POLITECNICO DI MILANO

DIPARTIMENTO DI ELETTRONICA, INFORMAZIONE E BIOINGEGNERIA

Skyline and Ranking Queries: a Reconciliation

Davide Martinenghi

Joint work with Paolo Ciaccia

Rome, April 28, 2017

Outline

- Finding interesting objects in a dataset
  - Rank aggregation and ranking queries
  - Skyline queries
  - Lexicographical approaches
- Restricted skylines
  - Unifying skyline and ranking queries
  - Revisiting dominance
  - Non-dominated objects
  - Potentially optimal objects
- Computing restricted skylines
  - The case of Lp norms
  - Algorithmic alternatives
- Ongoing and future work

Finding interesting objects in a dataset

Rank aggregation

[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
- Main applications of rank aggregation:
  - 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*

Rank aggregation

[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* judges/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
Judge 1	**Judge 2**	**Judge 3**	**Judge 4**	**Judge 5**

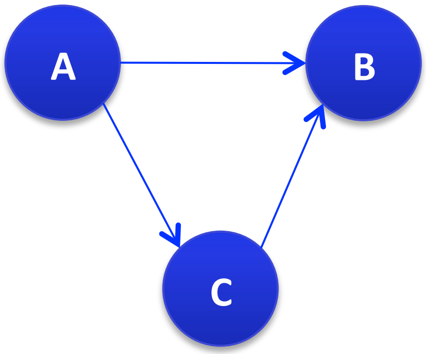
- What is the overall ranking according to all the judges?
  - No visible score assigned to candidates, only ranking
- Who is the best candidate?

Borda’s and Condorcet’s proposals

- Borda’s proposal
  - Election by order of merit
    - First place → 1 point
    - Second place → 2 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

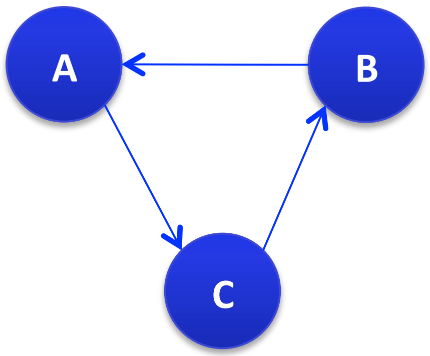
1	2	3	4	5	6	7	8	9	10
a	a	a	a	a	a	c	c	c	c
c	c	c	c	c	c	b	b	b	b
b	b	b	b	b	b	a	a	a	a



- 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

- Axiomatic approach
  - Desiderata of aggregation function formulated as "axioms"
  - By the classical result of Arrow, a small set of natural requirements cannot be simultaneously achieved by any nontrivial aggregation function

- Metric approach
  - Finding a new ranking R whose total distance to the initial rankings $R_1, \ldots, R_n$ is minimized
  - For several metrics, NP-hard to solve exactly
    - E.g., the **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
  - May admit efficient approximations

Combining opaque rankings

- Techniques using only the position of the elements in the ranking (no other associated score)
- We review MedRank, proposed by Fagin et al.
  - An algorithm for rank aggregation based on the notion of median

Input: *m* rankings of *n* elements

Output: the top *k* elements in the aggregated ranking

1. **Use sequential 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 sequential order, this algorithm is the best possible algorithm (to within a constant factor) on every input instance

MedRank example: hotels in Paris

Hotels by price	Hotels by rating
Ibis	Crillon
Etap	Novotel
Novotel	Sheraton
Mercure	Hilton
Hilton	Ibis
Sheraton	Ritz
Crillon	Lutetia
…	…

Top 3 hotels
Novotel
Hilton
Ibis

- Strategy:
  - Make one sequential access at a time in each ranking
  - Look for hotels that appear in both rankings

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

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.

