POLITECNICO DI MILANO

DIPARTIMENTO DI
ELETTRONICA,
INFORMAZIONE
E BIOINGEGNERIA

Flexible Score Aggregation

Davide Martinenghi

Bolzano, November 12, 2018

Outline

- Finding interesting objects in a dataset
  - Multi-objective optimization
- Historical perspective
  - Rank aggregation
  - Classical approaches and their limitations
- Combining opaque rankings
  - Median ranking with MedRank
- Ranking queries
  - Fagin’s Algorithm and Threshold Algorithm
- Skyline queries
  - Block-Nested-Loop and Sort-Filter-Skyline Algorithms
- Restricted skylines
  - Reconciling Ranking Queries and Skyline Queries
  - Reconciling Fagin’s Algorithm and Threshold Algorithm

Finding interesting objects in a dataset

Multi-objective optimization

- Simultaneous optimization of different criteria
  - E.g., different attributes of objects in a dataset
- Main scenarios:
  - Combination of user preferences expressed by multi-criteria queries
    - Example: ranking restaurants by combining criteria about culinary preference, driving distance, stars, …
  - Meta-search
    - For a given query, combine the results from different search engines
  - Nearest neighbor problem (e.g., similarity search)
    - Given a database D of n points in some metric space, and a query q in the same space, find the point (or the k points) in D closest to q

Multi-objective optimization

- Simultaneous optimization of different criteria
  - E.g., different attributes of objects in a dataset
- Main approaches:
  - Ranking queries
    - Top k objects according to a given scoring function
  - Skyline queries
    - Set of non-dominated objects
  - Lexicographic queries
    - strict priority among different attributes
    - even the smallest difference in the most important attribute can never be compensated by the other attributes

Historical perspective

Rank aggregation: the original problem

[Borda, 1770][Marquis de Condorcet, 1785]

- Rank aggregation is the problem of combining several ranked lists of objects in a robust way to produce a single consensus ranking of the objects

Rank aggregation: the original problem

[Borda, 1770][Marquis de Condorcet, 1785]

- Rank aggregation is the problem of combining several ranked lists of objects in a robust way to produce a single consensus ranking of the objects
  - Old problem (social choice theory) with lots of open challenges
  - Given: *n* candidates, *m* voters

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?
  - No visible score assigned to candidates, only ranking
- Who is the best candidate? (point of view of the buyer)

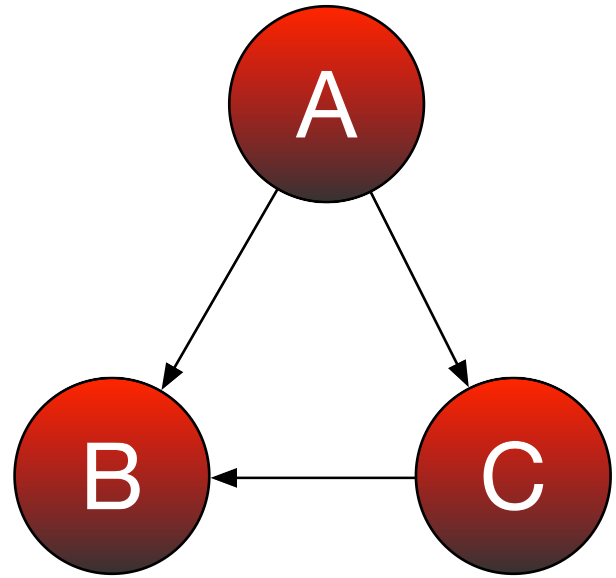
Borda's and Condorcet's proposals

- Borda's proposal
  - Election by order of merit
    - First place → 1 point
    - Second place → 2 points
    - ...
    - n-th place → n points

  - Candidate's score: sum of points
- Borda winner: lowest scoring candidate
- Condorcet winner:
  - A candidate who defeats every other candidate in pairwise majority rule election

Borda winner <> Condorcet winner

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



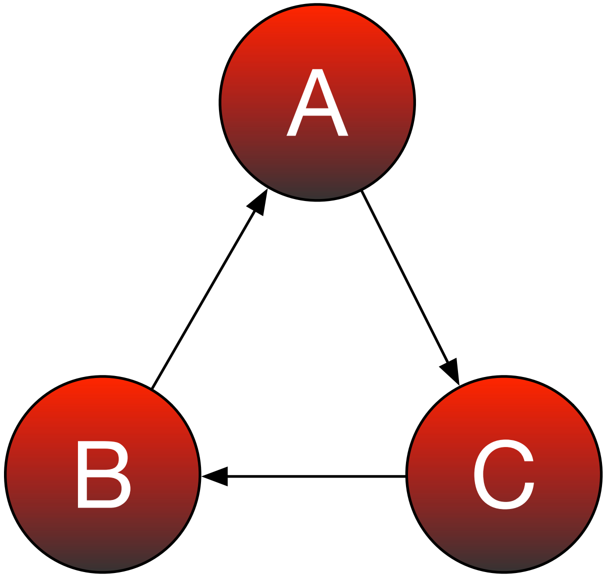
- Borda scores:
  - A: 1x6+3x4 = 18
  - B: 3x6+2x4 = 26
  - C: 2x6+1x4 = 16 ← Borda winner

- Condorcet's criterion:
  - A beats both B and C in pairwise majority
  - A is Condorcet's winner

Condorcet's paradox

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

- Condorcet's winner may not exist
  - Cyclic preferences



Main approaches to rank aggregation

[Arrow, 1950]

- Axiomatic approach
  - Desiderata of aggregation formulated as “axioms”
  - By the classical result of Arrow, a small set of natural requirements cannot be simultaneously achieved by any nontrivial aggregation function
  - Arrow’s paradox: no rank-order electoral system can be designed that always satisfies these three "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

Main approaches to rank aggregation

- Metric approach
  - Finding a new ranking R whose total distance to the initial rankings $R_1, \ldots, R_n$ is minimized

- Several ways to define a distance between rankings
  - Kendall tau distance $K(R_1, R_2)$, defined as the number of exchanges in a bubble sort to convert R_1 to R_2
  - Spearman's footrule distance $F(R_1, R_2)$, which adds up the distance between the ranks of the same item in the two rankings

- Finding an exact solution is
  - NP-hard with Kendall tau
  - PTIME with Spearman's footrule
  - It is known that

$$K(R_1, R_2) \leq F(R_1, R_2) \leq 2\, K(R_1, R_2)$$

  - $F(R_1, R_2)$ admits efficient approximations (e.g., median ranking)

Combining opaque rankings

Combining opaque rankings

[Fagin, Kuvar, Sivakumar, SIGMOD 2003]

- Techniques using only the position of the elements in the ranking (no other associated score)
- We review MedRank, proposed by Fagin et al.
  - Based on the notion of median, it provides a(n approximation of) Footrule-optimal aggregation

Input: *m* rankings of *n* elements

Output: the top *k* elements according to median ranking

1. Use sorted accesses in each ranking, one element at a time, until there are *k* elements that occur in more than *m*/2 rankings
2. These are the top *k* elements

- MedRank is instance-optimal
  - Among the algorithms that access the rankings in sorted order, this is the best possible algorithm (to within a constant factor) on every input instance

An aside: instance optimality

- A form of optimality aimed at when standard optimality is unachievable
- Formally:
  - Let **A** be a family of algorithms
  - Let **I** be a set of problem instances
  - Let *c* be a cost metric applied to an algorithm-instance pair
  - Algorithm A^* is instance-optimal wrt. **A** and **I** for the cost metric *c* if there exist constants k_1 and k_2 such that, for all $A \in \mathbf{A}$ and $I \in \mathbf{I}$,

$$c(A^*, I) \leq k_1 \cdot c(A, I) + k_2$$

MedRank example: hotels in Paris

price	rating	distance
Ibis	Crillon	Le Roch
Etap	Novotel	Lodge In
Novotel	Sheraton	Ritz
Mercure	Hilton	Lutetia
Hilton	Ibis	Novotel
Sheraton	Ritz	Sheraton
Crillon	Lutetia	Mercure
…	…	

Top 3 hotels	Median rank

- Strategy:
  - Make one sorted access at a time in each ranking
  - Look for hotels that appear in at least 2 rankings

NB: price, rating and distance are opaque, only the position matters

MedRank example: hotels in Paris

price	rating	distance
Ibis	Crillon	Le Roch
Etap	Novotel	Lodge In
Novotel	Sheraton	Ritz
Mercure	Hilton	Lutetia
Hilton	Ibis	Novotel
Sheraton	Ritz	Sheraton
Crillon	Lutetia	Mercure
…	…	

Top 3 hotels	Median rank

- Strategy:
  - Make one sorted access at a time in each ranking
  - Look for hotels that appear in at least 2 rankings

NB: price, rating and distance are opaque, only the position matters

MedRank example: hotels in Paris

price	rating	distance
Ibis	Crillon	Le Roch
Etap	Novotel	Lodge In
Novotel	Sheraton	Ritz
Mercure	Hilton	Lutetia
Hilton	Ibis	Novotel
Sheraton	Ritz	Sheraton
Crillon	Lutetia	Mercure
…	…	

Top 3 hotels	Median rank

- Strategy:
  - Make one sorted access at a time in each ranking
  - Look for hotels that appear in at least 2 rankings

NB: price, rating and distance are opaque, only the position matters

MedRank example: hotels in Paris

price	rating	distance
Ibis	Crillon	Le Roch
Etap	Novotel	Lodge In
Novotel	Sheraton	Ritz
Mercure	Hilton	Lutetia
Hilton	Ibis	Novotel
Sheraton	Ritz	Sheraton
Crillon	Lutetia	Mercure
…	…	

Top 3 hotels	Median rank
Novotel	median{2,3,?}=3

- Strategy:
  - Make one sorted access at a time in each ranking
  - Look for hotels that appear in at least 2 rankings

NB: price, rating and distance are opaque, only the position matters

MedRank example: hotels in Paris

price	rating	distance
Ibis	Crillon	Le Roch
Etap	Novotel	Lodge In
Novotel	Sheraton	Ritz
Mercure	Hilton	Lutetia
Hilton	Ibis	Novotel
Sheraton	Ritz	Sheraton
Crillon	Lutetia	Mercure
…	…	

Top 3 hotels	Median rank
Novotel	median{2,3,?}=3

- Strategy:
  - Make one sorted access at a time in each ranking
  - Look for hotels that appear in at least 2 rankings

NB: price, rating and distance are opaque, only the position matters

MedRank example: hotels in Paris

price	rating	distance
Ibis	Crillon	Le Roch
Etap	Novotel	Lodge In
Novotel	Sheraton	Ritz
Mercure	Hilton	Lutetia
Hilton	Ibis	Novotel
Sheraton	Ritz	Sheraton
Crillon	Lutetia	Mercure
…	…	

Top 3 hotels	Median rank
Novotel	median{2,3,5}=3
Hilton	median{4,5,?}=5
Ibis	median{1,5,?}=5

When the median ranks are all distinct (unlike here), we have the Footrule-optimal aggregation

- Strategy:
  - Make one sorted access at a time in each ranking
  - Look for hotels that appear in at least 2 rankings

NB: price, rating and distance are opaque, only the position matters

Ranking queries

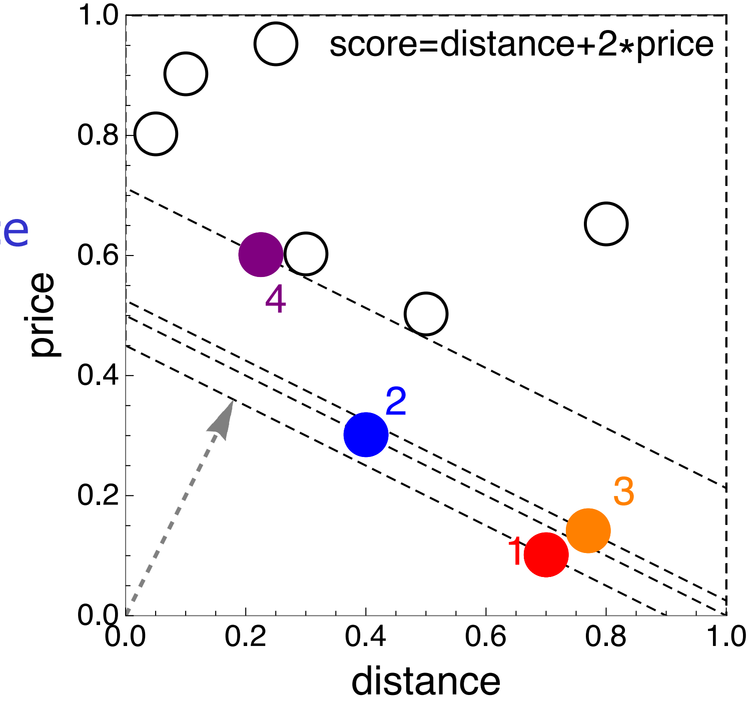
Ranking queries with a scoring function

- Several studies consider rankings where the objects, besides the position, also include a score (usually in the [0, 1] interval)
- Traditionally, two ways of accessing data:
  - Sorted (sequential) access: access, one by one, the next element (together with its score) in a ranked list, starting from top
  - Random access: given an element, retrieve its score (position in the ranked list or other associated value)
- Main interest in the top k elements of the aggregation
  - Need for algorithms that quickly obtain the top results
  - … without having to read each ranking in its entirety
- Several algorithms developed in the literature to minimize the accesses when determining the top k elements
  - Main works by Fagin et al.

