

Online Skyline Queries

Agenda

- Motivation: Top N vs. Skyline
- „Classic“ Algorithms
 - Block-Nested Loop Algorithm
 - Divide & Conquer Algorithm
- Online Algorithm
 - Motivation
 - NN Algorithm
- Summary

Top N Queries

- Examples:
 - *The five cheapest hotels?*
 - *How rich are the top 10 percent on an average?*
 - *Increase the salary of the ten best goalies!*
- Top N in SQL (almost) not possible
- Extended SQL
[Carey & Kossmann 1997]
- Algorithms and optimization techniques
[Carey & Kossmann 1997, 1998]

Top N Queries

- Example: The five cheapest hotels

```
SELECT *  
FROM Hotels  
ORDER BY price  
STOP AFTER 5;
```

- Almost all commercial DBMS support this (syntax varies; semantics for ties varies)
- **What happens if you have several criteria?**

Nearest Neighbor Search

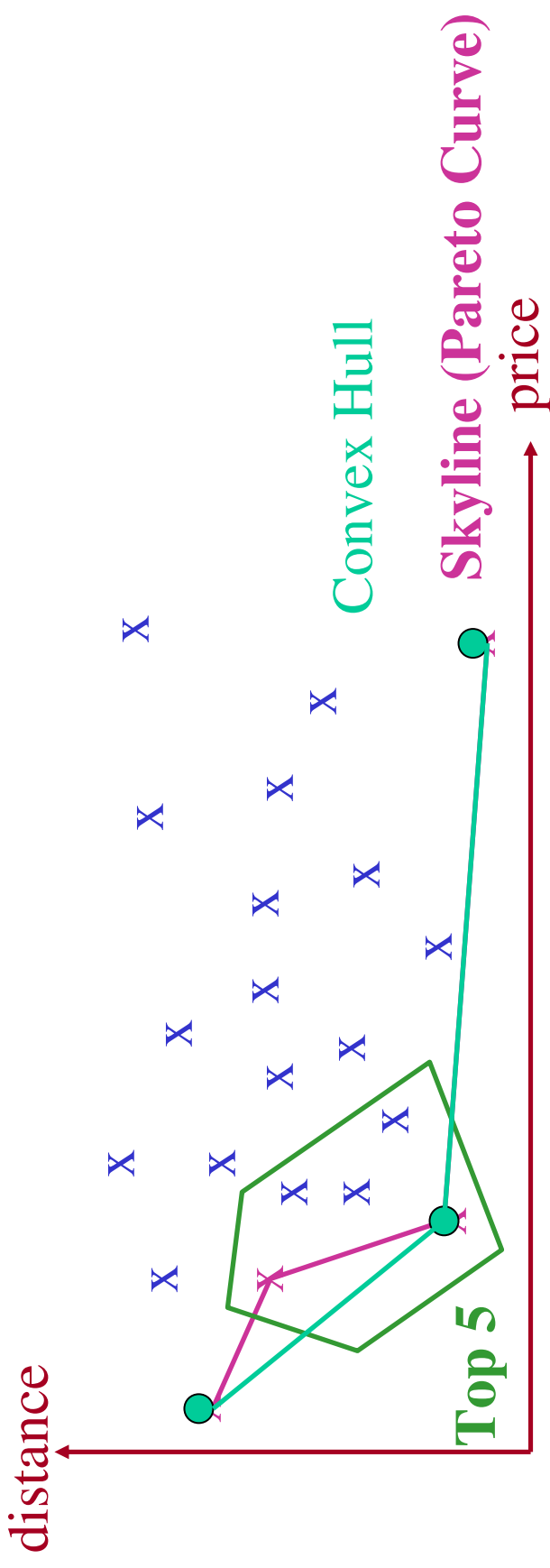
- Cheap and close to the beach

```
SELECT *  
FROM Hotels  
ORDER BY distance * x + price * y  
STOP AFTER 5;
```

- How to set x and y ?

Skyline Queries

- Hotels which are **close to the beach** and **cheap**.



Literatur: Maximum Vector Problem. [Kung et al. 1975]

Syntax of Skyline Queries

- Additional **SKYLINE OF** clause
[Börszönyi, Kossmann, Stocker 2001]
- Cheap & close to the beach

```
SELECT *  
FROM Hotels  
WHERE city = 'Nassau'  
SKYLINE OF distance MIN, price MIN;
```

Flight Reservation

- Book flight from Washington DC to San Jose

```
SELECT *
FROM Flights
WHERE depDate < `Nov-13`
SKYLINE OF price MIN,
distance (27750, dept) MIN,
distance (94000, arr) MIN,
(`Nov-13` - depDate) MIN;
```

Visualisation (VR)

- Skyline of NY (visible buildings)

```
SELECT * FROM Buildings  
WHERE city = `New York`  
SKYLINE OF h MAX, x DIFF, z MIN;
```



Location-based Services

- Cheap Italian restaurants that are close
- Query with current location as parameter

```
SELECT *  
FROM Restaurants  
WHERE type = `Italian`  
SKYLINE OF price MIN, d(addr, ?) MIN;
```

Skyline and Standard SQL

- Skyline can be expressed as nested Queries

```
SELECT *
FROM Hotels h
WHERE NOT EXISTS (
  SELECT * FROM Hotels
  WHERE h.price >= price AND h.d >= d
  AND (h.price > price OR h.d > d) )
```

- Such queries are quite frequent in practice
- **The response time is disastrous**

Naive Algorithm

- Nested Loops
- Compare each point with all other points

```
FOR i=1 TO N
  D = FALSE;
  j = 1;
  WHILE (NOT D) AND (j <= N)
    D = dominate(a[j], a[i]);
    j++;
  END WHILE
  IF (NOT D) output(a[i]);
END FOR
```

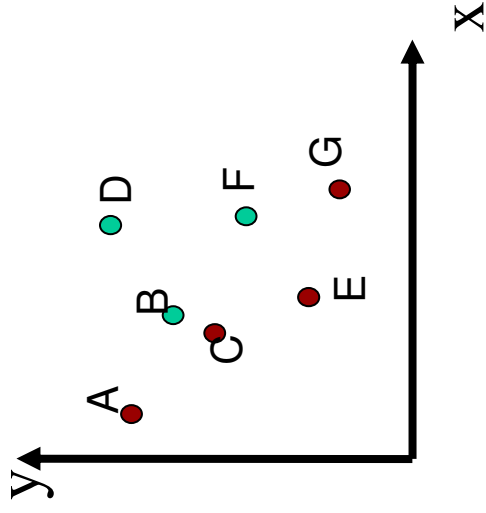
Block Nested-Loops Algorithm

- Problem of naive algorithm
 - N iterations through whole database (database often does not fit into main memory)
 - Each pair of points is compared twice
- Block Nested Loops Algorithm
 - Keep **window** of *incomparable* points
 - If window does not fit in memory, spill points to disk
- Assessment
 - N / window iterations through database
 - No double-comparisons

BNL Input

Input: ABCDEFG

Window size = 2



Step	Window	Input	Temp	Output
1,2	AB	CDEFG		
3	AC	DEFG		
4-7	AC		EFG	
8		EFG		AC
9-11	EG			AC
12				ACEG

Organisation of Window

- „Self-organizing List“
 - Move „Hits“ to the beginning
 - Reduces CPU cost for comparisons (early stop)
- „Replacement“
 - Maximize the „Volume“ of window
 - Additional CPU overhead for replacement policy
 - Less iterations because points in window are a better filter

Divide & Conquer Algorithm

- [Kung et al. 1975]
- Idea:
 - Partition the data into two sets
 - Apply algorithm recursively to both sets
 - Merge the results from both sets
- Magic is in „merge“
- Best algorithm for „worst case“
 $O(n * (\log n)^{(d-2)})$
- Poor in best case: $O(n * (\log n)^{(d-2)})$ vs. $O(n)$
- Poor performance if DB does not fit in memory

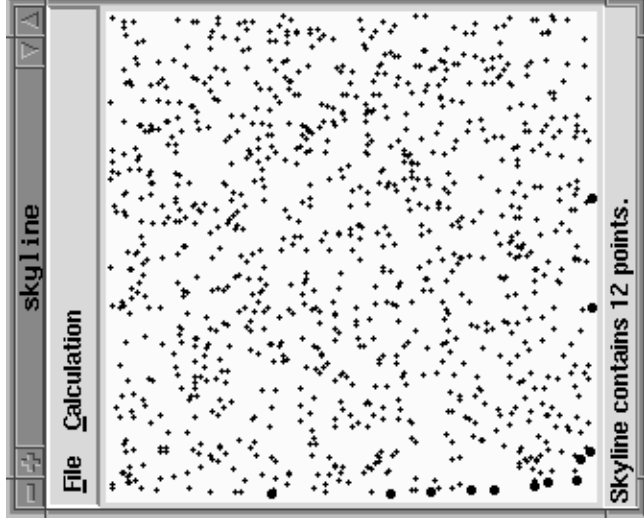
Variants of D&C Algos

- M-way Partitioning
 - Partition into M sets ($M > 2$)
 - Extended merge algorithm
 - Optimized „Merge Tree“
 - