Finding the Best Objects in Large Datasets

Davide Martinenghi

Eurecom, 28 June 2024

Disclaimer


Gemini
GPT
Claude
Ceci n'est pas une présentation
sur les LLMs
Magritte

Buzzwords, languages, and tools

NoSQL

GraphQL

neo4j

SQL

Datalog

Google Cloud

Amazon S3

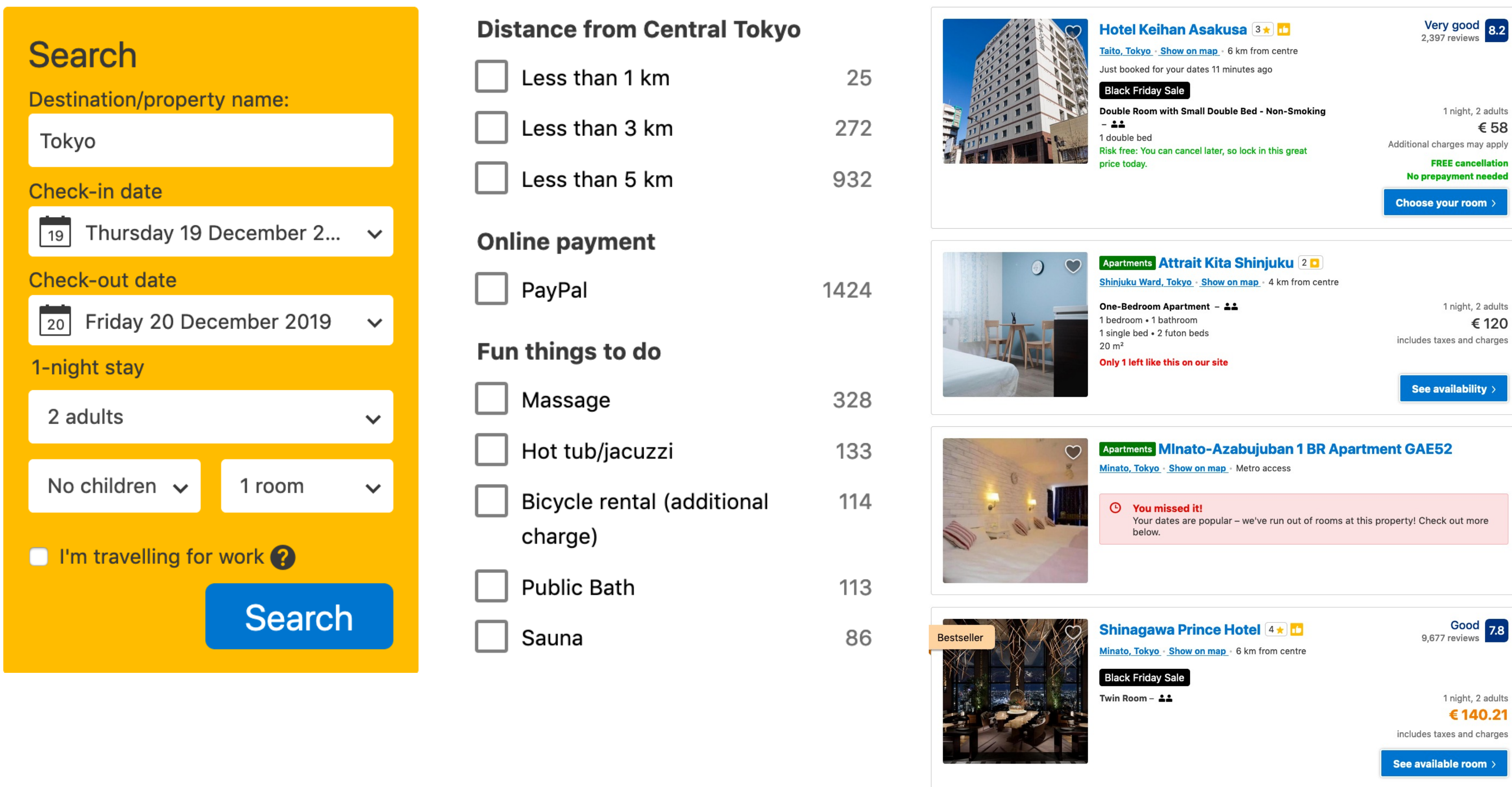
- …but how to get the best results out of the data?
- What does “best” even mean?

A general concern: multi-objective optimization

- Simultaneous optimization of different criteria
  - E.g., different attributes of objects in a dataset
- A general problem formulation:
  - Given *N* objects described by *d* attributes
  - Find the best *k* objects
    - wrt some notion of "goodness"
- Relevant in many applications

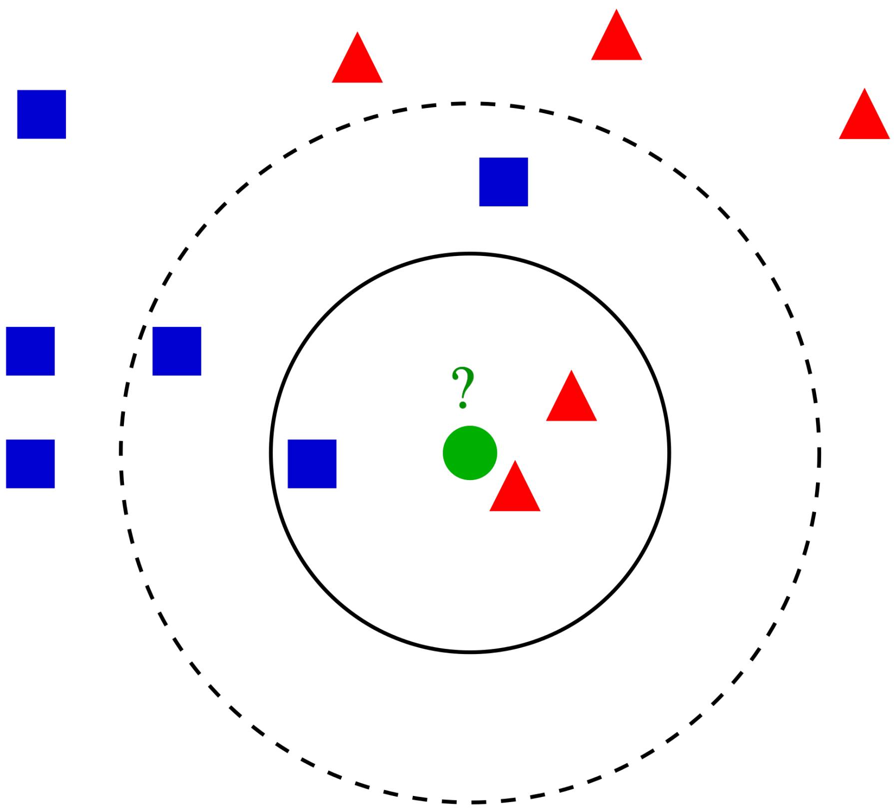
Application: multi-criteria queries

– Example: ranking hotels by combining criteria about available facilities, driving distance, stars, …



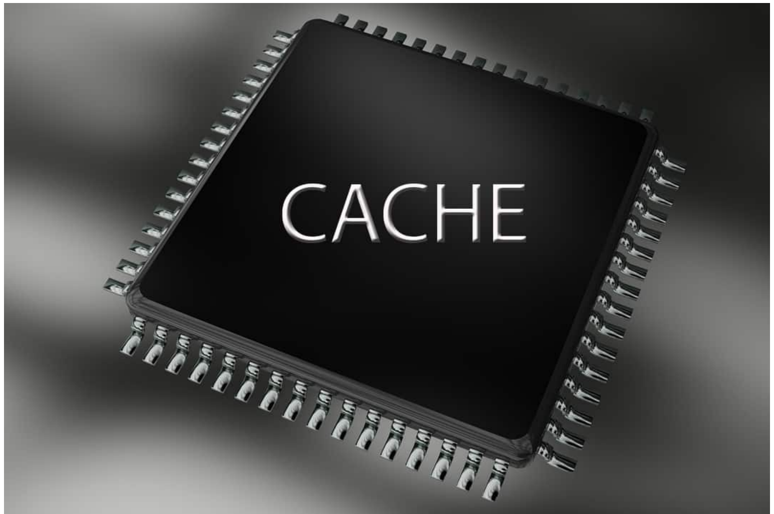
Application: k-nearest neighbors

- (e.g., similarity search)
  - Given *N* points in some metric (*d*-dimensional) space, and a query point *q* in the same space, find the *k* points closest to *q*
  - Used for classification in Machine Learning



Application: caching

- Select the objects (memory cells, pages, files, ...) that are most likely to be accessed soon to minimize the ***miss rate*** among a very large set of *N* objects
- Each such object is described by *d* different attributes, each providing an estimate of the likelihood of reuse
- Goal:
  - What are the most promising *k* objects to be retained/brought to main memory so as to minimize the miss rate?



Many more applications

- Candidate hiring
- Sports ranking, university ranking, …
- Recommender systems
- Feature selection
- Ensemble learning
- …
- Essential aspect in (Big) Data Preparation
  - For subsequent use in, e.g., ML…

Outline

- Historical perspective
- Classical approaches
  - Top-k queries
  - Skyline queries
- New approaches
  - Hybridization of skyline and top-k queries
  - Uncertainty in top-k queries
  - Balance in top-k queries
- Outlook

Historical perspective

Rank aggregation

[Borda, 1770][Marquis de Condorcet, 1785][Llull, 13th century]

- Goal: combining several ranked lists of objects into a single consensus ranking of the objects

Jean-Charles de Borda

Marie Jean Antoine Nicolas de Caritat, Marquis de Condorcet

Ramon Llull

Rank aggregation

- A problem from social choice theory
- Given: *N* candidates, *d* voters
  - No visible score assigned to candidates, only rank

Candidate	Candidate	Candidate	Candidate	Candidate
A	B	D	E	C
B	D	B	A	E
C	E	E	C	A
D	A	C	D	B
E	C	A	B	D
Voter 1	Voter 2	Voter 3	Voter 4	Voter 5

- What is the overall ranking according to all the Voters?
- Who wins? (top-*k* candidates, with *k*=1)

Classical proposals

10 voters, 3 candidates

1	2	3	4	5	6	7	8	9	10
A	A	A	A	A	A	C	C	C	C
C	C	C	C	C	C	B	B	B	B
B	B	B	B	B	B	A	A	A	A

- Borda's proposal
  - n-th place → n points of penalty
  - winner (C): lowest overall penalty

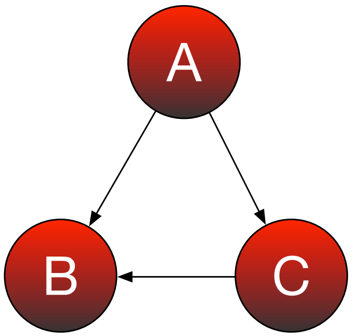
Borda scores:

A: 1x6+3x4 = 18

B: 3x6+2x4 = 26

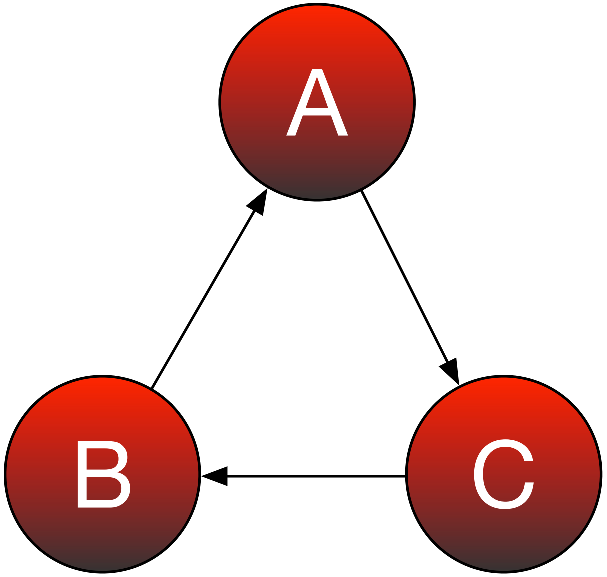
C: 2x6+1x4 = 16

- Condorcet's proposal:
  - winner (A): defeats everyone in pairwise majority rule election



Condorcet’s paradox

1	2	3
C	B	A
B	A	C
A	C	B



- A winner may not exist

More paradoxes

[Arrow, 1950]

Axioms for aggregation may not work out:

Arrow's paradox: no rank-order electoral system can be designed that always satisfies reasonable "fairness" criteria:

- No dictatorship (nobody determines, alone, the group's preference)
- If all prefer X to Y, then the group prefers X to Y
- If, for all voters, the preference between X and Y is unchanged, then the group preference between X and Y is unchanged

Kenneth Arrow

Perfect democracy is unattainable!