Ranking queries

- Objects are ranked by using a scoring function
  - Weights may express relative importance of attributes
  - The problem reduces to single-objective optimization
  - Typically the function is monotone
- Algorithmic focus is on different kinds of access to data and optimality wrt. number of accesses

score=distance+2*price



Fagin's Algorithm (FA, also known as A0)

[Fagin, PODS 1998]

Input: a monotone query combining rankings $R_1, \ldots, R_n$

Output: the top k <object, score> pairs

1. Extract the same number of objects by sorted accesses in each ranking until there are at least k objects in common
2. For each extracted object, compute its overall score by making random accesses wherever needed
3. Among these, output the k objects with the best overall score

- Complexity is sub-linear in the number N of objects
  - Proportional to the square root of N when combining two rankings
  - The stopping criterion is independent of the scoring function
  - Not instance-optimal

Example cont'd: hotels in Paris

Hotels	Cheapness	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

- Query: hotels with best price and rating
  - Scoring function: 0.5*cheapness+0.5*rating
- Strategy:
  - Make one sorted access at a time in each ranking
  - Look for hotels that appear in both rankings

Example cont'd: hotels in Paris

Hotels	Cheapness	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

- Query: hotels with best price and rating
  - Scoring function: 0.5*cheapness+0.5*rating
- Strategy:
  - Make one sorted access at a time in each ranking
  - Look for hotels that appear in both rankings

Example cont'd: hotels in Paris

Hotels	Cheapness	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

- Query: hotels with best price and rating
  - Scoring function: 0.5*cheapness+0.5*rating
- Strategy:
  - Make one sorted access at a time in each ranking
  - Look for hotels that appear in both rankings

Example cont'd: hotels in Paris

Hotels	Cheapness	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

- Query: hotels with best price and rating
  - Scoring function: 0.5*cheapness+0.5*rating
- Strategy:
  - Make one sorted access at a time in each ranking
  - Look for hotels that appear in both rankings

Example cont'd: hotels in Paris

Hotels	Cheapness	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

- Query: hotels with best price and rating
  - Scoring function: 0.5*cheapness+0.5*rating
- Strategy:
  - Make one sorted access at a time in each ranking
  - Look for hotels that appear in both rankings

Example cont'd: hotels in Paris

Hotels	Cheapness	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

- Query: hotels with best price and rating
  - Scoring function: 0.5*cheapness+0.5*rating
- Strategy:
  - Make one sorted access at a time in each ranking
  - Look for hotels that appear in both rankings

Example cont'd: hotels in Paris

Hotels	Cheapness	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

- Query: hotels with best price and rating
  - Scoring function: 0.5*cheapness+0.5*rating
- Strategy:
  - Now complete the score with random accesses

Threshold Algorithm (TA)

[Fagin, Lotem, Naor, PODS 2001]

Input: a monotone query combining rankings $R_1, \ldots, R_n$

Output: the top k <object, score> pairs

1. Do a sorted access in parallel in each ranking R_i
2. For each object o, do random accesses in the other rankings R_j, thus extracting score s_j
3. Compute overall score $f(s_1, \ldots, s_n)$. If the value is among the k highest seen so far, remember o
4. Let s_{Li} be the last score seen under sorted access for R_i
5. Define threshold $T=f(s_{L1}, \ldots, s_{Ln})$
6. If the score of the k-th object is worse than T, go to step 1
7. Return the current top-k objects

- TA is instance-optimal among all algorithms that use random and sorted accesses (FA is not)
  - The stopping criterion depends on the scoring function
- The authors of TA received the Gödel prize in 2014 for the design of innovative algorithms

Example cont'd: hotels in Paris with TA

Hotels	Cheapness	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 price and rating
  - Scoring function: 0.5*cheapness+0.5*rating

Example cont'd: hotels in Paris with TA

Hotels	Cheapness	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: τ =(.92,.9)

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

Example cont'd: hotels in Paris with TA

Hotels	Cheapness	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: τ =(.91,.9)

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

Example cont'd: hotels in Paris with TA

Hotels	Cheapness	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 price and rating
  - Scoring function: 0.5*cheapness+0.5*rating
- Strategy:
  - Stop when the score of the *k*-th hotel is no worse than the threshold

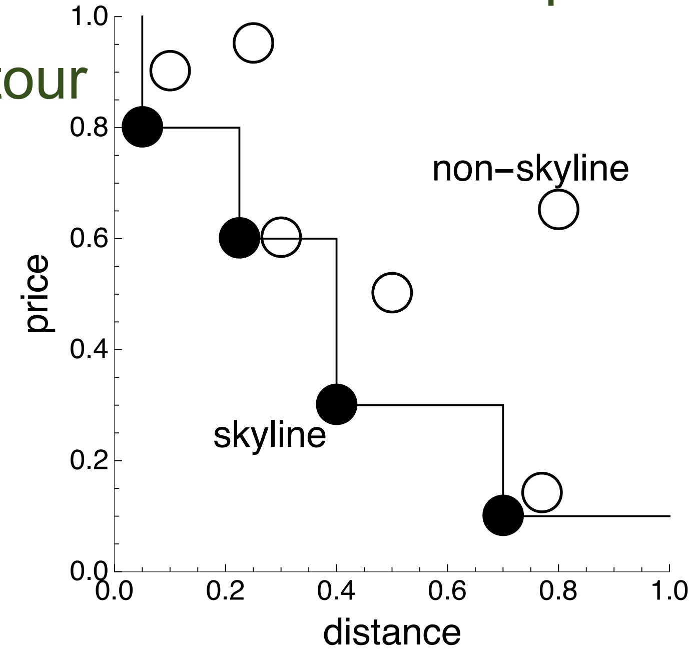
Ranking queries – main aspects

- Effective in identifying the best objects
  - Wrt. a specific scoring function
- Excellent control of the cardinality of the result
  - k is an input parameter of a top-k query
- For a user, it is difficult to specify a scoring function
  - E.g., the weights of a weighted sum
- Computation is very efficient
  - E.g., $N \log k$ for local, unordered datasets of N elements
  - Many different results for different settings
- Easy to express the relative importance of attributes

Skyline queries

Skylines

- Find good objects according to several different perspectives
  - e.g., attribute values $A_1,\ldots,A_d$
  - Based on the notion of dominance
- Tuple t dominates tuple s, indicated $t \prec s$, iff
  - $\forall i.\ 1 \le i \le d \rightarrow t[A_i] \le s[A_i]$ (*t* is nowhere worse than *s*)
  - $\exists j.\ 1 \le j \le d \wedge t[A_j] < s[A_j]$ (and better at least once)
- The skyline of a relation is the set of its non-dominated tuples
- In 2D, the shape resembles the contour of the dataset (hence the name)



Skylines – Block Nested Loop (BNL)

[Börzsönyi et al., ICDE 2001]

Input: a dataset D of multi-dimensional points
Output: the skyline of D

1. Let $W = \emptyset$
2. for every point p in D
3. if p not dominated by any point in W
4. remove from W the points dominated by p
5. add p to W
6. return W

- Computation is $O(n^2)$ where $n=|D|$
- Very inefficient for large datasets

Skylines – Sort-Filter-Skyline (SFS)

[Chomicki et al., ICDE 2003]

Input: a dataset D of multi-dimensional points
Output: the skyline of D

```
1.  Let S = D sorted by a monotone function of D's attributes
2.  Let W = Ø
3.  for every point p in S
4.      if p not dominated by any point in W
5.          add p to W
6.  return W
```

- Pre-sorting pays off for large datasets, thus SFS performs much better than BNL

Example cont'd: hotels in Paris with SFS

Hotels	Cost	Reviews
Ibis	.08	.3
Novotel	.15	.1
Hilton	.175	.3
Crillon	.25	.1
Sheraton	.2	.2

- Dataset
  - (low values are good)

Example cont'd: hotels in Paris with SFS

Hotels	Cost	Reviews
Novotel	.15	.1
Crillon	.25	.1
Ibis	.08	.3
Sheraton	.2	.2
Hilton	.175	.3

