School of
Information Systems

SMU
SINGAPORE MANAGEMENT UNIVERSITY

Geometric Top-k Processing: Updates since MDM'16

[Advanced Seminar]

Kyriakos Mouratidis

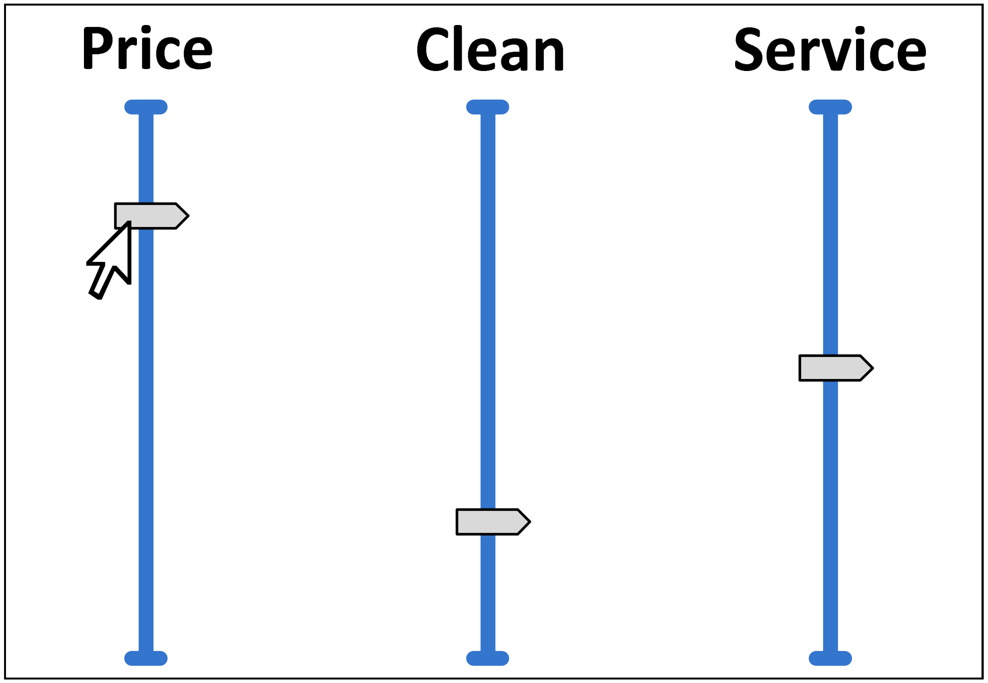
Singapore Management University

MDM 2019

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:

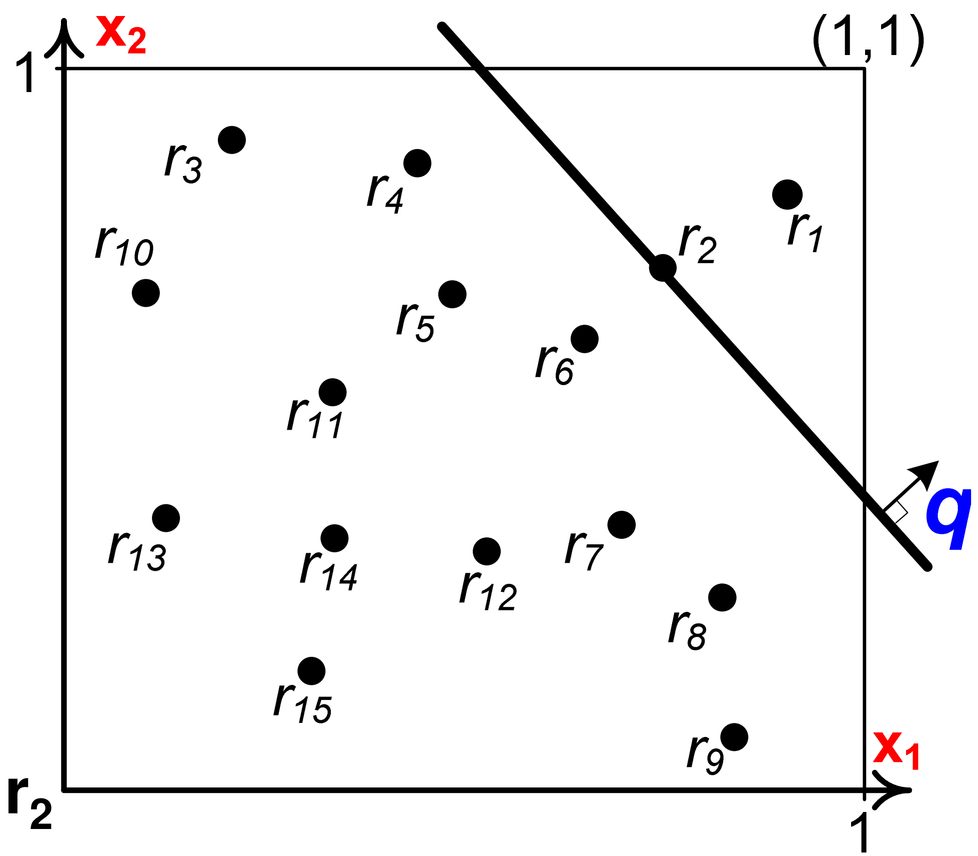


Introduction

- Slide-bar locations → **numerical weights**
- We call **q** = <0.8, 0.3, 0.5> the ***query vector***
  - and its domain **query space** or **preference space**
- Linear function ranks hotels (i.e. **options**)
  - *score* = **0.8**·*price* + **0.3**·*clean* + **0.5**·*service*
  - if option **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 **option 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 an option is met ↔ order in ranking!
  - E.g. top-2 = { r_1, r_2 }
- At current position: ∀ option above (below) the line, higher (lower) score than $\mathbf{r_2}$

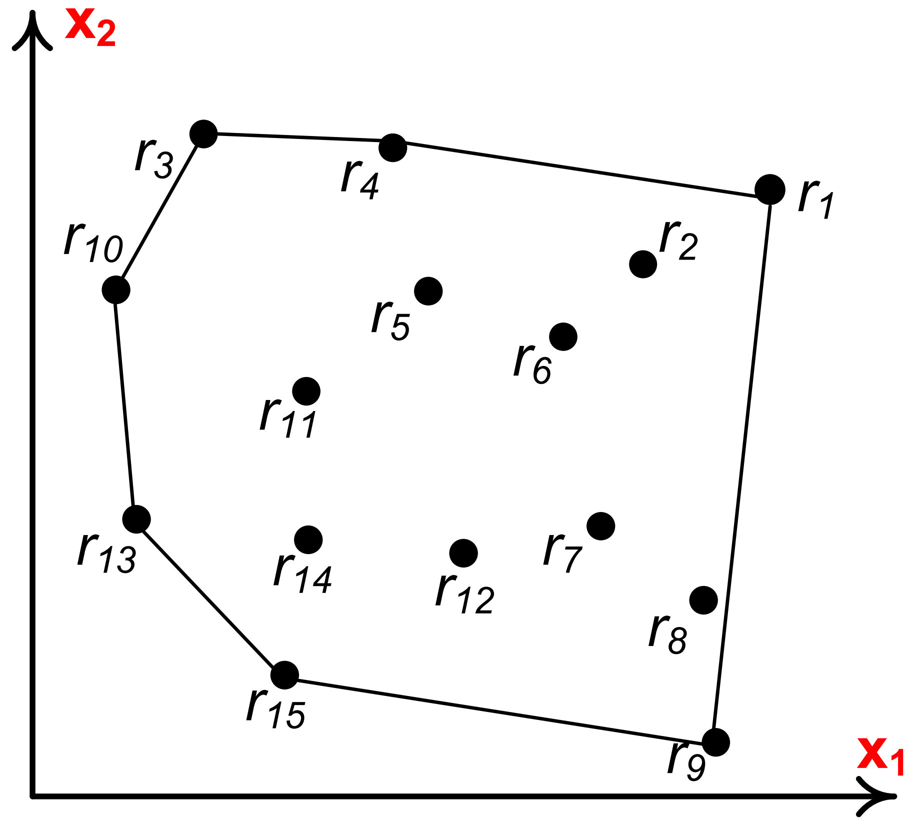


Notes on dim/nality of query domain

- Ranking of 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

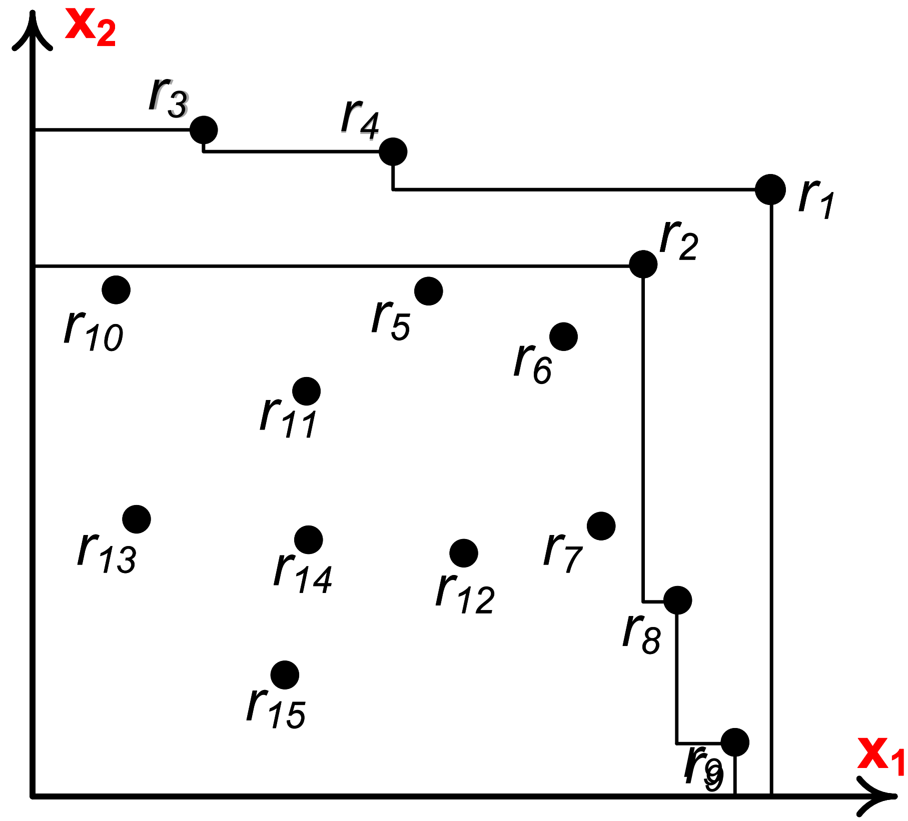
Relationship to Convex Hull

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



[Börzsönyi01, Papadias03]: Skyline

- **Dominance:** option $\mathbf{r_1}$ 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 opts. that aren't dominated
- Includes top-1 $\forall$ **q**
- **k-skyband**: all opts. not dominated by k or more others
- Includes top-k $\forall$ **q**

