Preference queries

Jan Chomicki
University at Buffalo

Preference relations

Preference relation $\succ$

- binary relation between objects
- $x \succ y \equiv x$ *is_better_than* $y \equiv x$ dominates y
- an abstract, uniform way of talking about (relative) desirability, worth, cost, timeliness,..., and their combinations
- strong partial order: transitive, irreflexive
- preference relations used in preference queries

Preference specification

Explicit preference relations

Finite sets of pairs: bmw $\succ$ mazda, mazda $\succ$ kia,...

Implicit preference relations

- can be infinite but finitely representable
- defined using logic formulas in some constraint theory:

$$(m_1, y_1, p_1) \succ_1 (m_2, y_2, p_2) \equiv y_1 > y_2 \vee (y_1 = y_2 \wedge p_1 < p_2)$$

  for relation $Car(Make, Year, Price)$.
- defined using real-valued scoring functions: $F(m, y, p) = \alpha \cdot y + \beta \cdot p$
  $(m_1, y_1, p_1) \succ_2 (m_2, y_2, p_2) \equiv F(m_1, y_1, p_1) > F(m_2, y_2, p_2)$

Skylines

Skyline

Given single-attribute total preference relations $\succ_{A_1}, \ldots, \succ_{A_n}$ for a relational schema $R(A_1, \ldots, A_n)$, the skyline preference relation $\succ^{sky}$ is defined as

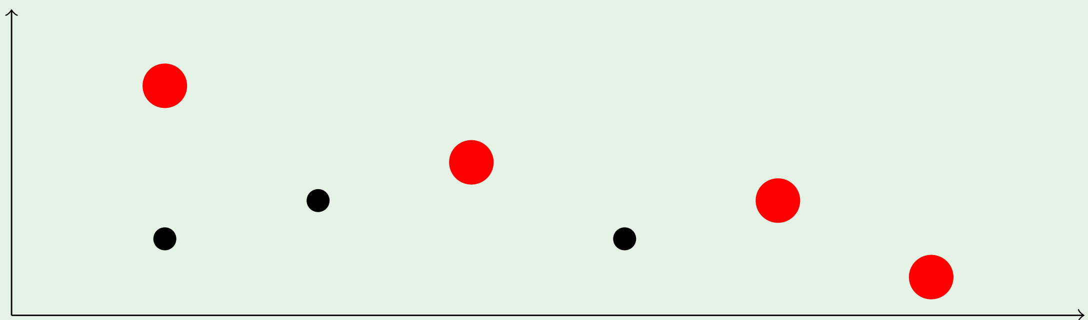
$$(x_1, \ldots, x_n) \succ^{sky} (y_1, \ldots, y_n) \equiv \bigwedge_i x_i \succeq_{A_i} y_i \wedge \bigvee_i x_i \succ_{A_i} y_i.$$

Euclidean space

Two-dimensional Euclidean space

$$(x_1, x_2) \succ^{sky} (y_1, y_2) \equiv x_1 \geq y_1 \wedge x_2 > y_2 \vee x_1 > y_1 \wedge x_2 \geq y_2$$

Skyline consists of $\succ^{sky}$-maximal vectors



Winnow

Winnow

- new relational algebra operator ω)
- retrieves the non-dominated (best) elements in a database relation
- can be expressed in terms of other operators

Definition

Given a preference relation $\succ$ and a database relation r:

$$\omega_{\succ}(r) = \{t \in r \mid \neg \exists t' \in r.\ t' \succ t\}.$$

Skyline query

$\omega_{\succ^{sky}}(r)$ computes the set of maximal vectors in r (the skyline set).

Example of winnow

Relation $Car(Make, Year, Price)$

Preference relation:

$$(m, y, p) \succ_1 (m', y', p') \equiv y > y' \vee (y = y' \wedge p < p').$$

Make	Year	Price
mazda	2009	20K
ford	2009	15K
ford	2007	12K

Computing winnow using BNL

Require: SPO $\succ$, database relation r

```
initialize window W and temporary file F to empty
repeat
   for every tuple t in the input do
      if t is dominated by a tuple in W then
         ignore t
      else if t dominates some tuples in W then
         eliminate them and insert t into W
      else if there is room in W then
         insert t into W
      else
         add t to F
      end if
   end for
   output tuples from W that were added when F was empty
   make F the input, clear F
until empty input
```

BNL in action

Preference relation: a $\succ$ c, a $\succ$ d, b $\succ$ e.