- Sorted dataset
  - E.g., by Cost + Reviews

Example cont'd: hotels in Paris with SFS

Hotels	Cost	Reviews
Novotel	.15	.1
Crillon	.25	.1
Ibis	.08	.3
Sheraton	.2	.2
Hilton	.175	.3

- Sorted dataset
  - Add if not dominated by any point in the window

Hotels	Cost	Reviews
Novotel	.15	.1

- Window

Example cont'd: hotels in Paris with SFS

Hotels	Cost	Reviews
Novotel	.15	.1
Crillon	.25	.1
Ibis	.08	.3
Sheraton	.2	.2
Hilton	.175	.3

- Sorted dataset
  - Add if not dominated by any point in the window

Hotels	Cost	Reviews
Novotel	.15	.1

- Window

Example cont'd: hotels in Paris with SFS

Hotels	Cost	Reviews
Novotel	.15	.1
Crillon	.25	.1
Ibis	.08	.3
Sheraton	.2	.2
Hilton	.175	.3

- Sorted dataset
  - Add if not dominated by any point in the window

Hotels	Cost	Reviews
Novotel	.15	.1
Ibis	.08	.3

- Window

Example cont'd: hotels in Paris with SFS

Hotels	Cost	Reviews
Novotel	.15	.1
Crillon	.25	.1
Ibis	.08	.3
Sheraton	.2	.2
Hilton	.175	.3

- Sorted dataset
  - Add if not dominated by any point in the window

Hotels	Cost	Reviews
Novotel	.15	.1
Ibis	.08	.3

- Window

Example cont'd: hotels in Paris with SFS

Hotels	Cost	Reviews
Novotel	.15	.1
Crillon	.25	.1
Ibis	.08	.3
Sheraton	.2	.2
Hilton	.175	.3

- Sorted dataset
  - Add if not dominated by any point in the window

Hotels	Cost	Reviews
Novotel	.15	.1
Ibis	.08	.3

- Window

Example cont'd: hotels in Paris with SFS

Hotels	Cost	Reviews
Novotel	.15	.1
Crillon	.25	.1
Ibis	.08	.3
Sheraton	.2	.2
Hilton	.175	.3

- Sorted dataset
  - Add if not dominated by any point in the window

Hotels	Cost	Reviews
Novotel	.15	.1
Ibis	.08	.3

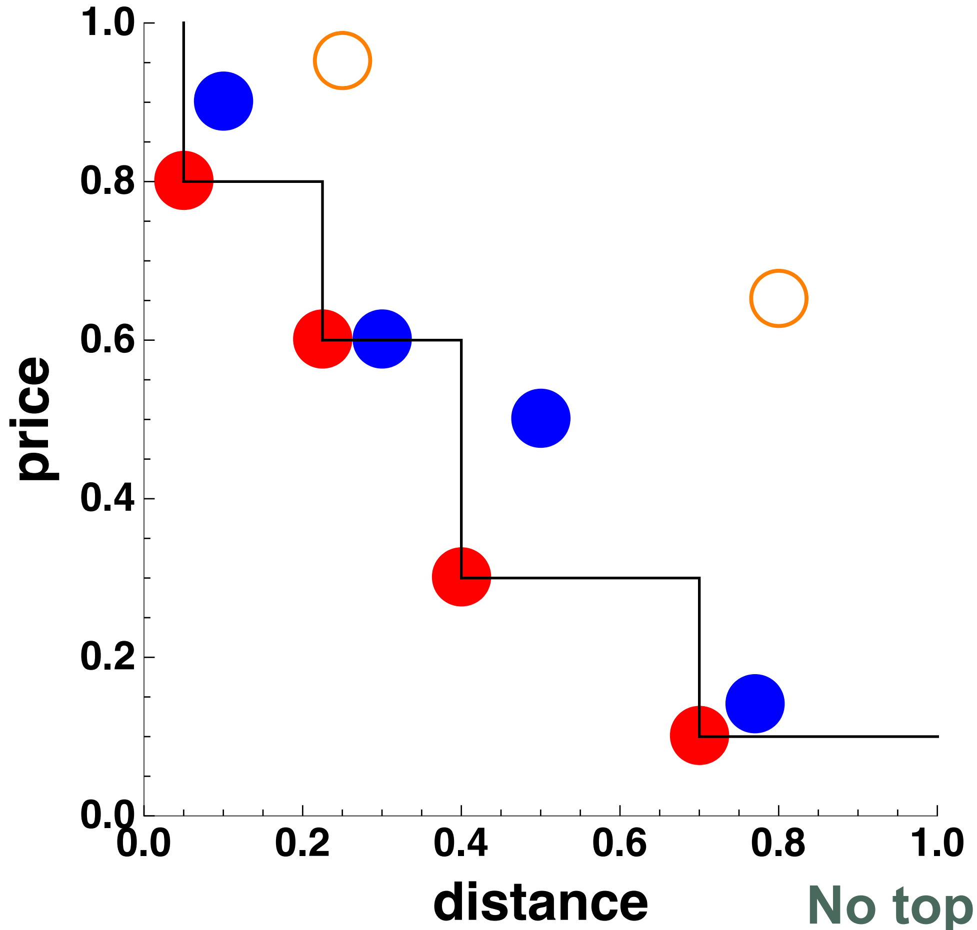
- This is the skyline

Skylines – main aspects

- Effective in identifying potentially interesting objects if nothing is known about the preferences of a user
- Very simple to use (no parameters needed!)
- Too many objects returned for large, anti-correlated datasets
- Computation is essentially quadratic in the size of the dataset (and thus not so efficient)
- Agnostic wrt. known user preferences (e.g., price is more important than distance)
- Extension: k-skyband = set of tuples dominated by < k tuples
    - Every top-k result set is contained in the k-skyband

Comparing different approaches

Example: skyline/k-skyband query

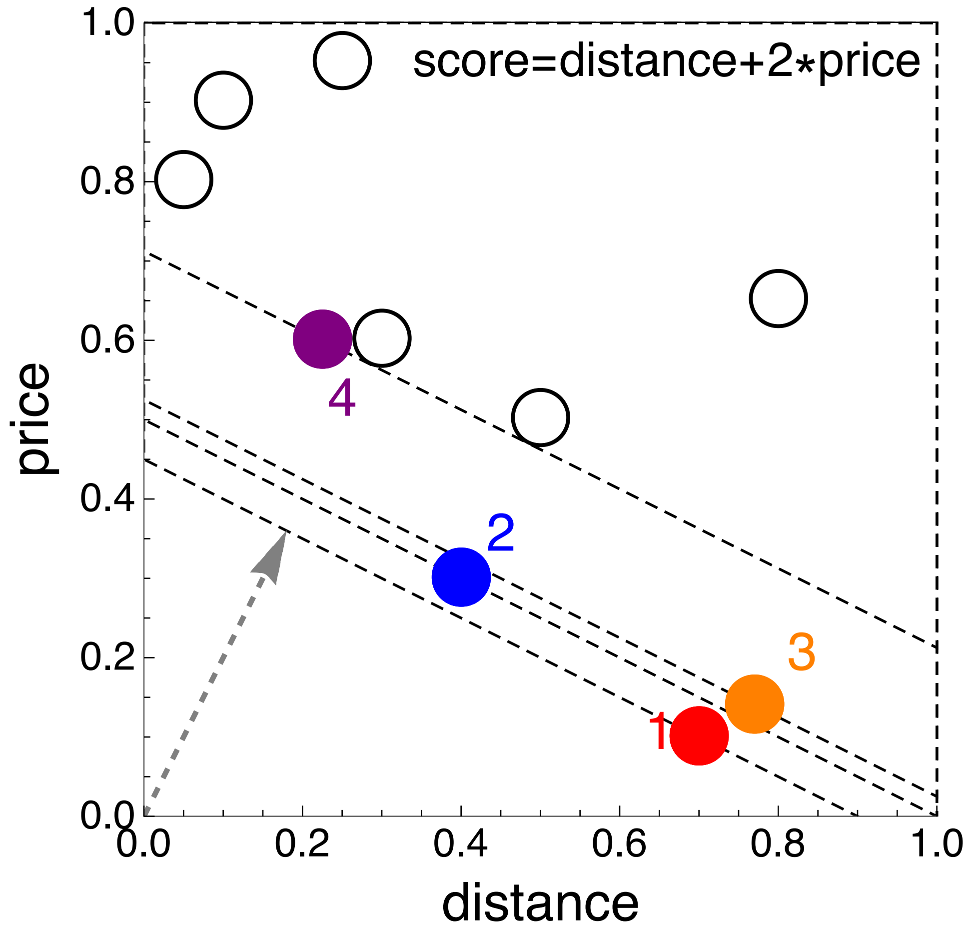


skyline

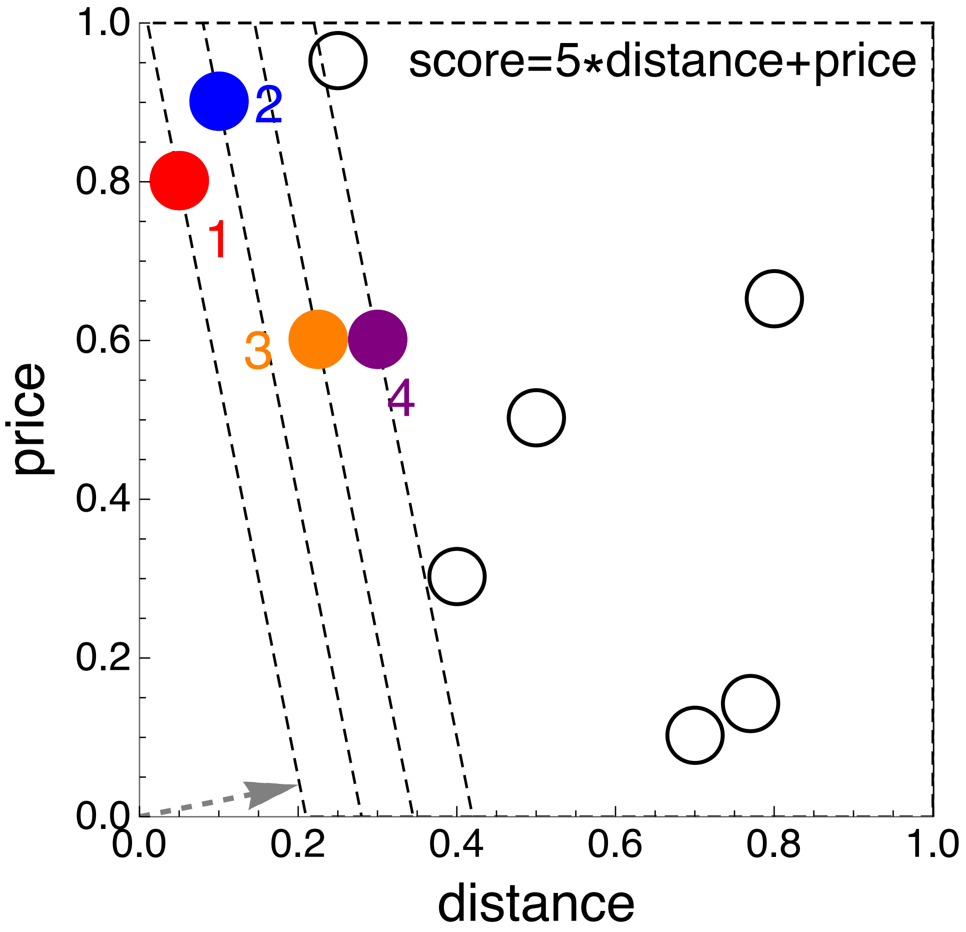
2-skyband = 3-skyband

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

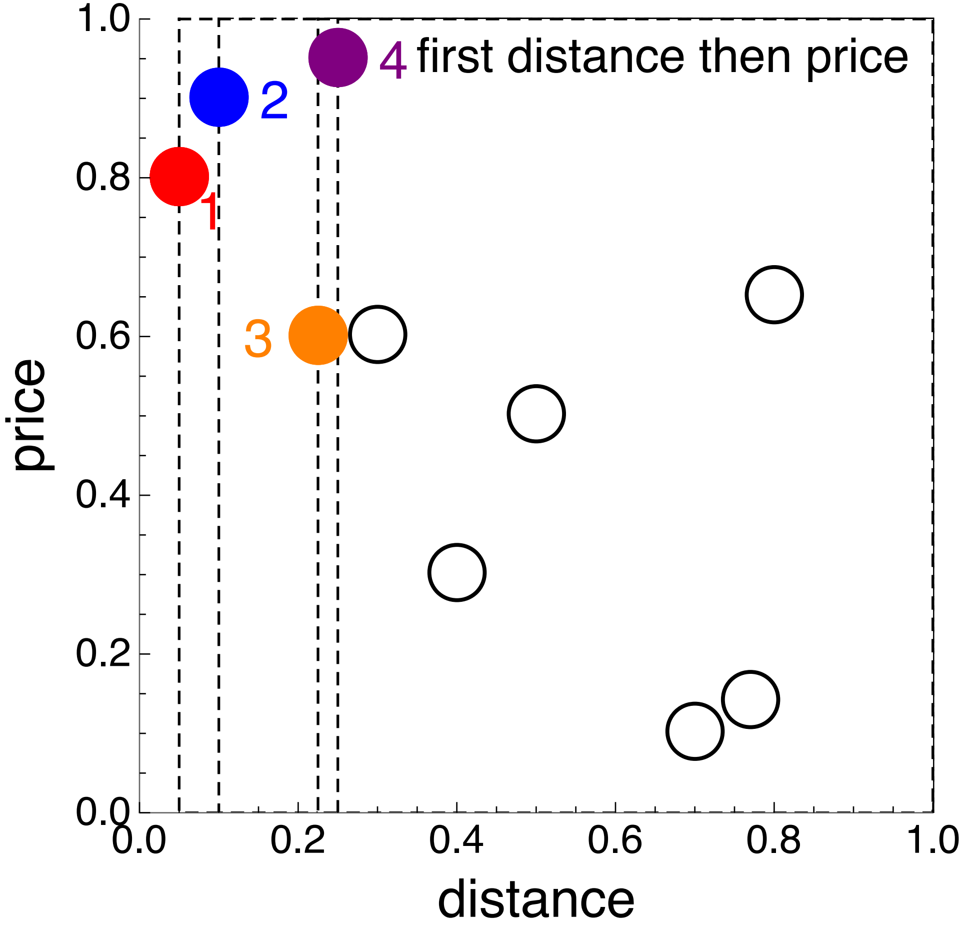
Example: ranking query



Example: another ranking query



Example: lexicographic query favoring price



Comparing different approaches

	Ranking queries	Lexicographic approach	Skyline queries
Simplicity	No	Yes	Yes
Overall view of interesting results	No	No	Yes
Control of cardinality	Yes	Yes	No
Trade-off among attributes	Yes	No	No
Relative importance of attributes	Yes	Yes	No

Restricted skylines

Restricted skylines

[Ciaccia and Martinenghi, VLDB 2017]

- A reconciliation between skyline and ranking queries
  - Take into account different importance of different attributes
    - no strict priority as in the lexicographic approach
- Extreme cases:
  - Skyline queries: dominance across all monotone functions M
  - Ranking queries: one single scoring function $f \in M$
- Idea: consider a family of scoring functions $F \subseteq M$ to characterize the interesting objects
    - may be specified by means of constraints on the weights

F-dominance:
- Tuple t F-dominates tuple $s \neq t$, denoted by $t \prec_F s$, iff $\forall f \in F.\ f(t) \leq f(s)$, with at least one strict inequality

- When $F \equiv M$ then $\prec_F \equiv \prec$

ND and PO

- Skyline as non-dominated tuples:

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

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

$$\text{ND}(r; \mathcal{F}) = \{t \in r \mid \nexists s \in r.\ \ s \prec_{\mathcal{F}} t\}$$

- Skyline as tuples optimal wrt a monotone scoring function:

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

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

$$\text{PO}(r; \mathcal{F}) = \{t \in r \mid \exists f \in \mathcal{F}.\ \forall s \in r.\ \ s \neq t \rightarrow f(t) < f(s)\}$$

- Extreme cases:
  - *F*≡*M* → ND=PO=Sky
  - *F*={ *f* } → ND≈PO≈top-1 query wrt. scoring function *f*

ND and PO

- *k*-Skyband as non-dominated tuples:
  - Tuples dominated by less than *k* tuples

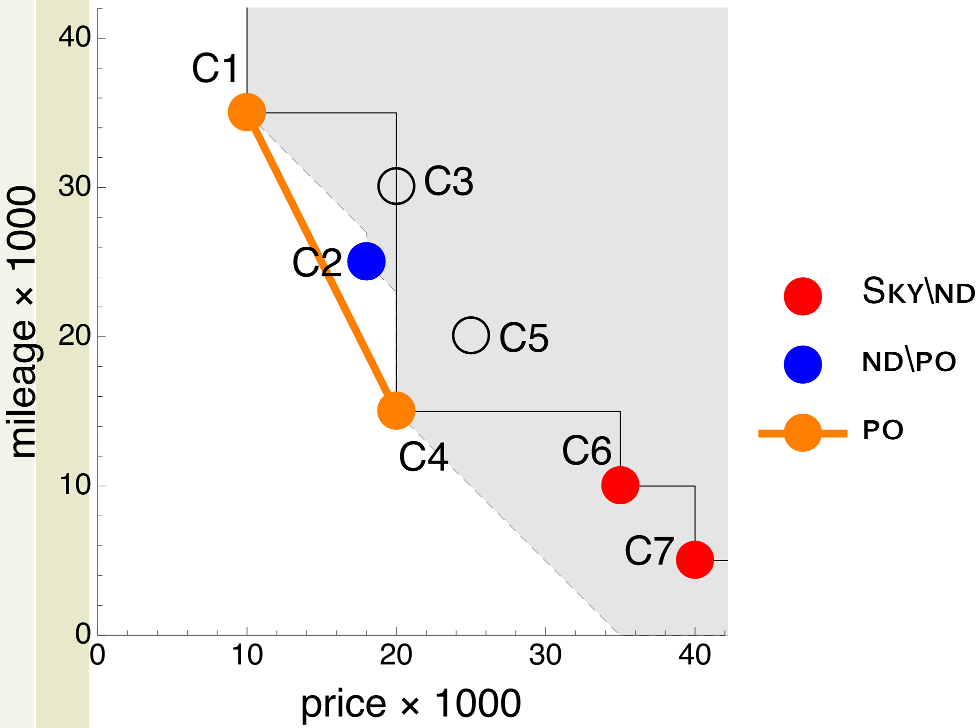
- Non-Dominated k-Skyband (ND_k), given *F*:
  - Tuples *F*-dominated by less than *k* tuples

- *k*-Skyband as tuples optimal wrt a monotone scoring function:
  - Tuples that are top *k* for some monotone scoring function

- Potentially Optimal *k*-Skyband (PO_k), given *F*:
  - Tuples that are top *k* for some monotone scoring function in *F*

- Extreme cases:
  - $F \equiv M \rightarrow ND_k = PO_k = Sky_k$
  - $F = \{ f \} \rightarrow ND_k \approx PO_k \approx$ top-k query wrt. scoring function *f*

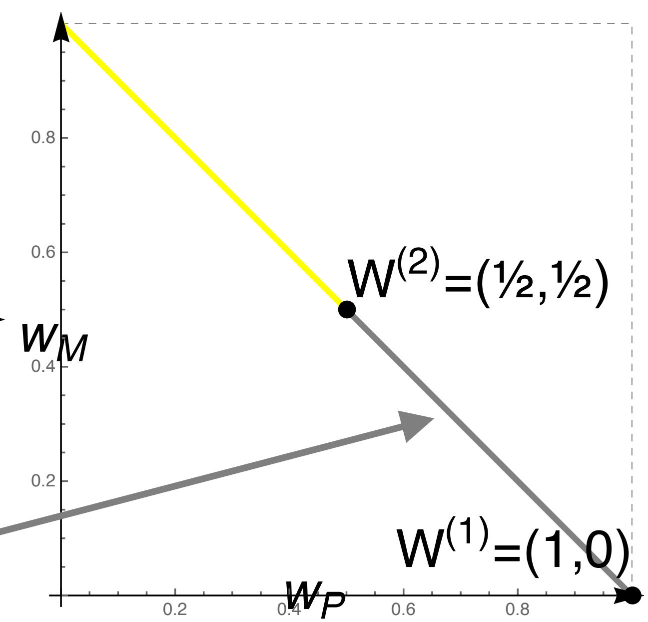
Restricted skylines – example (cars)



- Sky = {C1, C2, C4, C6, C7}
  - C2 ≺ C3 and C4 ≺ C5
- ND = {C1, C2, C4}
  - C4 $\prec_F$ C6 and C4 $\prec_F$ C7
- PO = {C1, C4}

  No allowed combination of weights makes C2 the top car

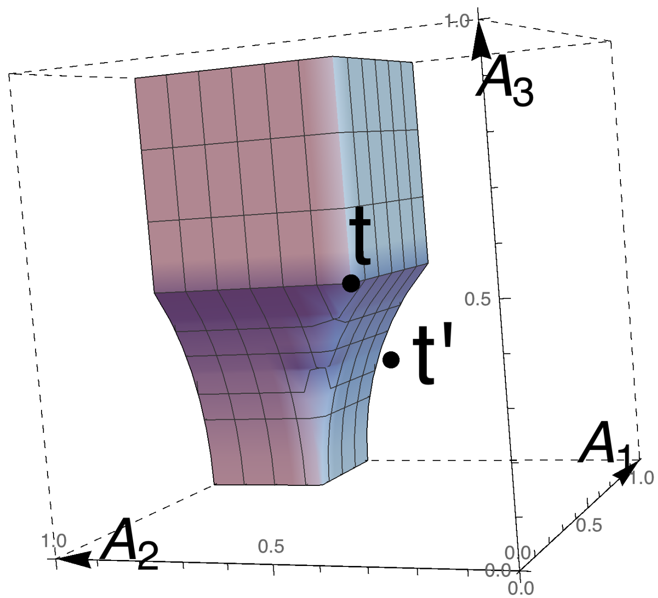
$$\mathcal{F} = \{w_P \texttt{Price} + w_M \texttt{Mileage} \mid w_P \geq w_M\}$$

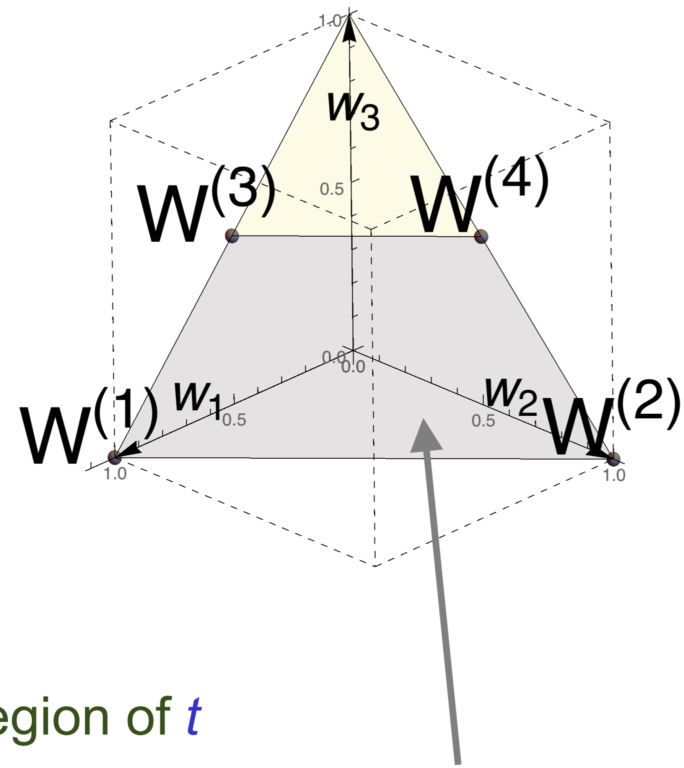


Allowed weights:
convex polytope in the weight space

F-dominance regions

- The *F*-dominance region of *t*
  - set of all points *F*-dominated by *t*
- Example: *F* = {quadratic functions with $w_1 + w_2 \geq w_3$}





  - *t'* is not in the *F*-dominance region of *t*
    - and thus not *F*-dominated by it

Allowed weights:
convex polytope in the weight space

Checking *F*-dominance

- Common scoring functions are linear in the weights:

$$f(p)=\sum_i w_i g_i(p[i])$$

- For these functions and linear constraints on the weights, checking $p \prec_F q$ can be done in two ways:
  1. by solving a linear program, or
  2. by verifying if q is in the F-dominance region of p
- The second approach is faster, but requires computing the vertices of a convex polytope in the weight space
  - But this has to be done just once for a query
- Let $W^{(j)}$ be the j-th vertex of the polytope. Then:

$$p \prec_F q \quad \text{iff } \forall j \; \underbrace{\sum_i w_i^{(j)} g_i(p[i])} \leq \sum_i w_i^{(j)} g_i(q[i])$$

j-th "vertex score" of *p*

Computing ND_k in multi-source scenarios

[Ciaccia and Martinenghi, CIKM 2018]

- In a centralized setting, ND_k is essentially computed as Sky_k by replacing dominance with *F*-dominance
- In a distributed (multi-source) setting, we reconcile the FA and TA algorithms through *F*-dominance:

Flexible Score Aggregation (FSA)

- Do a sorted access and corresponding random accesses for tuple *t*
- Keep *t* if less than *k* objects *F*-dominate *t*
  - i.e., *t* belongs to the current $ND_k(r;F)$ set
- Stop when $t \prec_F \tau$ holds for *k* objects (τ is the threshold point)

- FSA is instance-optimal for any family *F*
  - When $F = \{f\}$ it reduces to TA
  - When $F = M$ it reduces to FA

Wrap-up

Wrap-up

- All approaches to multi-criteria queries have pros and cons
- Reconciling ranking queries and skylines offers improvements:
  - Control over the importance of attributes
  - Much better control over the cardinality of the result
  - Easier specification of functions than top-k queries
  - Efficiency often better than skylines (but not top-k queries)
- But there is much more. For instance:
  - Cases of uncertainty in the ranking (what to do when scores or weights are not a precise value but an interval?)
  - Ranking heterogeneous objects across different sources (rank join problem)
  - Including notions such as proximity and diversification of objects in the ranking
  - Ranking queries from the point of view of the seller: which weights make my candidates win (reverse top-k)?

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
  Fuzzy Queries in Multimedia Database Systems. PODS 1998: 1-10
- Ronald Fagin, Amnon Lotem, Moni Naor
  Optimal Aggregation Algorithms for Middleware. PODS 2001

Skylines

- 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

Main References

Extensions of skylines: restricted skylines, k-skybands

- 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
- Dimitris Papadias, Yufei Tao, Greg Fu, Bernhard Seeger
  Progressive skyline computation in database systems. ACM Trans. Database Syst. 30(1): 41-82 (2005)

Rank join

- Ihab F. Ilyas, Walid G. Aref, Ahmed K. Elmagarmid
  Supporting Top-k Join Queries in Relational Databases. VLDB 2003: 754-765
- Karl Schnaitter, Neoklis Polyzotis
  Evaluating rank joins with optimal cost. PODS 2008: 43-52