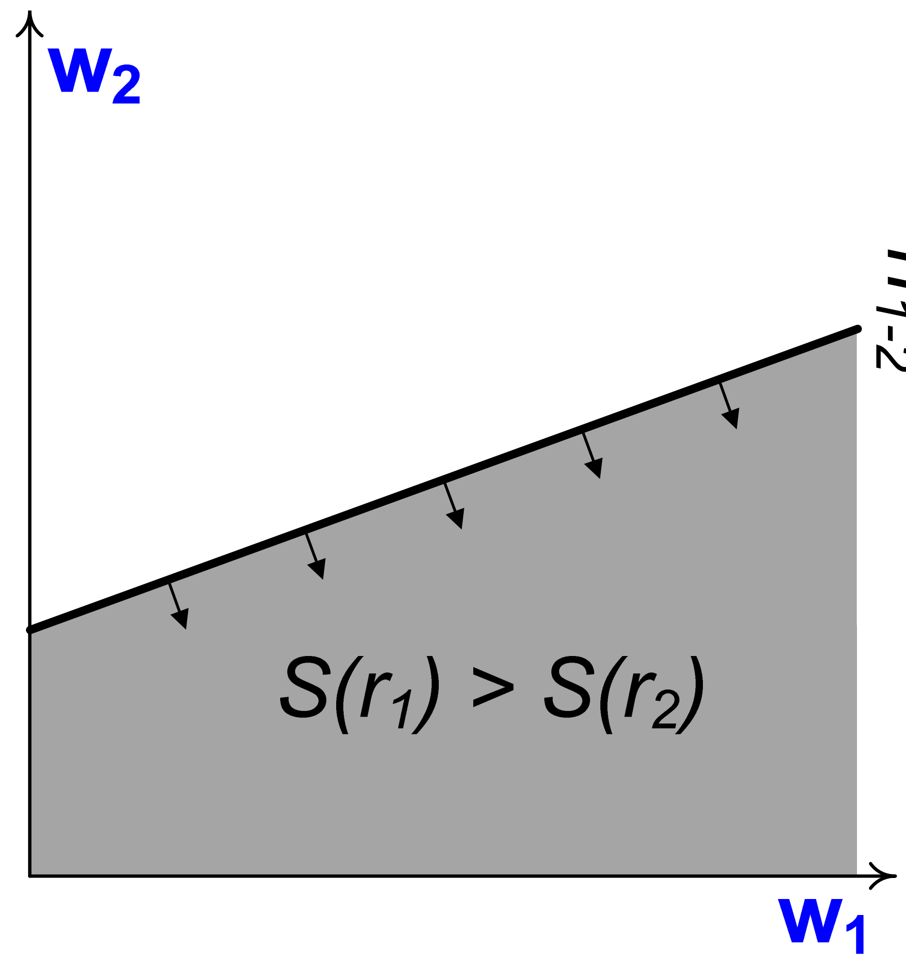


[Zhang14]: Global Immutable Region

- **Global Immutable Region (GIR)**
  - The **maximal** region around query vector ***q*** where the top-*k* result remains the same
- **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 $\leftrightarrow$ a half-space!

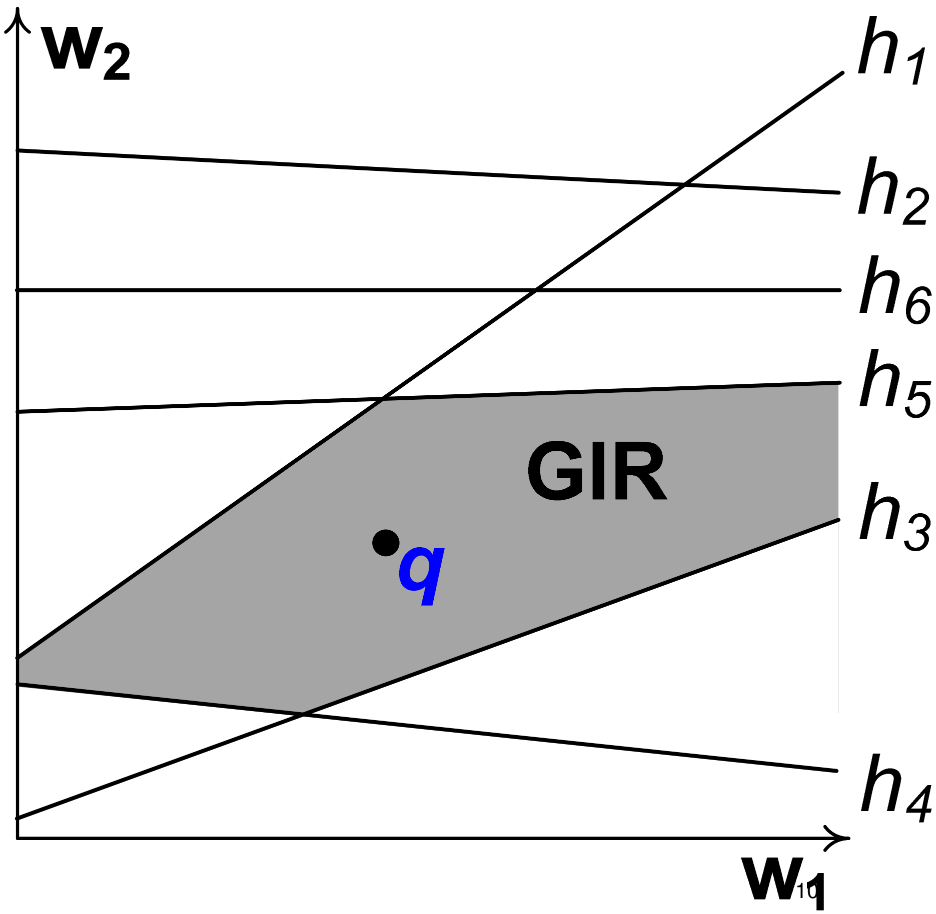
[Zhang14]: Global Immutable Region

- Each condition ↔ a **half-space**!
- Intersect all half-spaces
- Cost: $O(n^{d/2})$
- Problem: Too expensive
- Idea: limit no. of NR-conditions!



[Zhang14]: Global Immutable Region

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



[Asudeh18]: Result stability

- Given a total ranking of the dataset w.r.t. **q**
- They use GIR volume as a measure of stability

- Allowing **q** to move in a region R in pref. space
- They report total rankings in decreasing stability order (i.e., decreasing GIR volume)
- Their approach relies on sampling (i.e., is approximate) with a probabilistic accuracy analysis

[Mouratidis15]: MaxRank

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

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

  2. All the regions in the preference space where that rank is attained

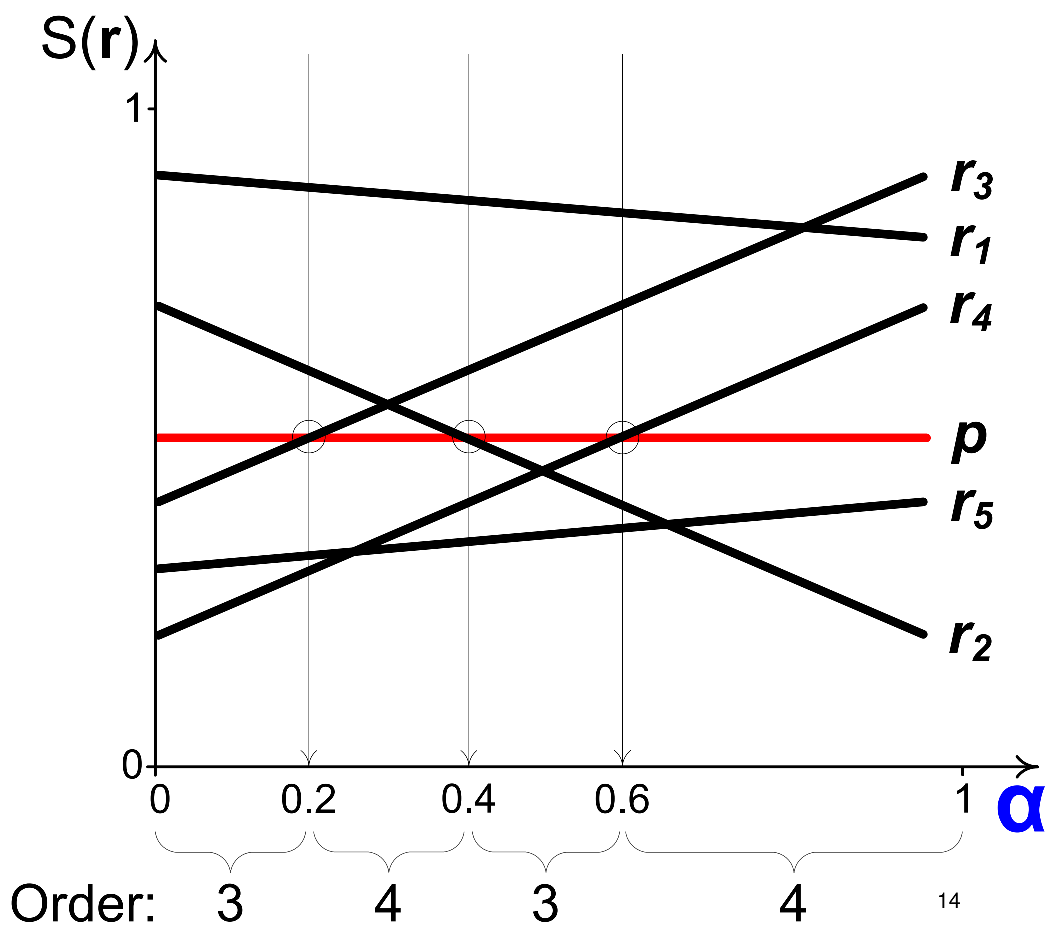
[Vlachou10 & 11]: Reverse top-k query

- **Bichromatic** (main focus): Given a focal option **p**, a set of options, **and a set of top-k queries**, identify the queries that have **p** in their result
  - Algebraic bounds based on MBRs

- **Monochromatic**:
  Given a focal option **p** and a set of options, find **all regions in pref. space** where **p** is in the top-k result
  - Solution only for 2-D

[Vlachou10 & 11]: Reverse top-k query

- Monochromatic RTOP-k in 2-D
- $S(r) = \alpha \cdot x_1 + (1-\alpha) \cdot x_2$
- Every intersection of scoreline of **p** ↔ reordering
- Plane sweep algo.

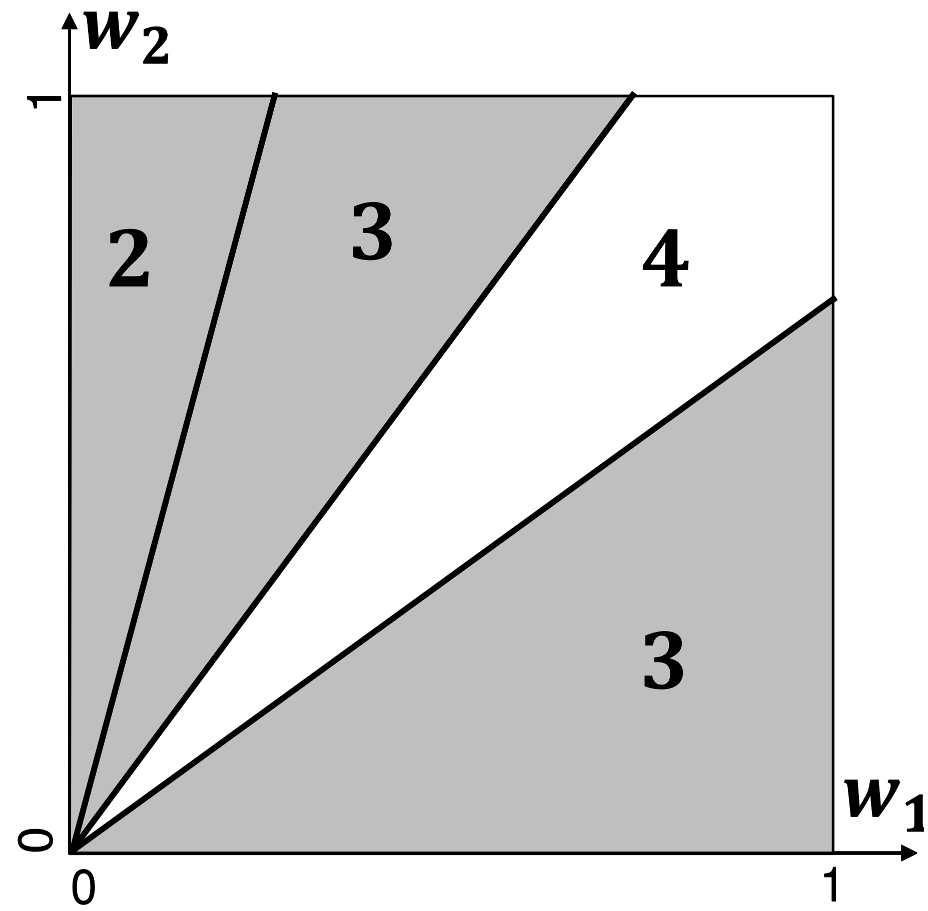


[Tang17]: k-Shortlist Preference Regions

- Monochromatic RTOP-k **for d ≥ 2**
- **aka: k-Shortlist Preference Regions (kSPR):**
  – All regions in preference space where a given focal option **p** belongs to the top-k result

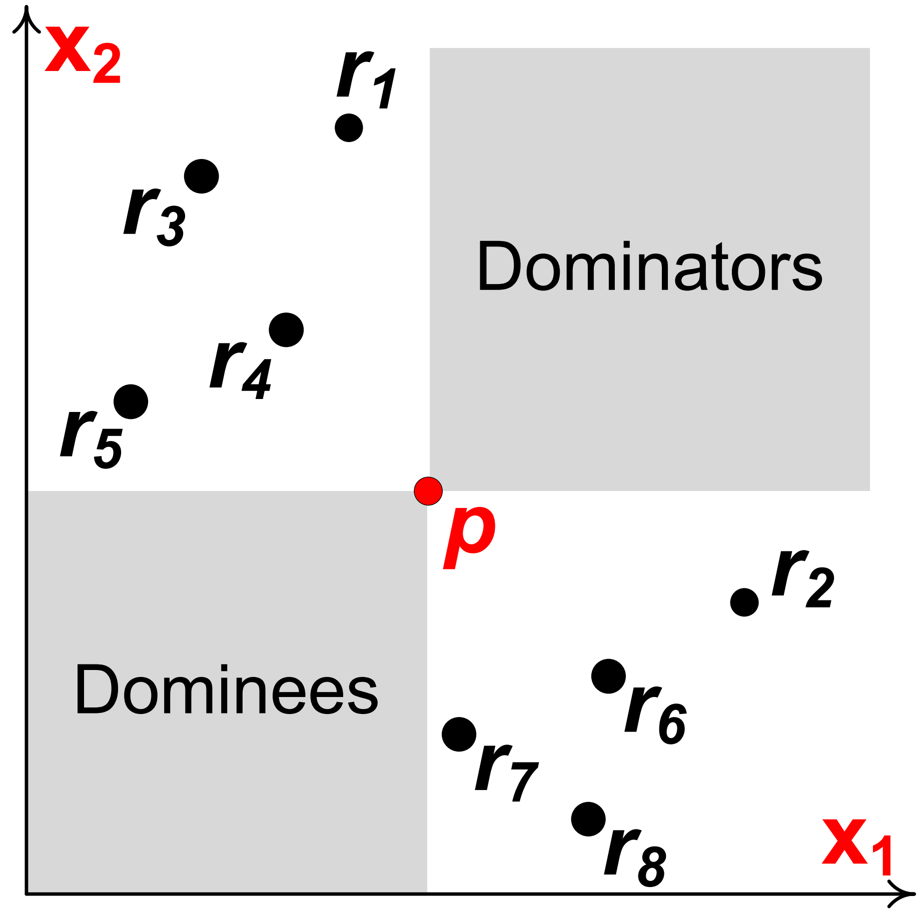
[Tang17]: kSPR Example

- 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]: Fast pruning

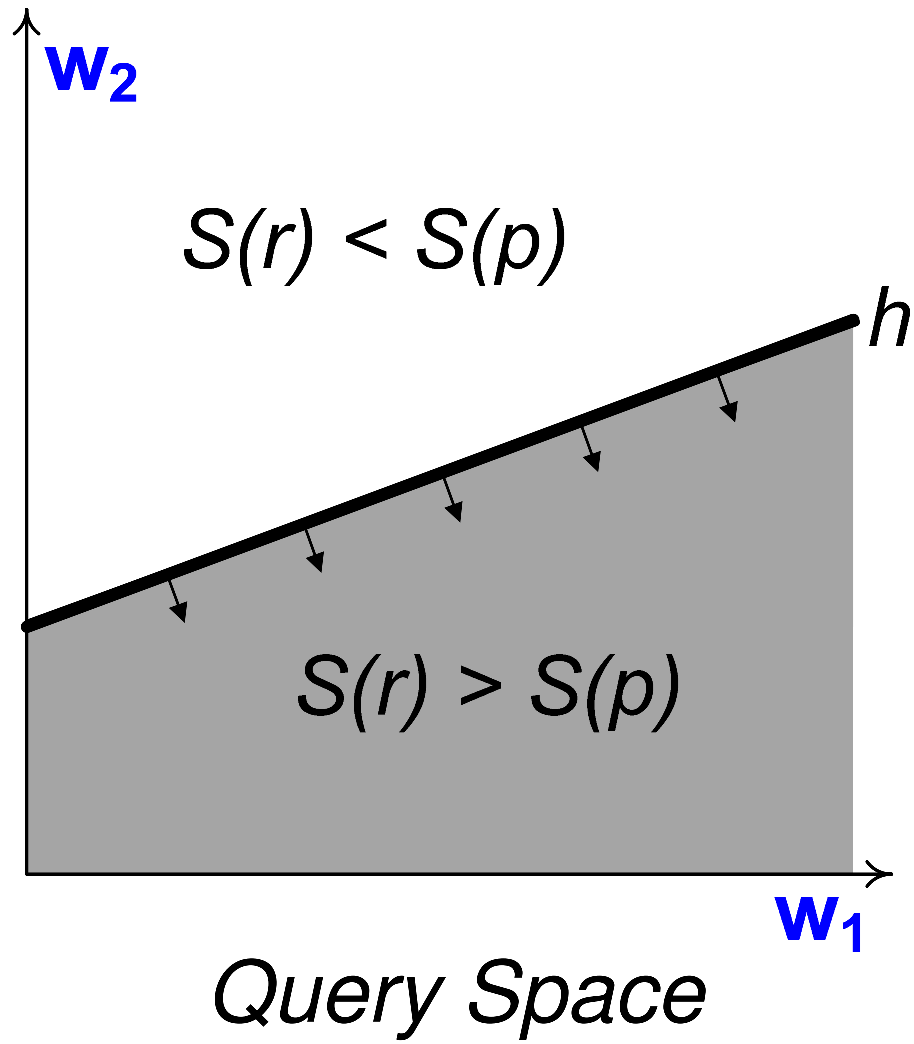
- *Dominees*
  - – ignore
- *Dominators*
  - – simply increment *k**
- *Incomparable*
  - – How to deal with them?



Data Space

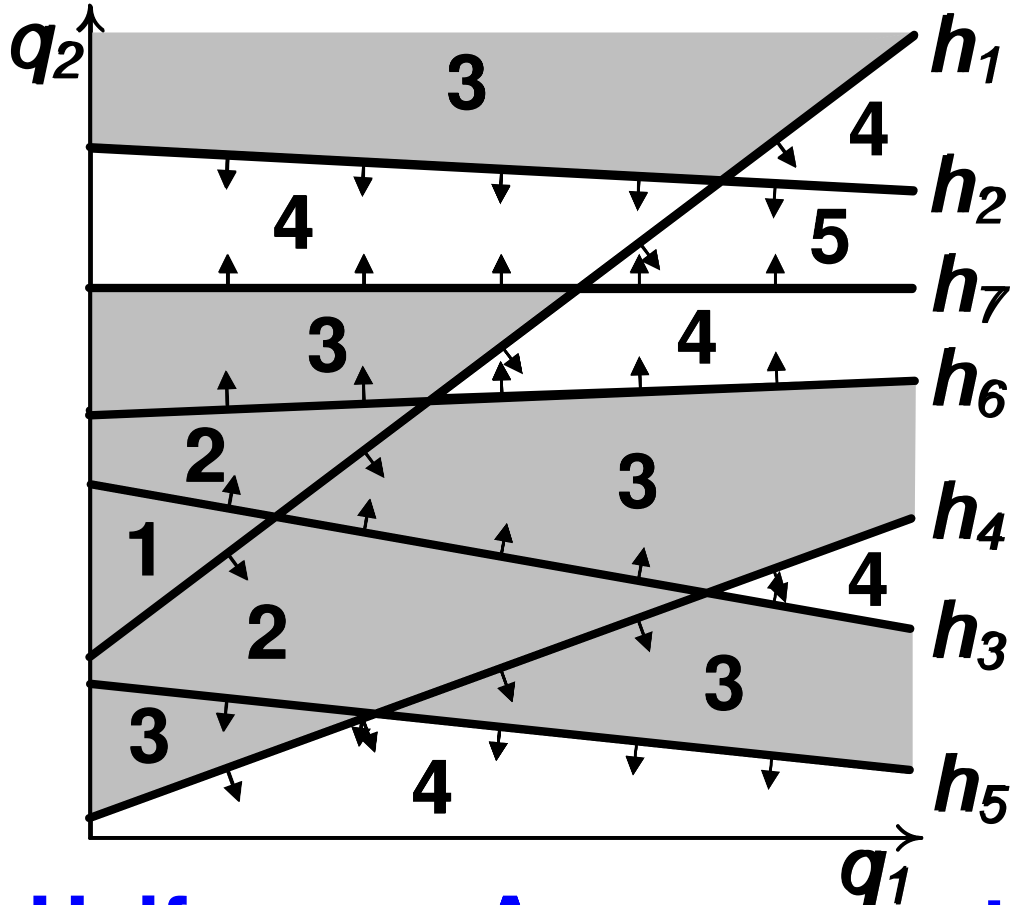
[Tang17]: kSPR

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



[Tang17]: Fundamentals

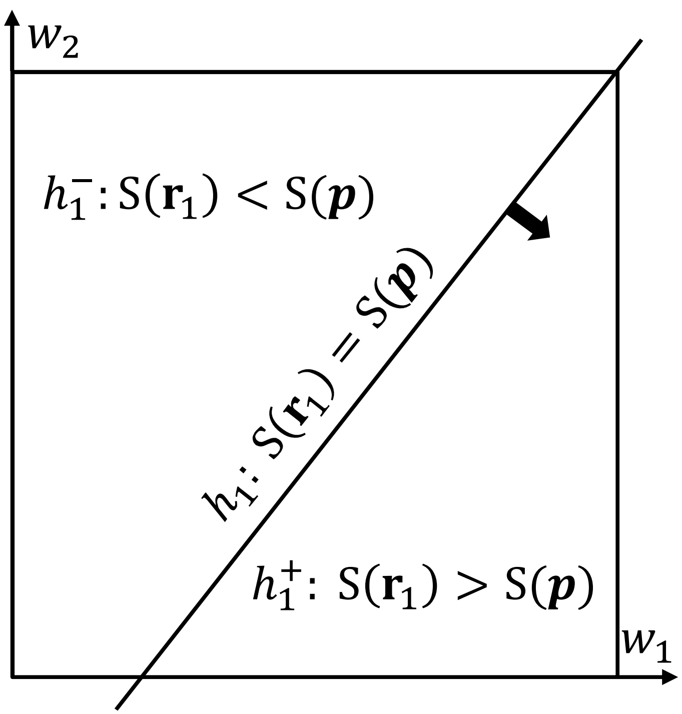
- **Idea: map each incomp. option to a h/s**
- Set of h/s including cell = set of options scoring higher than **p**
- Count in each cell = no. of options that score higher than **p**
- kSPR result for k=4: cells with count ≤ 3

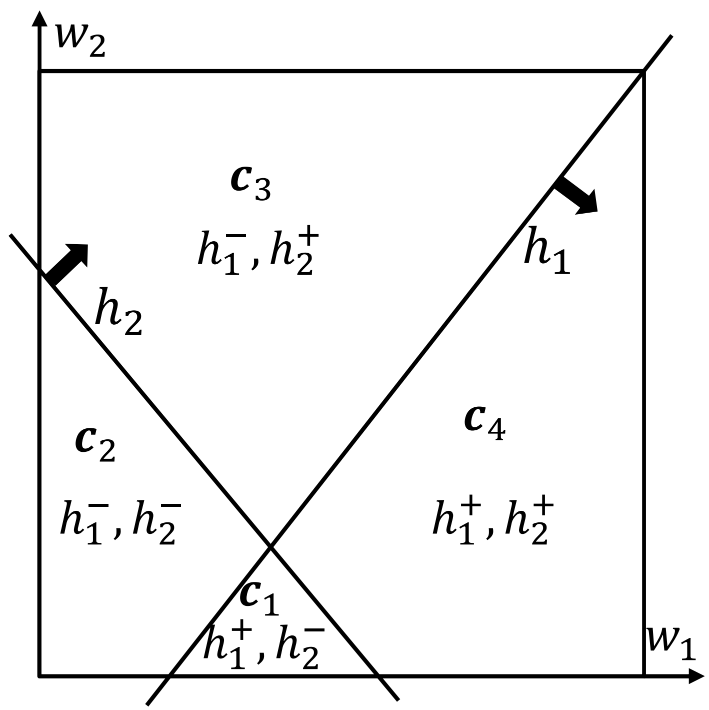


Half-space Arrangement

[Tang17]: Cell Tree

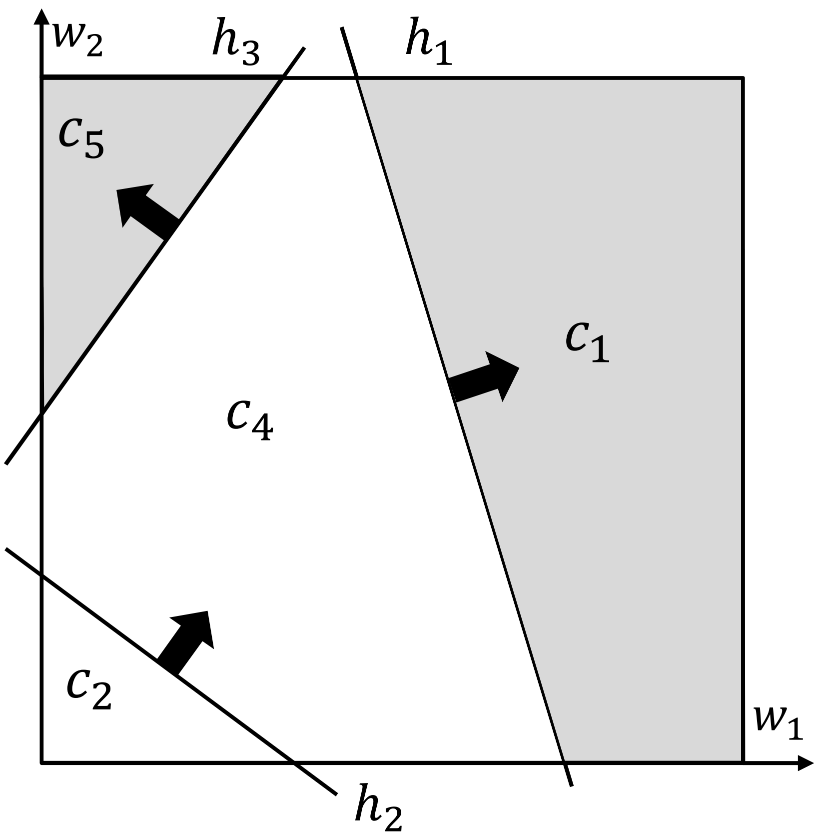
- Insert h/s one by one into a binary tree to maintain the arrangement
- Insertion of h_1 (root split into 2 leaves)
- Insertion of h_2 (each leaf split into two)