Ranking queries
(a.k.a. top-k queries)

Top-*k* queries

- Focus on the best *k* out of *N* items
  - Best = most important/interesting/relevant/…
- Items described by (*d*) numerical attributes
  - not just the rank
- Preferences through a *scoring function*
  - assigns an overall score for ranking tuples
  - E.g., S(t) = t.Points + t.Rebounds

Top-*k* queries in SQL

```
SELECT *
FROM NBA_STATS
ORDER BY Points + Rebounds
LIMIT 5
```

Player	Points	Rebounds	…
Antetokounmpo	28.1	11.0	…
Embiid	28.5	10.6	…
Jokić	26.4	10.6	…
Dončić	27.7	8.0	…
Towns	24.8	10.6	…

Top-*k* queries in SQL

- Standard in SQL since 2008

```
SELECT *
FROM NBA_STATS
ORDER BY Points + Rebounds
FETCH FIRST 5 ROWS ONLY
```

Player	Points	Rebounds	…
Antetokounmpo	28.1	11.0	…
Embiid	28.5	10.6	…
Jokić	26.4	10.6	…
Dončić	27.7	8.0	…
Towns	24.8	10.6	…

- If input already **sorted**: $O(k)$
- Else perform in-memory sort (through a *heap*): $O(N \log k)$
  - Better: $O(N + k \log k)$

Top-*k* join queries in SQL

- Generally, many relations may be involved, e.g.,

```
SELECT *
FROM RESTAURANTS R, HOTELS H
WHERE R.City = H.City
ORDER BY R.Price + H.Price
FETCH FIRST 2 ROWS ONLY
```

- Many algorithms focus on **top-*k* 1-1 join queries**
  - All joins on a common key attribute
  - Practically relevant in two main scenarios:
    - There is an index for retrieving tuples according to each preference
    - The relation is vertically distributed (the “middleware” scenario)

Threshold Algorithm (TA)

[Fagin, Lotem, Naor, PODS 2001]

Input: integer k, a monotone function S combining ranked lists R_1, …, R_d
Output: the top k <object, score> pairs

1. Descend in parallel in each list R_i
2. For each found object o, extract its score s_j in the other lists R_j
3. Compute score $S(s_1, \ldots, s_d)$. If top k so far, remember o
4. Threshold $T=S(L_1, \ldots, L_d)$ where L_i is the last score seen for R_i
5. If the score of the k-th object is worse than T, go to step 1
6. Return the current top-k objects

- TA is not strictly optimal, but cannot be beaten by an arbitrarily large factor (instance optimality)
- The authors of TA received the Gödel prize in 2014 for the design of innovative algorithms

Hotels	Cleanliness	Hotels	Rating
Ibis	.92	Crillon	.9
Etap	.91	Novotel	.9
Novotel	.85	Sheraton	.8
Mercure	.85	Hilton	.7
Hilton	.825	Ibis	.7
Sheraton	.8	Ritz	.7
Crillon	.75	Lutetia	.6
…		…	

Top 2	Score

Threshold
value: T = ??
point: τ =(??,??)

- Query: hotels with best cleanliness and rating
  - Scoring function: 0.5*cleanliness+0.5*rating

Hotels	Cleanliness	Hotels	Rating
Ibis	.92	Crillon	.9
Etap	.91	Novotel	.9
Novotel	.85	Sheraton	.8
Mercure	.85	Hilton	.7
Hilton	.825	Ibis	.7
Sheraton	.8	Ritz	.7
Crillon	.75	Lutetia	.6
…		…	

Top 2	Score
Crillon	.825
Ibis	.81

Threshold value: $T = .91$
point: $\tau = (.92, .9)$

- Query: hotels with best cleanliness and rating
  - Scoring function: 0.5*cleanliness+0.5*rating
- Strategy:
  - Make one sorted access at a time in each list
  - Then make a random access for each new hotel

Hotels	Cleanliness	Hotels	Rating
Ibis	.92	Crillon	.9
Etap	.91	Novotel	.9
Novotel	.85	Sheraton	.8
Mercure	.85	Hilton	.7
Hilton	.825	Ibis	.7
Sheraton	.8	Ritz	.7
Crillon	.75	Lutetia	.6
…		…	

Top 2	Score
Novotel	.875
Crillon	.825

Threshold value: $T = .905$
point: $\tau = (.91, .9)$

- Query: hotels with best cleanliness and rating
  - Scoring function: 0.5*cleanliness+0.5*rating
- Strategy:
  - Make one sorted access at a time in each list
  - Then make a random access for each new hotel

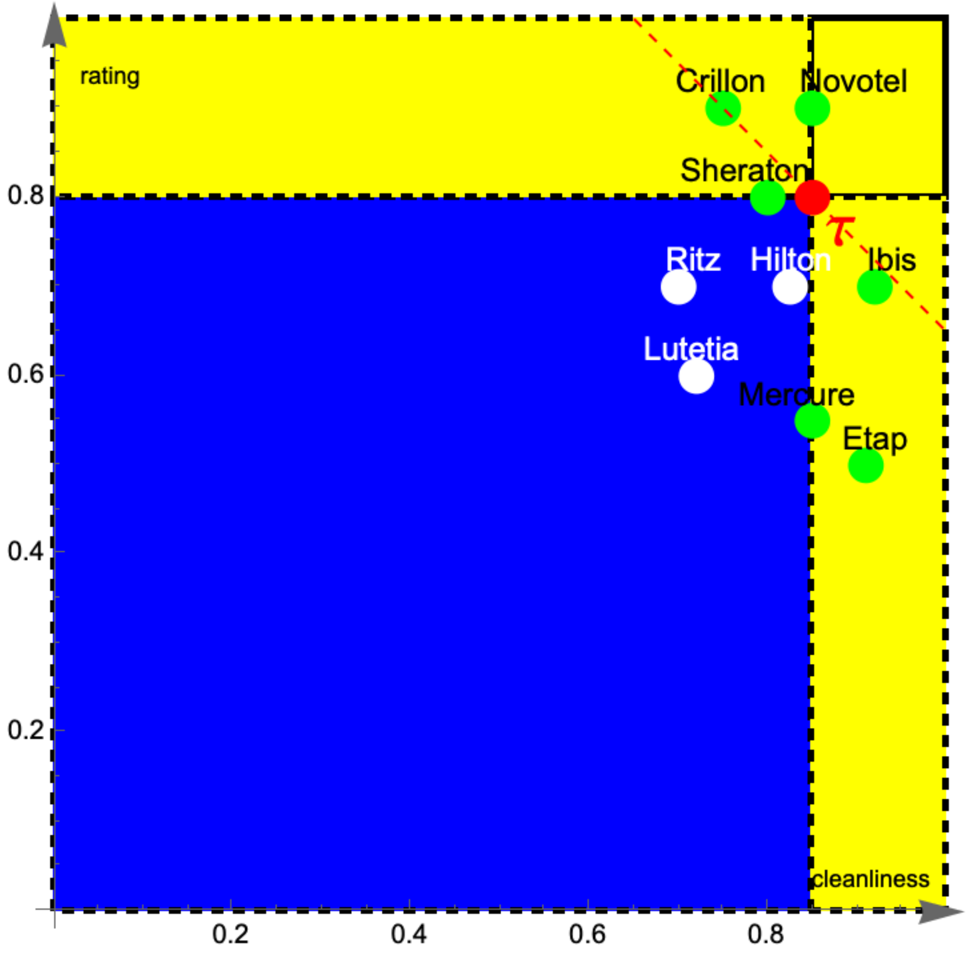
Hotels	Cleanliness	Hotels	Rating
Ibis	.92	Crillon	.9
Etap	.91	Novotel	.9
Novotel	.85	Sheraton	.8
Mercure	.85	Hilton	.7
Hilton	.825	Ibis	.7
Sheraton	.8	Ritz	.7
Crillon	.75	Lutetia	.6
…		…	

Top 2	Score
Novotel	.875
Crillon	.825

Threshold
value: T = .825
point: τ =(.85,.8)

- Query: hotels with best cleanliness and rating
  - Scoring function: 0.5*cleanliness+0.5*rating
- Strategy:
  - Stop when the score of the *k*-th hotel is no worse than the threshold

Why does TA work?



- τ is the threshold point
- TA stops when the yellow region (fully seen points) contains at least k points at least as good as τ
- None of the points in the blue region (unseen points) can beat τ
- The dashed red line separates the region of points with a higher score than τ from the rest
  - Now, Crillon is as good as τ and Novotel is better

Ranking queries – main aspects

- Pros:
  - Very effective in identifying the best objects
    - Wrt. a specific scoring function
  - Very efficient
  - Excellent control of the cardinality of the result (*k*)
  - Easy to express the relative importance of attributes
- Cons:
  - For a user, it is difficult to specify a scoring function
    - E.g., the weights of a weighted sum (magic numbers…)

Skyline queries

Skylines

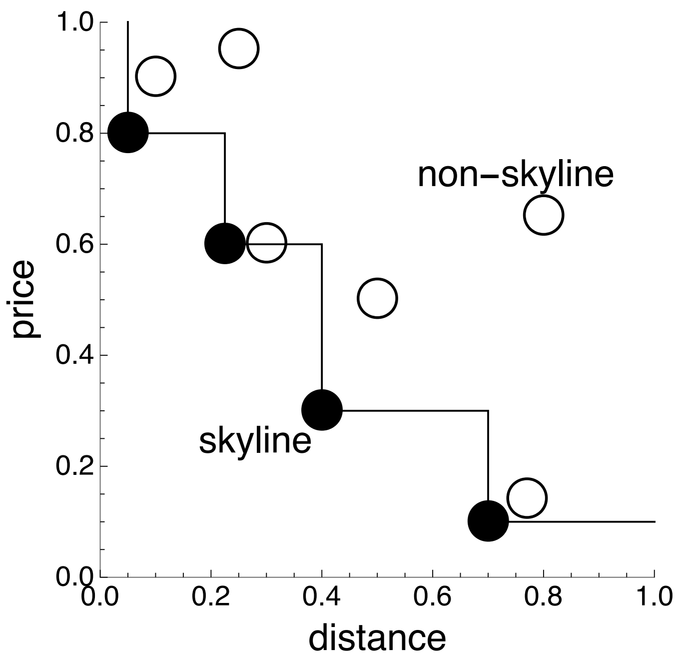
- The skyline of a relation is the set of its non-dominated tuples. Aka:
  - Maximal vectors problem (computational geometry)
  - Pareto-optimal solutions (multi-objective optimization)
- Tuple t dominates tuple s, indicated $t \prec s$, iff

  $\forall i.\ 1 \leq i \leq m \rightarrow t[A_i] \leq s[A_i]$

  (t is nowhere worse than s)

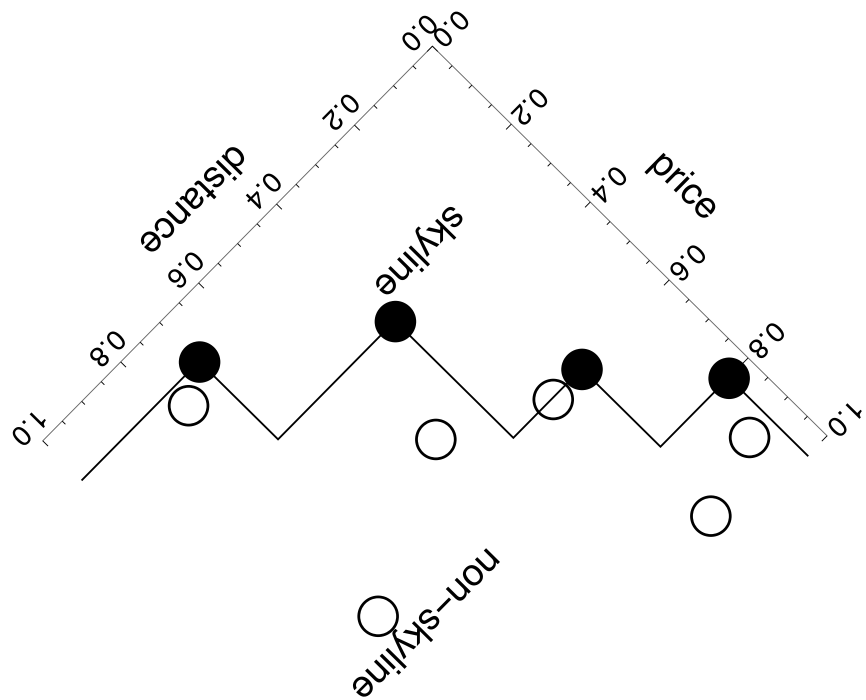
  $\exists j.\ 1 \leq j \leq m \wedge t[A_j] < s[A_j]$

  (and better at least once)



Skylines

- In 2D, the shape resembles the contour of the dataset (hence the name)



Skyline queries in SQL

[Börzsönyi et al., ICDE 2001]

- No standard notation
- Can be easily rendered in SQL:

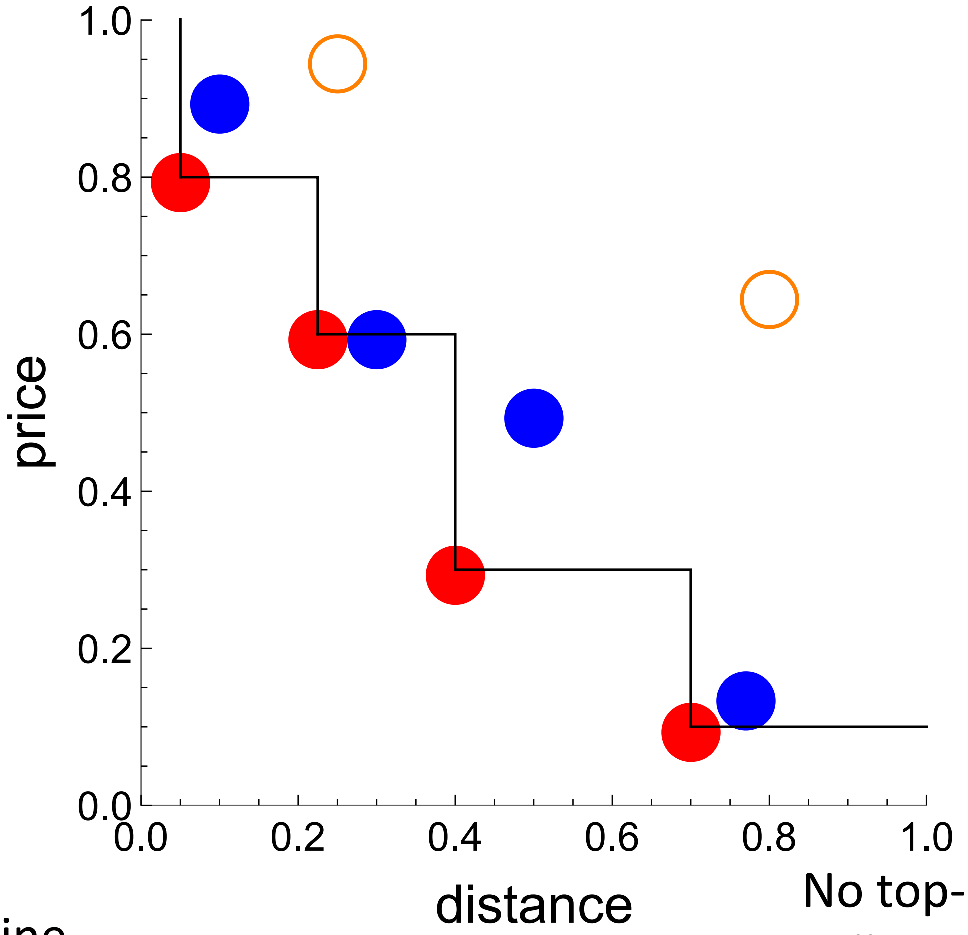
```
SELECT * FROM Hotels h
WHERE h.city = 'Paris' AND NOT EXISTS (
    SELECT * FROM Hotels h1
    WHERE h1.city = h.city AND
     h1.distance <= h.distance AND
     h1.price    <= h.price AND
    (h1.distance <  h.distance OR
     h1.price    <  h.price))
```