Extensions of ranking queries: uncertainty, proximity, diversity, reverse top-k

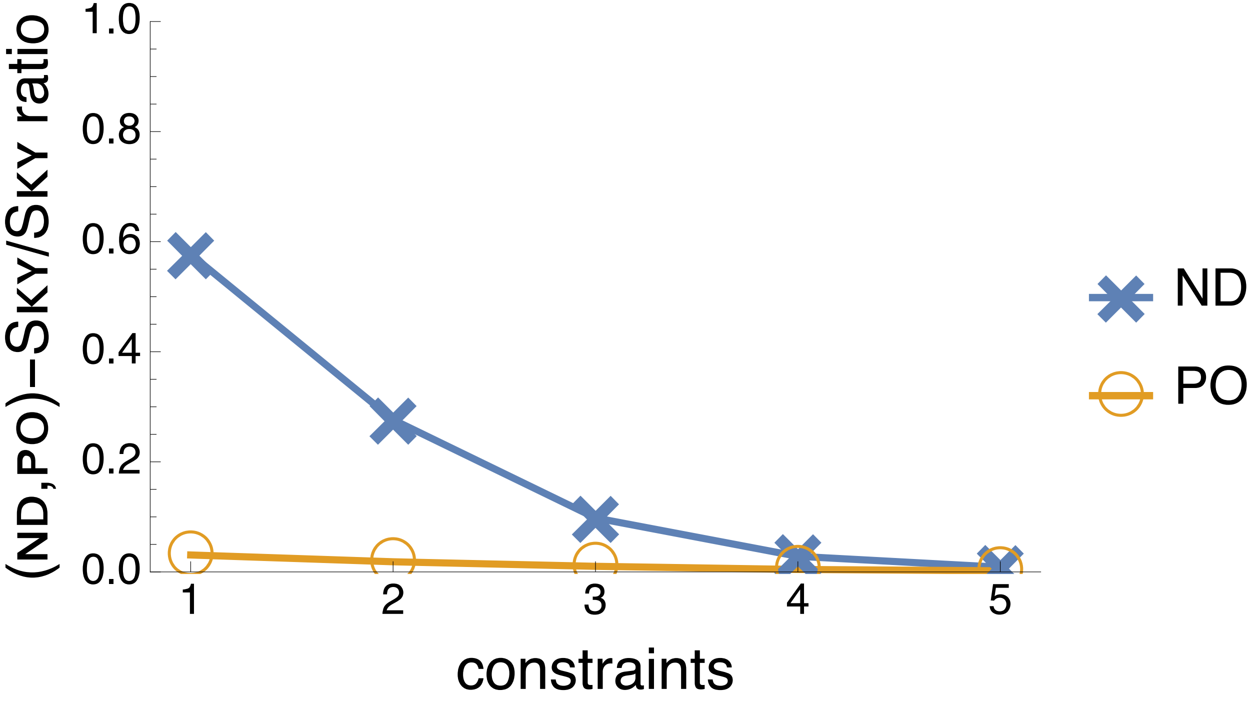
- Mohamed A. Soliman, Ihab F. Ilyas, Davide Martinenghi, Marco Tagliasacchi
  Ranking with uncertain scoring functions: semantics and sensitivity measures. SIGMOD Conference 2011: 805-816
- Davide Martinenghi, Marco Tagliasacchi
  Proximity Rank Join. PVLDB 3(1): 352-363 (2010)
- Piero Fraternali, Davide 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

Extras

Complexity of computing ND and PO

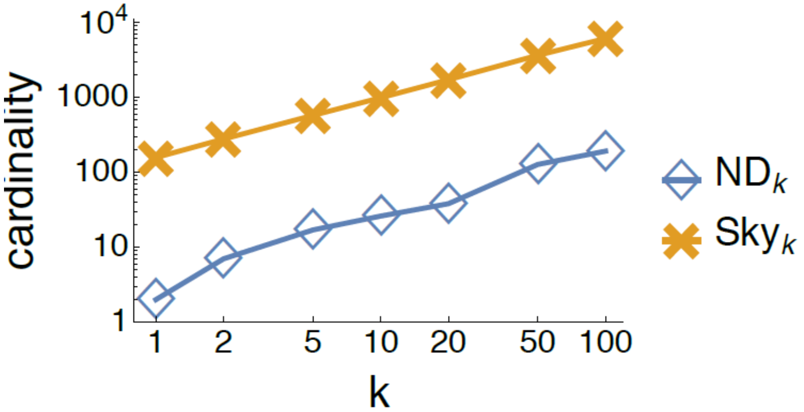
- Algorithmic variants for ND
  - unsorted vs. sorted
  - Linear Programming vs. Vertex Enumeration
  - 1 phase (check $\prec_F$ directly) vs. 2 (first check $\prec$ then $\prec_F$)
- Parameters
  - c (constraints), d (dimensions), N (tuples), q (vertices) — at most $\mathcal{O}(c^{\lfloor d/2 \rfloor})$
  - ve(c) = complexity of vertex enumeration given c constraints (NP-hard)
  - lp(x, y) = complexity of Linear Programming with x inequalities and y variables
- ND: $\mathcal{O}(\text{ve}(c) + N \cdot (\log N + |\text{ND}| \cdot q))$
- PO: $\mathcal{O}(|\text{ND}| \cdot \log |\text{ND}| \cdot \text{lp}(q, |\text{ND}|))$

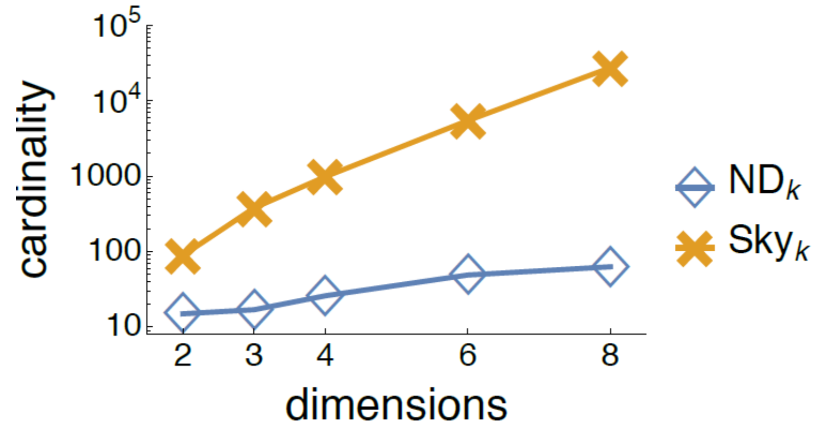
Effectiveness of restricted skylines vs skylines

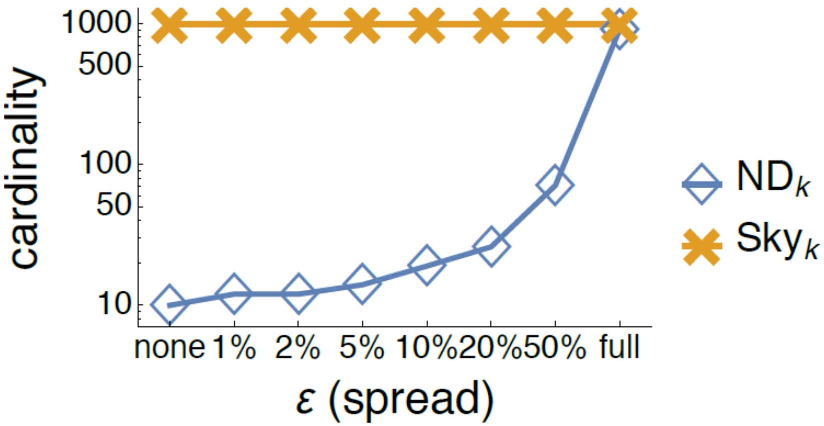


Effectiveness wrt Sky_k (NBA dataset)

NBA dataset (190,862 points)







constraints: $(1-\varepsilon)/d \leq w_i \leq (1+\varepsilon)/d$

Default values:

$k = 10$

$N = 100K$

$d = 4$

$\varepsilon = 20\%$