Fagin's algorithm for monotone queries

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 sequential accesses in each ranking until there are at least *k* objects that match the query
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

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 3	Score

- Query: hotels with best price and rating
  - Aggregation function: 0.5*cheapness+0.5*rating
- Strategy:
  - Make one sequential 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 3	Score

- Query: hotels with best price and rating
  - Aggregation function: 0.5*cheapness+0.5*rating
- Strategy:
  - Make one sequential 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 3	Score

- Query: hotels with best price and rating
  - Aggregation function: 0.5*cheapness+0.5*rating
- Strategy:
  - Make one sequential 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 3	Score

- Query: hotels with best price and rating
    - Aggregation function: 0.5*cheapness+0.5*rating
- Strategy:
    - Make one sequential 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 3	Score

- Query: hotels with best price and rating
  - Aggregation function: 0.5*cheapness+0.5*rating
- Strategy:
  - Make one sequential 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 3	Score

- Query: hotels with best price and rating
  - Aggregation function: 0.5*cheapness+0.5*rating
- Strategy:
  - Make one sequential 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 3	Score
Novotel	.875
Crillon	.825
Ibis	.81

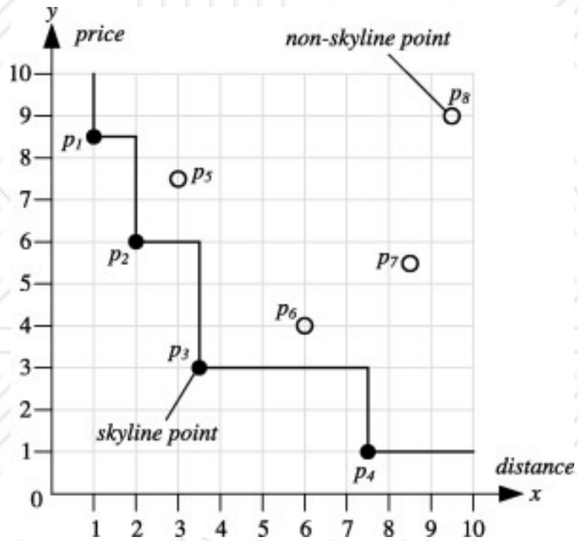
- Query: hotels with best price and rating
  - Aggregation function: 0.5*cheapness+0.5*rating
- Strategy:
  - Now complete the score with random accesses

Ranking queries – wrap-up

- Effective in identifying the best objects according to 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
  - Many different results for different settings
- The scoring function allows the user to trade-off between different attributes
  - E.g., relative importance of attributes

Skylines

- Used in *multi-objective optimization*:
  - find objects that are good according to several different perspectives (e.g., attribute values $A_1,\dots,A_d$)
  - 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 r is the set of non-dominated tuples
- In 2D, the shape resembles the contour of the dataset (hence the name)
- Skylines are agnostic wrt user preferences



Skylines – wrap-up

- 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 for large, anti-correlated datasets
- Computation is essentially quadratic in the size of the dataset (and thus not so efficient)
- Can’t leverage known user preferences wrt attributes (e.g., price is more important than distance)

The lexicographical approach

- Used in *multi-objective optimization*:
  - find objects that are good according to several different perspectives (e.g., attribute values $A_1,\ldots,A_d$)
  - a strict priority among different attributes is established
- Point of view too narrow:
  - linear priority between attributes
  - even the smallest difference in the most important attribute can never be compensated by the other attributes
- Prioritized skylines:
  - combination of skylines with the lexicographic approach
  - aim: reducing the size of the result
  - no trade-off between attributes possible
  - still no explicit control on the result cardinality

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

Skylines, revisited

- Two equivalent points of view:
  - Non-dominated tuples:

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

  - Tuples optimal according to a monotone scoring function:

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

(*M* is the set of all monotone scoring functions)

Restricted skylines

- A combination (or, better, reconciliation) of skyline and ranking queries
  - Take into account different importance of different attributes, without a strict priority as in the lexicographic approach
  - Allow a family of scoring functions *F* instead of a single one to characterize the interesting objects
    - *F* is possibly specified by means of constraints on the weights
  - Notion of dominance generalized to *F*-dominance
- For a set of monotone functions *F*, $[0,1]^d \rightarrow R^+$, tuple t *F*-dominates tuple s<>t, denoted by $t \prec_F s$, iff, $\forall f \in F.\ f(t) \leq f(s)$
- Observe that, when *F* is the set of all monotonic functions *M*, then $\prec_F$ coincides with standard dominance $\prec$
- Idea: generalize the two views of skylines when $F \subseteq M$