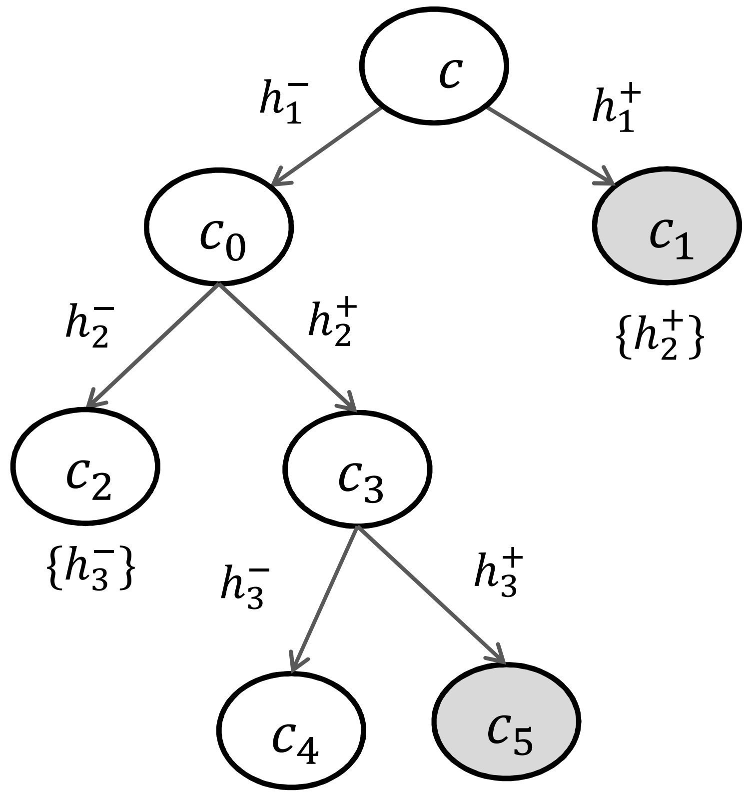




[Tang17]: Cell Tree (3 h/s, k = 2)

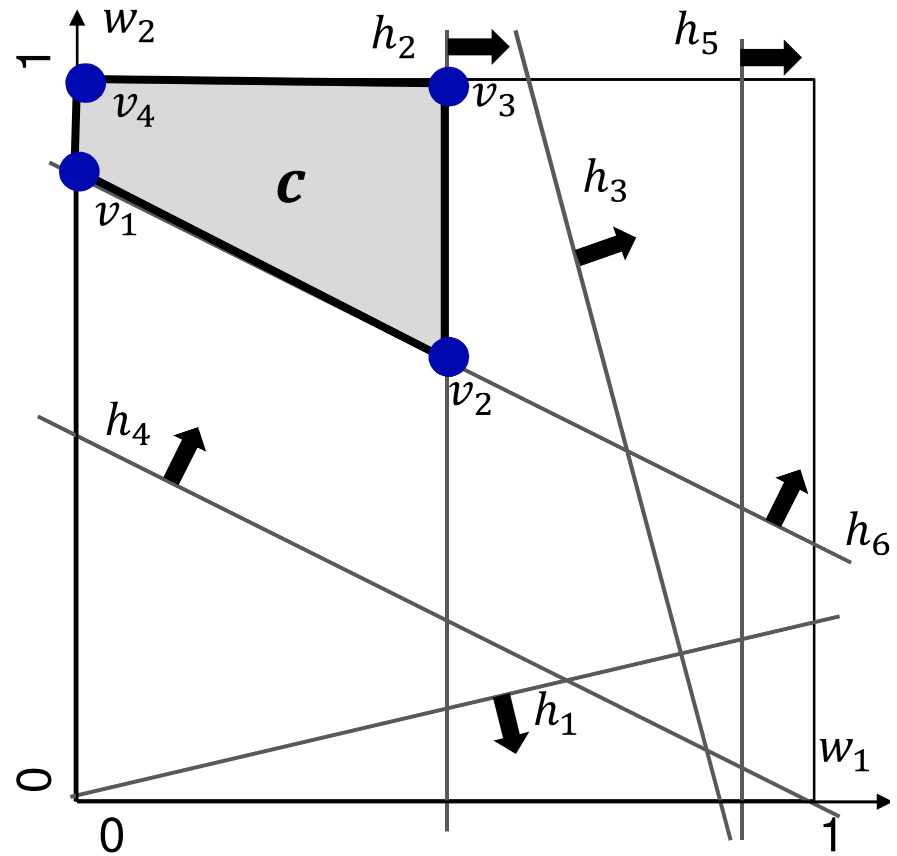
- Assume 3 h/s as shown below:
- Cell Tree looks like:





[Tang17]: Cell Representation (implicit)

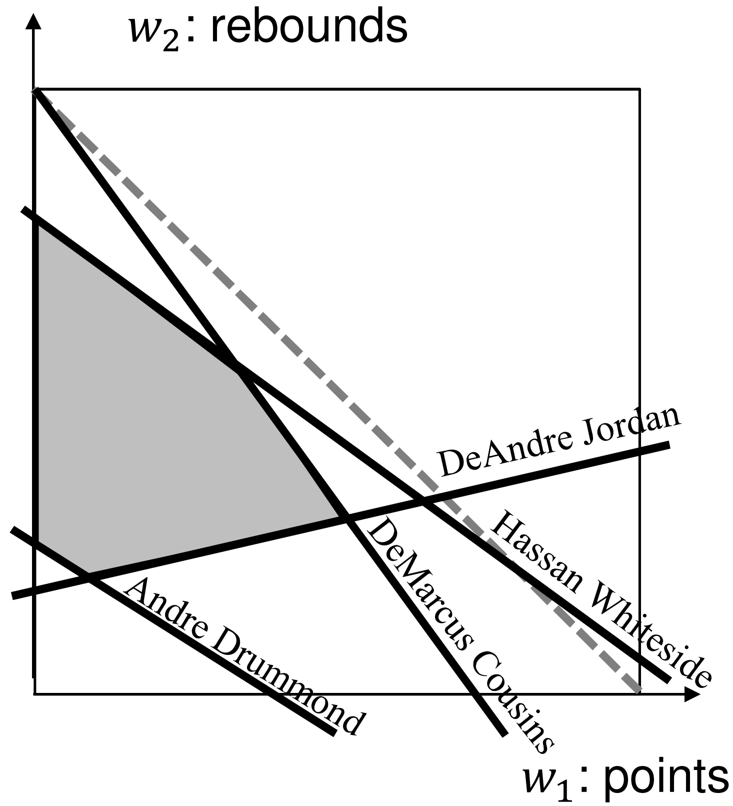
- Cell computation takes $O(n^{d/2})$
- Implicit representation by defining halfspaces: $\{\boldsymbol{h_1^-},\boldsymbol{h_2^-},\boldsymbol{h_3^-},\boldsymbol{h_4^+},\boldsymbol{h_5^-},\boldsymbol{h_6^+}\}$
- …even better, just the bounding ones: $\{\boldsymbol{h_2^-},\boldsymbol{h_6^+}\}$
- Trouble: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



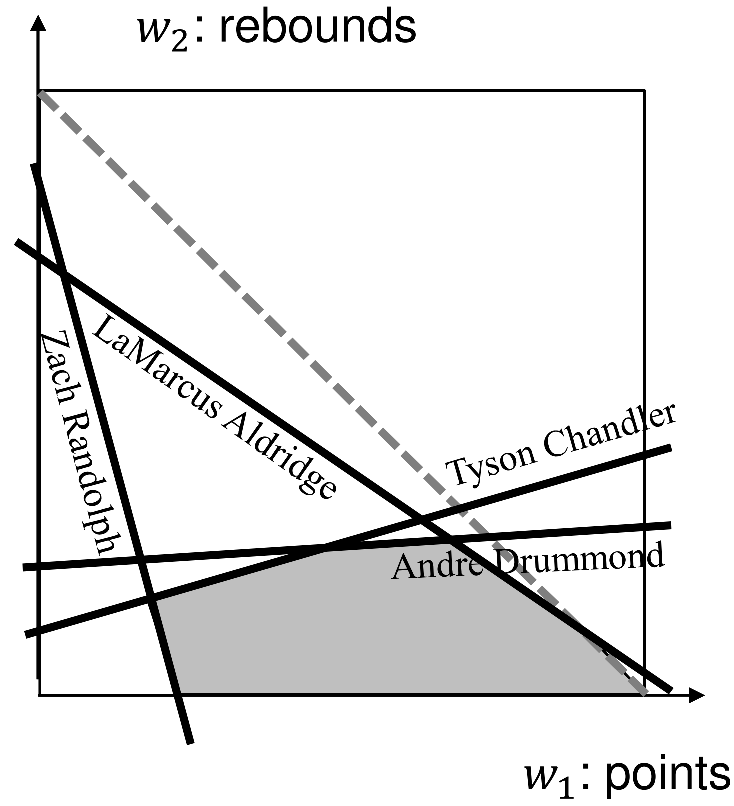
[Tang17]: Case Study

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

Season: 2014-15



Season: 2015-16



Uncertain Preferences

- Literature assumes **q** is given and exact, but…
- …whether manually input or mined, it could only be taken as a mere indication
- If only approximate prefs., instead of exact **q**, use a region *R* in pref. space to allow for inaccuracies
- [Ciaccia&Martinenghi17]:
  identify all possible **top-1** options ($k = 1$)
- [Mouratidis&Tang18]:
  identify all possible **top-k** options ($k \geq 1$)

[Mouratidis&Tang18]: Uncertain Top-k

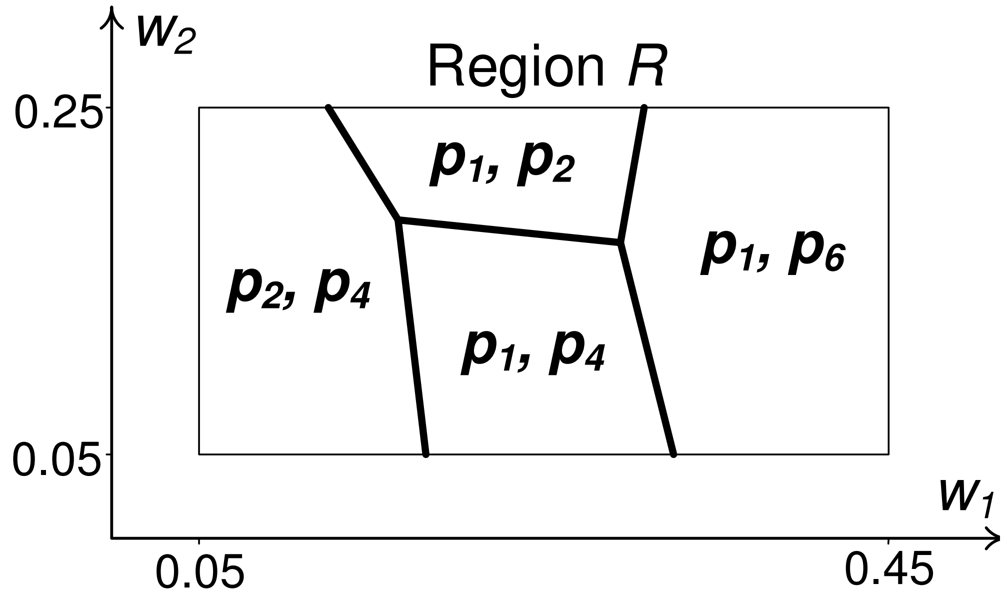
- Given:
  approx. preferences $\leftrightarrow$ region *R* in pref. space

- **UTK$_1$**: report all options that may be among the top-k when $\mathbf{q} \in R$

- **UTK$_2$**: report specific top-k set for any $\mathbf{q} \in R$

UTK: Example

Hotel	Svc.	Cln.	Loc.
p_1	8.3	9.1	7.2
p_2	2.4	9.6	8.6
p_3	5.4	1.6	4.1
p_4	2.6	6.9	9.4
p_5	7.3	3.1	2.4
p_6	7.9	6.4	6.6
p_7	8.6	7.1	4.3

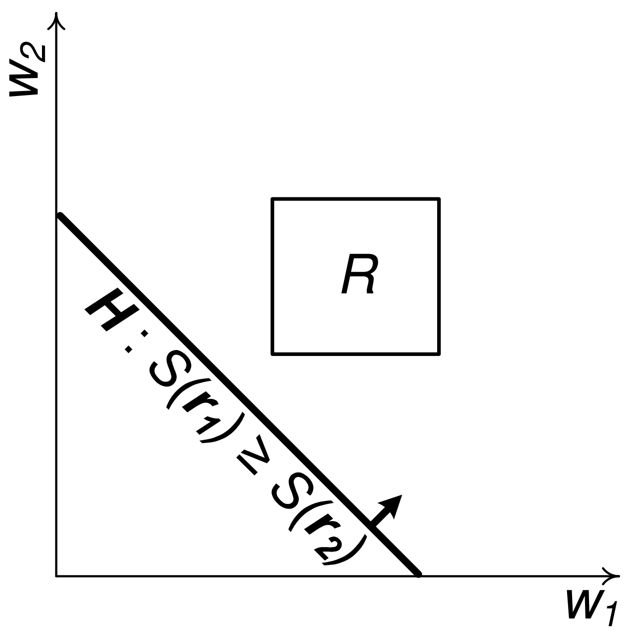


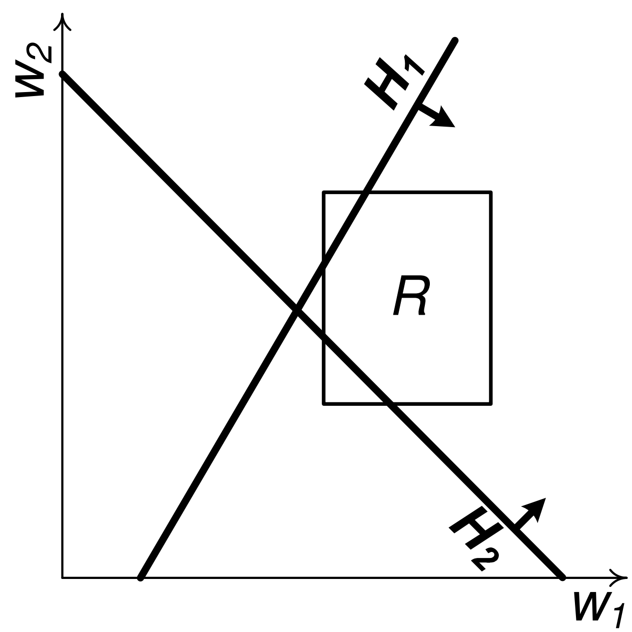
Dataset

UTK output for $k = 2$ (in preference space)

r-dominance; r-skyband

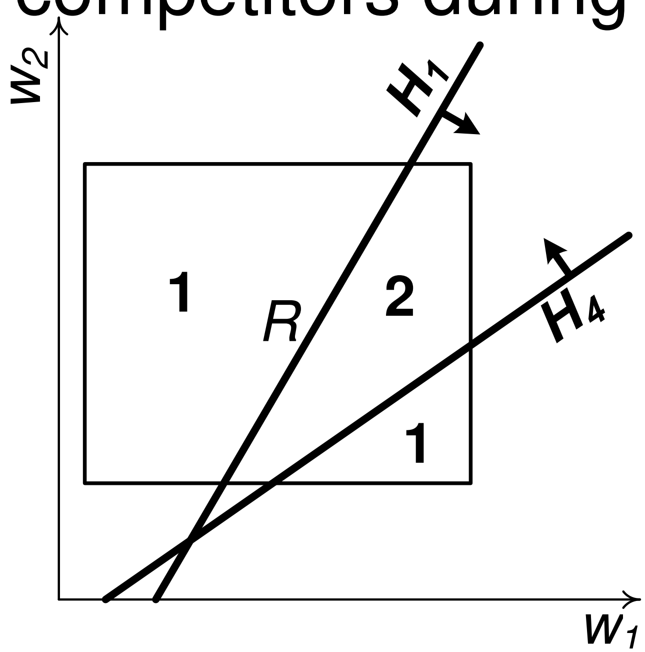
- Consider options $\mathbf{r_1}$ and $\mathbf{r_2}$
- $\forall \mathbf{q}$ in R, $S(\mathbf{r_1}) > S(\mathbf{r_2})$: $\mathbf{r_1}$ **r-dominates** $\mathbf{r_2}$
- **r-skyband**: options r-dominated by <k others
- Good filtering, but still superset of UTK options

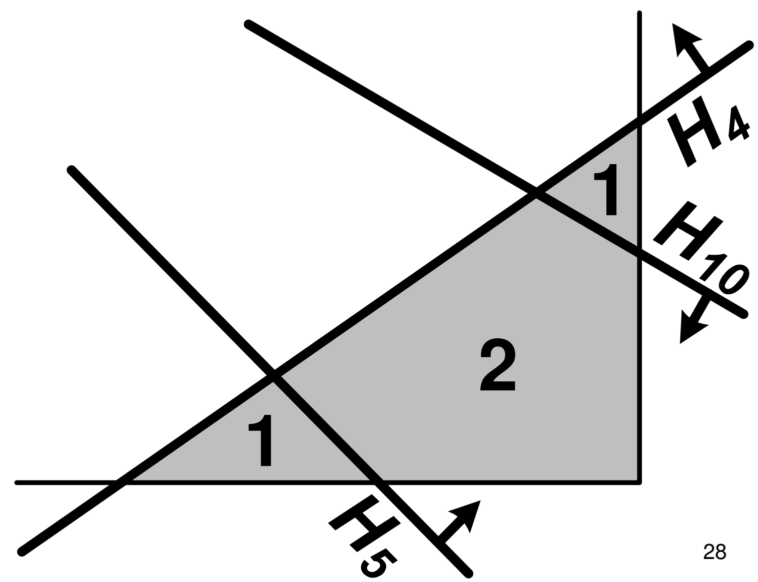




UTK_1 – Refinement (RSA)

- ∀ remaining candidate **r** determine if there is position in *R* where **r** is in top-k
- Progressively consider competitors and **recursively partition** *R* by focusing only on promising regions
- Use r-dominance relationships to prioritize competitors during verification of **r**





UTK_1 – Drill optimization

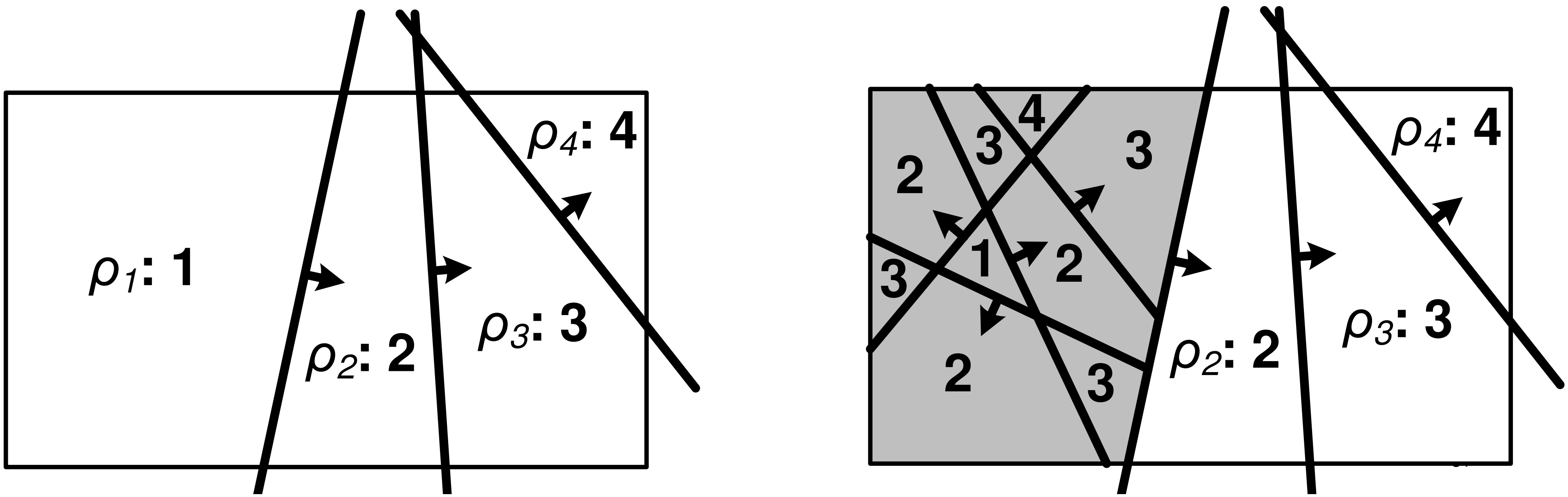
- When a promising partition is examined, we first perform a regular top-k query for a ***drill vector***, i.e., a vector inside the partition
- If candidate **r** is in top-k, it is part of UTK_1 result
- Drill vector must be inside the partition
- We compute it using LP as the vector **q*** in the partition that maximizes score of **r**

UTK_2 – Refinement (JAA)

- Choose a candidate **p** as anchor and produce a **single partitioning** of *R* for all candidates, i.e., determine the rank of **p** anywhere in *R*
- If its rank is different than *k* in some partitions, choose a different anchor **p'** for them
- …anchor choice: make sure it's the k-th somewhere in the partition at hand

UTK$_2$: Refinement Example

- Let k=2
- Choose an option as **anchor**
- Determine its rank in *R*
- *equal-to*, *less-than*, and *greater-than* partitions
- E.g., for ρ_1 (less-than) choose **different anchor**



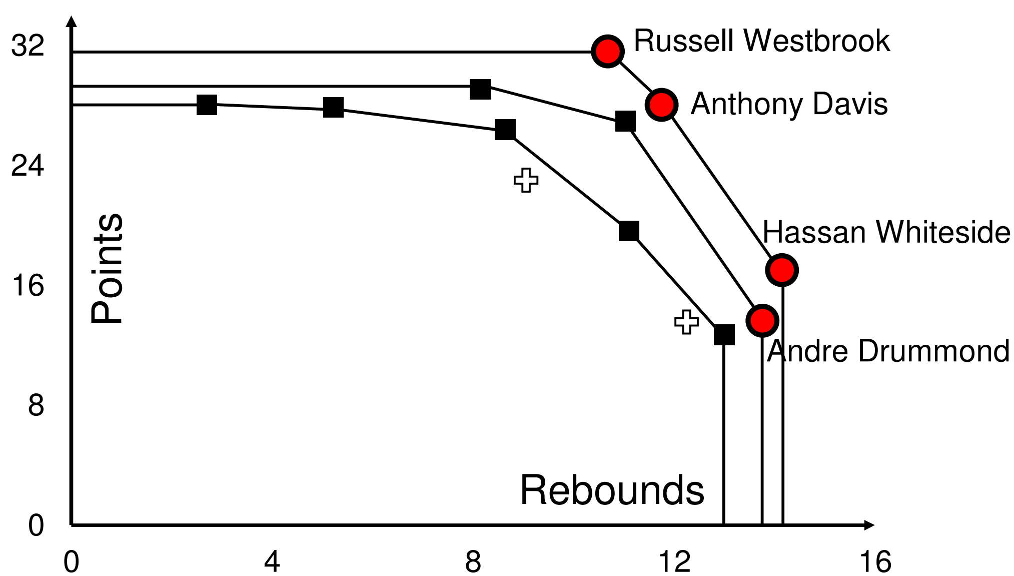
Case Study

UTK (k=3) on NBA data for 2016-17 (2D and 3D)

2D: (rebounds, points)

k = 3 and R = [0:64, 0:74]

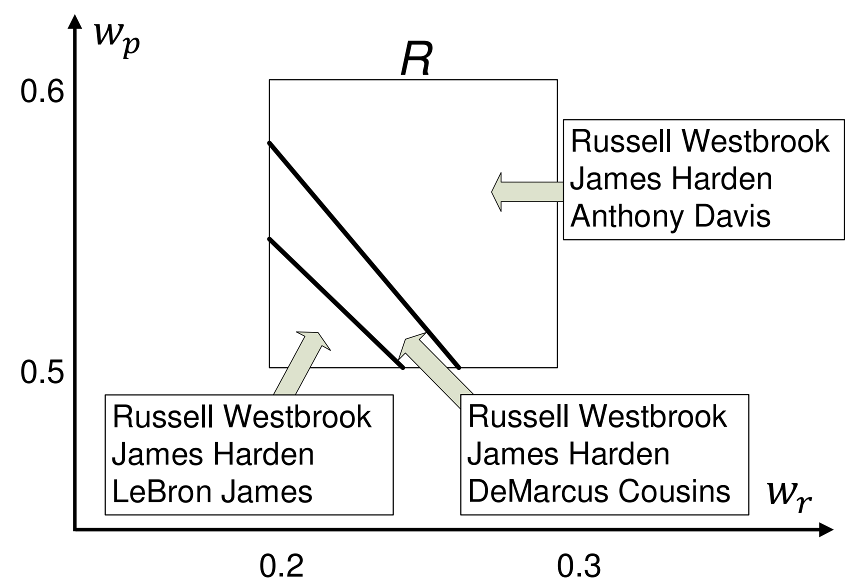
Data Space



3D: (rebounds, points, assists)

R = [0:64, 0:72] × [0:72, 0:74]

Preference Space



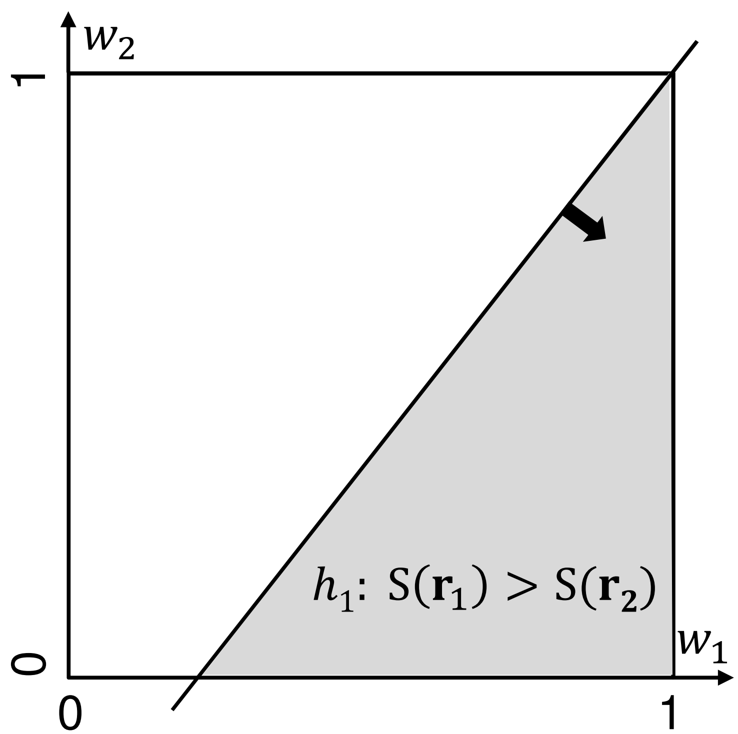
Related in spirit

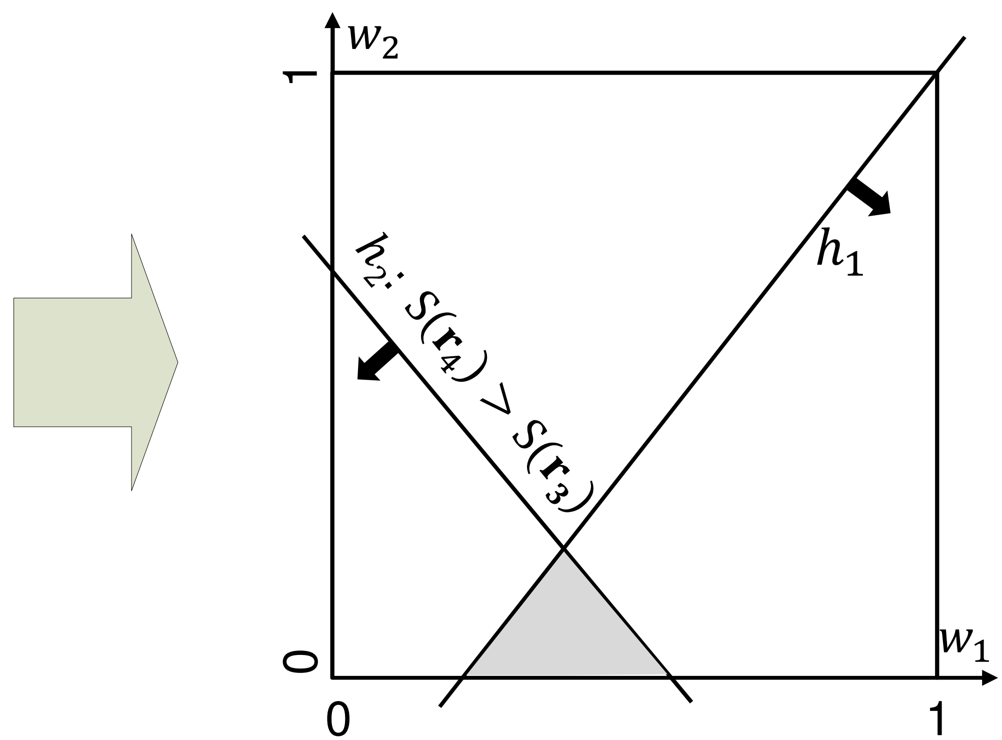
- **[Ciaccia&Martinenghi18]:**
  - – Assuming data indexed by **sorted lists**…
  - – they compute the **r-skyband**…
  - – following the <u>threshold algorithm</u> paradigm
  - – aiming to reduce random/sorted accesses to lists

- **[Qian15]:**
  - – Learn approx. user preferences (i.e., a region *R*)…
  - – by iterative pairwise comparisons

[Qian15]: Iterative pairwise comparisons

- **1st probe: r$_1$** vs. **r$_2$** (user chooses **r$_1$**)
- **2nd probe: r$_3$** vs. **r$_4$** (user chooses **r$_4$**)





[Liu16]: Why-not RTOP-k

- Given a focal option **p**, and…
- a set of query vectors **Q** (for which **p** is not in top-k set)
- Compute the **minimum perturbation** to
  - (attribute values of) **p**, or
  - the query vectors **and** value k, or
  - all of the above (focal option, vector set, value k)
  - s.t. **p** is among the top-k for every vector in **Q**

[Liu16]: Why-not RTOP-k

- Exact solution for 1st problem; improving **p**
- Key idea:
  - Let $\mathbf{p}_{i\text{-}k}$ be the current k-th opt. for query vector $\mathbf{q}_i$
  - To be in top-k for $\mathbf{q}_i$, the updated **p** must outscore $\mathbf{p}_{i\text{-}k}$ for $\mathbf{q}_i$ $\leftrightarrow$ $\mathbf{q}_i \cdot \mathbf{p} \geq \mathbf{q}_i \cdot \mathbf{p}_{i\text{-}k}$
  - This inequality defines a **half-space** $\mathbf{h}_i$ in <u>data space</u>!
  - The new **p** must be in the intersection of the half-spaces $\mathbf{h}_i$ defined for each $\mathbf{q}_i$ in **Q**

[Yang16]: Influence optimization

- Problem: improve **p** so that it is **top-1** for at least m query vectors in set **Q**

- Key idea:
  - Let $\mathbf{p_i}$ be the current k-th opt. for query vector $\mathbf{q_i}$
  - To be top-1 for $\mathbf{q_i}$, the updated **p** must outscore $\mathbf{p_i}$ for $\mathbf{q_i}$ $\leftrightarrow$ $\mathbf{q_i} \cdot \mathbf{p} \geq \mathbf{q_i} \cdot \mathbf{p_i}$
  - This inequality defines a **half-space** $\mathbf{h_i}$ in <u>data space</u>!
  - The new **p** must be in the intersection of <u>at least m</u> half-spaces $\mathbf{h_i}$ defined by vectors $\mathbf{q_i}$ in **Q**

[Yang&Cai17]: Improvement strategies

- Similar objective to prev. problem
- Given focal opt. **p** and a set of query vectors **Q**
- Compute the minimum perturbation (improvement) to values of **p** so that it appears in **top-k** set for at least m vectors in **Q**
- Problem is hard; **heuristic solutions** proposed

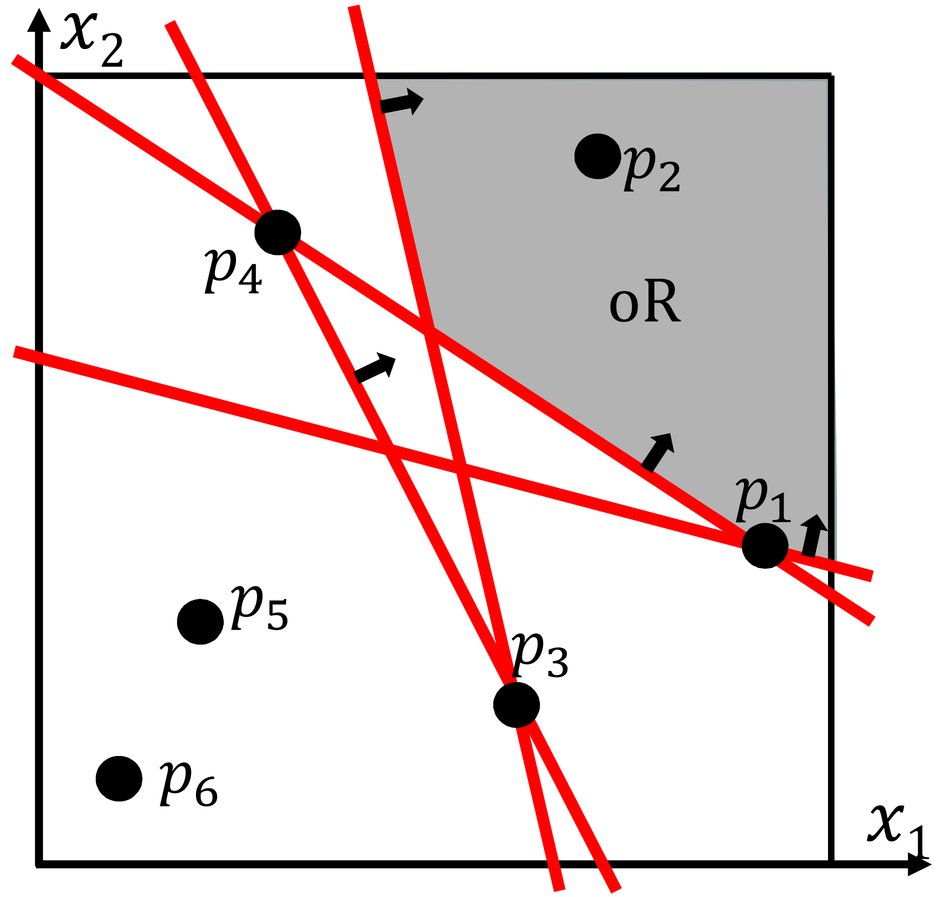
[Tang19]: Top Ranking Region (TopRR)

- Input: dataset & a region R in pref. space (representing our target clientele)
- Query: where should we build a new option **p** s.t. it is in **top-k** set for **any** query vector in R?
- Challenge: dealing with a **continuous** region in pref. space (R) **and** a **continuous** region in data space (the output)
- Key idea: beat continuity by reducing it to a **finite number** of critical points, while **retaining exactness**!

TopRR: Example

Laptop	Speed	Battery
$\boldsymbol{p}_1$	0.9	0.4
$\boldsymbol{p}_2$	0.7	0.9
$\boldsymbol{p}_3$	0.6	0.2
$\boldsymbol{p}_4$	0.3	0.8
$\boldsymbol{p}_5$	0.2	0.3
$\boldsymbol{p}_6$	0.1	0.1

Dataset



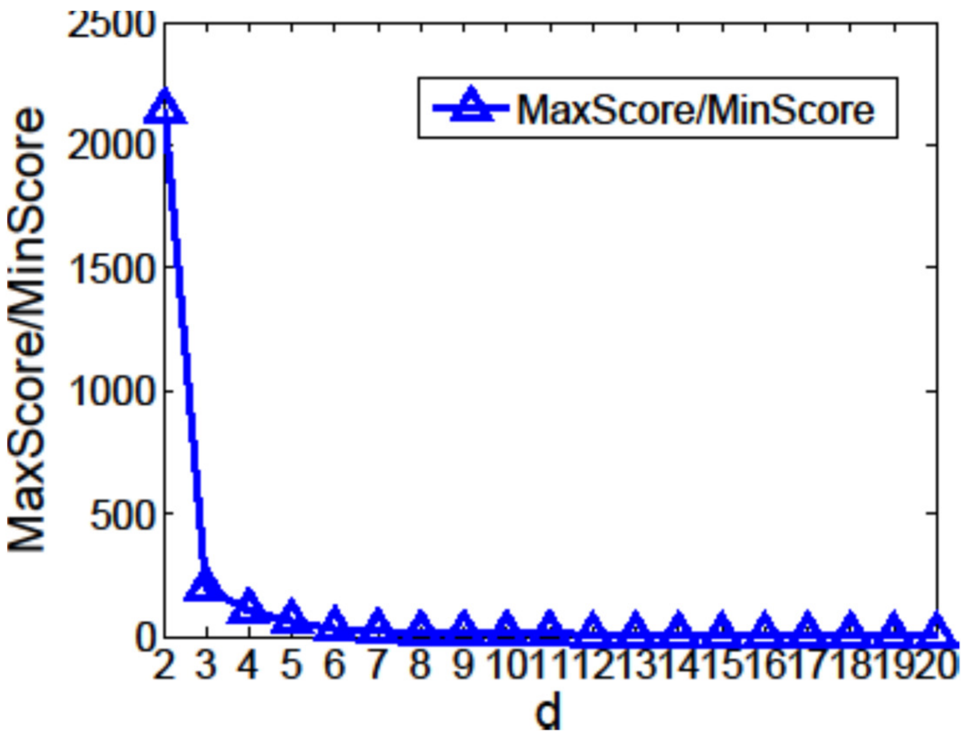
TopRR output for $k = 3$
(in **data space**)

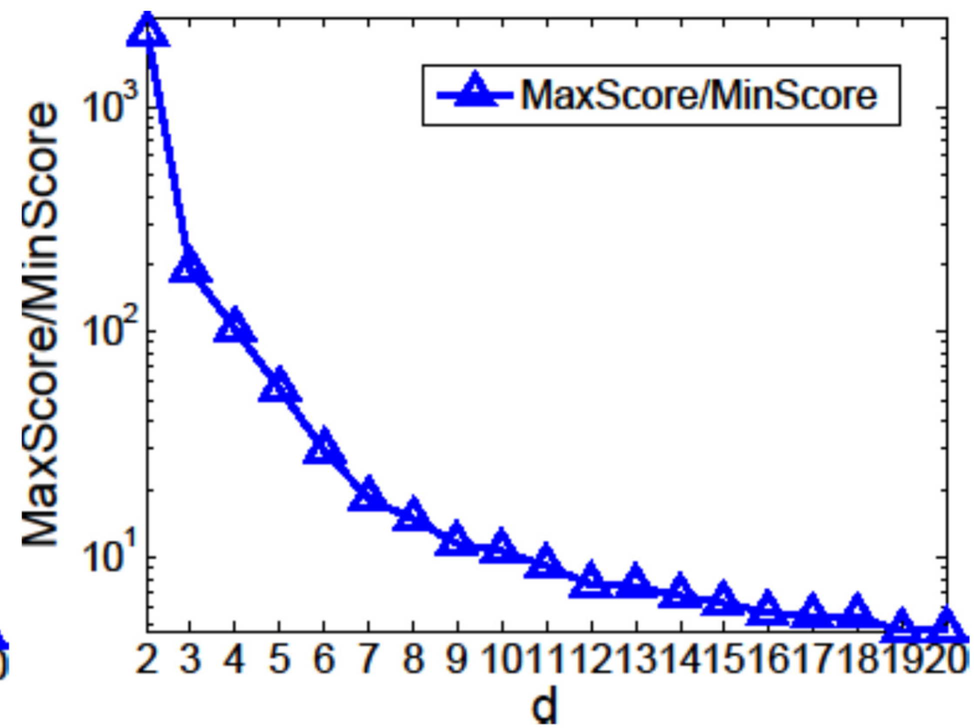
Top-k in High-D?

- 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!