Order-Based Retrieval

Derek Bridge

Presented by : K. Prasanna, AIDB Lab

Recommendation System

- Solution for information overload problem
- Content based
  - Recommend items similar to the items a user liked in the past
- Collaborative
  - Recommend items that similar users also liked
- Retrieval
  - Filter based
  - Similarity based

Filter-Based Retrieval

- Database approach of exact matching
- User preferences and requirements encoded as filters using < , = and >
- Products either satisfy them or not
- Empty result set may be returned

Similarity-Based Retrieval

- CBR approach of inexact matching using similarity measures
- User supplies ideal values for some or all attributes
- Cases most similar to ‘ideal’ case retrieved
- Result set may not be diverse

Order-Based Retrieval

- Similarity measures order the cases rather than filter them
- Order the cases based on user requirements on individual attributes
- Combine the orders and find the maxima of resulting order
- Operators for construction of partial orders from filters and similarity measures

Operators for OBRQL

- Filter Ordering
- Similarity Ordering
- About Ordering
- Non-Contradiction Ordering
- Less Strict Prioritisation Ordering

Filter Ordering

- To produce an ordering from a filter
- Given a unary predicate p, we can construct an ordering from p as:

$$x <_{\mathrm{FO}(p)} y \triangleq \neg p(x) \wedge p(y)$$

  - x is lower than y iff y satisfies p but x does not.

Similarity Ordering

- To produce an ordering from similarity measure
- Given a similarity measure σ and ideal value v order as:

$$x <_{SO(\sigma,v)} y \hat{=} \sigma(x,v) < \sigma(y,v)$$

  - x is lower than y iff x's similarity to v is lower than y's similarity to v.

About Ordering

- To define new order from existing order
- Given a partial order < and ideal value v, new ordering is defined as

$$x <_{\mathrm{AO}(<,v)} y \triangleq (x < y \leq v) \vee (x > y \geq v)$$

  - Values ordered by there distance from v
- Obtained by ‘breaking the back’ of existing order

Non-Contradiction Ordering

- To combine two partial orders without priorities
- For x to be less than y in the compound ordering x should be lower in at least one of the two partial orders and not higher in the other

$$x <_{\mathrm{NCO}(<_1,<_2)} y \triangleq$$
$$(x <_1 y \wedge x \not>_2 y) \vee (x <_2 y \wedge x \not>_1 y)$$

Less Strict Prioritisation Ordering

- To combine two partial orders with priorities
- $<_1$ takes precedence over $<_2$
- It is defined as:

$$x <_{\mathrm{LSPO}(<_1,<_2)} y \triangleq$$
$$x <_1 y \vee ((x \not<_1 y \wedge y \not<_1 x) \wedge x <_2 y)$$

- Ordering is based on $<_1$ but when values are incomparable or equal in $<_1$,$<_2$ breaks the tie.

Attribute Types

- Ordered
  - There exists non-trivial partial order of its values called as base ordering
  - Example:
    - Numeric valued attributes such as price
- Unordered
  - No inherent base order exists
  - Example:
    - Attributes such as location
    - Need similarity measure to construct orderings
- This distinction is not precise.Whether attribute month is ordered or not!!

Examples

- Assume a holiday case base with attributes duration, accomm, price, season...
- duration, accomm, price - ordered
- season – unordered (need to use similarity)