ND-Sky and PO-Sky

- Skyline as non-dominated tuples:

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

- Non-Dominated Skyline (ND-Sky) :

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

- Skyline as tuples optimal wrt a monotone scoring function:

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

- Potentially Optimal Skyline (PO-Sky):

$$\text{PO-SKY}(r; \mathcal{F}) =$$

$$\{t \in r \mid \exists f \in \mathcal{F}.\ \forall s \in r.\ \ s \neq t \to f(t) < f(s)\}$$

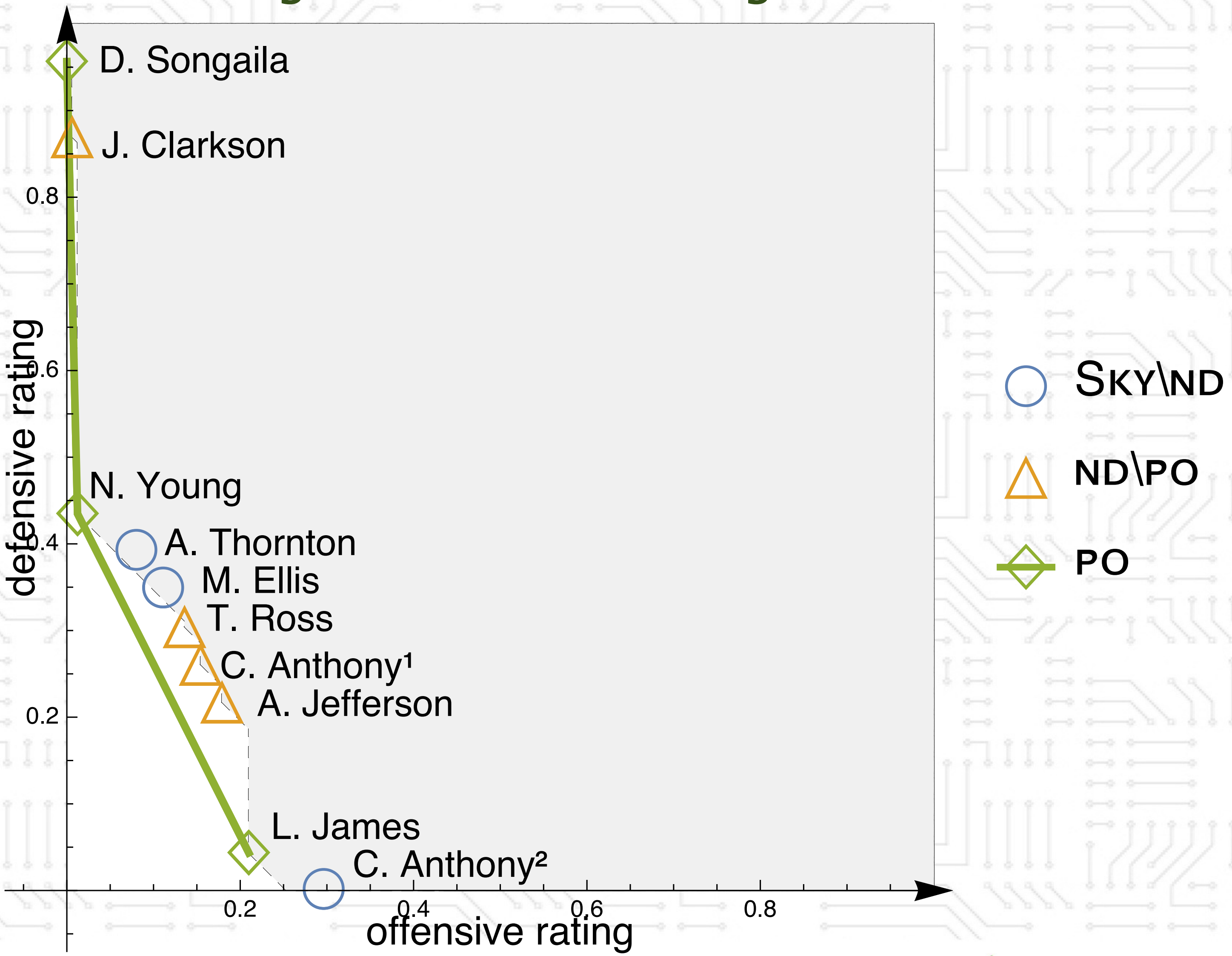
Restricted skylines - example

CarID	Price $(\times 10^3)$	Mileage $(\times 10^3)$
C1	10	35
C2	18	25
C3	20	30
C4	20	15
C5	25	20
C6	35	10
C7	40	5

- Sky returns C1, C2, C4, C6, C7
  - C3 dominated by C2 and C5 by C4