- Computation is $O(N^2)$
  - Presorting the dataset helps, but still quadratic

Skylines – main aspects

- Pros:
  - Effective in identifying potentially interesting objects if nothing is known about the preferences of a user
  - Very simple to use (no parameters needed!)
- Cons:
  - May return too many results
  - Computation is essentially quadratic in the size of the dataset
  - No preferences (e.g., price is more important than distance)
- Extension: *k*-skyband = set of tuples dominated by less than *k* tuples
  - Skyline = 1-skyband
  - <u>Every top-*k* result set is contained in the *k*-skyband</u>

Example: skyline/*k*-skyband query

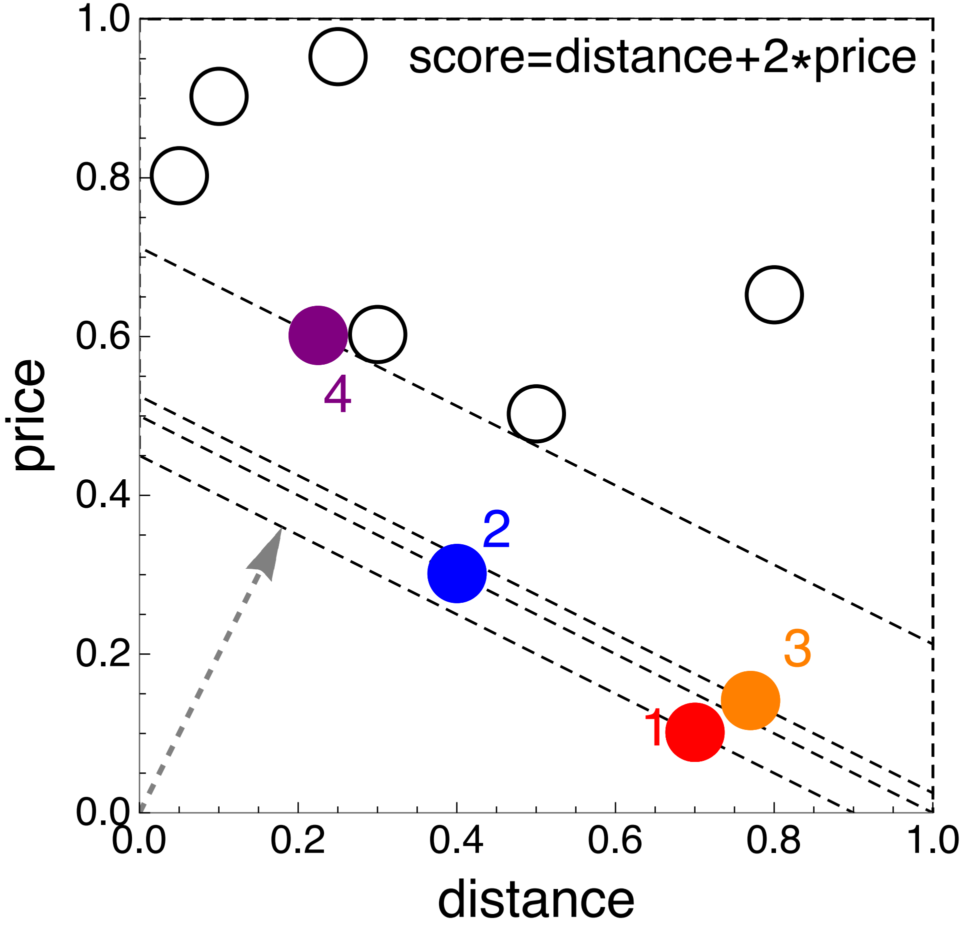


skyline

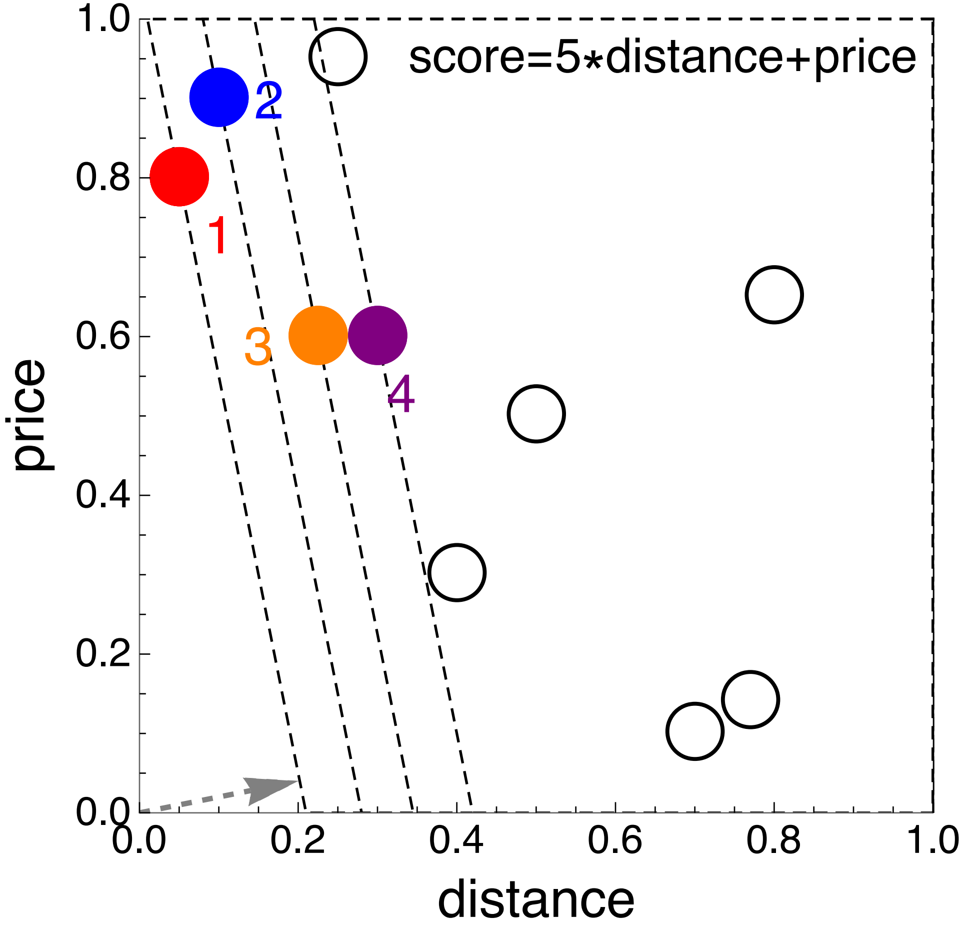
2-skyband = 3-skyband

No top-2 or top-3 query will return a point

Example: ranking query



Example: another ranking query



Extending skylines

Skylines, revisited

- Two equivalent points of view:
  - Tuples that are non-dominated:

    $$\textsc{Sky}(r) = \{t \in r \mid \nexists s \in r.\ \ s \prec t\}$$

  - Tuples that are optimal according to some monotone scoring function:

    $$\textsc{Sky}(r) = \{t \in r \mid \exists f \in \mathcal{M}.\ \forall s \in r.\ \ s \neq t \to f(t) < f(s)\}$$

Skylines, revisited

- Two equivalent points of view:
  - Tuples that are non-dominated:

    $$\textsc{Sky}(r) = \{t \in r \mid \nexists s \in r.\ \ s \prec t\}$$

  - Tuples that are optimal according to some monotone scoring function:

    $$\textsc{Sky}(r) = \{t \in r \mid \exists f \in \mathcal{M}.\ \forall s \in r.\ \ s \neq t \rightarrow f(t) < f(s)\}$$

Idea: accommodate preferences by using a subset of *M* (all monotone functions)

Dominance, revisited

- Consider a set of monotone functions *F*:

$$t \prec_F s, \text{ iff, } \forall f \in \mathit{F}.\ f(t) \leq f(s)$$

- *F*-dominance = standard dominance if *F* = *M*

Flexible skylines: ND and PO
[VLDB 2017, TODS 2020]

- Skyline as non-dominated tuples:

$$\textsc{Sky}(r) = \{t \in r \mid \nexists s \in r.\ s \prec t\}$$

- Skyline as optimal tuples:

$$\textsc{Sky}(r) = \{t \in r \mid \exists f \in \mathcal{M}.\ \forall s \in r.\ s \neq t \rightarrow f(t) < f(s)\}$$

Flexible skylines: ND and PO

[VLDB 2017, TODS 2020]

- Skyline as non-dominated tuples:

$$\textsc{Sky}(r) = \{t \in r \mid \nexists s \in r.\ s \prec t\}$$

- Non-Dominated *F*-Skyline (ND):

$$\textsc{nd-Sky}(r;\mathcal{F}) = \{t \in r \mid \nexists s \in r.\ s \prec_{\mathcal{F}} t\}$$

- Skyline as optimal tuples:

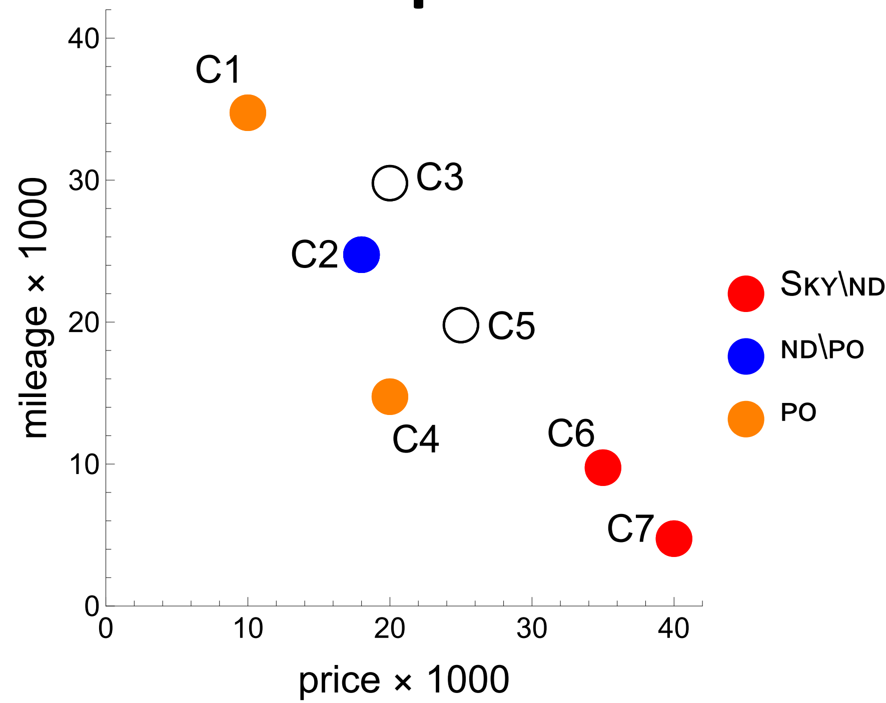
$$\textsc{Sky}(r) = \{t \in r \mid \exists f \in \mathcal{M}.\ \forall s \in r.\ s \neq t \to f(t) < f(s)\}$$

- Potentially Optimal *F*-Skyline (PO):

$$\textsc{po-Sky}(r;\mathcal{F}) = \{t \in r \mid \exists f \in \mathcal{F}.\ \forall s \in r.\ s \neq t \to f(t) < f(s)\}$$

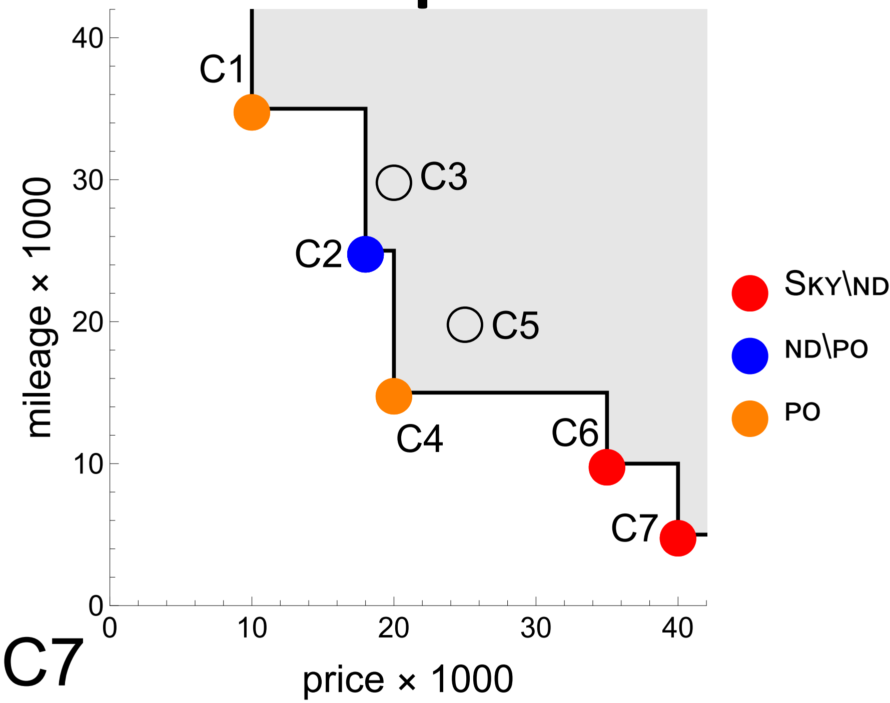
Flexible skylines - example

A dataset of used cars



Flexible skylines - example

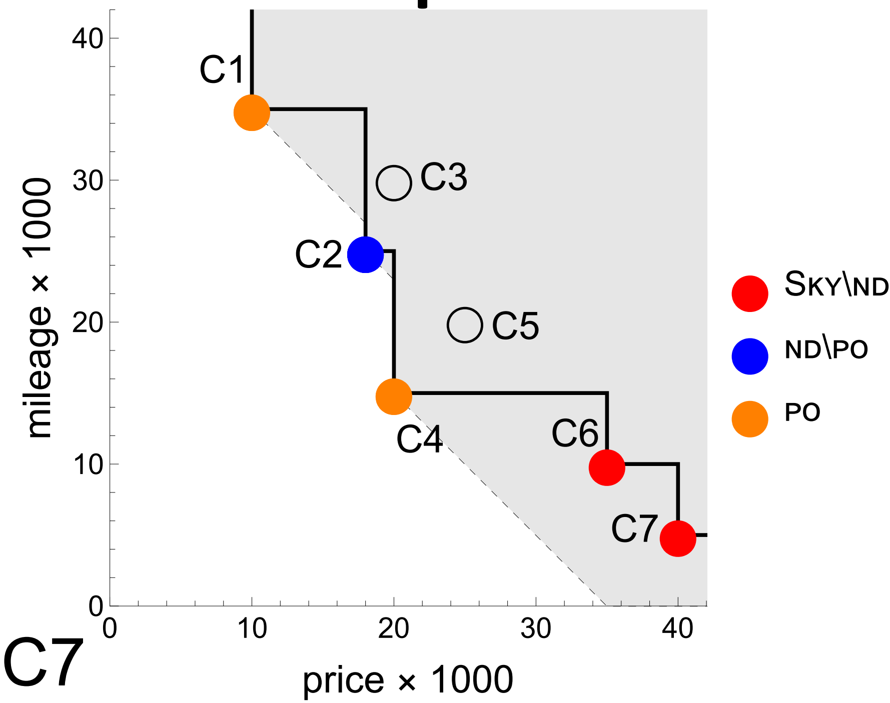
A dataset of used cars



- Sky returns C1, C2, C4, C6, C7

Flexible skylines - example

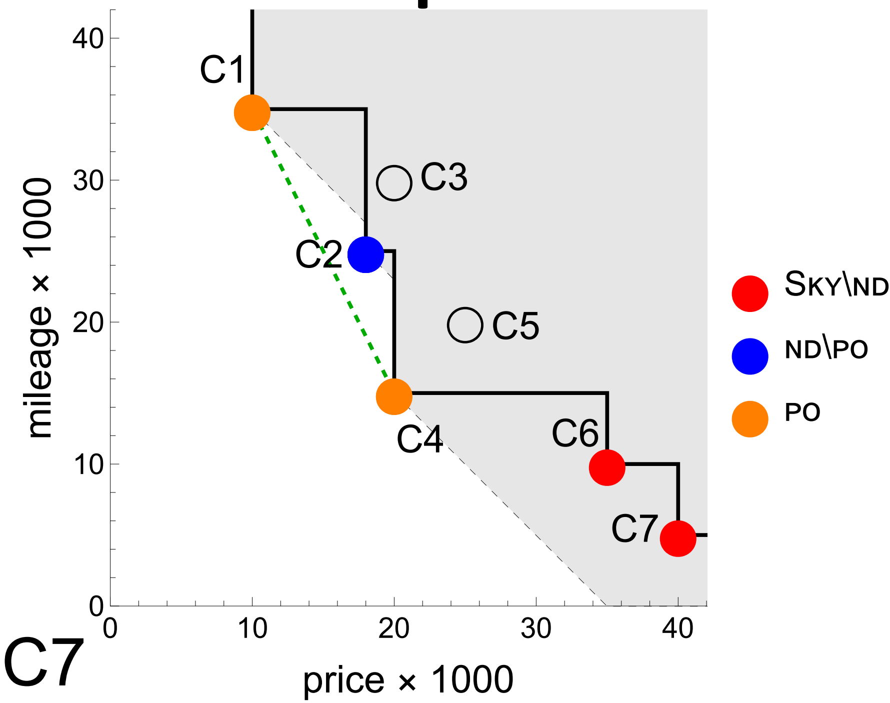
A dataset of used cars



- Sky returns C1, C2, C4, C6, C7
- Consider $\mathcal{F} = \{w_P \texttt{Price} + w_M \texttt{Mileage} \mid w_P \geq w_M\}$
- ND-Sky returns C1, C2, C4
  - C6 and C7 are *F*-dominated by C4

Flexible skylines - example

A dataset of used cars

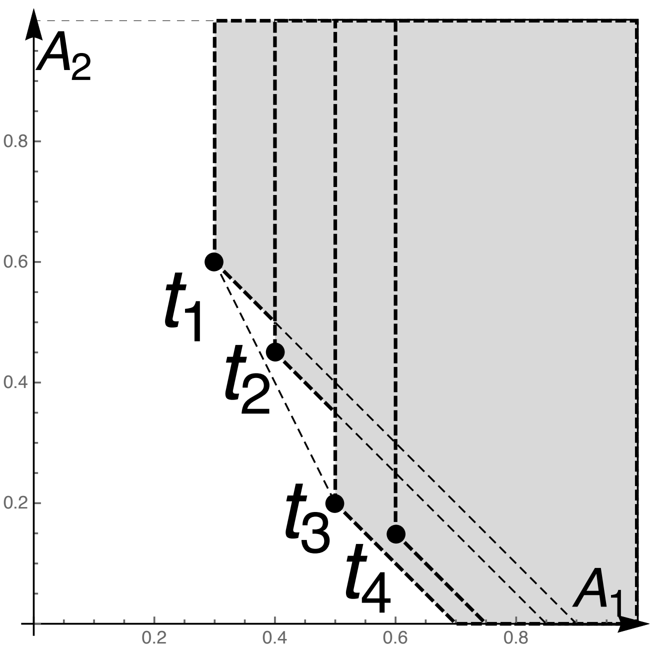


- Sky returns C1, C2, C4, C6, C7
- Consider $\mathcal{F} = \{w_P\texttt{Price} + w_M\texttt{Mileage} \mid w_P \geq w_M\}$
- ND-Sky returns C1, C2, C4
  - C6 and C7 are *F*-dominated by C4
- PO-Sky returns C1, C4
  - No allowed combination of weights can make C2 the top car

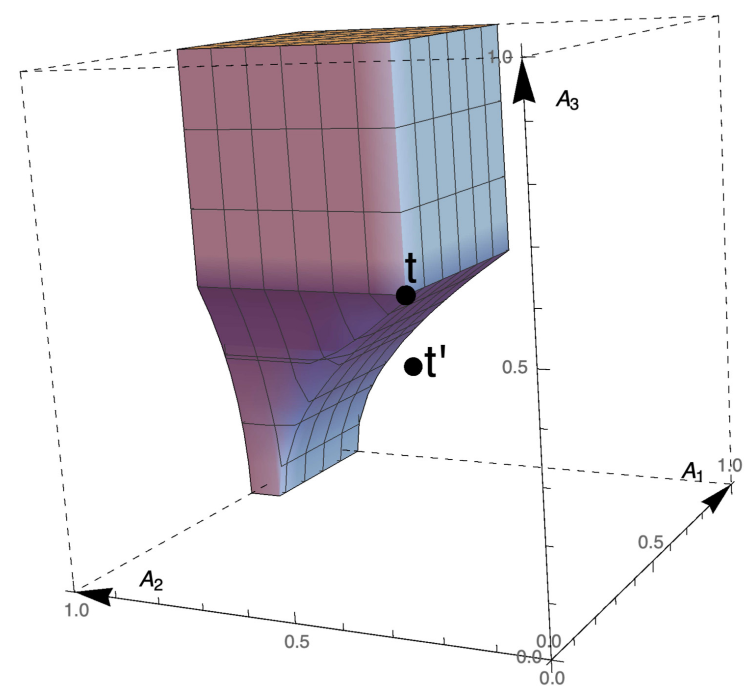
F-dominance regions

- The *F*-dominance region of t
  - set of all points *F*-dominated by t

Linear, $w_1 \geq w_2$



Quadratic, $w_1+w_2 \geq w_3$



Extensions of Flexible Skylines

[CIKM 2018] [Mouratidis&Tang, VLDB 2018]

- Idea: leverage the *k*-skyband to target the potential top *k* (instead of just top 1)
- $ND_k(r;F)$ = tuples *F*-dominated by less than *k* tuples in *r*
- $PO_k(r;F)$ = top-*k* tuples in *r* for at least one function in *F*
- Both ND and PO coincide with
  - Top-*k* query if *F* is a single scoring function
  - *k*-Skyband if *F* = *M* (all monotone functions)

Pros and cons of Flexible Skylines

- Pros:
  - User preferences (via constraints on weights)
    - More robust with respect to magic numbers
  - Reduced output size
  - Computed efficiently for L_p norms with linear constraints on weights
- Cons:
  - Cardinality of output not directly controllable
    - Even less so for extensions based on *k*-skybands
  - Still slow for loose constraints

Speaking of magic numbers…

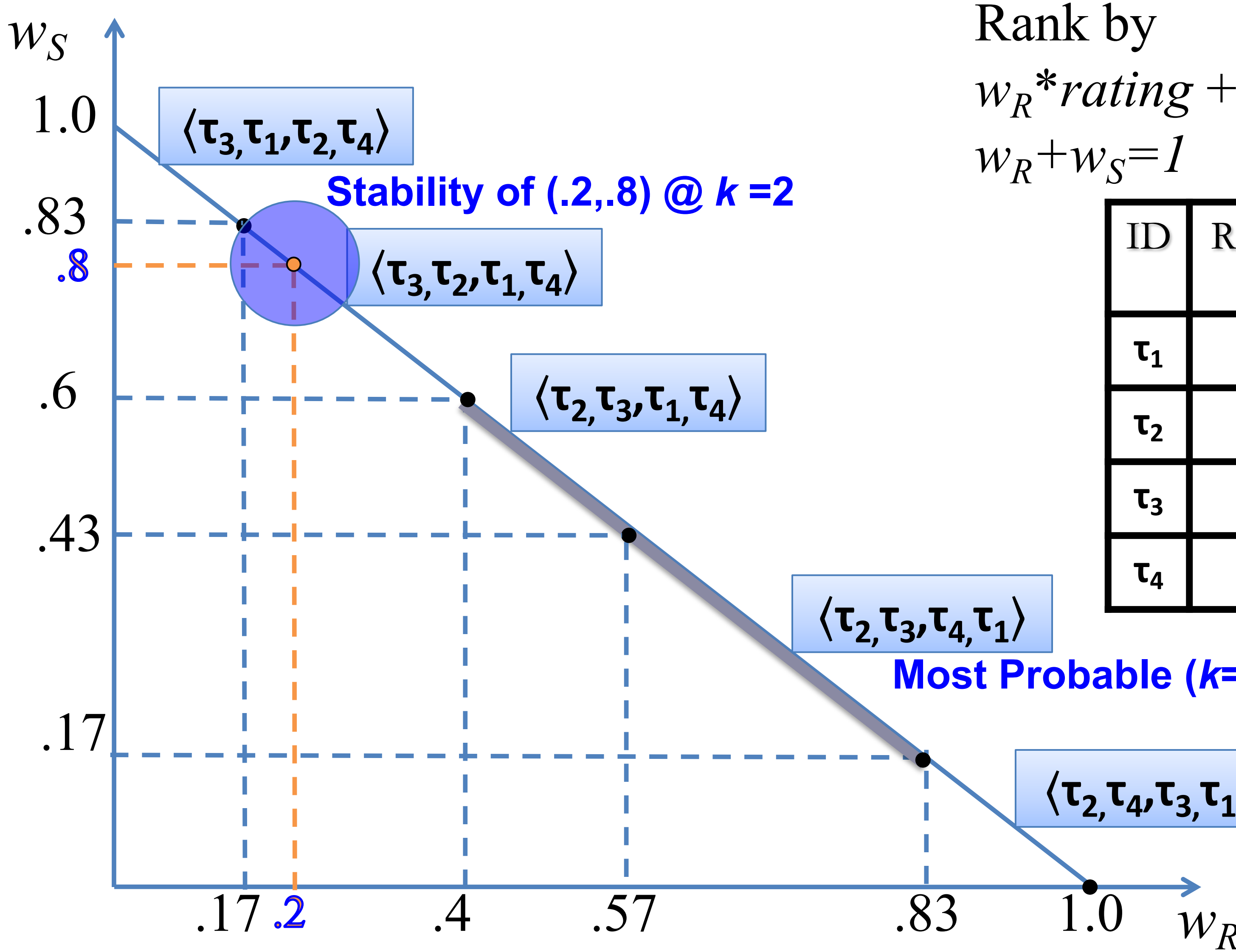
- FIFA World Ranking system 2006-2018
  - Teams ranked by a combination of their previous performance (p_x = performance *x* years ago)

    score = $p_0 + 0.5p_1 + 0.3p_2 + 0.2p_3$
  - A very unstable scoring function
    - Tiny weight changes heavily affect the final ranking
      - These weights were just magic numbers
  - NB: France was never #1 in that period

1	•	Argentina
2	•	France
3	•	Belgium
4	↑ 1	Brazil
5	↓ 1	England
6	•	Portugal
7	•	Netherlands
8	•	Spain
9	↑ 1	Croatia
10	↓ 1	Italy

Stability

[SIGMOD 2011]