Example -FO

- We want a holiday lasting no fewer than 7 days

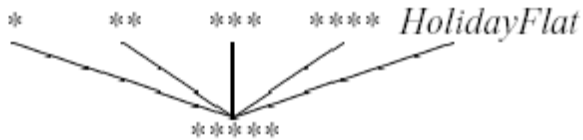
$<_{FO(\lambda x[x \geq 7])}$

- We cannot take more than 12 days holiday

$<_{FO(\lambda x[x \leq 12])}$

- We don't want a five star hotel

$<_{FO(\lambda x[x \neq *****])}$



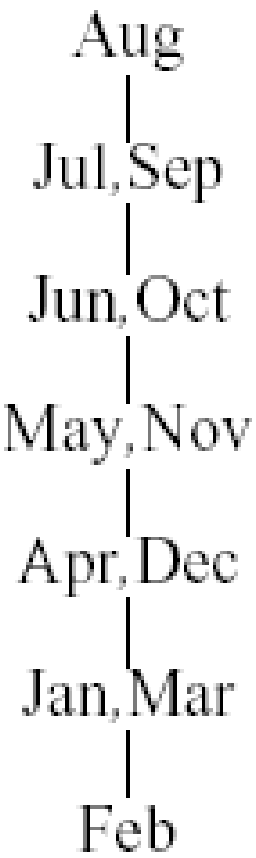
Example-SO

- We want vacation in months like August

Similarity table w.r.t August

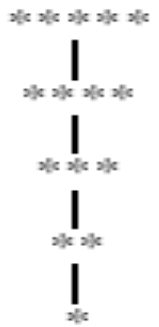
Jan	0.17	*Apr*	0.33	*Jul*	0.83	*Oct*	0.67
Feb	0.0	*May*	0.5	*Aug*	1.0	*Nov*	0.5
Mar	0.17	*Jun*	0.67	*Sep*	0.83	*Dec*	0.33

$<_{SO(\sigma_{season}, Aug)}$



Example-AO

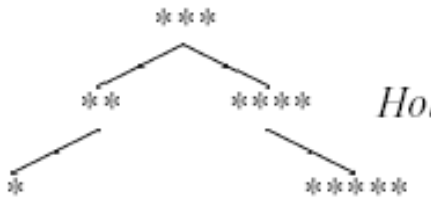
- The base ordering of accomm, $<_{accomm}$



HolidayFlat

- We want something like 3-star hotel

$<_{AO(<accomm,***)}$



HolidayFlat

- This is obtained by ‘breaking’ the base ordering at node ***

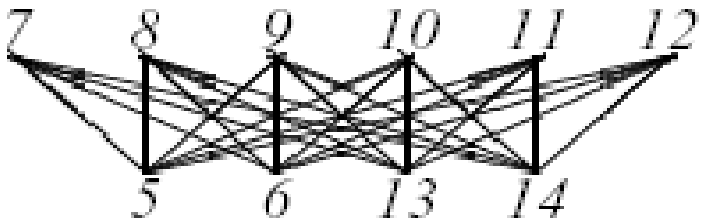
Example -combining orderings

- We want no fewer than 7 days, no more than 12 days

$<_{FO(\lambda x[x \geq 7])}$

$<_{FO(\lambda x[x \leq 12])}$

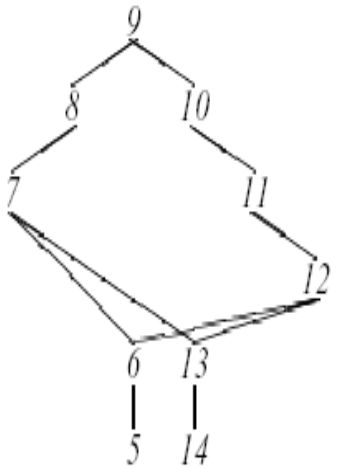
$<_{NCO(<_{FO(\lambda x[x \geq 7])}, <_{FO(\lambda x[x \leq 12])})}$



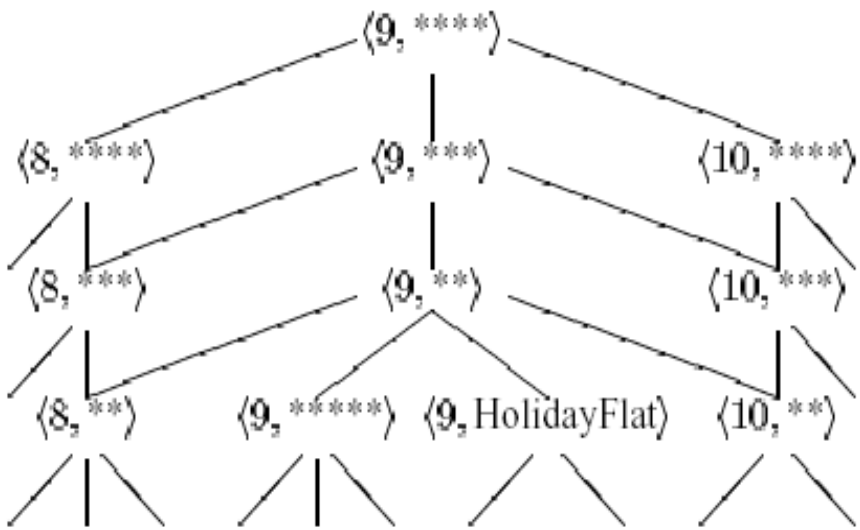
Combining orderings on different attributes

• We want a holiday no fewer than 7 days, no more than 12 but around 9 and hotel should not be a flat, 1-star, 5 star but otherwise best hotel possible

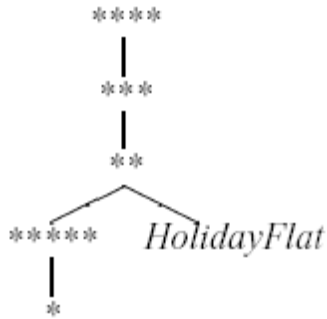
Ordering on duration



LSPO



Ordering on hotel-type



One more example

- Consider the property case base below

Identifier	Price	Bdrms	Location
A	325	3	Clapham
B	330	2	Hounslow
C	400	2	Chelsea
D	400	3	Hounslow
E	500	3	Chelsea
F	550	1	Richmond
G	600	1	Richmond
H	600	4	Clapham

- Price, bdrms are ordered, location not ordered
- Customer desires a two bedroom property in Battersea

OBR

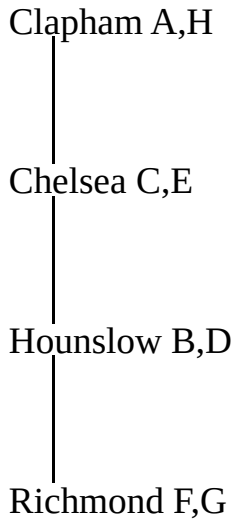
- bdrms is an ordered type attribute
- From this new ordering is $<_{AO}(<,2)$
- Location is unordered type ,similarity measure is used to construct new ordering $<_{SO}(Sim_{location},Battersea)$
- Final query expression

  $<NCO(<_{AO}(<,2)\ ,\ <_{SO}(Sim_{location},Battersea))$

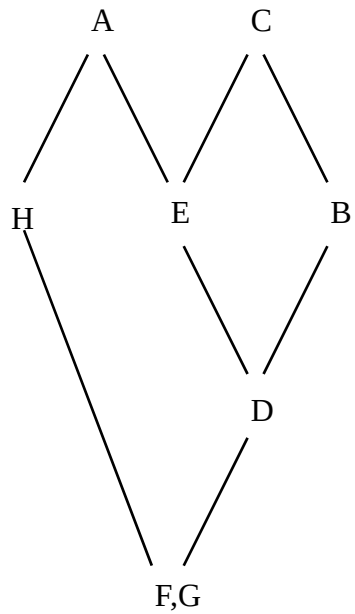
OBR

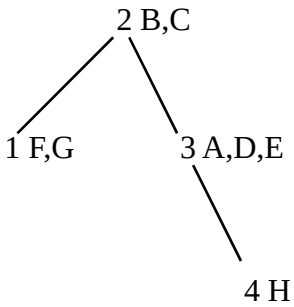
- Similarity with Battersea

Battesea	1.0
Clapham	0.7
Chelsea	0.5
Hounslow	0.3
Richmond	0.0



<SO(Sim$_{location}$,Battersea)





<AO(<,2)

<NCO(<AO(<,2) , <SO(Sim$_{location}$,Battersea))

Algorithm for finding maxima

```
maxima := {}
for each c ∈ CB
    shouldInsert = true
    for each case in maxima, m
        if c < m
            shouldInsert = false
            break
        else if m < c
            remove m from maxima
        end if
    end for
    if shouldInsert
        insert c into maxima
    end if
end for
```

Fig. 6. Naïve query evaluation algorithm applying $<$ to CB

Advantages of OBR

- Enable soft constraints
- Better semantics to tweaks
- Can have weighted similarity metrics
- Diversity among retrieved results

Soft Constraints

- Need not specify only 'ideal' values
- Constraint such as maximum/minimum can be used to construct order using FO
- Can have higher priority to this FO while combining with other orderings
- Sample final query expression for the requirement 2 bedroom property in Battersea with price <= 400