- 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

Restricted skylines – example from a real dataset

- Offensive rating ≥ defensive rating



Basic properties

- Everything collapses to Sky, when *F*=*M*

$$\textsc{po-Sky}(r;\mathcal{M}) = \textsc{nd-Sky}(r;\mathcal{M}) = \textsc{Sky}(r)$$

- Otherwise there is an inclusion relationship:

$$\textsc{po-Sky}(r;\mathcal{F}) \subseteq \textsc{nd-Sky}(r;\mathcal{F}) \subseteq \textsc{Sky}(r)$$

- Smaller sets of functions determine smaller result sets

$$\textsc{nd-Sky}(r;\mathcal{F}_1) \subseteq \textsc{nd-Sky}(r;\mathcal{F}_2)$$

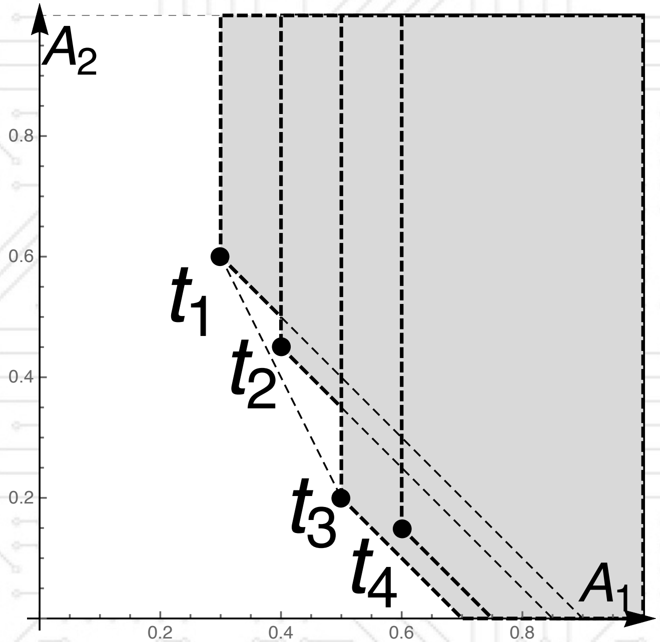
for $F_1 \subseteq F_2$

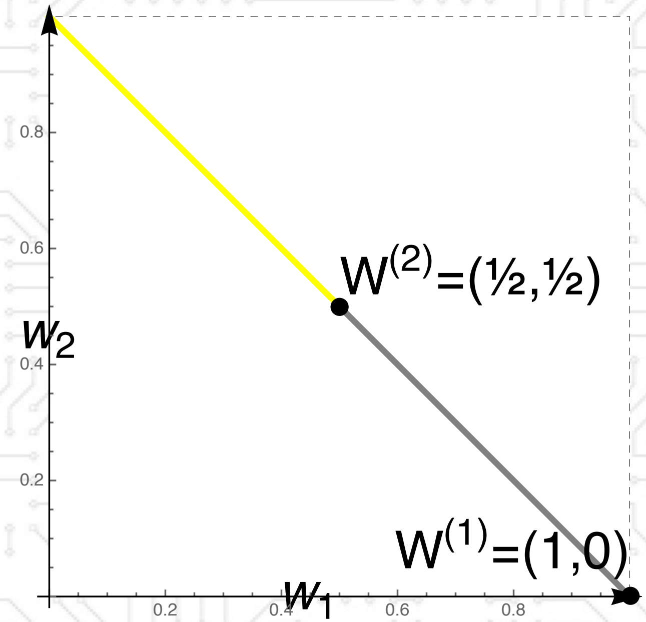
$$\textsc{po-Sky}(r;\mathcal{F}_1) \subseteq \textsc{po-Sky}(r;\mathcal{F}_2)$$