Rank by

w_R*rating + w_S*stars

$w_R + w_S = 1$

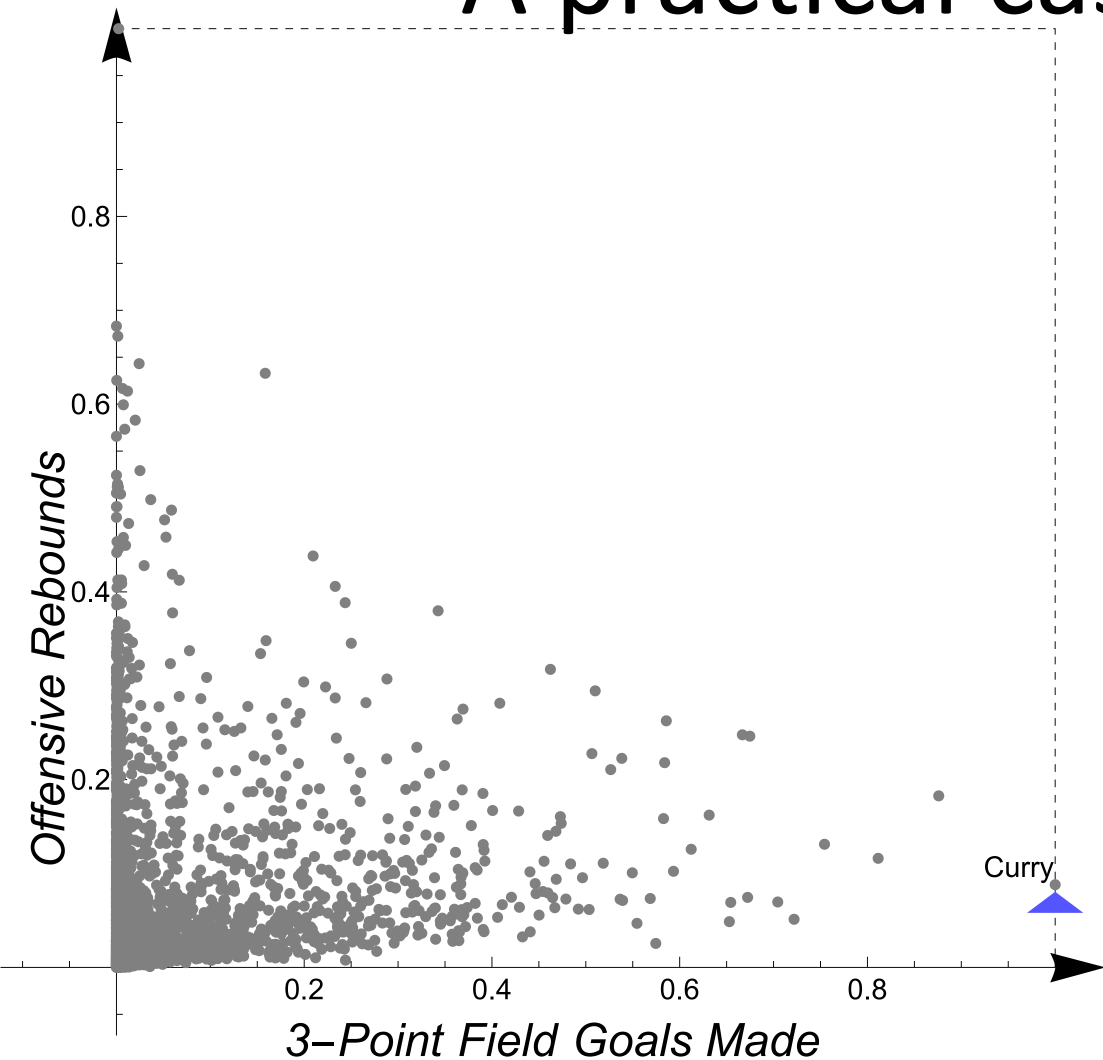
ID	Rating	Stars
τ_1	2	6
τ_2	7	5
τ_3	4	7
τ_4	5	2

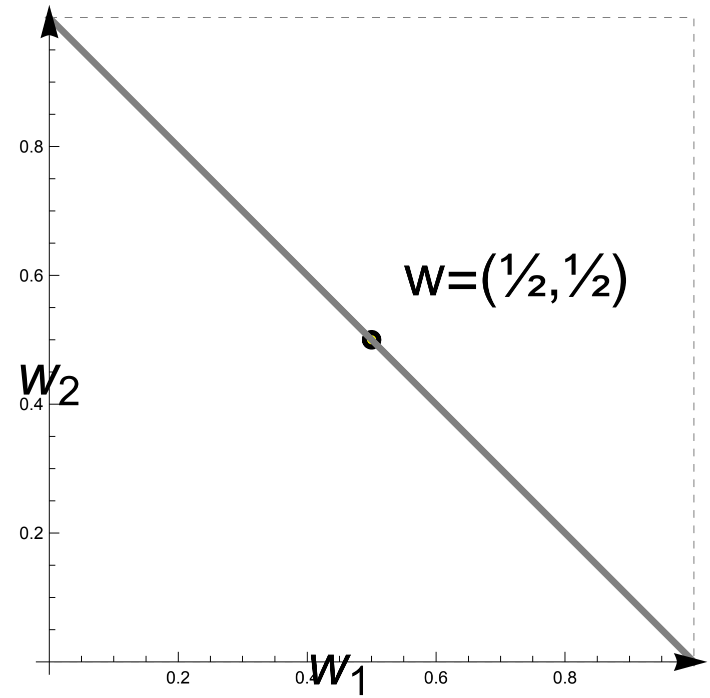
Adding cardinality control

[Mouratidis et al., SIGMOD 2021]

- Aim: Output-Size Specified (OSS) operators
- Idea:
  - Collect user preferences (weight vector **w**)
  - Apply either ND_k (called ORD) or PO_k (ORU)
  - Limit their output size to a user-defined number *m*
    - Use a set *F* of linear scoring functions whose weight vector **w'** is at a distance at most ϱ from **w** so that the output size is exactly *m*

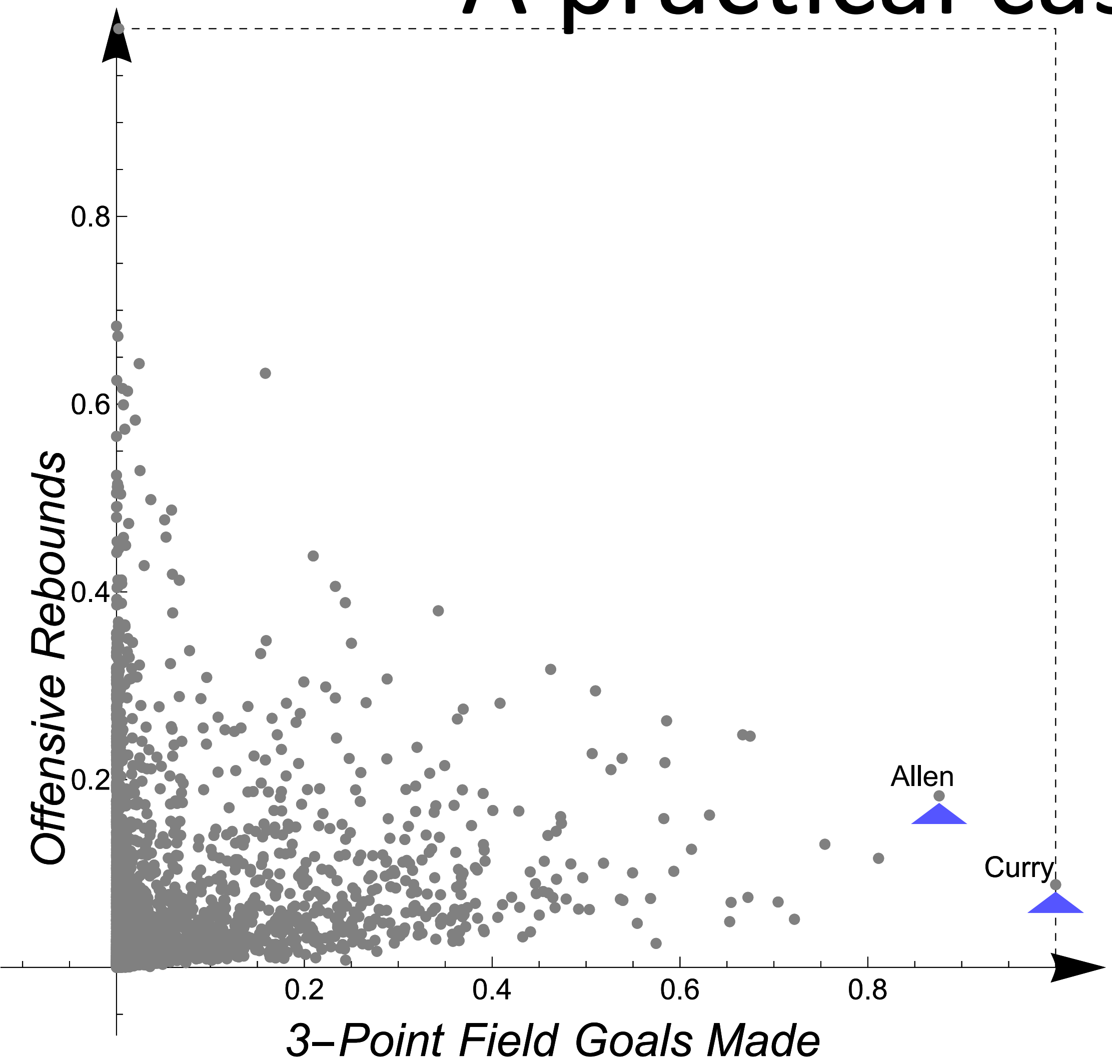
A practical case: NBA



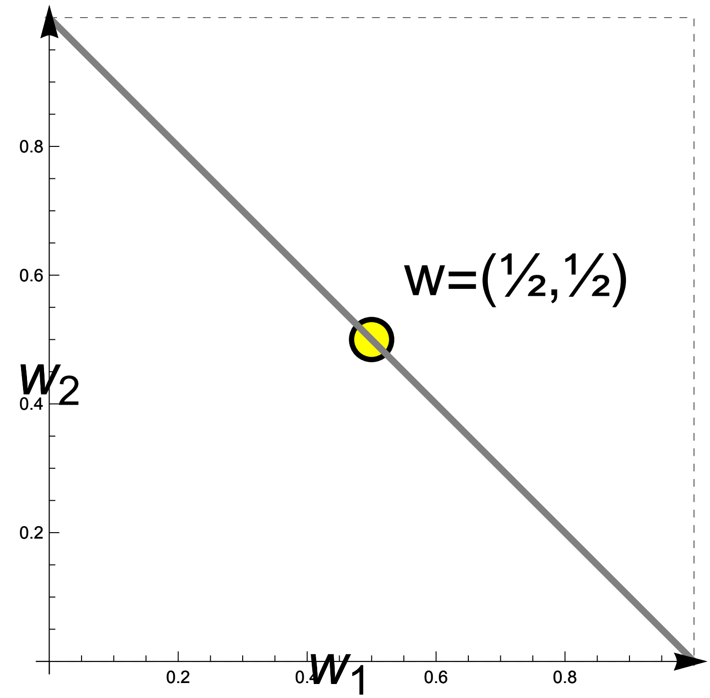


Top-1 result

A practical case: NBA

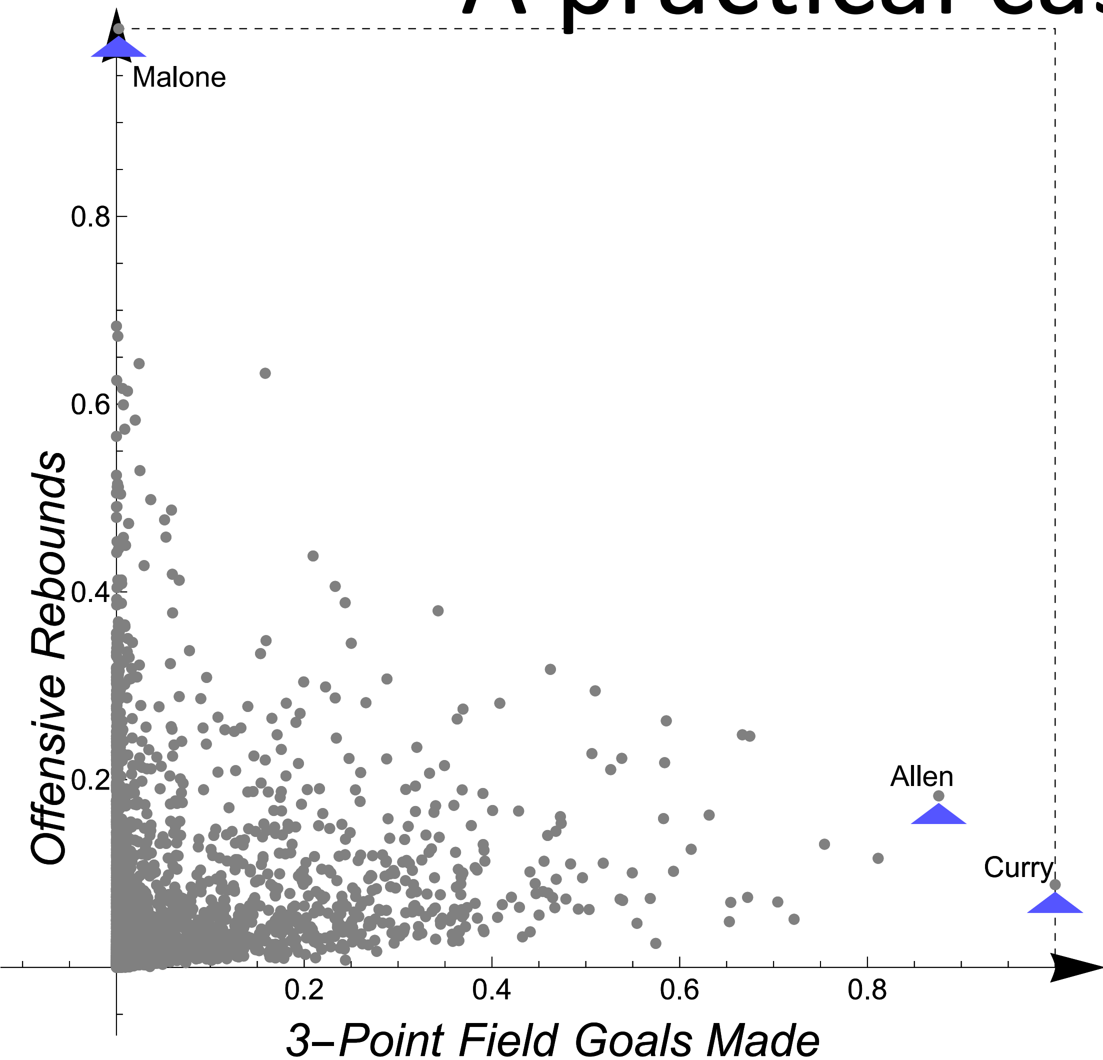


Top-2 results

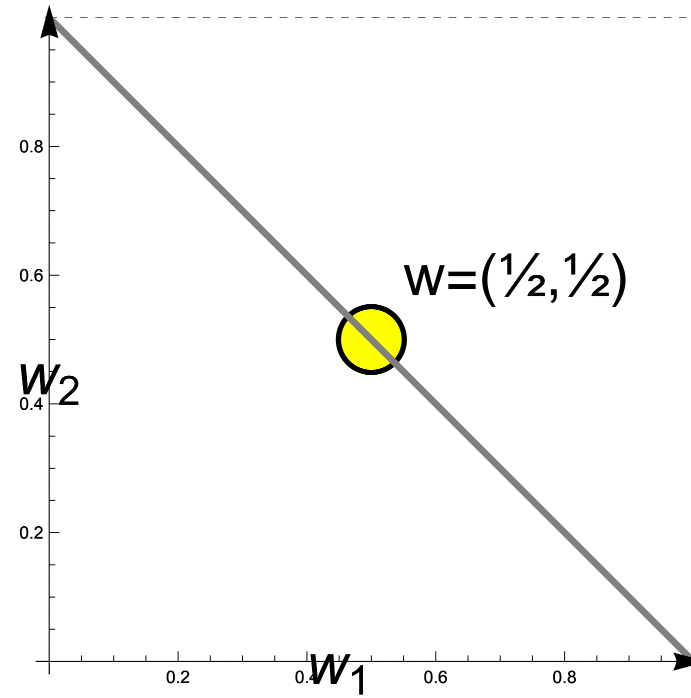


Increasing the radius…

A practical case: NBA

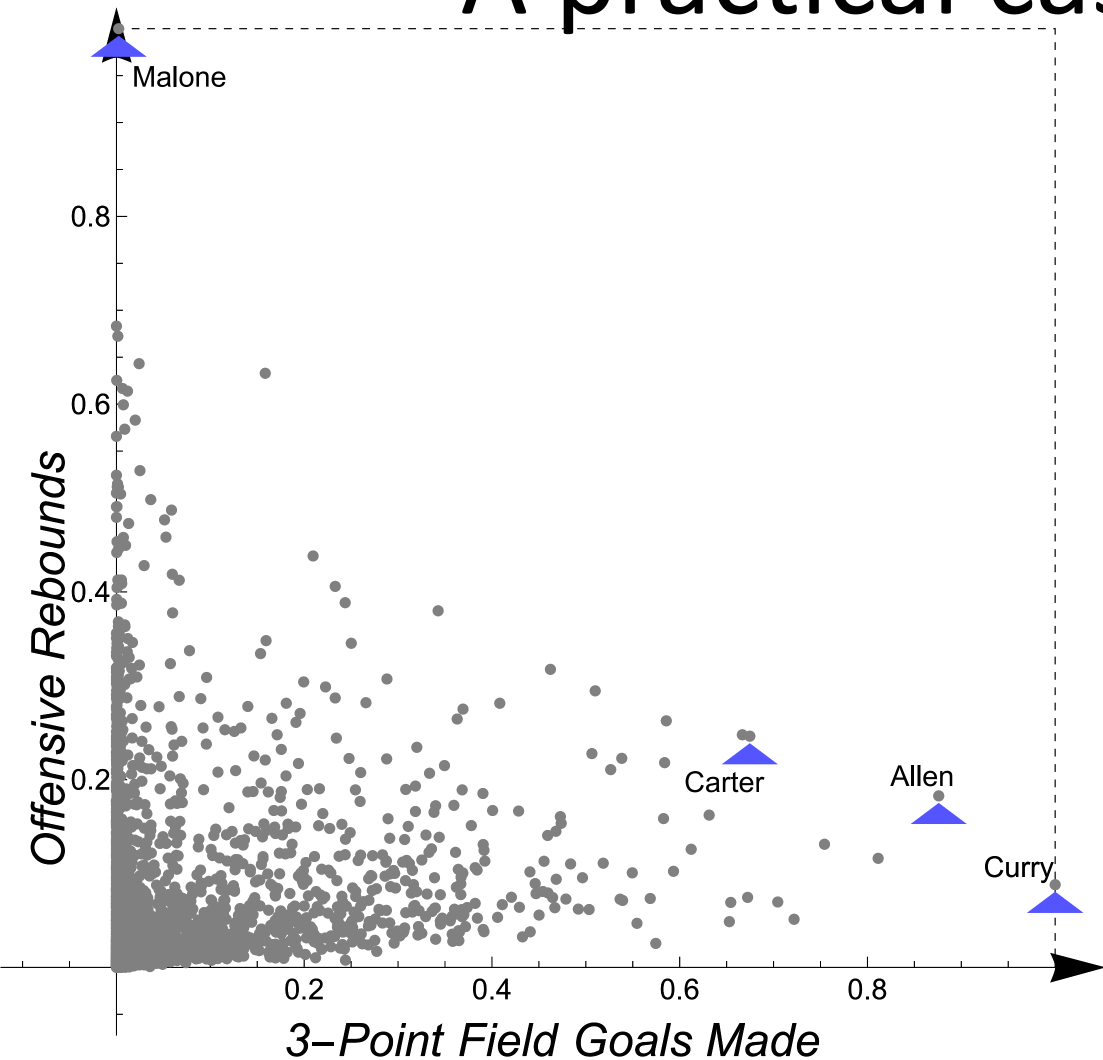


Top-3 results

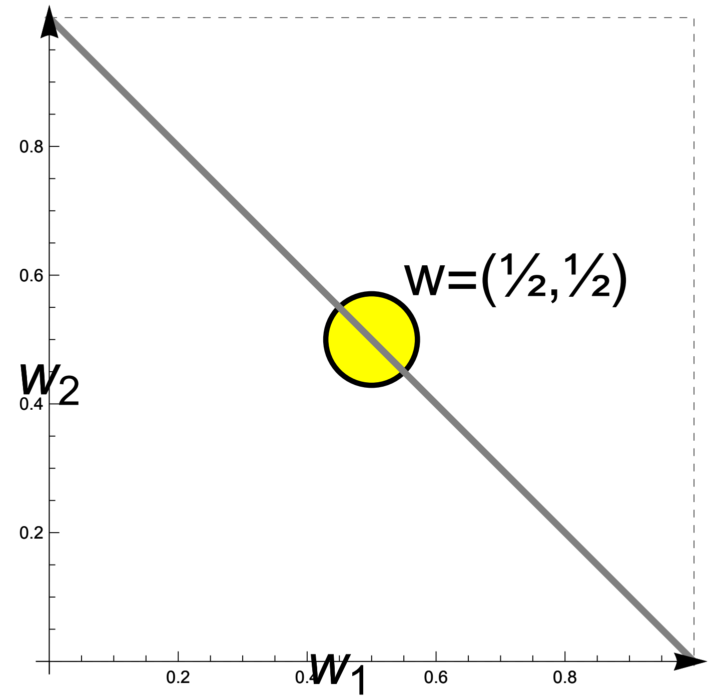


Increasing the radius…

A practical case: NBA

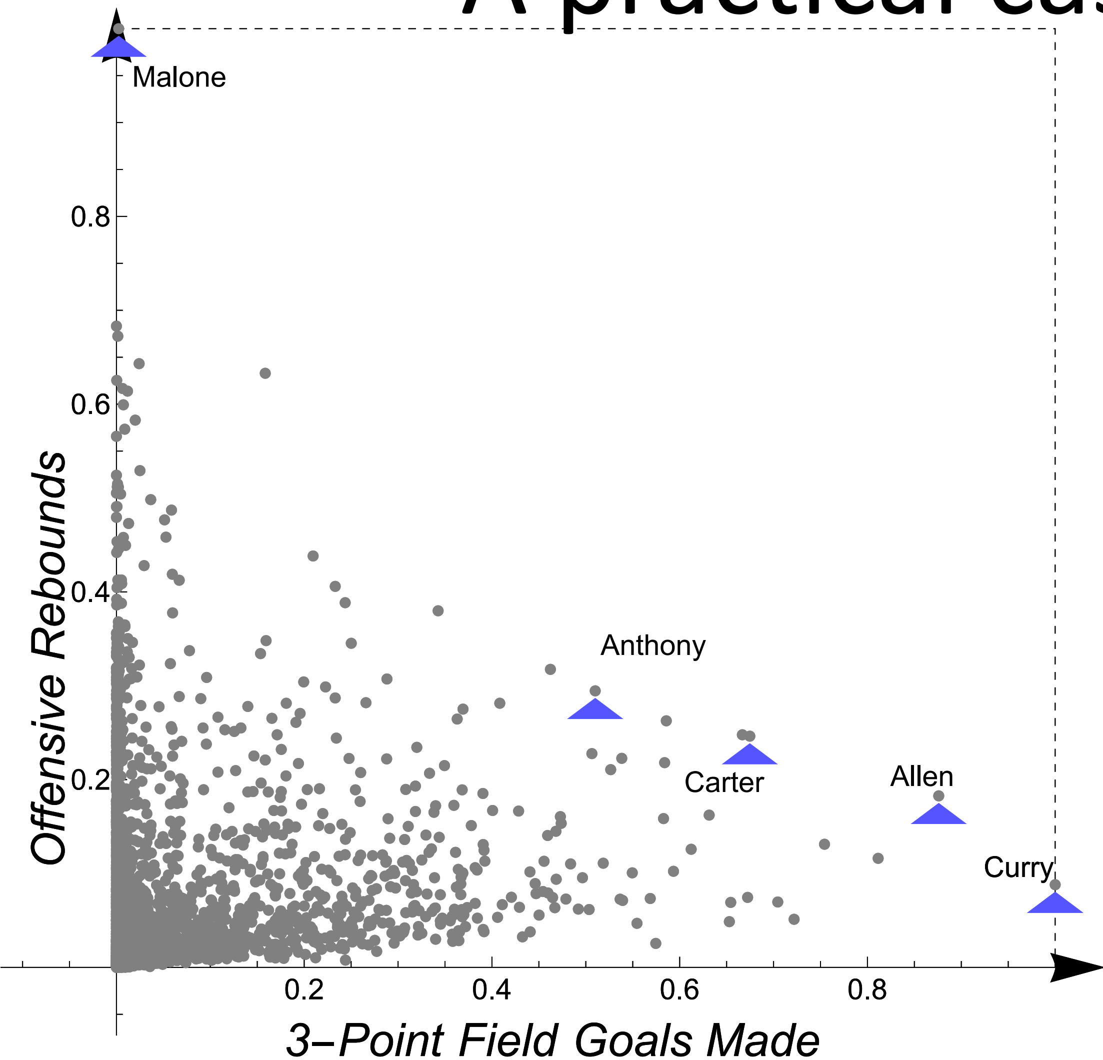


Top-4 results

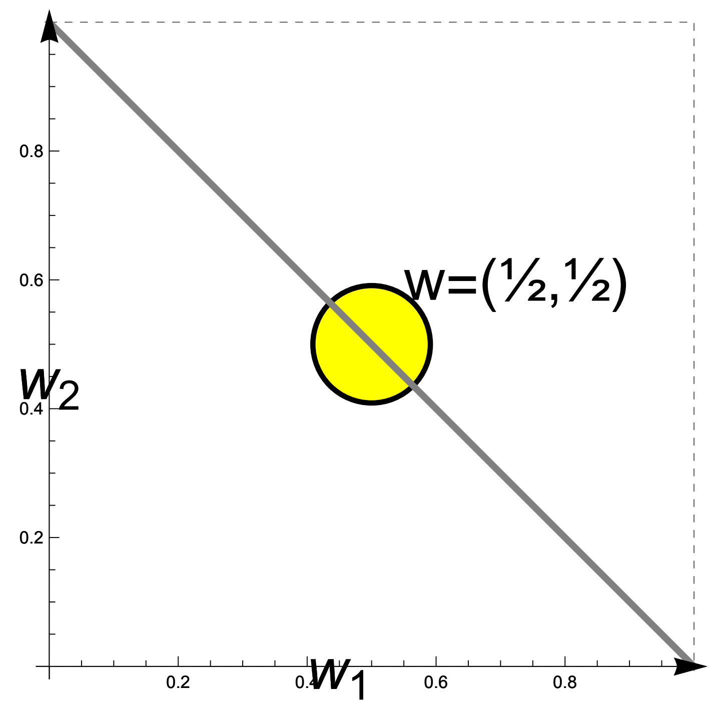


Increasing the radius…

A practical case: NBA



Top-5 results



Increasing the radius…

Features of ORD and ORU

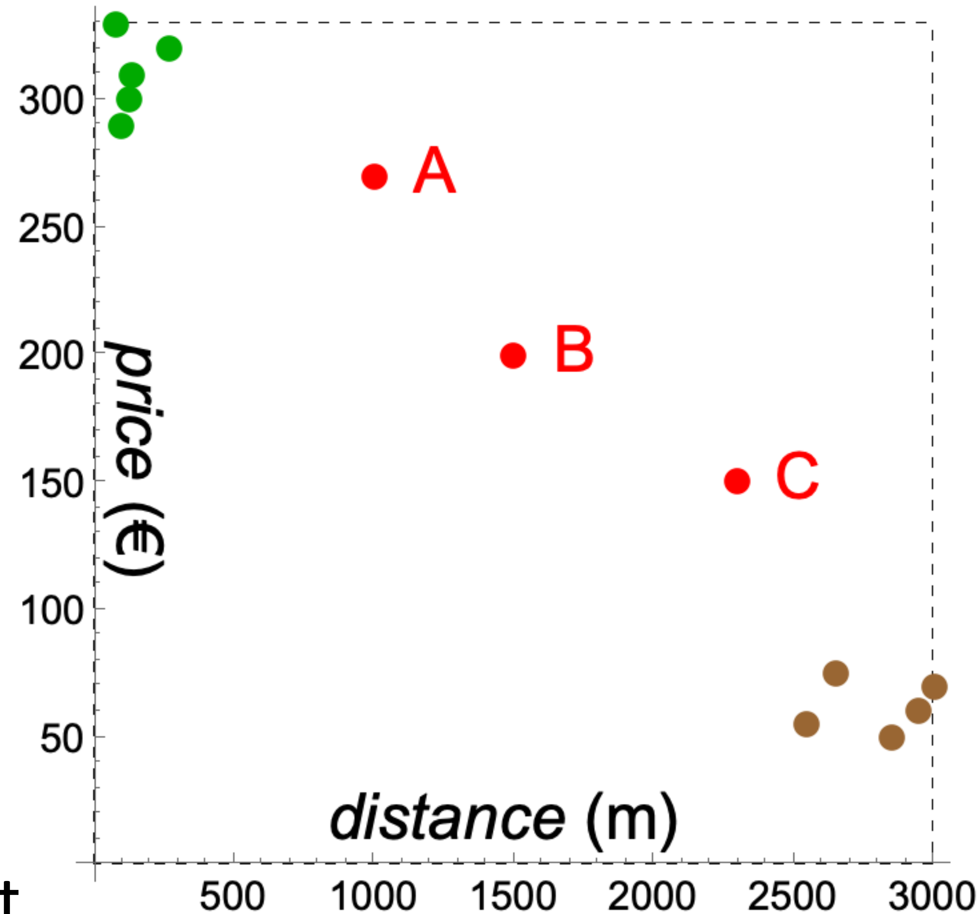
[Mouratidis et al., SIGMOD 2021]

- Pros:
  - OSS operators
    - output size may be easier than constraints on weights
  - Personalized with user preferences (weights)
  - Flexible (weights used loosely)
- Cons:
  - Too many size parameters (*k* and *m*)
    - When *k*=*m*, it's a standard linear top-k query ($\varrho = 0$)
  - Restricted to linear functions
    - The most common choice, but…

Beyond linear queries

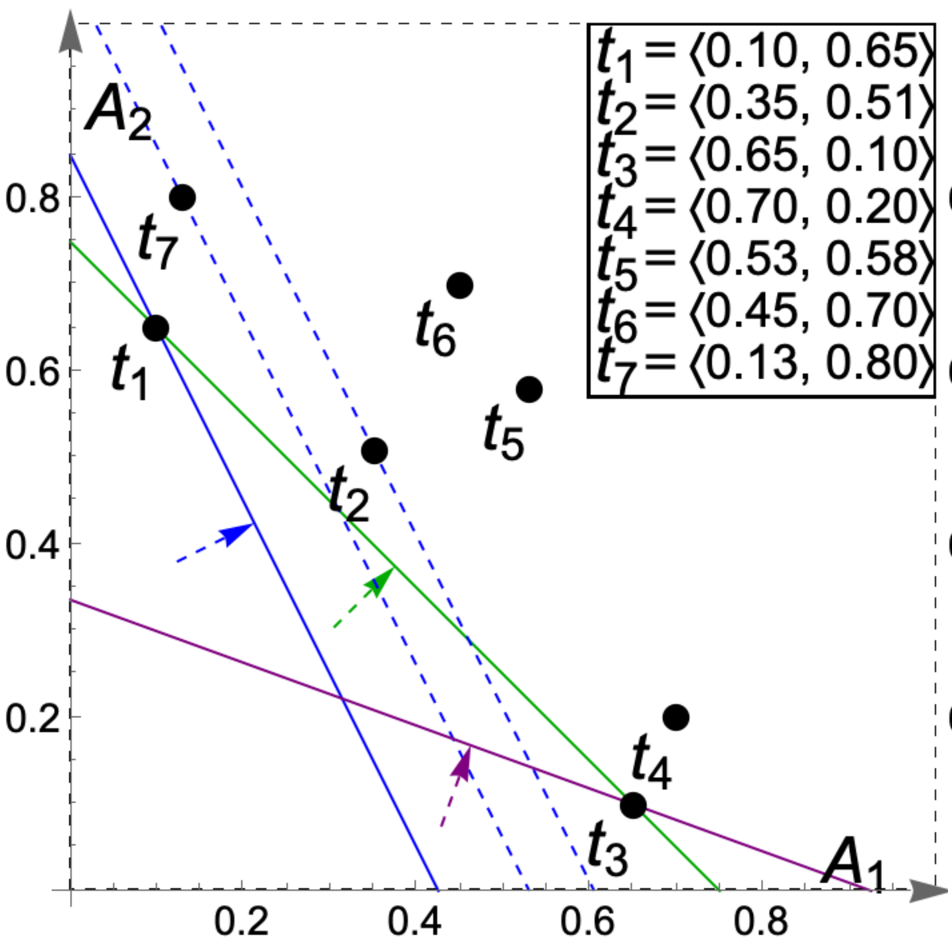
The limits of linear top-*k* queries

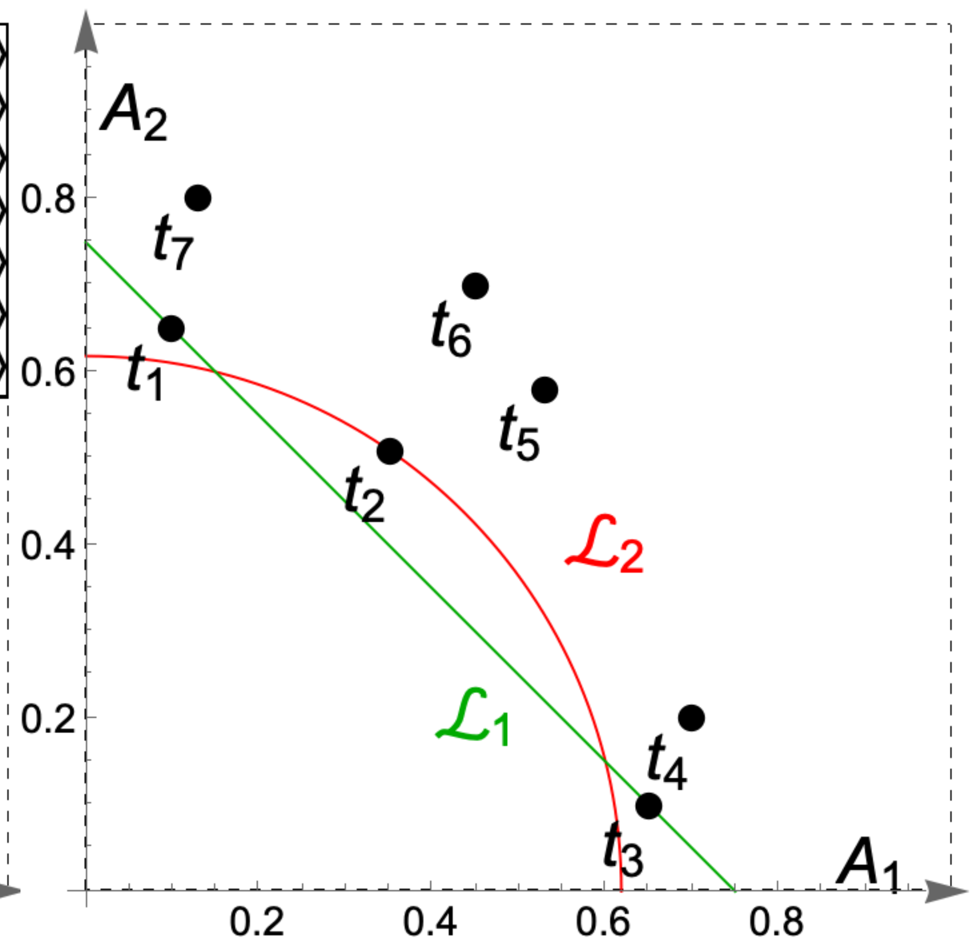
		weights		
	w_d	0.3	0.5	0.7
	w_p	0.7	0.5	0.3
hotel ranks	A	11	10	6
	B	7	8	7
	C	6	13	9



- No linear function ranks A, B or C as top
  - Interesting results but difficult to retrieve