Temporary file

d

Window

c
a

eb

Input

c,e,d,a,b e,d,a,b d,a,b a,b b d

Computing winnow with presorting

SFS: adding presorting step to BNL

- topologically sort the input:
  - if x dominates y, then x precedes y in the sorted input
  - window contains only winnow points and can be output after every pass
- for skylines: sort the input using a monotonic scoring function, for example $\prod_{i=1}^{k} x_i$.

SFS in action

Preference relation: a $\succ$ c, a $\succ$ d, b $\succ$ e.

Temporary file

Window

a
b

Input

a,b,c,d,e b,c,d,e c,d,e d,e e

Algebraic laws

Commutativity of winnow with selection

If the formula

$$\forall t_1, t_2.[\alpha(t_2) \wedge \gamma(t_1, t_2)] \Rightarrow \alpha(t_1)$$

is valid, then for every r

$$\sigma_\alpha(\omega_\gamma(r)) = \omega_\gamma(\sigma_\alpha(r)).$$

Under the preference relation

$$(m, y, p) \succ_{C_1} (m', y', p') \equiv y > y' \wedge p \leq p' \vee y \geq y' \wedge p < p'$$

the selection $\sigma_{Price<20K}$ commutes with ω_{C_1} but $\sigma_{Price>20K}$ does not.

Top-K queries

Scoring functions

- each tuple t in a relation has numeric scores $f_1(t), \ldots, f_m(t)$ assigned by numeric component scoring functions $f_1, \ldots, f_m$
- the combined score of t is $F(t) = E(f_1(t), \ldots, f_m(t))$ where E is a numeric-valued expression
- F is monotone if $E(x_1, \ldots, x_m) \leq E(y_1, \ldots, y_m)$ whenever $x_i \leq y_i$ for all i

Top-K queries

- return K elements having top F-values in a database relation R
- query expressed in an extension of SQL:

```
SELECT *
FROM R
ORDER BY F DESC
LIMIT K
```


Top-K sets

Definition

Given a scoring function F and a database relation r, s is a Top-K set if:

- $s \subseteq r$
- $|s| = \min(K, |r|)$
- $\forall t \in s.\ \forall t' \in r - s.\ F(t) \geq F(t')$

There may be more than one Top-K set: one is selected non-deterministically.

Example of Top-2

Relation $Car(Make, Year, Price)$

- component scoring functions:

$$f_1(m, y, p) = (y - 2005)$$

$$f_2(m, y, p) = (20000 - p)$$

- combined scoring function: $F(m, y, p) = 1000 \cdot f_1(m, y, p) + f_2(m, y, p)$

Make	Year	Price	Combined score
mazda	2009	20000	4000
ford	2009	15000	9000
ford	2007	12000	10000

Computing Top-K

Naive approaches

- sort, output the first K-tuples
- scan the input maintaining a priority queue of size K
- ...

Better approaches

- the entire input does not need to be scanned...
- ... provided additional data structures are available
- variants of the threshold algorithm

Threshold algorithm (TA)

Inputs

- a monotone scoring function $F(t) = E(f_1(t), \ldots, f_m(t))$
- lists S_i, $i = 1, \ldots, m$, each sorted on f_i (descending) and representing a different ranking of the same set of objects

1. For each list S_i in parallel, retrieve the current object w in sorted order:
   - (random access) for every $j \neq i$, retrieve $v_j = f_j(w)$ from the list S_j
   - if $d = E(v_1, \ldots, v_m)$ is among the highest K scores seen so far, remember w and d (ties broken arbitrarily)
2. Thresholding:
   - for each i, w_i is the last object seen under sorted access in S_i
   - if there are already K top-K objects with score at least equal to the threshold $T = E(f_1(w_1), \ldots, f_m(w_m))$, return collected objects sorted by F and terminate
   - otherwise, go to step 1.

TA in action

Combined score

$$F(t) = P_1(t) + P_2(t)$$

OID	P_1
5	50
1	35
3	30
2	20
4	10

OID	P_2
3	50
2	40
1	30
4	20
5	10

Priority queue

5:60
3:80

5:60
1:65

5:60

2:60

T=100
T=75

TA in databases

- objects: tuples of a single relation r
- single-attribute component scoring functions
- sorted list access implemented through secondary indexes
- random access to all lists implemented by primary index access to r that retrieves entire tuples