- Note that sets of functions may be determined by constraints on weights

F-dominance regions

- The *F*-dominance region of t
  - set of all points *F*-dominated by t
- Example: linear scoring functions, weights w_1 and w_2, $w_1 \geq w_2$

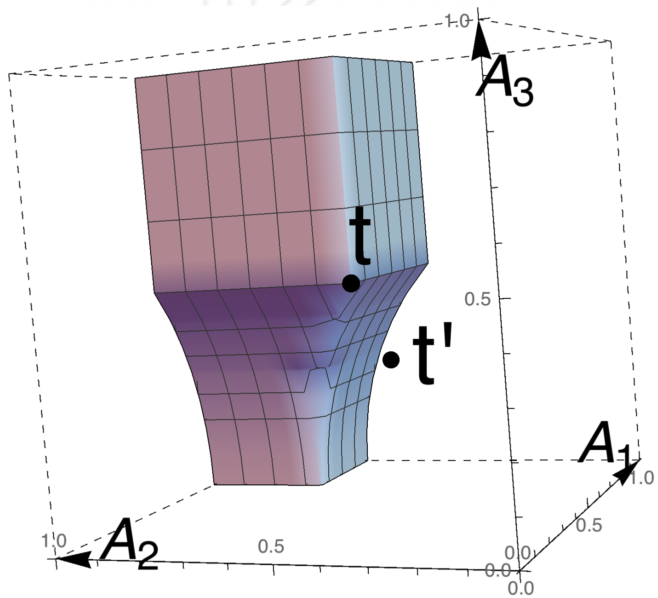


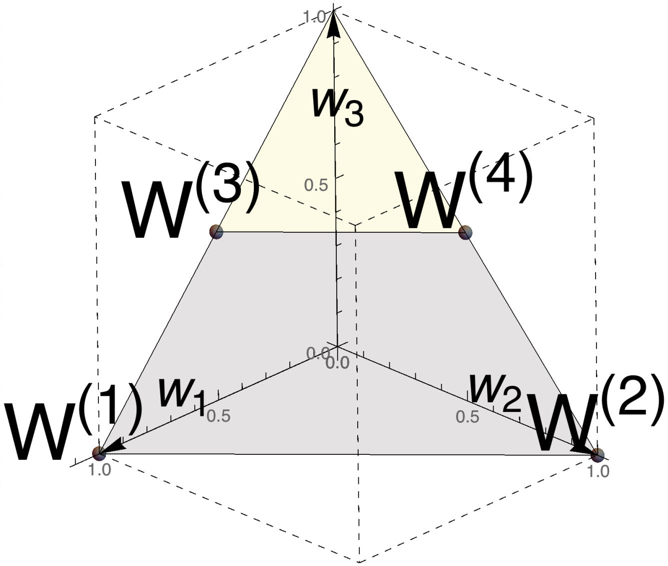


  - All are in Sky
  - t_4 is not in ND-Sky (*F*-dominated by t_3) and thus not in PO-Sky
  - t_2 is not in PO-Sky (no allowed linear function can make it top)

F-dominance regions

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

Computing restricted skylines

Lp Norms

- Common scoring functions are characterized by a weight vector W=(w_1,…,w_d):

$$L_p^W(t) = \left(\sum_{i=1}^{d} w_i t[A_i]^p \right)^{1/p}, \quad p \in \mathbb{N}$$

- thus defining a family of scoring functions:

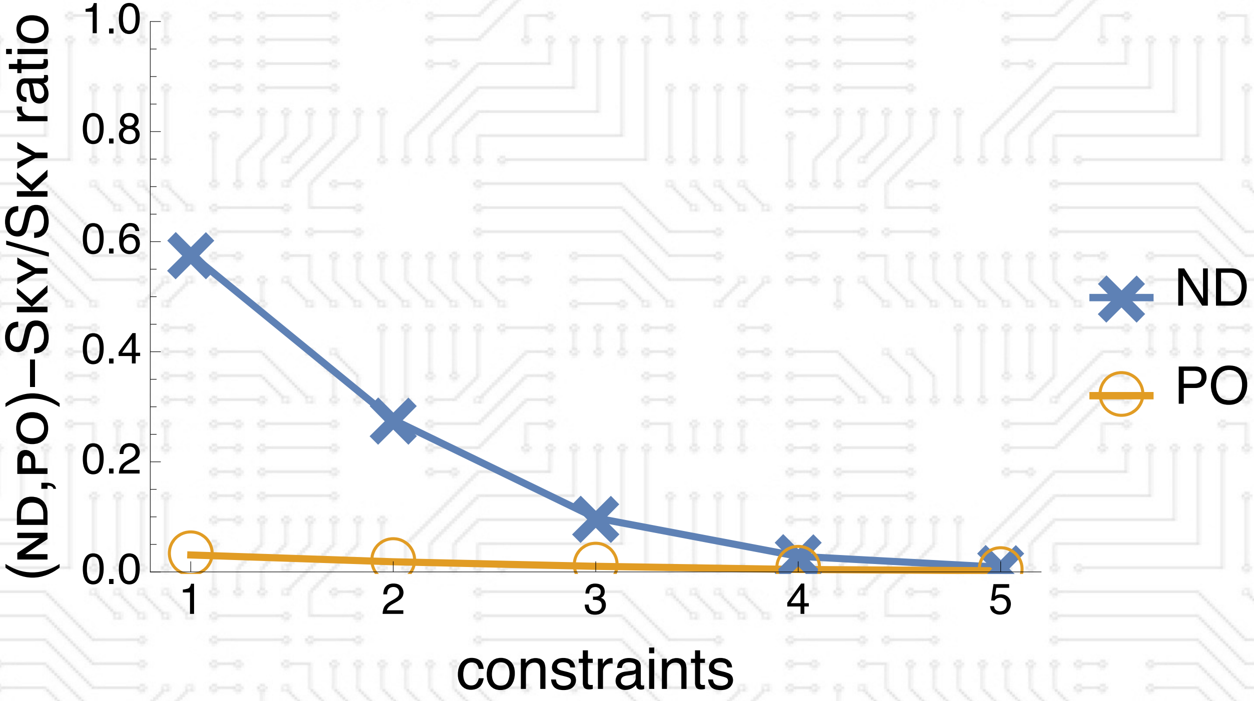
$$\mathcal{L}_p = \{L_p^W \mid W \in \mathcal{W}\}, \quad p \in \mathbb{N}$$

- For these functions, the F-dominance test t $\prec_F$ s can be checked in two ways:
  1. by solving a linear program, or
  2. by checking if s is in the F-dominance region of t
- The second approach is simpler, but requires computing the vertices of a polytope (vertex enumeration problem)

Algorithmic alternatives

- ND-Sky requires checking *F*-dominance for all pairs of tuples
- Appropriate pre-sorting of the dataset avoids lots of tests
- F-dominance regions need to be computed only once per candidate F-dominant tuple
  - Very efficient
- Although ND-Sky ⊆ Sky, first computing Sky and then removing F-dominated tuples is seldom beneficial
- A tuple t in ND-Sky is also in PO-Sky if it is not F-dominated by any convex combination of the other tuples in ND-Sky
  - Very costly
  - Sufficient conditions for pruning tuples may speed up the computation

Effectiveness of restricted skylines vs skylines



Ongoing and future work

Wrap-up

- All approaches to multi-criteria queries have pros and cons
- We have tried to reconcile ranking queries and skylines into a unifying framework
- Skylines have been generalized from two points of view:
  - Non-dominated objects
  - Potentially optimal objects
- Results
  - 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)

Future work

- Computation strategies specified for the Lp class
  - What happens with other classes?
- Restricted skylines generalize skylines (not k-skybands) and top-k queries (for k=1, not for k>1)
  - How to address these cases?

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

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