Tweaks-Query Refinement

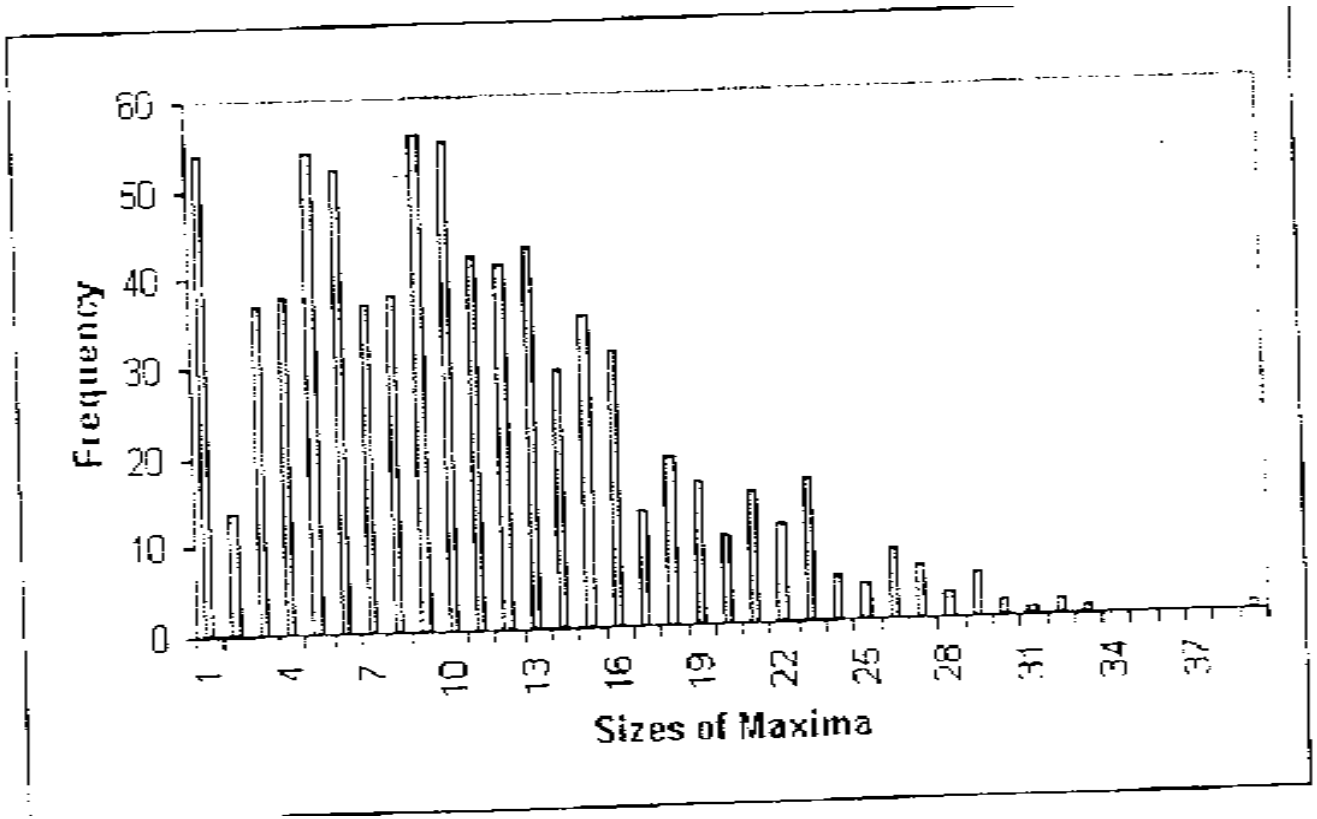
- Customer selects a product from result set and poses a new query to retrieve similar products which satisfy the tweak (products "like this but...")
- Tweaks can be encoded as filters and converted to FO
- Example:Customer can select property E and want something similar but cheaper

Diversity in OBR

- Diversity a property of result set
- Combining orderings enable diverse results to be selected
- $<_1$ and $<_2$ be two orderings such that $c_1 <_1 c_2$ and $c_2 <_2 c_1$
- Combination of these two yield a new ordering in which c_1 and c_2 are incomparable, they will be as good as each other
- If c_1 is in maxima then c_2 will also be in maxima and vice versa

