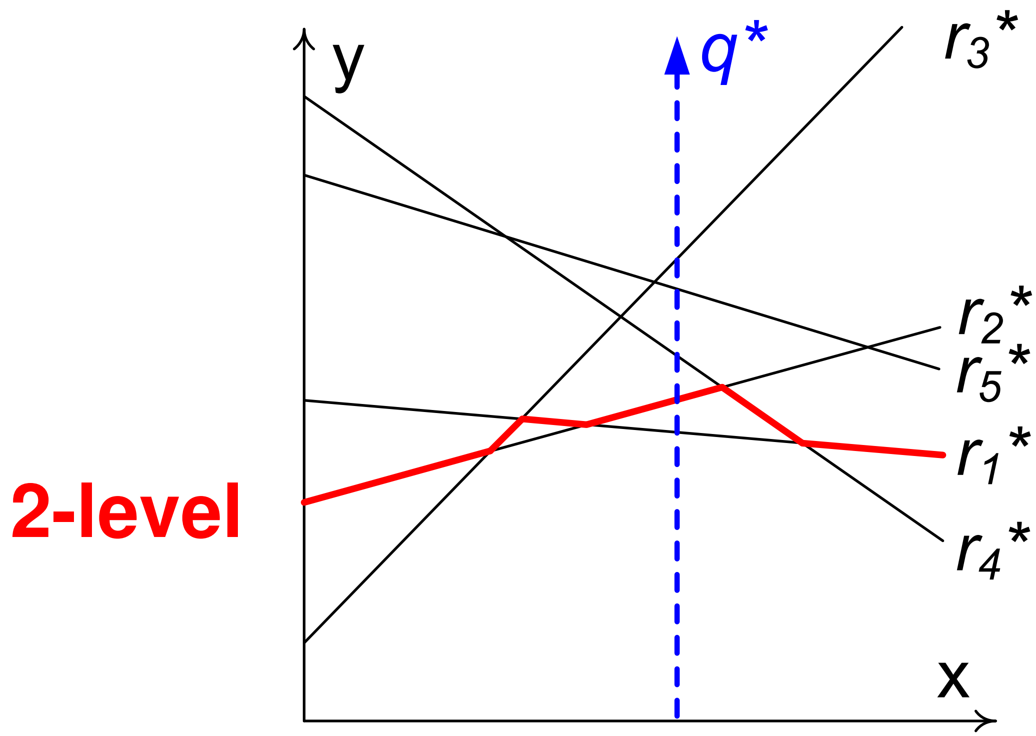
[Yu12]: Duality, higher-D

- Overview: **dual transformation** used to process **continuous** top-k queries on a dynamic buffer (e.g. sliding window)

- Insertions and deletions made to the buffer
- Continuous top-k queries posed
- Objective: refresh the top-k results as fast as possible

[Yu12]: Duality, higher-D

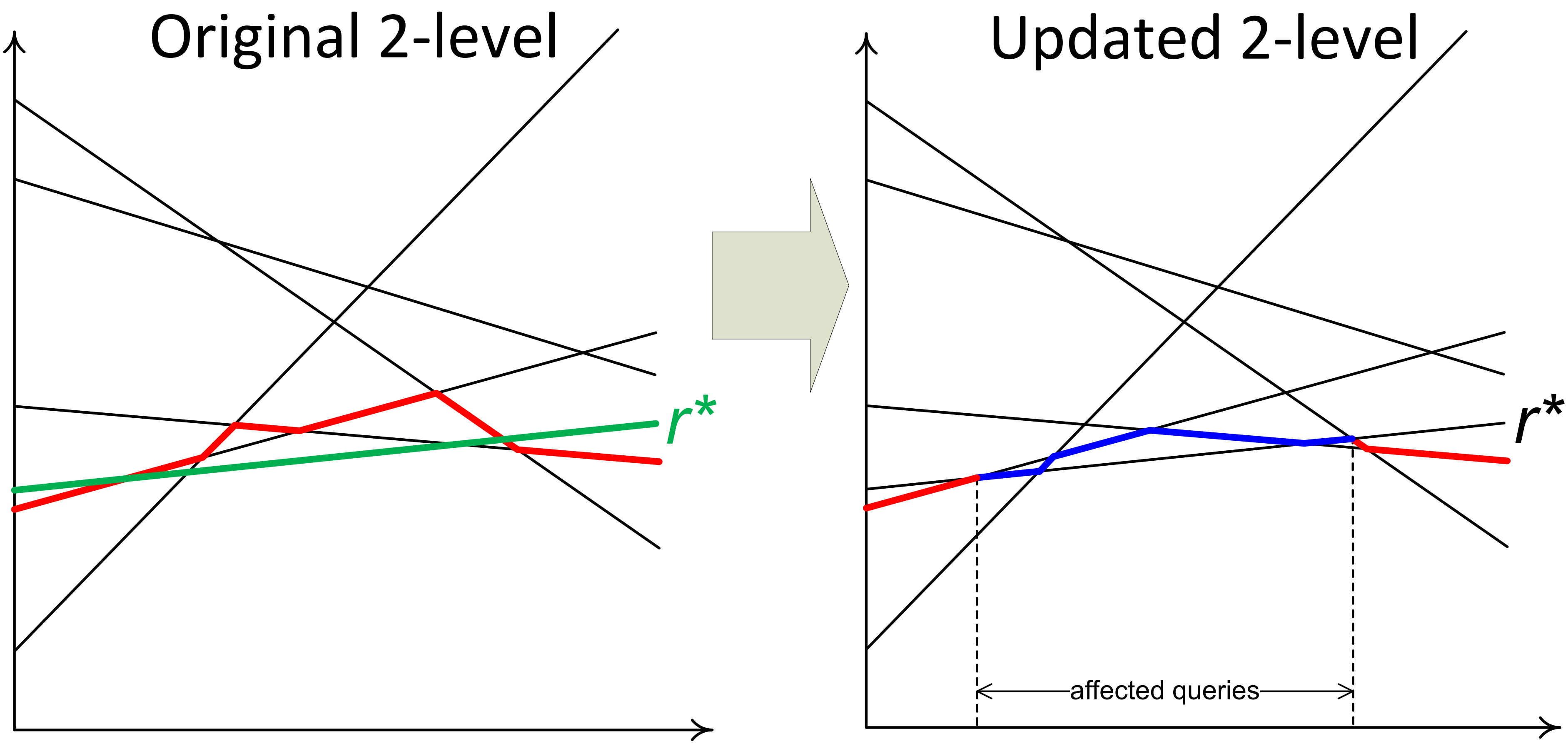
- **k-level**: set of edges (facets) in the arrangement w/ exactly k-1 others below them
- **k-level captures the k-th result of any query!**



2nd top record of q is r_2

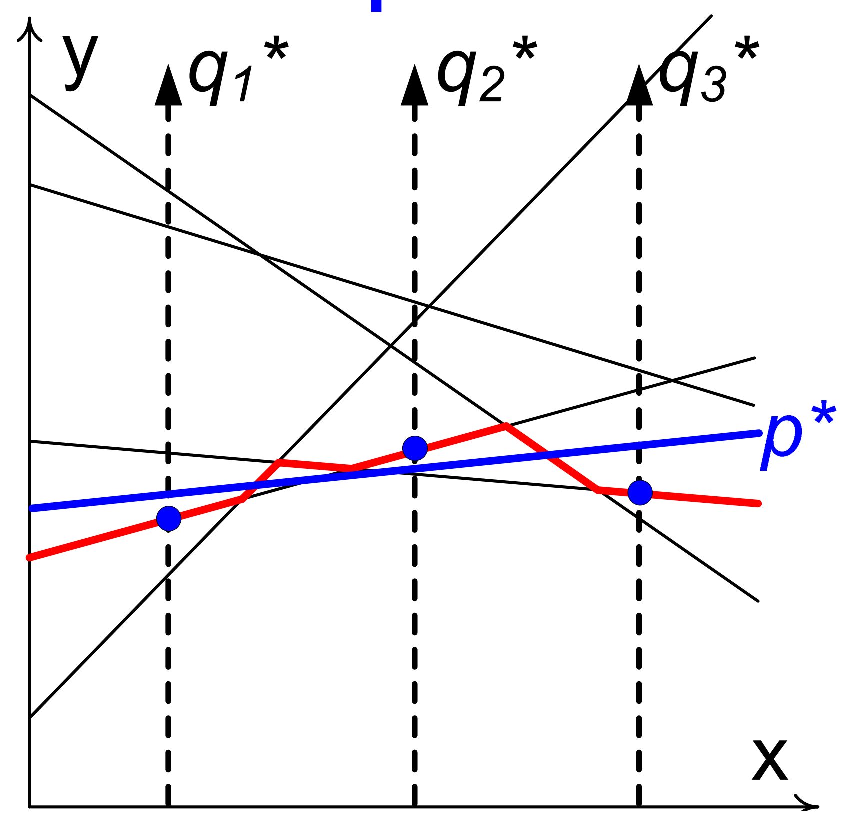
[Yu12]: Duality, higher-D

- Consider record **r** insertion (deletion is similar)
  - Affected queries = those under new edges in k-level



[Yu12]: Duality, higher-D

- A by-product: preprocessing method for **(bichromatic) reverse top-k queries (RTOP-k)** [Vlachou10 & 11]
- Given a **focal record p**, a set of records, and a set of top-k queries, find the queries that have **p** in the result
- Prep: Find top-k points of all queries, i.e., intersections of query rays and the k-level
- Index these points
- Posed a RTOP-k query for **p**, report those queries whose top-k point is above **p***
- Ex: RTOP-k includes only q_2



[Soliman11]: Repr/tives & measures

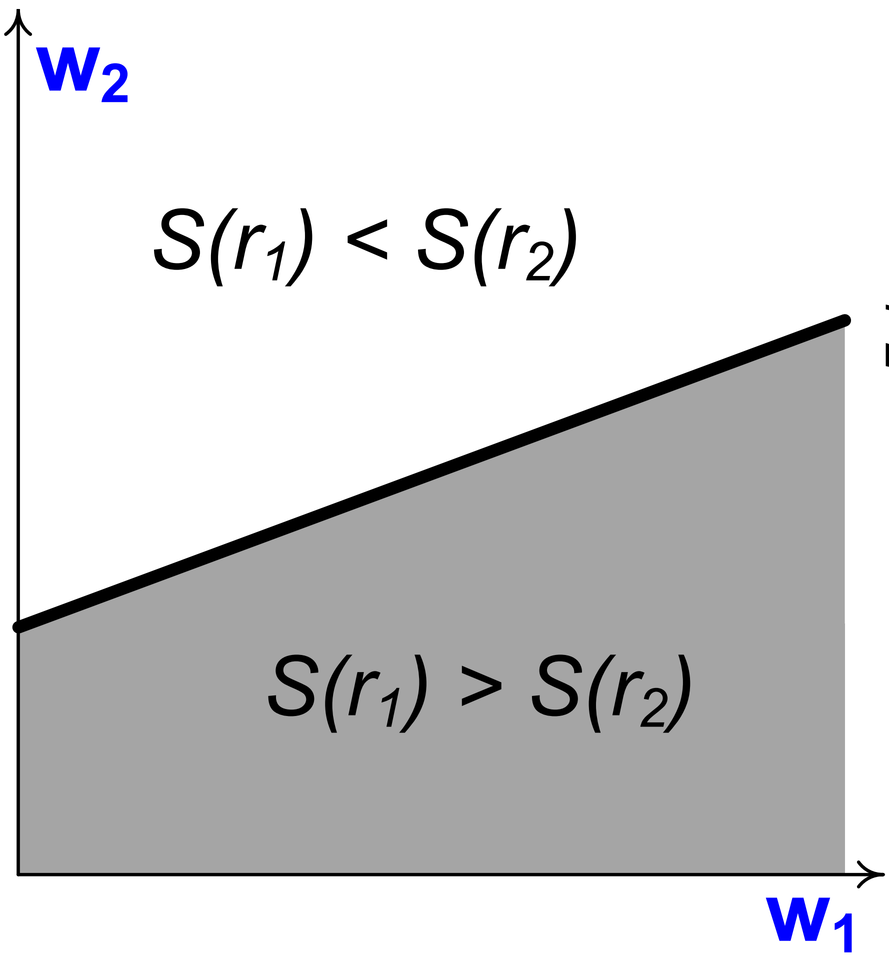
- Defines 4 problems:

1. **MPO**: Find the most probable top-k result (if query vector is randomly & uniformly chosen)
2. **ORA**: Find the top-k result with minimum summed distance from all others
3. **STB**: Find maximum radius ard. **q** where top-k result remains the same
4. **LIK**: Find probability that a randomly & uniformly chosen query has same result as **q**

MPO&ORA: Repr/tives; STB&LIK: Sensitivity!

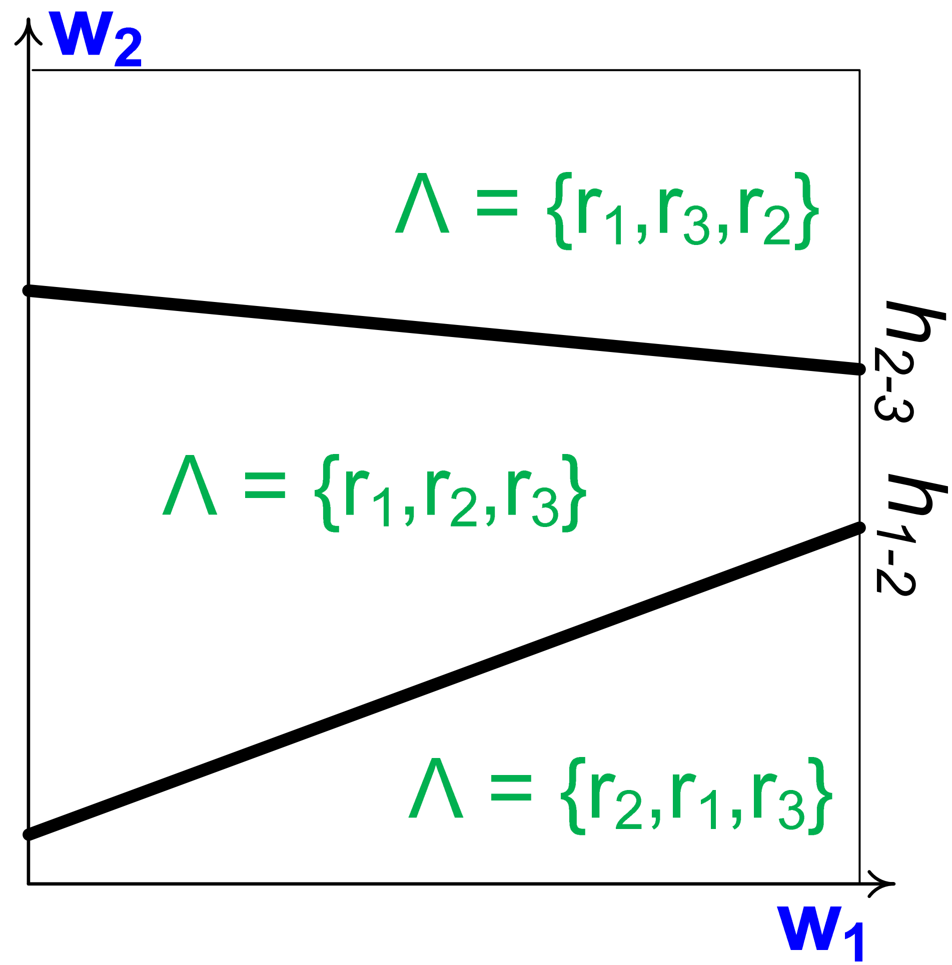
[Soliman11]: Repr/tives & measures

- MPO & ORA key idea:
- For $\mathbf{r_1}$, $\mathbf{r_2}$: equality $S(r_1) = S(r_2)$ maps into **hyperplane** in **query domain**!
- **Every pair** of records induces a hyperplane
- Producing an **arrangement**!



[Soliman11]: Repr/tives & measures

- Every **cell** corresponds to different **full ordering Λ** of the records!
- Possible orderings: $O(n^{2^{\wedge}(d-1)})$
- Top-k result ↔ k-prefix of **Λ**
- Enumerate, compute **volume**, report **MPO**
- Bottom-up or top-down processing



[Soliman11]: Repr/tives & measures

- Experiments for **MPO** only

- **ORA** solution utilizes specific characteristics of distance function (Kendall tau & Footrule)
- …and approximation/sampling (in the case of Kendall tau)

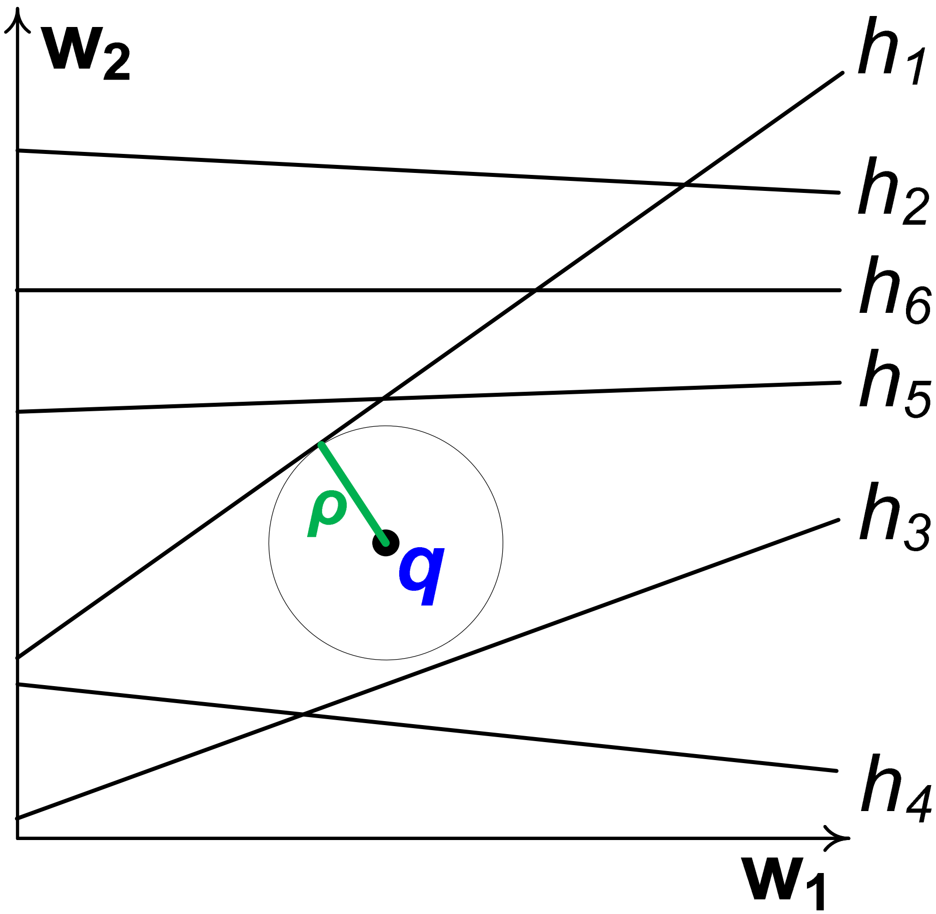
[Soliman11]: Repr/tives & measures

- **STB**: Given **q**, find max. radius **ρ** that vector **q** can move without changing top-k result:
- **Order** within result retained
  - i.e. $S(r_1) > S(r_2)$ and $S(r_2) > S(r_3)$ … $S(r_{k-1}) > S(r_k)$
  - k-1 conditions (**O-conditions**)
- **Non-results** cannot overtake $\mathbf{r_k}$
  - i.e. $S(r_k) > S(r)$ for every non-result **r**
  - n-k conditions (**NR-conditions**)
- **Observation:** each condition ↔ a hyperplane!

[Soliman11]: Repr/tives & measures

- **STB** solution: Compute dist. from **q** to **<u>each</u>** of the n-1 hyperplanes
- **ρ** is the min. of these distances!
- Cost: O(nd)

- **LIK**: compute **the cell** including **q** (and then its volume)
- Cost: $O(n^{2^{(d-2)}})$



[Zhang14]: Global Immutable Region

- [Zhang14]: Actually, with half-space intersection (n-1 O-conditions & NR-conditions):
- Cost: $O(n^{d/2})$
- Computes the **cell** enclosing **q** $\leftrightarrow$ **GIR**!

- **Global Immutable Region (GIR)**
  - The **maximal** region around query vector ***q*** where the top-*k* result remains the same

[Zhang14]: Global Immutable Region

- Hotels with attributes *location*, *service*

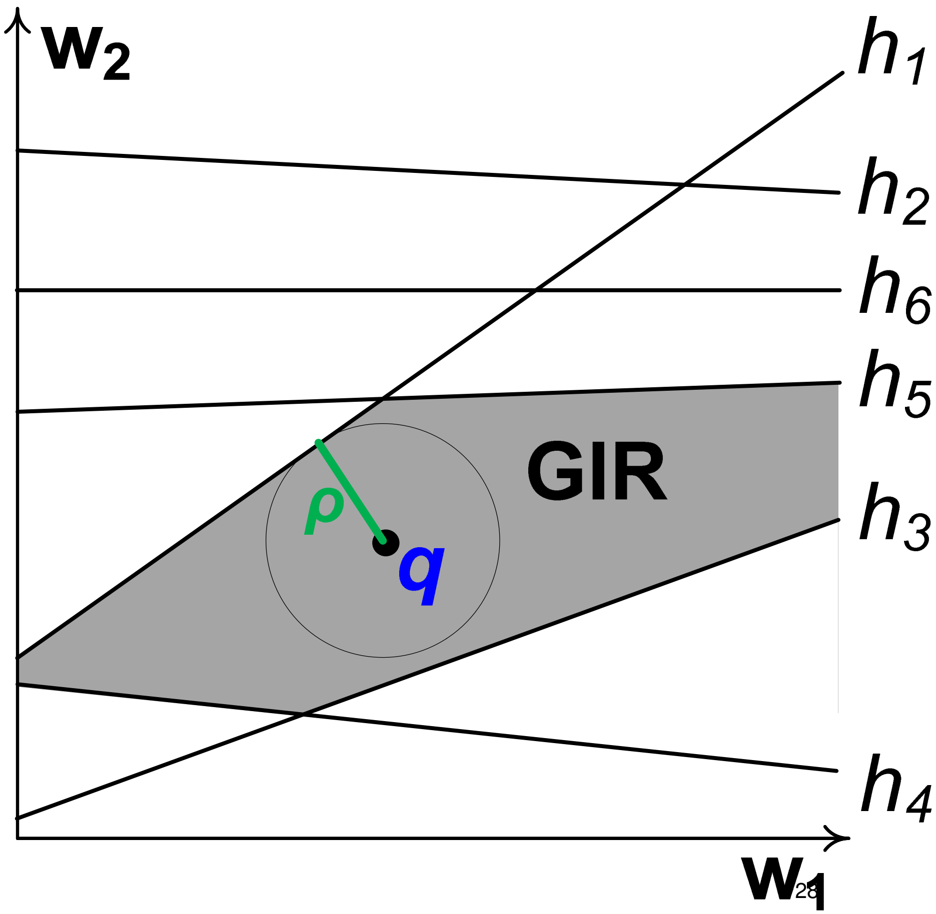
Option	Location	Service
1	0.8	0.9
2	0.2	0.7
3	0.9	0.4
4	0.7	0.2
5	0.4	0.3
6	0.5	0.5

- Query weights in [0,1]
- For $\boldsymbol{q}$ = <0.5, 0.5> top-3 result is: p_1, p_3, p_6
- Which other possible queries would have the same top-3?

[Zhang14]: Global Immutable Region

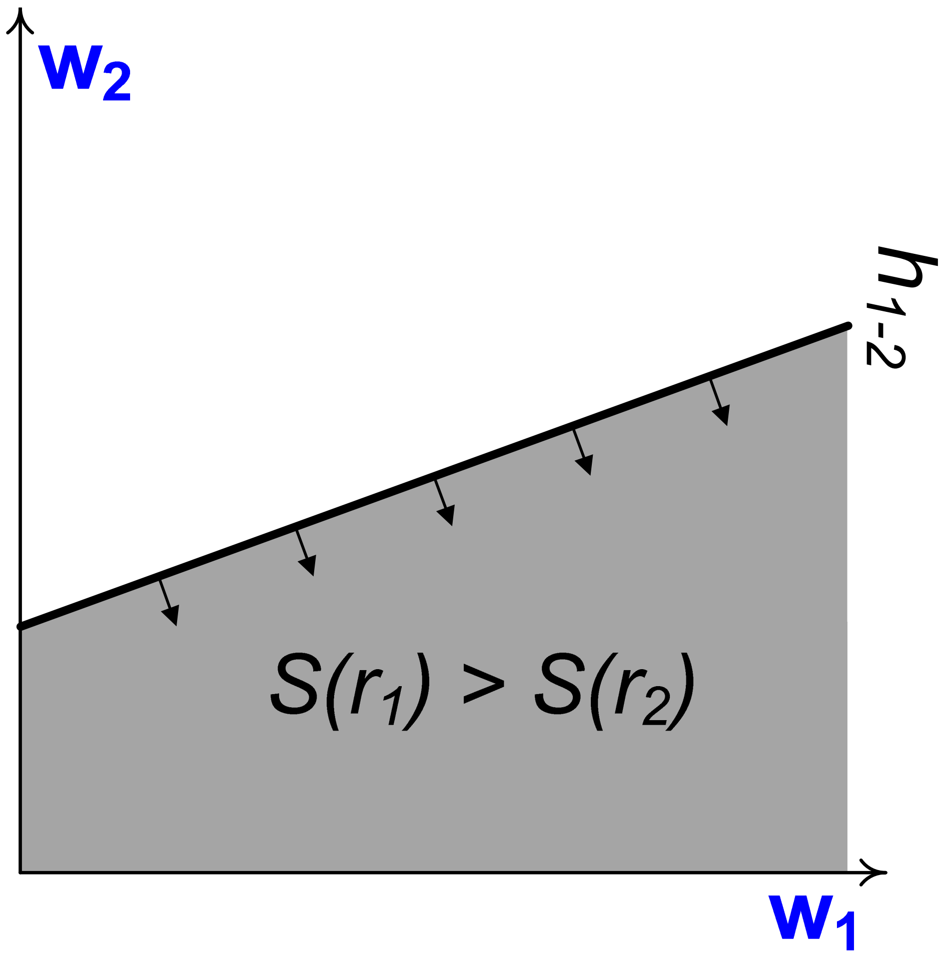
- Answer:
  Every query vector in shaded area (GIR)
- Applications:
  - Sensitivity analysis
    - E.g. volume of GIR equals to probability that a random query vector returns same result as *q*
  - Result caching
  - Weight readjustment

Observe difference from **STB**



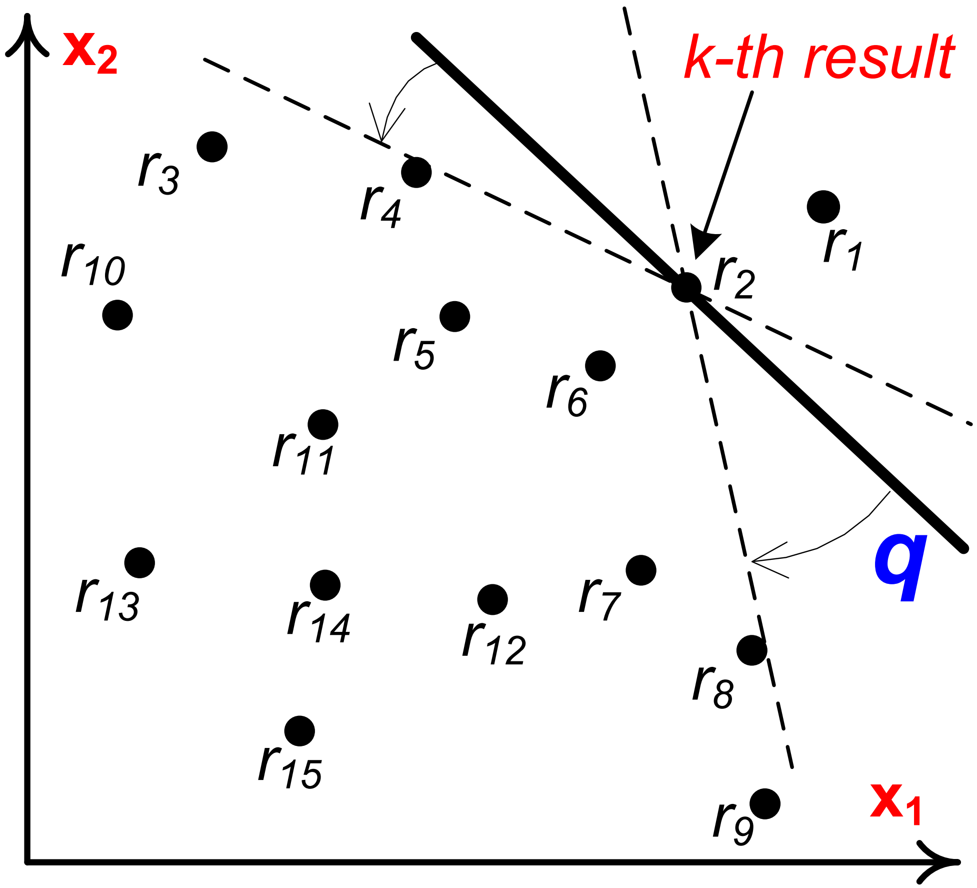
[Zhang14]: Global Immutable Region

- **Basic Alg.**: There are **k-1 O-cond/s** (e.g. $S(r_1) > S(r_2)$)
- …and **n-k NR-cond/s** ($S(r_k) > S(r)$ $\forall$ non-result **r**)
- Each condition $\leftrightarrow$ a **half-space**!
- Intersect all half-spaces
- Cost: $O(n^{d/2})$
- Problem: Too expensive
- Idea: limit no. of NR-conditions!
- …i.e. prune non-results!



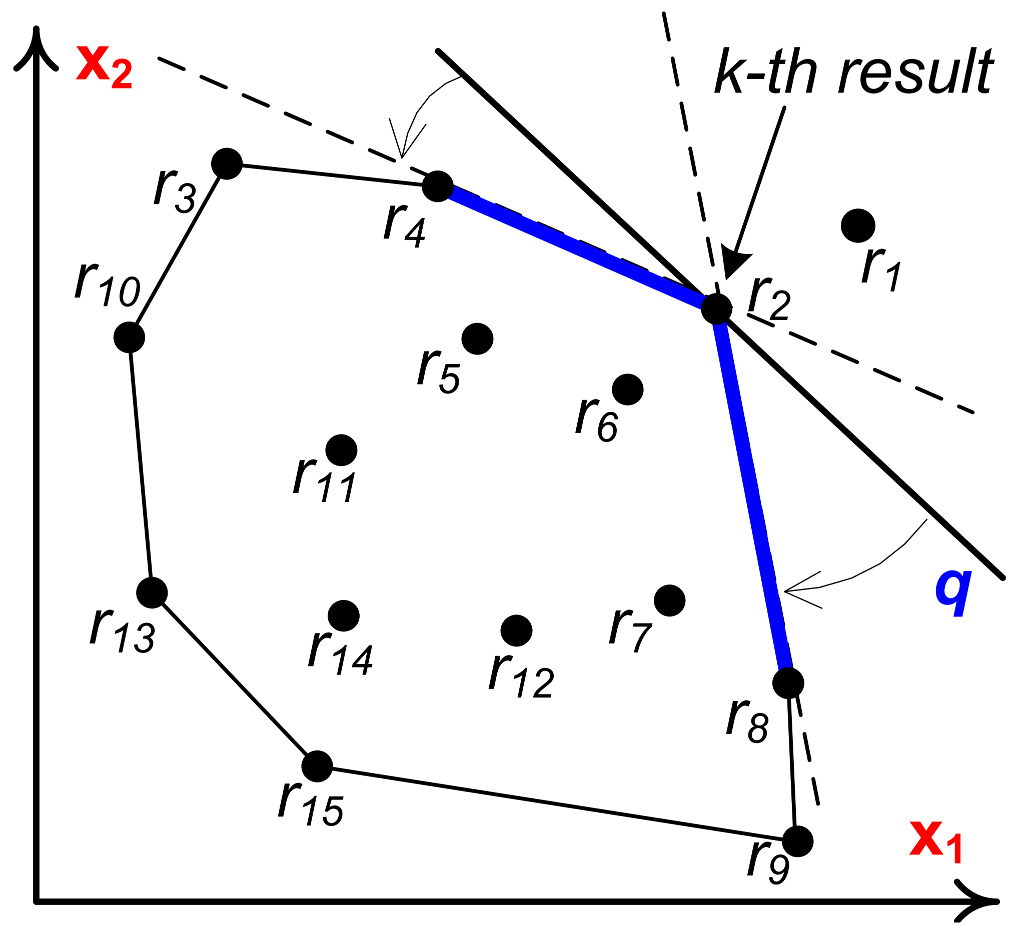
[Zhang14]: Global Immutable Region

- Observation: **pin** sweeping line at $\mathbf{r_k}$ and consider all orientations that keep NRs below it!
- Tilting bound **only by r_4 and r_8**
- NR conditions only for r_4 and r_8 !
- Formalize??



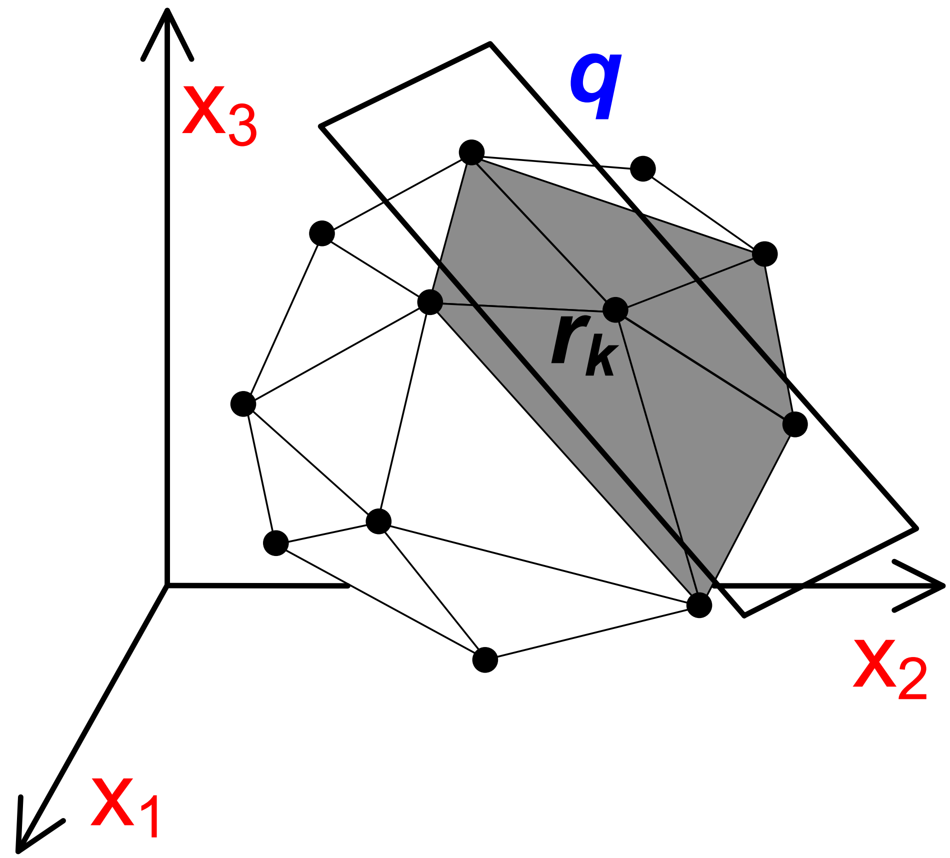
[Zhang14]: Global Immutable Region

- **<u>Facet pruning</u>**:
- Consider CH of r_k and NRs
- Only CH facets adjacent to r_k affect the GIR!
  - Consider only NRs on adj. facets
- Optimization: ONLY compute adj. facets (not entire CH)



[Zhang14]: Global Immutable Region

- The same applies to any dimension!
- E.g. for d = 3



[Mouratidis15]: MaxRank

- **MaxRank query**: given a focal record ***p***, find:

  1. The highest rank ***p*** may achieve under **any possible** user preference, and

  2. All the regions in the query vector's domain where that rank is attained

[Mouratidis15]: MaxRank

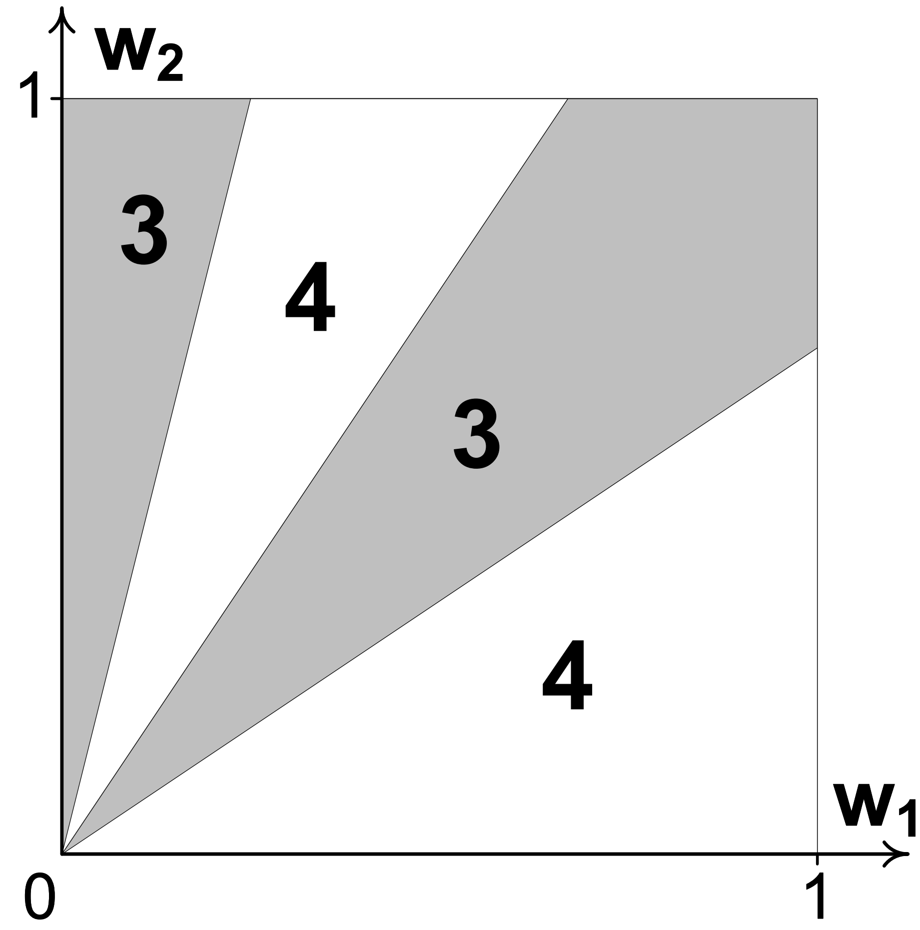
- Hotels with attributes *location*, *service*

Option	Location	Service
1	0.8	0.9
2	0.2	0.7
3	0.9	0.4
4	0.7	0.2
5	0.4	0.3
p (focal)	0.5	0.5

- Query weights in [0,1]
- If ***q*** = <0.7, 0.3>
  order of ***p*** is **4**
- If ***q*** = <0.1, 0.9>
  order of ***p*** is **3**

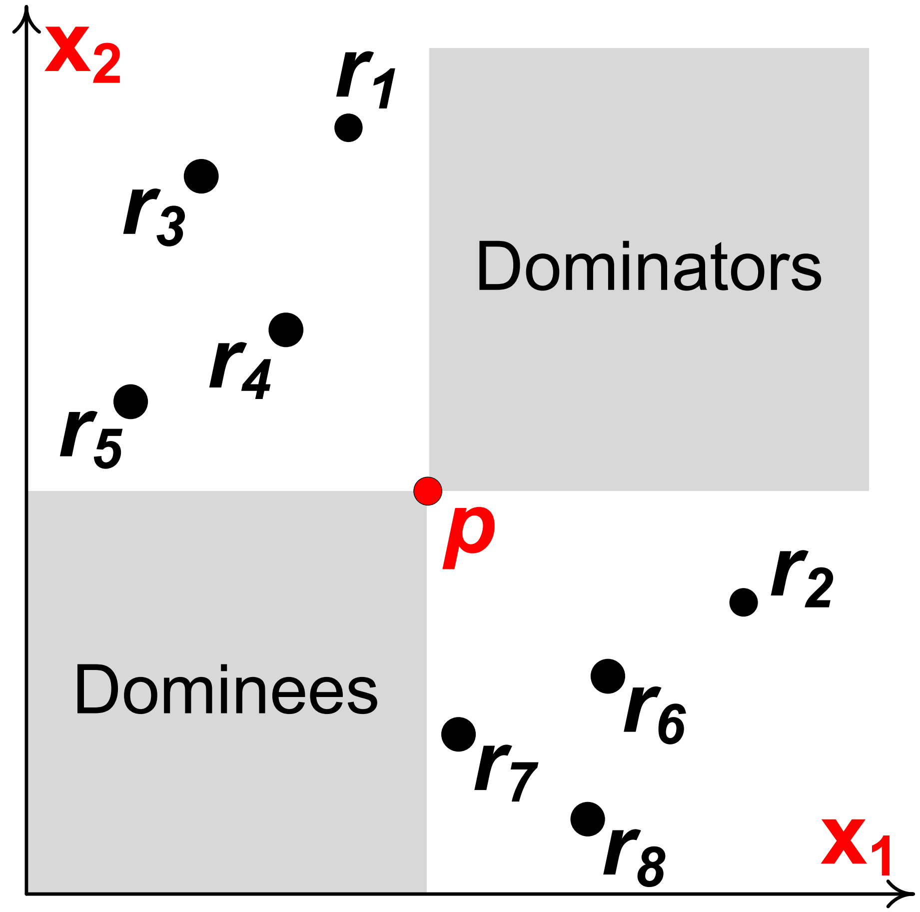
[Mouratidis15]: MaxRank

- Query domain
- Order of ***p***
- *MaxRank* result:
  - Min. order $k^* = 3$
  - *MaxRank* regions: shaded wedges
- Applications:
  - Market impact analysis
  - Customer profiling
  - Targeted advertising



[Mouratidis15]: MaxRank

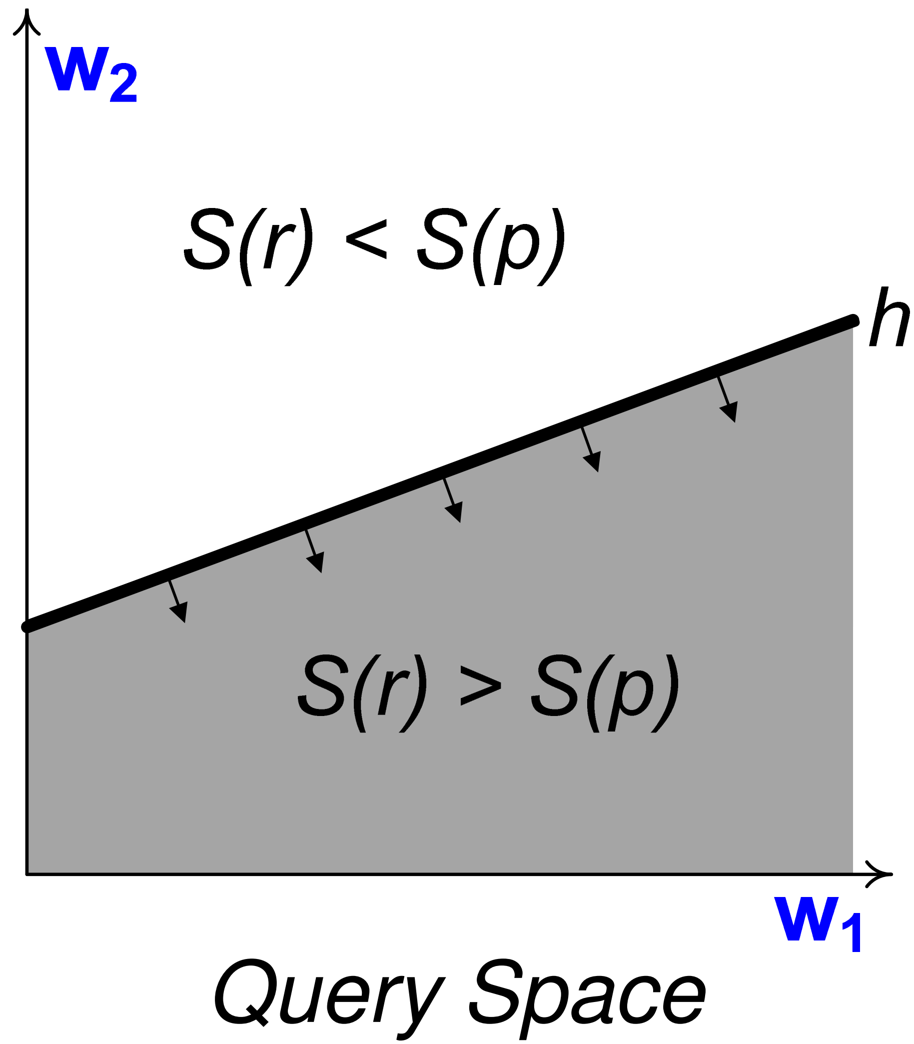
- *Dominees*
  - ignore
- *Dominators*
  - simply increment *k**
- *Incomparable*
  - How to deal with them?



Data Space

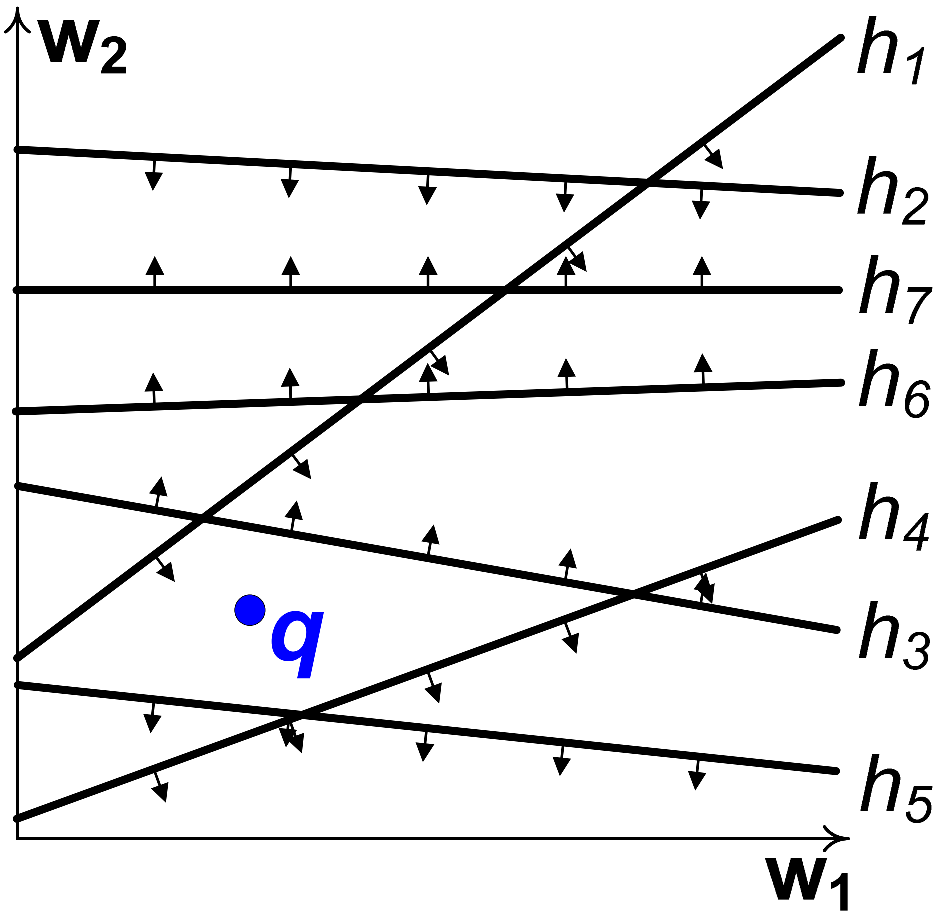
[Mouratidis15]: MaxRank

- Consider a single incomparable rec. ***r***
- Score of ***r*** higher than ***p*** iff query vector is inside a half-space
  - Inequality $S(\boldsymbol{r}) > S(\boldsymbol{p})$ maps into half-space in query space



[Mouratidis15]: MaxRank

- **Idea: map each incomp. record to a h/s**
- Recs. $\boldsymbol{r_1}$ to $\boldsymbol{r_7}$ map to h/s $\boldsymbol{h_1}$ to $\boldsymbol{h_7}$
- Consider a cell
- set of h/s including cell = set of recs. scoring higher than $\boldsymbol{p}$
- At cell of $\boldsymbol{q}$: $\boldsymbol{h_1}$ and $\boldsymbol{h_2}$ include it $\Leftrightarrow$ $\boldsymbol{r_1}$ and $\boldsymbol{r_2}$ score higher



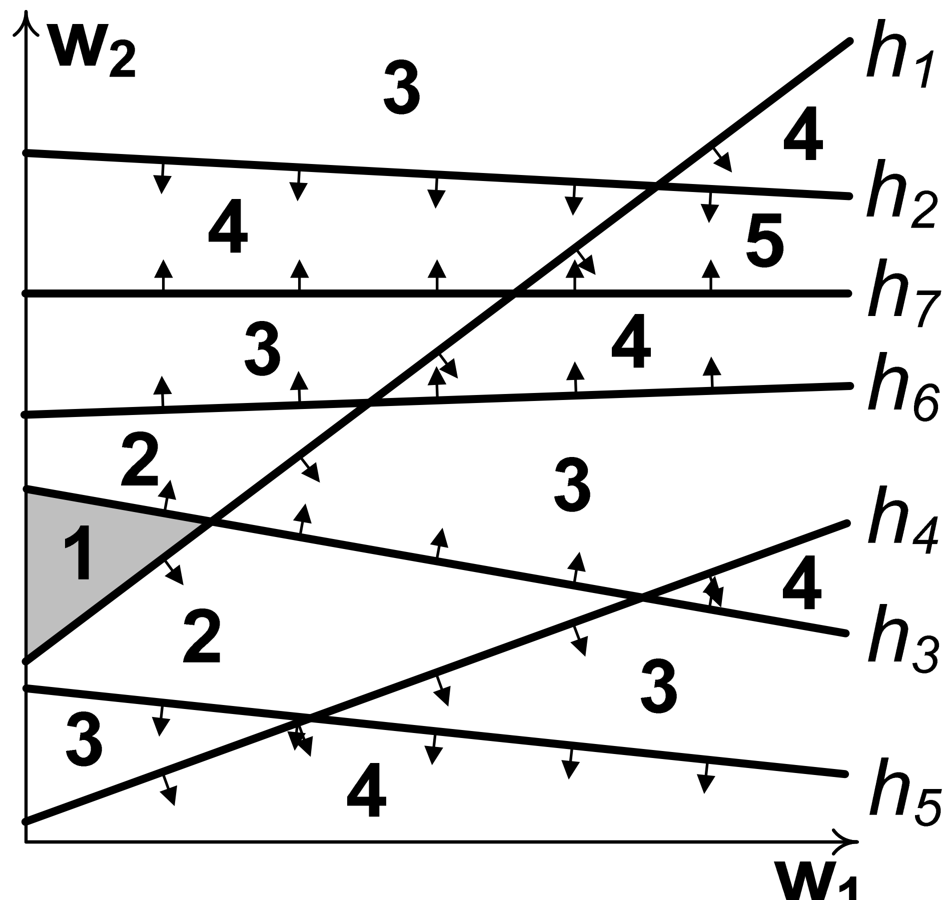
Half-space Arrangement

[Mouratidis15]: MaxRank

- Count in each cell = no. of h/s that include it

- Find the cell(s) with smallest count
  - These cell(s) = *MaxRank* regions
  - k^* = their count + no. of dominators + 1

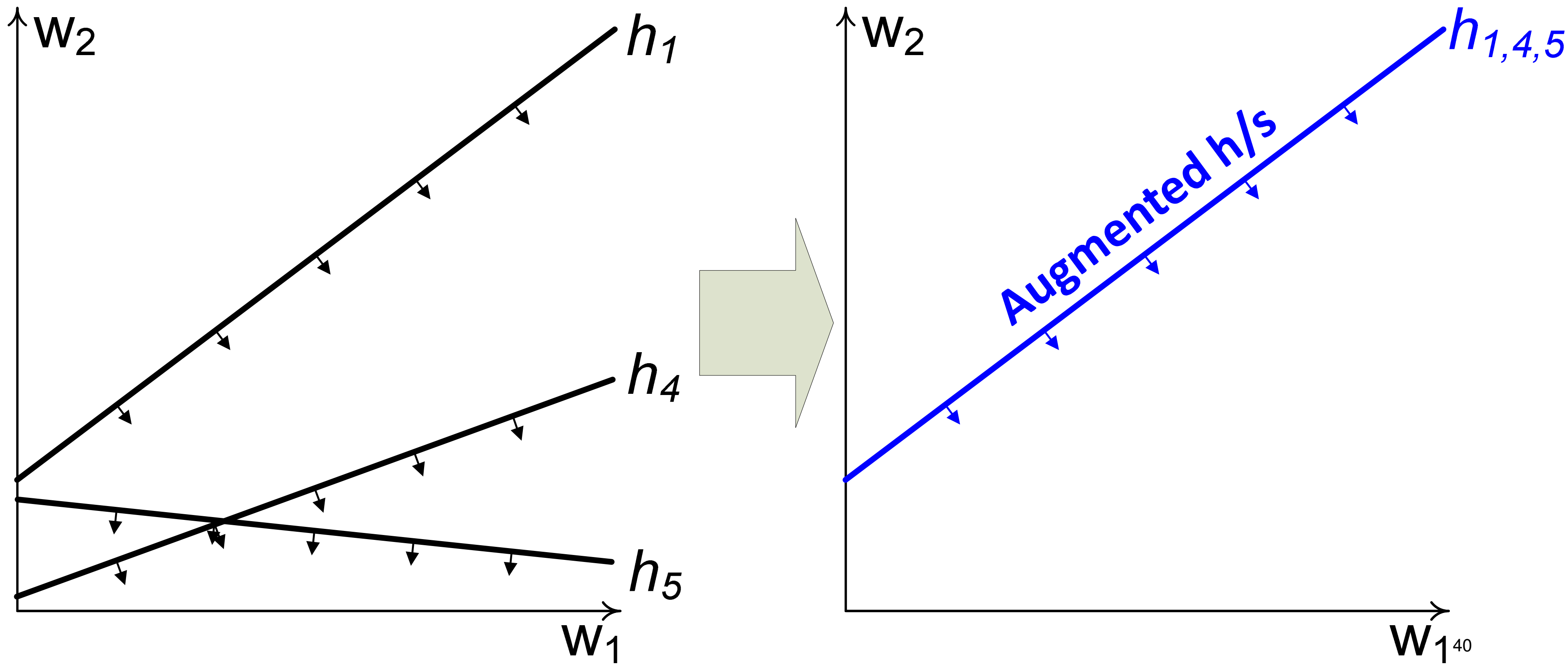
- **Trouble:** Arrangement comp. takes $O(n^d)$!!!



Half-space Arrangement

[Mouratidis15]: MaxRank

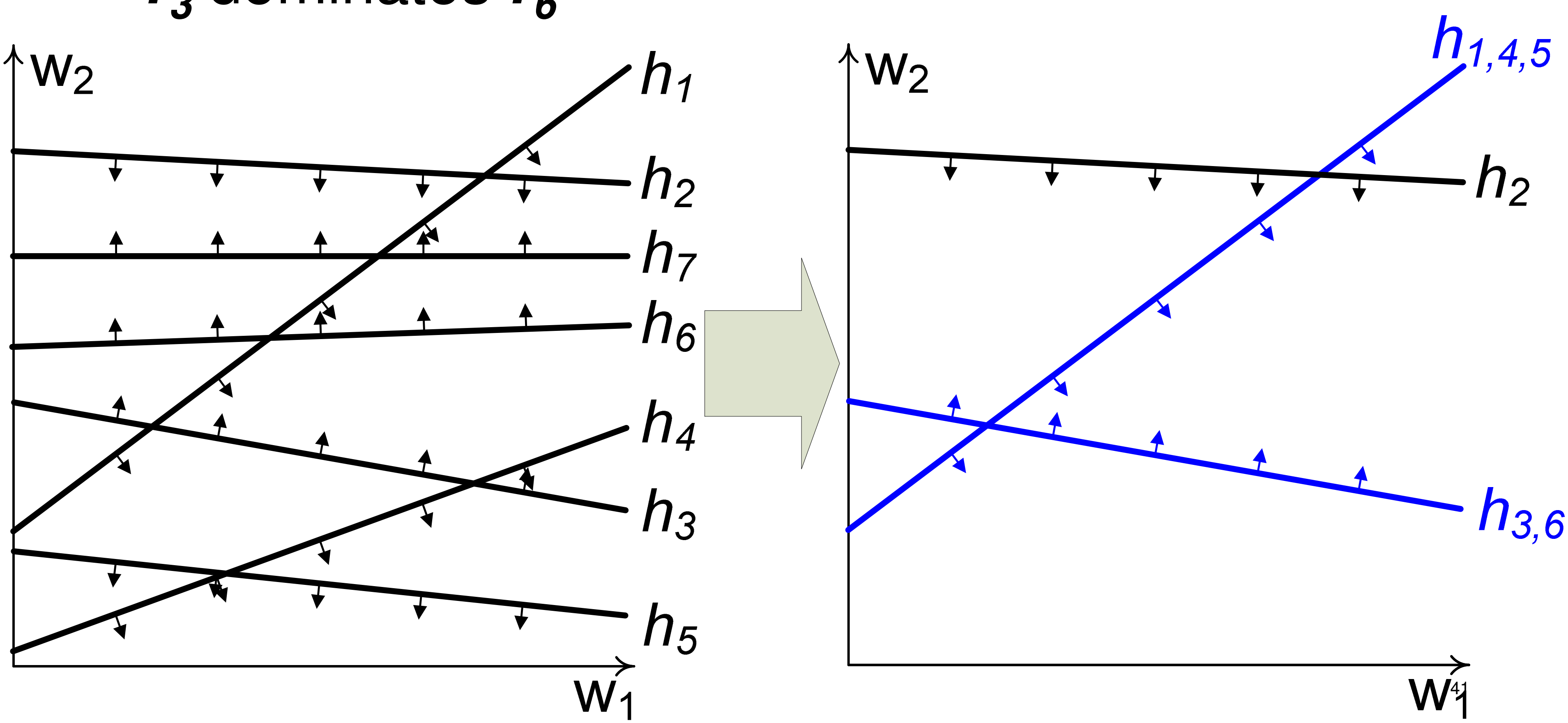
- Assume $\boldsymbol{r_1}$ dominates $\boldsymbol{r_4}$ and $\boldsymbol{r_5}$
- Subsume $\boldsymbol{h_4}$ and $\boldsymbol{h_5}$ under $\boldsymbol{h_1} \rightarrow$ augmented h/s



[Mouratidis15]: MaxRank

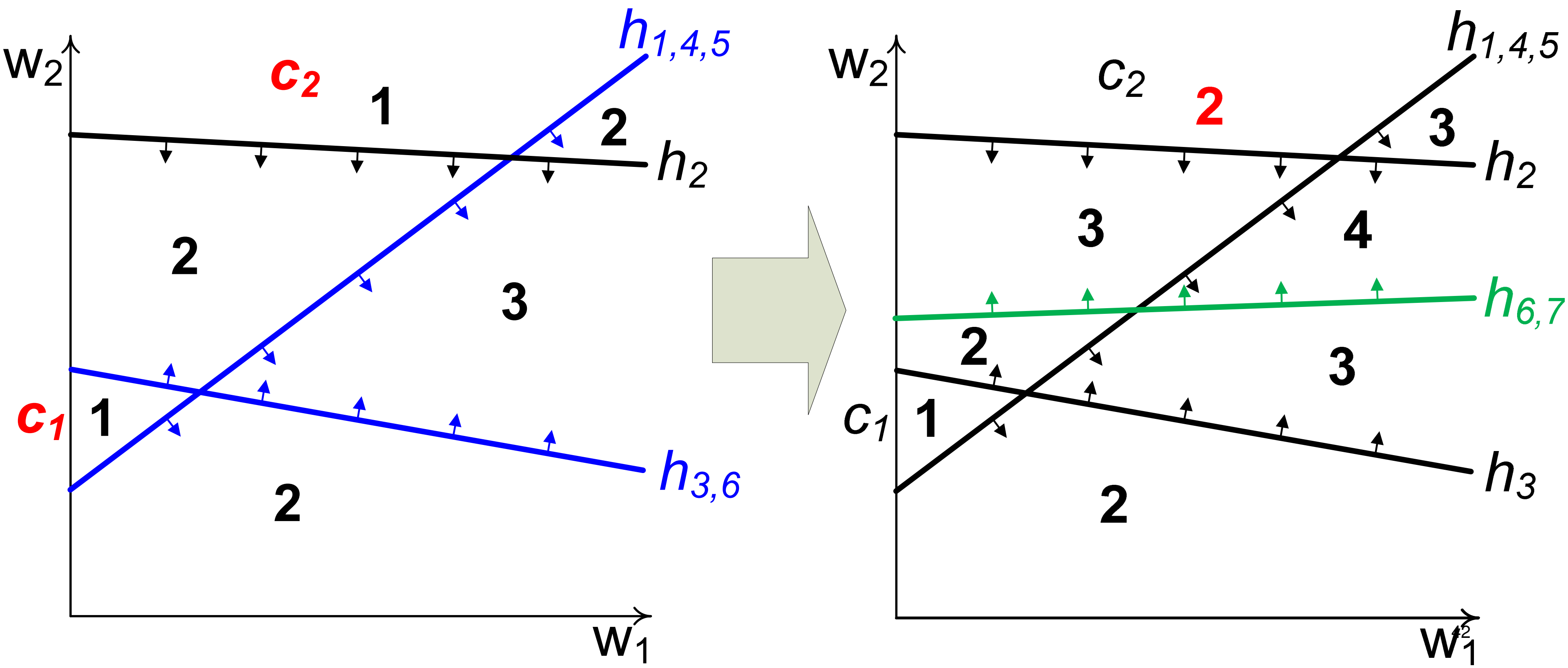
- In our example
  - r_1 dominates r_4 and r_5
  - r_3 dominates r_6

Mixed Arrangement



[Mouratidis15]: MaxRank

- Count is now a **lower bound** of the actual count if subsumed h/s were considered!
- c_1 **not in any** aug. h/s; but c_2 in $h_{3,6}$ → **expand** it!

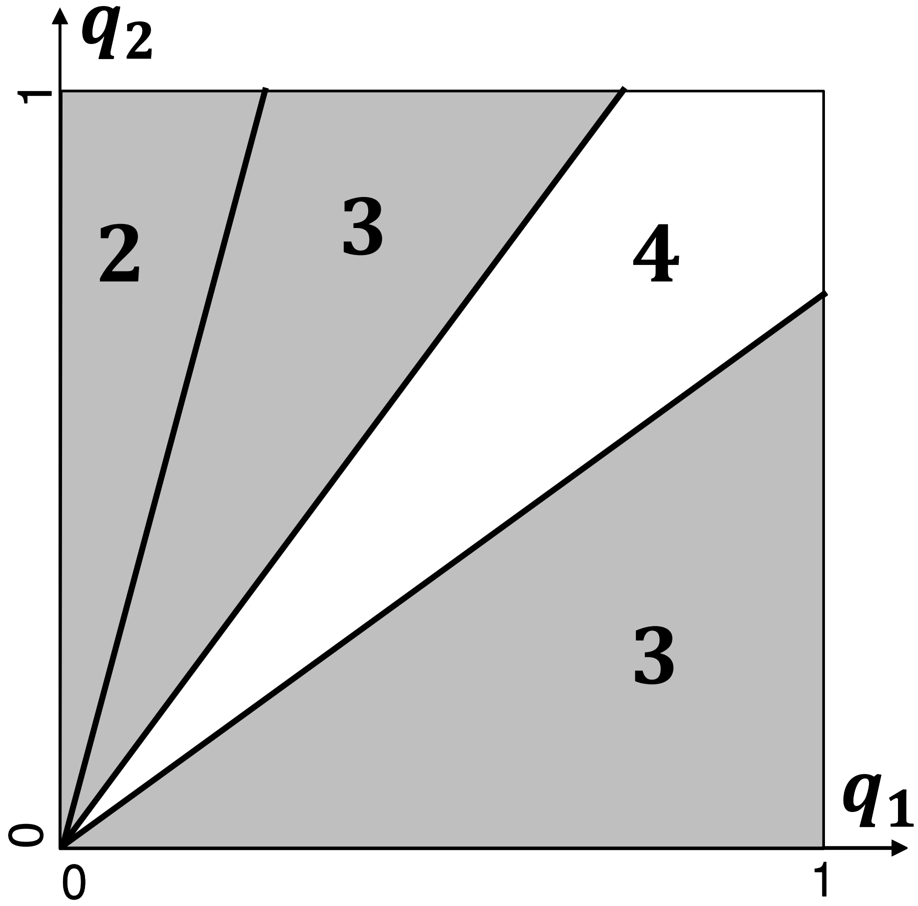


[Tang17]: k-Shortlist Preference Regions

- **k-Shortlist Preference Regions (kSPR):**
  - All regions in preference space where a given focal option ***p*** belongs to the top-k result
  - Previously defined as monochromatic reverse top-k query but only solved for the degenerate 2-D case [Vlachou10 & 11]

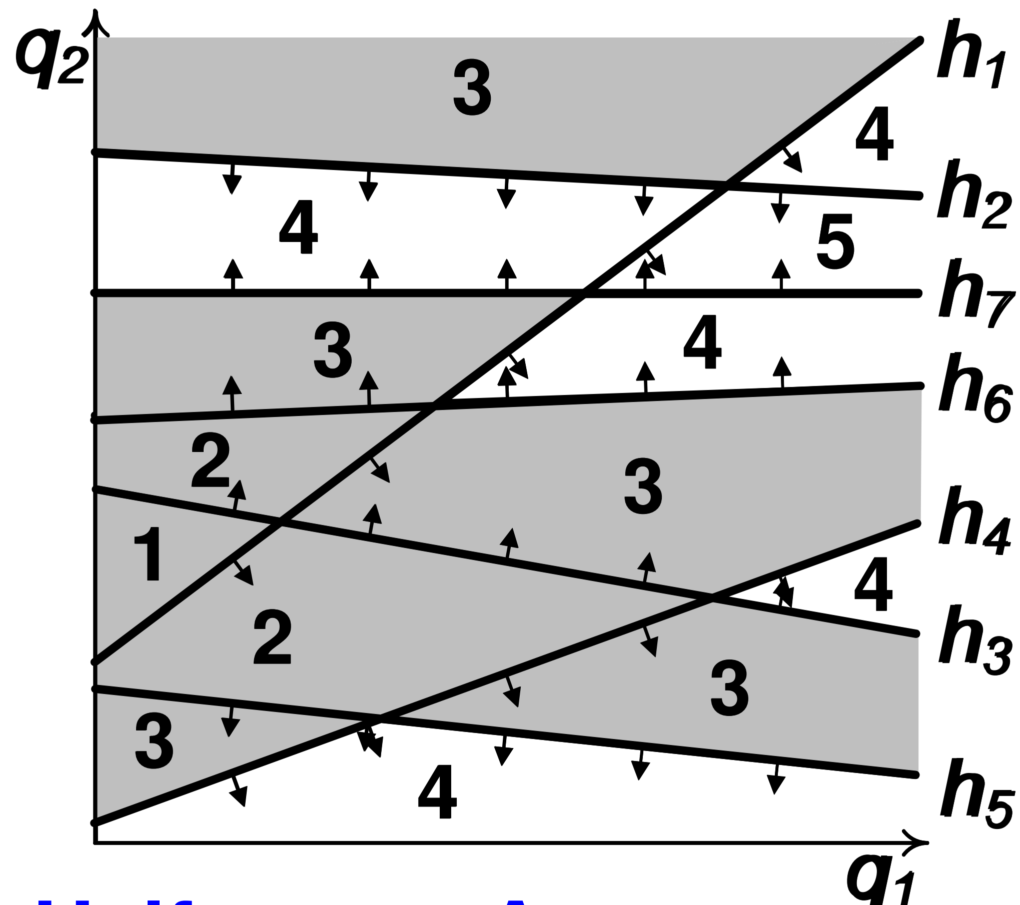
[Tang17]: kSPR Example

- Preference space
- Order of ***p***
- *kSPR* result for $k = 3$:
  - The shaded wedges
  - Every query vector in shaded area ranks ***p*** among the top-3 options



[Tang17]: Fundamentals

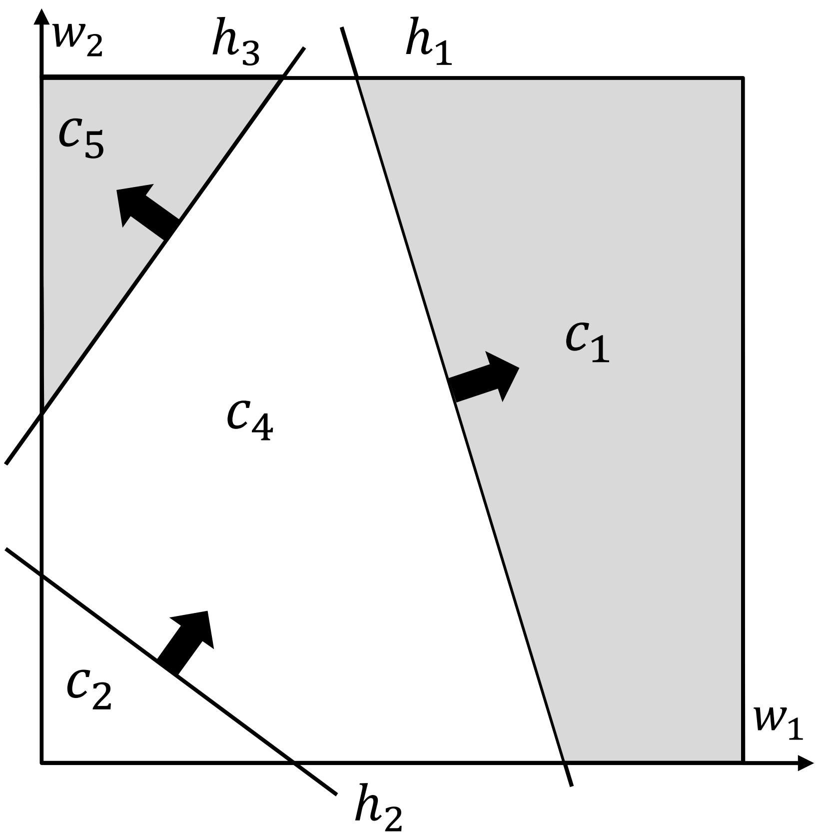
- **Again, we map each incomp. option to a h/s**
- Set of h/s including cell = set of options scoring higher than ***p***
- Count in each cell = no. of options that score higher than ***p***
- kSPR result for k=4: cells with count ≤ 3

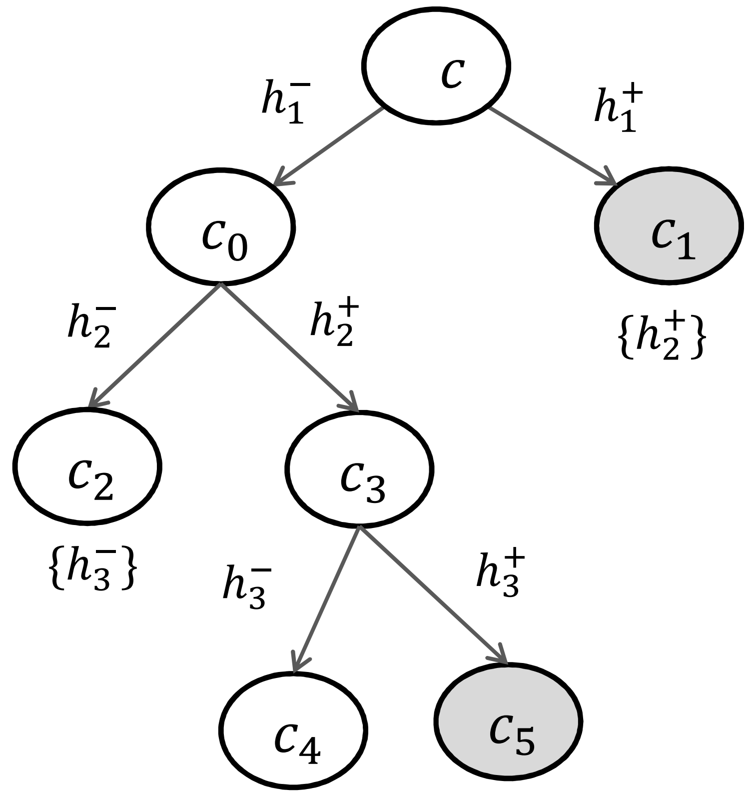


Half-space Arrangement

[Tang17]: Cell Tree (3 h/s, k = 2)

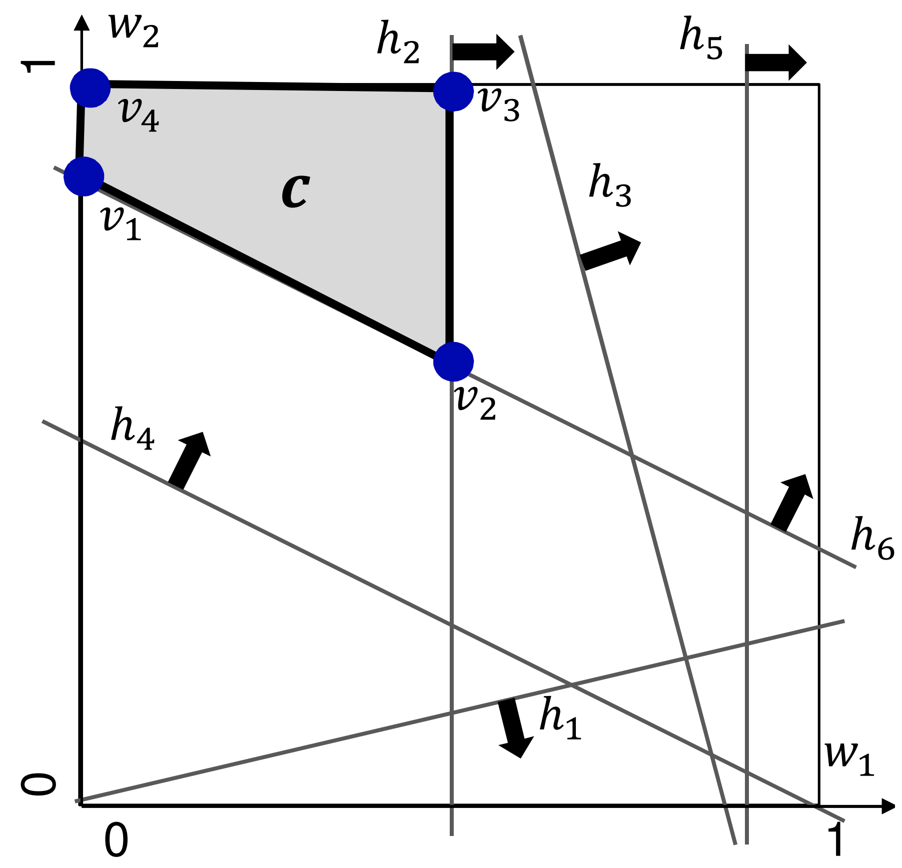
- Assume 3 h/s as shown below:
- Cell Tree looks like:





[Tang17]: Cell Representation (implicit)

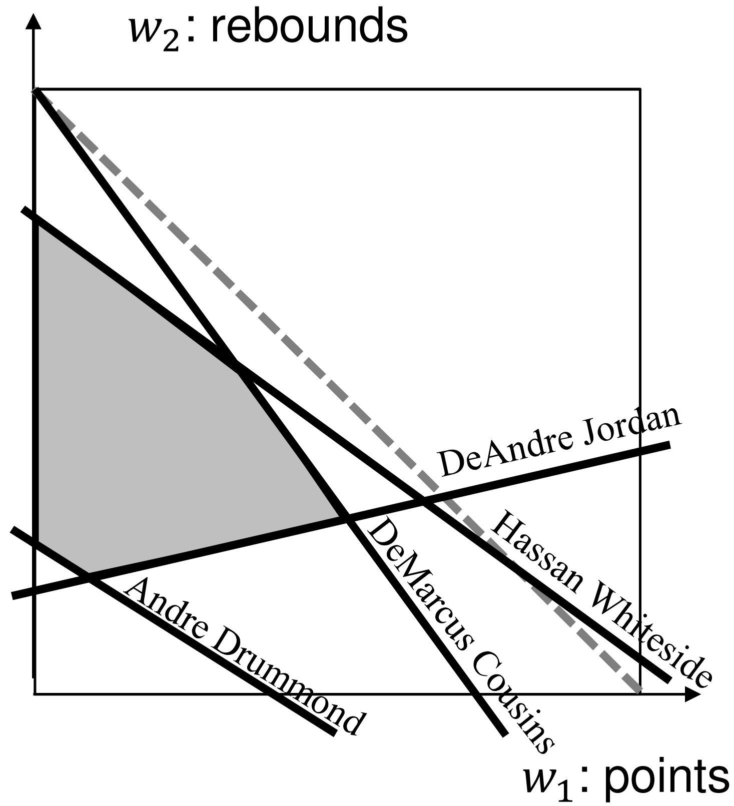
- Cell computation takes $O(n^{d/2})$
- Implicit representation by defining halfspaces: $\{h_1^-, h_2^-, h_3^-, h_4^+, h_5^-, h_6^+\}$
- …even better, just the bounding ones: $\{h_2^-, h_6^+\}$
- Trouble: how to detect infeasible cells?



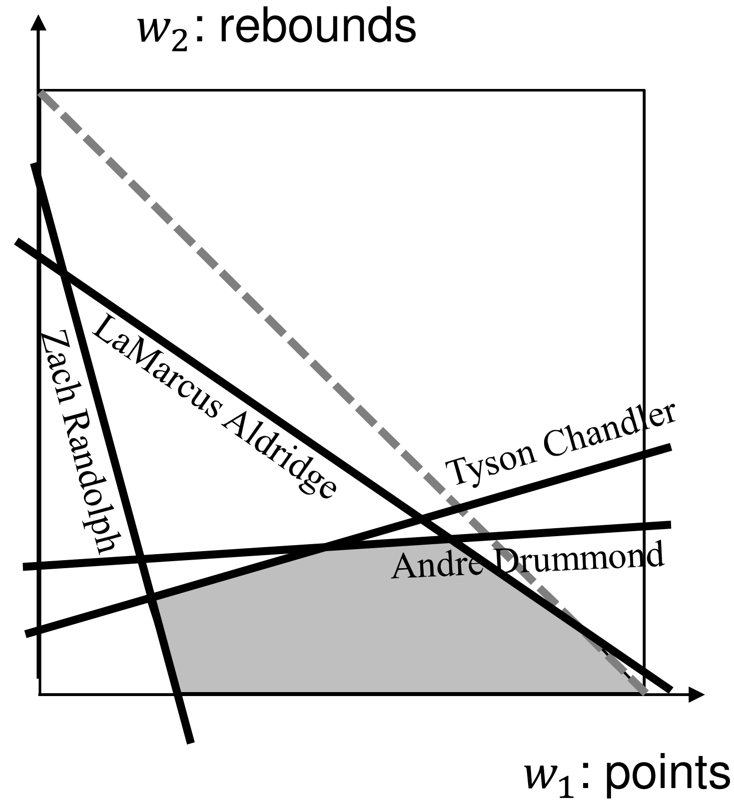
[Tang17]: Case Study

kSPR (k=3) on real NBA data for *Dwight Howard*

Season: 2014-15



Season: 2015-16



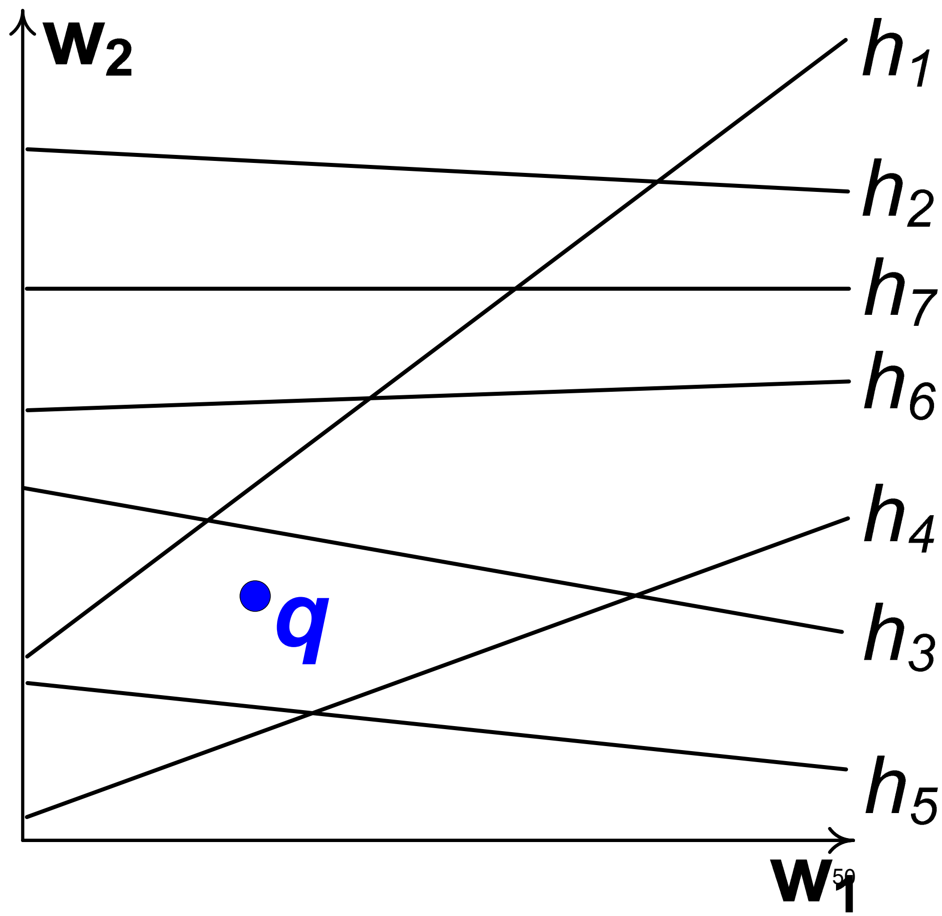
[He14]: “Why-not” query

- Given a query **q** and its top-k result
- How should we modify vector **q** and/or value k so that a record **p** is included in the result
- Defines a penalty function combining:

  (i) perturbation on **q** (Euclidean dist.) and
  (ii) increase in k
- Technique relies on **sampling** $\Rightarrow$ approximate answer
- However, there is an interesting geometric observation…

[He14]: “Why-not” query

- ∀ incomp. rec. **r** defines a hyper-plane w/ eqn. S(p) = S(r) ➔ Arrangement similar to **MaxRank**
- The optimal answer to the why-not query is proven to lie on the boundary of some cell!
- why-not **reverse** top-k query is defined in same spirit [Gao15]



[Peng15]: k-hit query

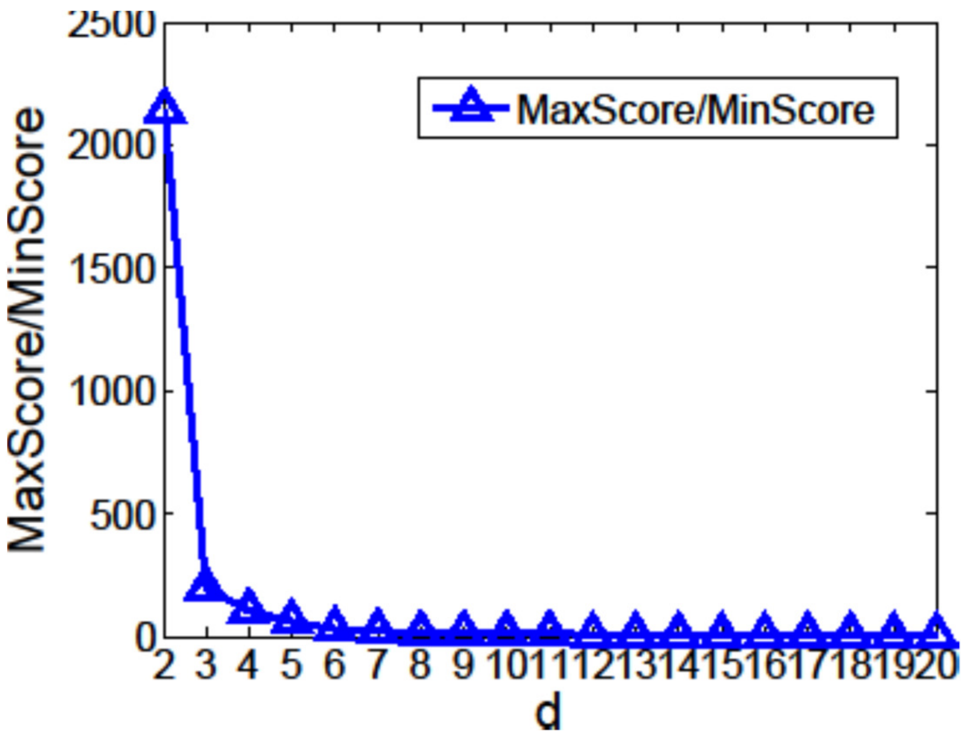
- Given: dataset + pdf of the query vector
- Select ***m*** recs. so that **top-1** rec. for a random query has highest probability to be among them
- Result belongs to the **convex hull**
- Computing probabilities = computing areas of cones (or wedges, in 2d), which is expensive.
- Thus **sampling ➔** approx. solutions w/ bounds
- **k-regret min. set** e.g. [Chester15]: subset of ***m*** recs s.t. top-1 rec. in subset scores the closest to the top-k^{th} rec. for any possible query

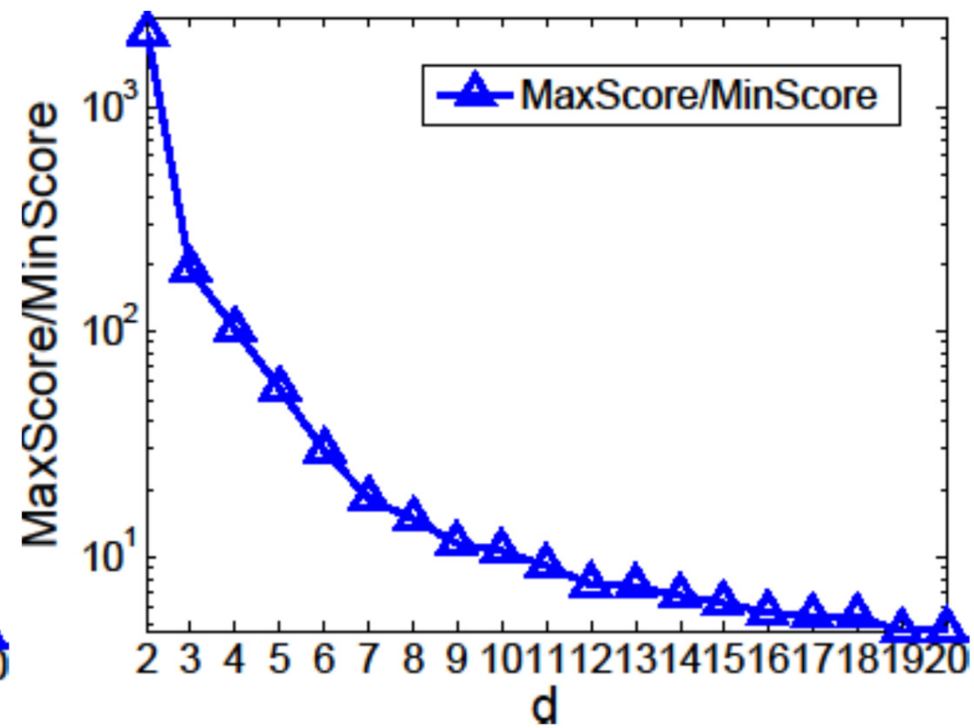
Top-k in High-D?

- Unless the data exhibit strong correlation, top-k is meaningless in more than 5-6 dimensions!
- As d grows, the **highest score** across the dataset approaches the **lowest score**!
- I.e. ranking by score no longer offers distinguishability $\leftrightarrow$ looses its usefulness
- Behaviour very similar to nearest neighbor query, known to suffer from the dimensionality curse [Beyer99]

Top-k in High-D?

- IND data
- …of fixed cardinality n = 100K
- …we vary data dimensionality





Thank you!