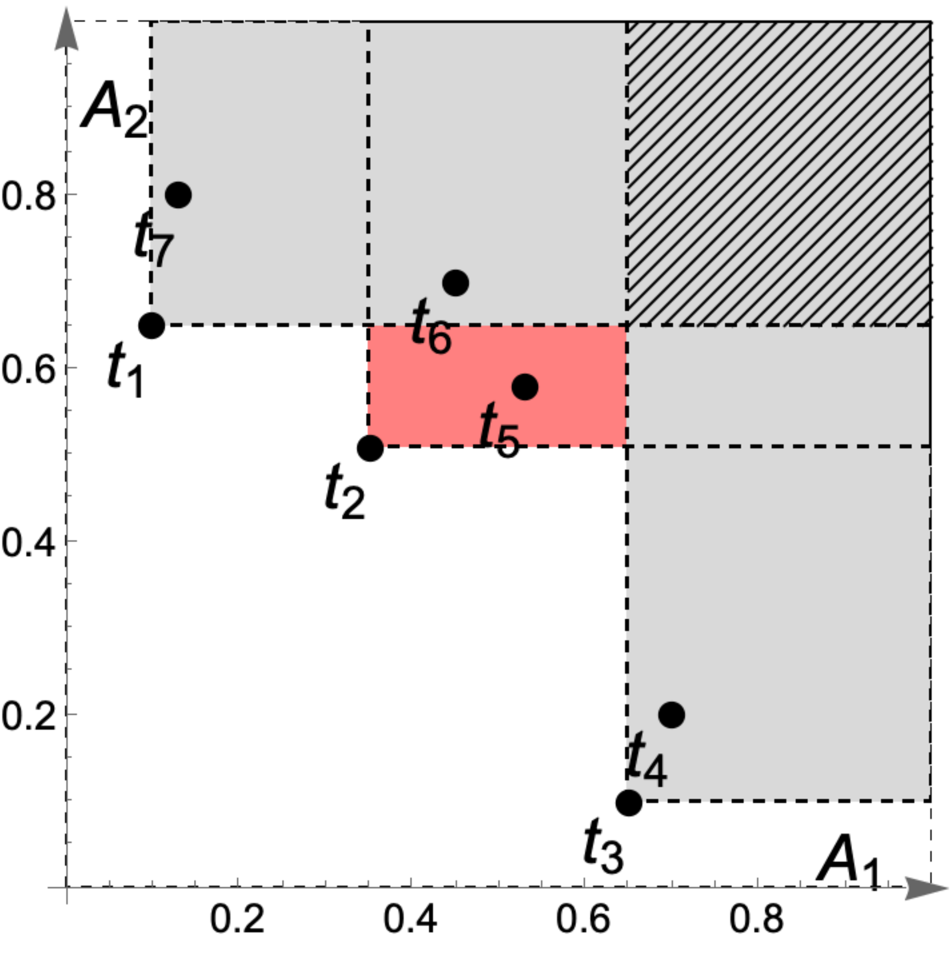
Indicators of difficulty



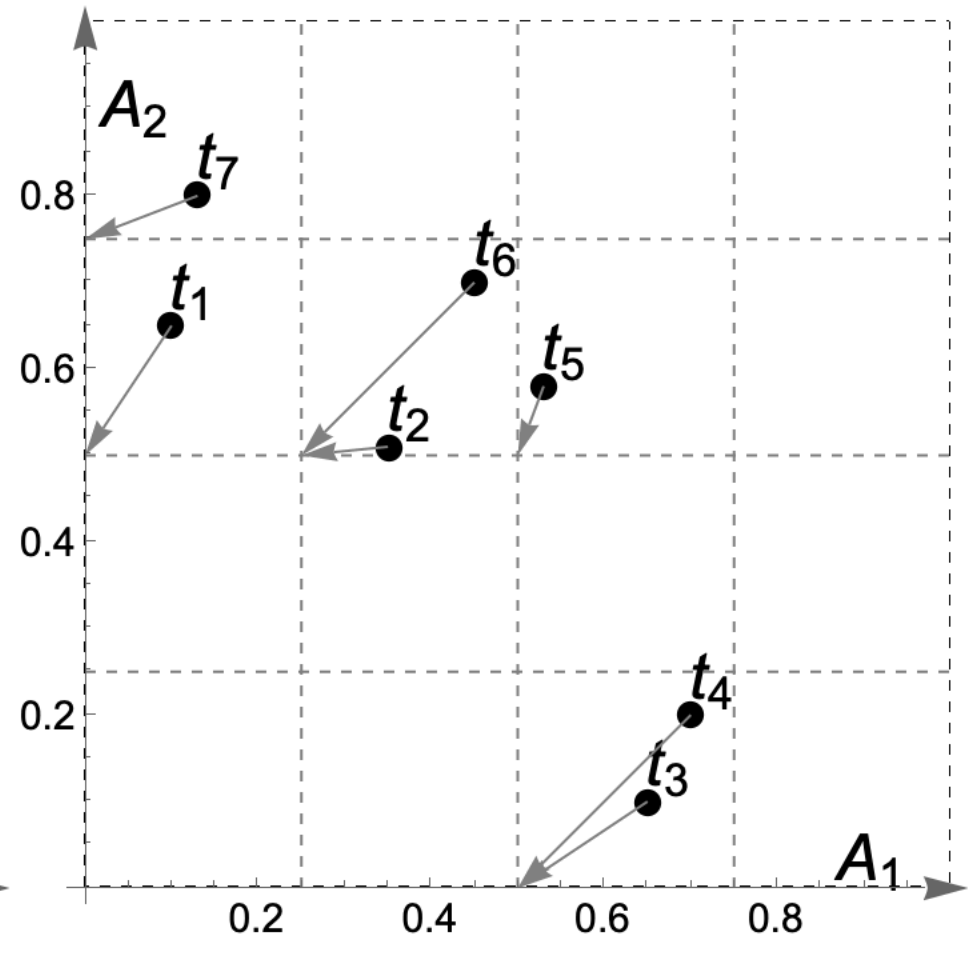


Best rank via linear query   Non-linearity to be top 1

Indicators of interest/robustness



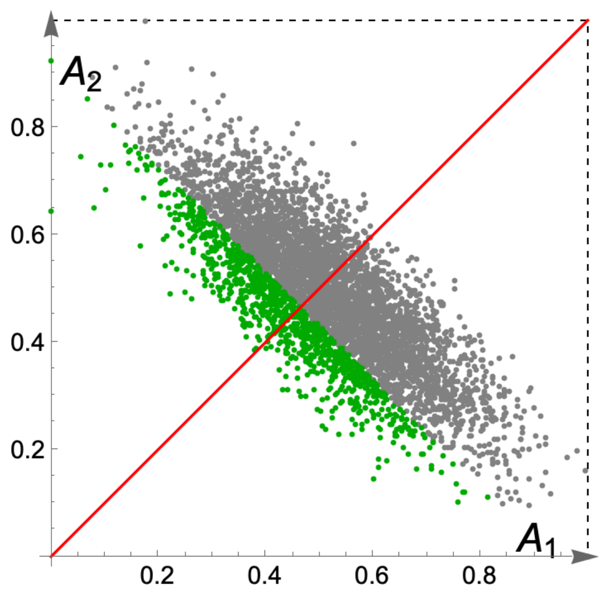
Exclusive volume



Grid resistance

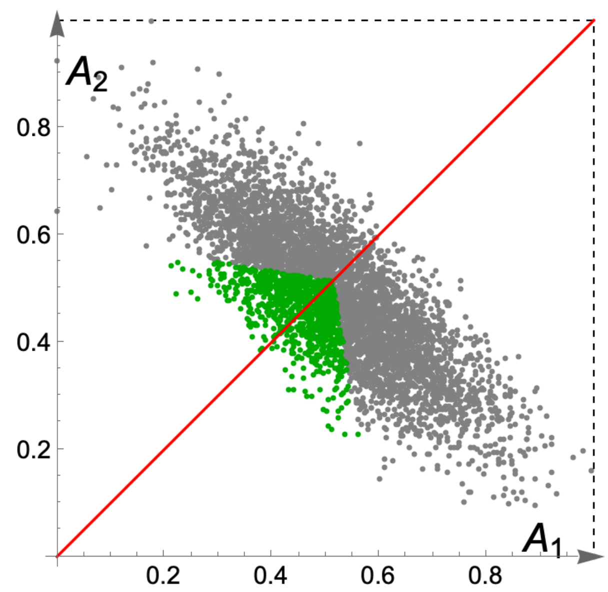
Balance and directional queries

- Weights induce a preference line
  - Balanced results are close to it
- Directional query = Linear query + balance



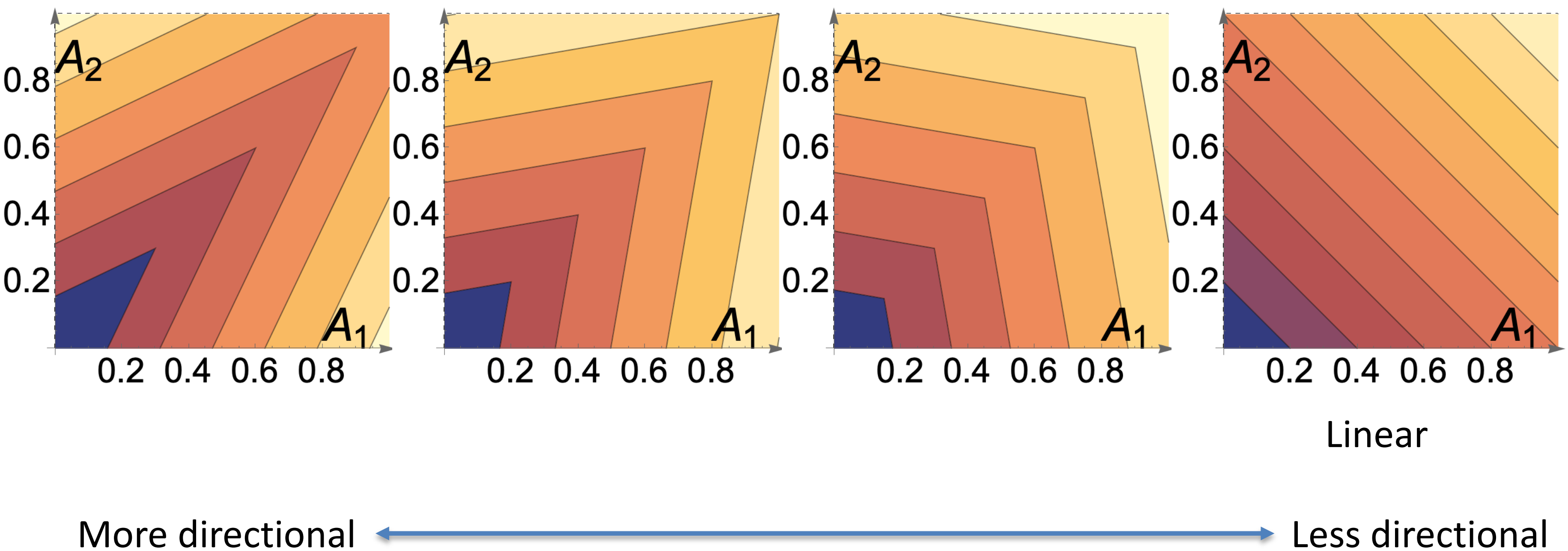
Linear query

Equal weights

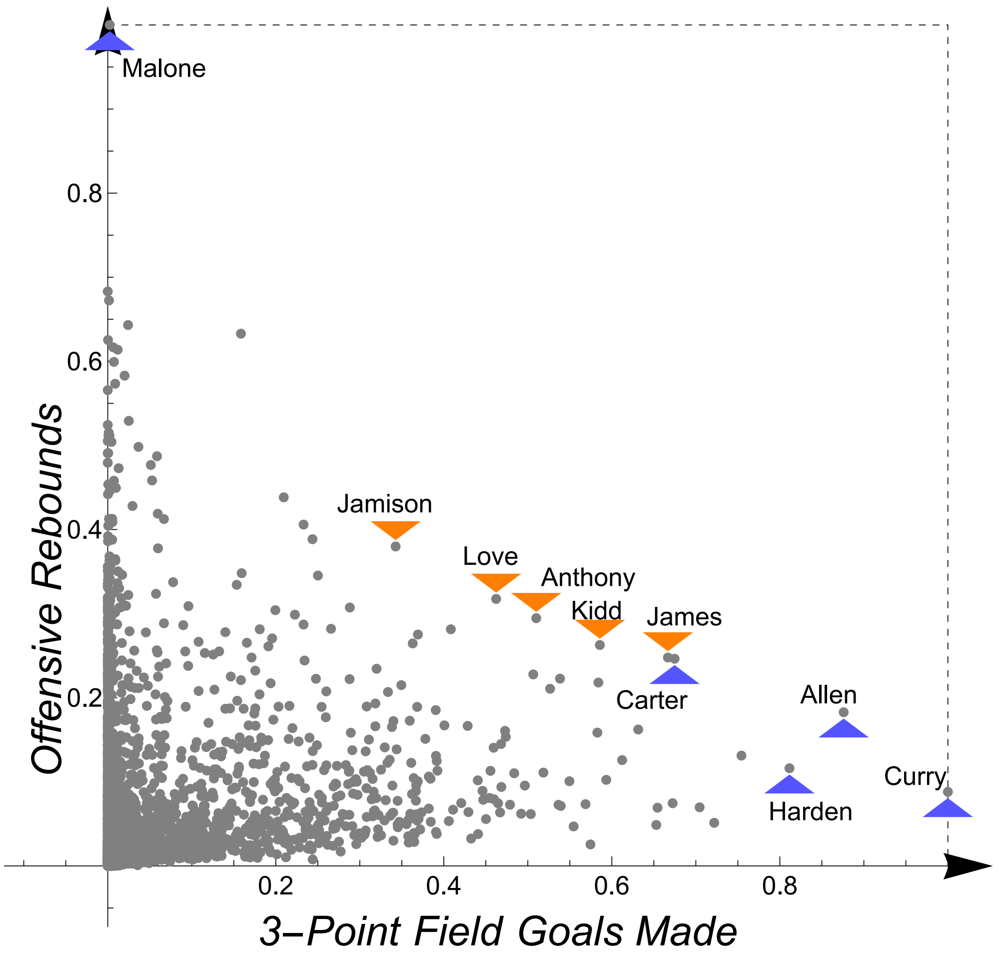


Directional query

The shape of directional queries



Directional vs linear queries



Features of Directional Queries

- Pros:
  - Increased balance of results
  - Increased overall exclusive volume
  - Increased overall grid resistance
  - Can retrieve all difficult tuples
  - Retains all standard advantages of top-*k* queries
- Cons:
  - What is the right mix of linear + balance?

Alternative approaches

Regret-Minimizing Sets

- For a scoring function f and a set D, let $Top_f(D) = max_{x \in D} f(x)$ (top score via f in D)
- The regret of $S \subset D$ is $(Top_f(D)\text{-}Top_f(S))/Top_f(D)$
- Find a set S of size k minimizing its maximum regret for any linear scoring function
- Pros:
  - may be used to add cardinality control to skylines
- Cons:
  - no preferences
  - only linear functions

Wrap up

Orthogonal aspects (not covered)

- Diversification of results
- Fairness of the selection process
  - Preserving the distribution of the input data
  - Changing scoring function / algorithm / data
- Uncertainty in the data
- Determining the true preferences of a user
- Point of view of the seller: which weights should I use so that my product becomes top?

Conclusions

- Ranking tools are still evolving towards the ultimate solution satisfying all desiderata
  - Preferences, output size control, efficiency, …
- Objective (and subjective) measures needed
  - Many datasets, no standard benchmark
  - User studies may come in handy

Thank you!

Main References

Historical papers

- Jean-Charles de Borda
  Mémoire sur les élections au scrutin. Histoire de l'Académie Royale des Sciences, Paris 1781
- Nicolas de Condorcet
  Essai sur l'application de l'analyse à la probabilité des décisions rendues à la pluralité des voix, 1785
- Kenneth J. Arrow
  A Difficulty in the Concept of Social Welfare. Journal of Political Economy. 58 (4): 328–346, 1950

Rank aggregation and ranking queries

- Ronald Fagin, Ravi Kumar, D. Sivakumar
  Efficient similarity search and classification via rank aggregation. SIGMOD Conference 2003: 301-312
- Ronald Fagin
  Combining Fuzzy Information from Multiple Systems. PODS 1996: 216-226
- Ronald Fagin
  Fuzzy Queries in Multimedia Database Systems. PODS 1998: 1-10
- Ronald Fagin, Amnon Lotem, Moni Naor
  Optimal Aggregation Algorithms for Middleware. PODS 2001

Skylines and k-Skybands

- Stephan Börzsönyi, Donald Kossmann, Konrad Stocker
  The Skyline Operator. ICDE 2001: 421-430
- Jan Chomicki, Parke Godfrey, Jarek Gryz, Dongming Liang
  Skyline with Presorting. ICDE 2003: 717-719
- Dimitris Papadias, Yufei Tao, Greg Fu, Bernhard Seeger
  Progressive skyline computation in database systems. ACM Trans. Database Syst. 30(1): 41-82 (2005)

Main References

Extensions of skylines: flexible skylines, ORD/ORU

- Paolo Ciaccia, Davide Martinenghi
  Reconciling Skyline and Ranking Queries. PVLDB 10(11): 1454-1465 (2017)
- Paolo Ciaccia, Davide Martinenghi
  FA + TA < FSA: Flexible Score Aggregation. CIKM 2018: 57-66
- Kyriakos Mouratidis, Bo Tang
  Exact Processing of Uncertain Top-k Queries in Multi-criteria Settings. Proc. VLDB Endow. 11(8): 866-879 (2018)
- Kyriakos Mouratidis, Keming Li, Bo Tang
  Marrying Top-k with Skyline Queries: Relaxing the Preference Input while Producing Output of Controllable Size. SIGMOD Conference 2021: 1317-1330

Regret queries

- Danupon Nanongkai, Atish Das Sarma, Ashwin Lall, Richard J. Lipton, Jun (Jim) Xu
  Regret-Minimizing Representative Databases. VLDB 2010.

Extensions of ranking queries: uncertainty, proximity, diversity

- Mohamed A. Soliman, Ihab F. Ilyas, Davide Martinenghi, Marco Tagliasacchi
  Ranking with uncertain scoring functions: semantics and sensitivity measures. SIGMOD Conference 2011: 805-816
- Davide Martinenghi, Marco Tagliasacchi
  Proximity Rank Join. PVLDB 3(1): 352-363 (2010)
- Piero Fraternali, Davide Martinenghi, Marco Tagliasacchi
  Top-k bounded diversification. SIGMOD Conference 2012: 421-432
- Akrivi Vlachou, Christos Doulkeridis, Yannis Kotidis, Kjetil Nørvåg
  Reverse top-k queries. ICDE 2010: 365-376