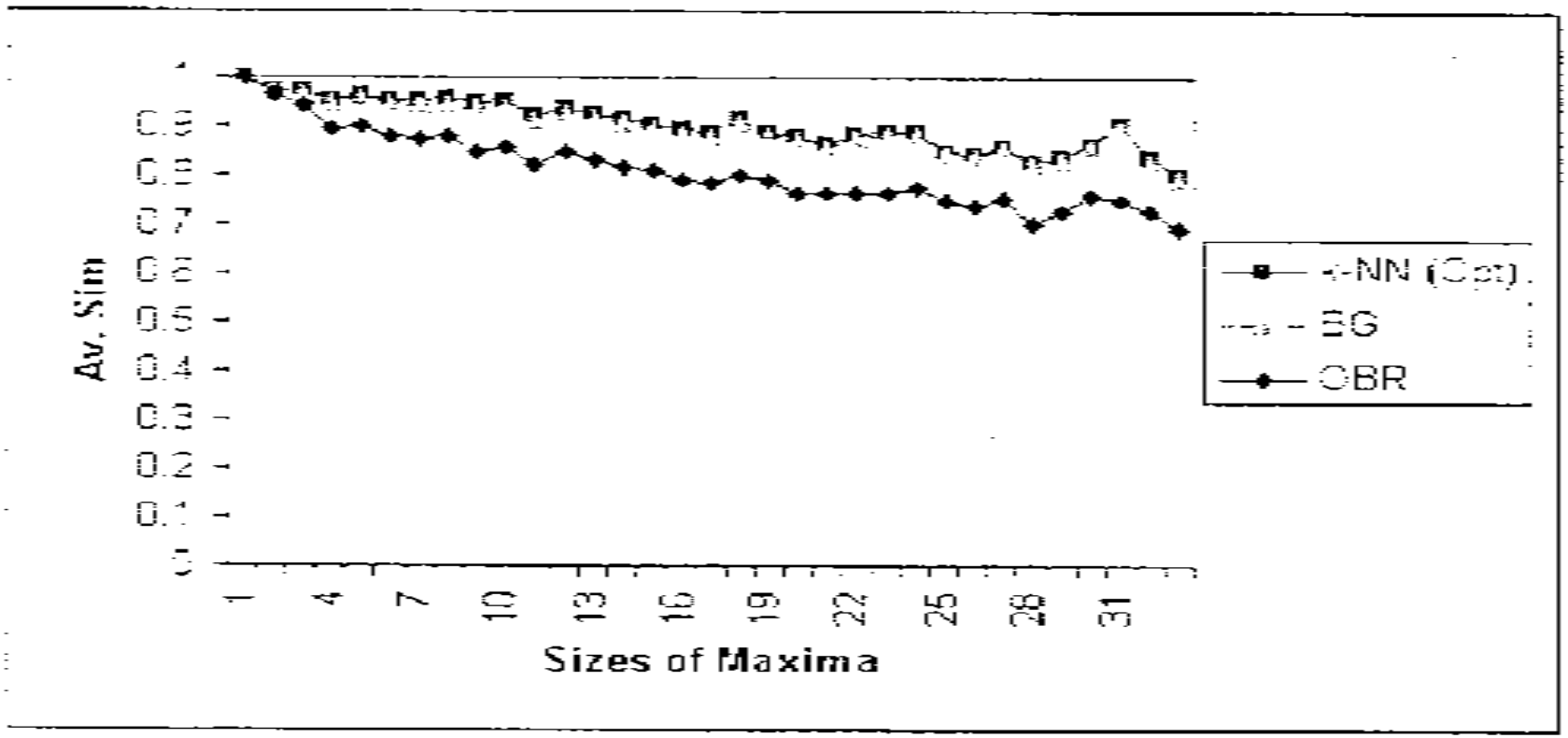
Experiments

- Case base has 794 cases with 6 attributes



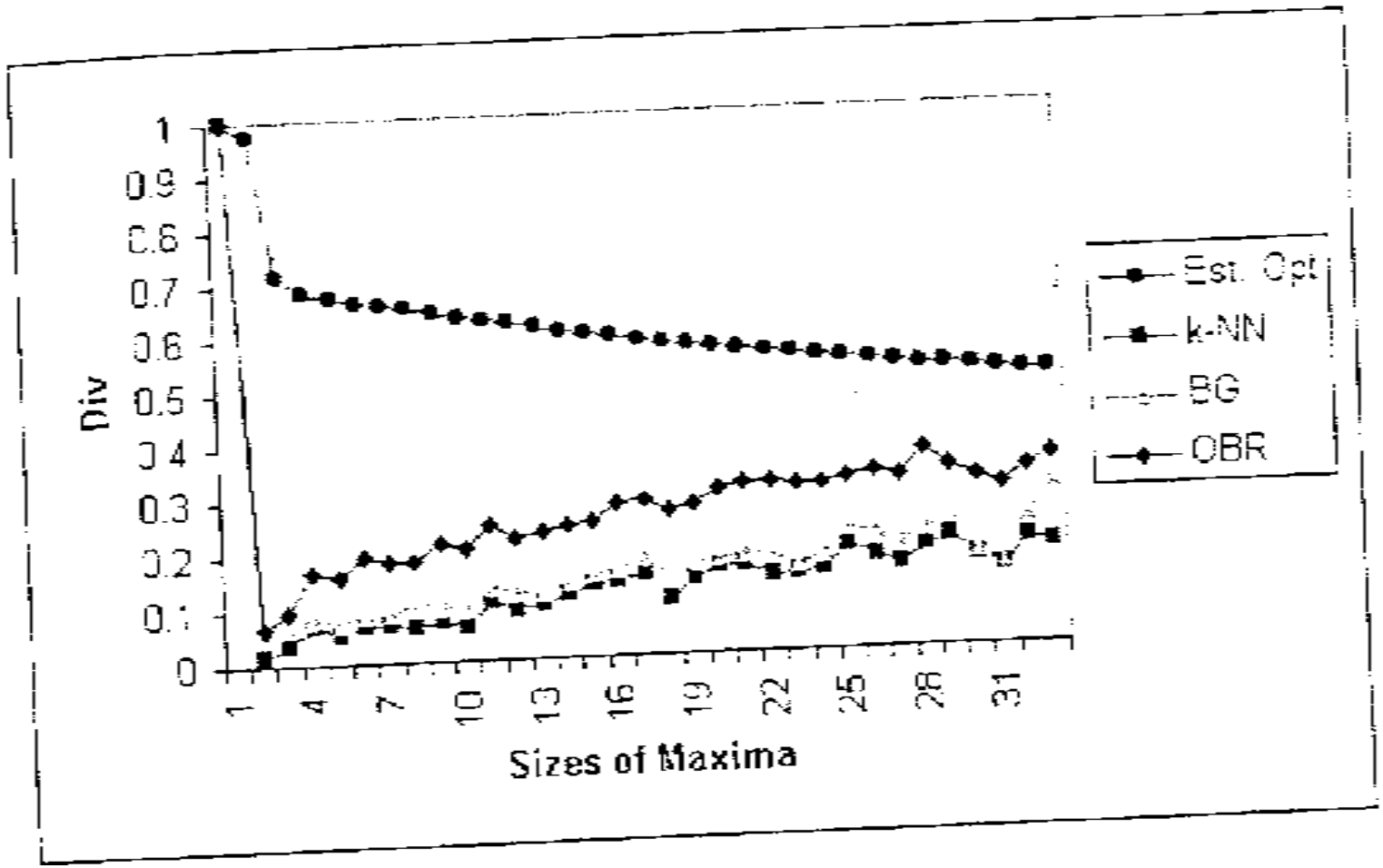
Average Similarity-Results

$$AvSim(R,q) \doteq \frac{\sum_{i=1...|R|} \mathrm{Sim}(r_i,q)}{|R|}$$



Diversity-Results

$$Diversity(c_1, ...c_n) = \frac{\sum_{i=1..n} \sum_{j=i..n} (1 - Similarity(c_i, c_j))}{\frac{n}{2} * (n-1)}$$



Conclusion

- Similarity induces ordering among cases which is more important than the similarity value itself
- Query language for OBR
- Advantages of OBR in recommendation systems
- Experiments and results

References

- Derek Bridge and Alex Ferguson: **Diverse Product Recommendations using an Expressive Language for Case Retrieval** accepted for publication by the *Sixth European Conference in Case-Based Reasoning*, 2002

- Derek Bridge: **Product Recommendation Systems: A New Direction**, R.Weber and C.G.von Wangenheim (eds.), *Procs. of the Workshop Programme at the Fourth International Conference on Case-Based Reasoning*, pp.79-86, 2001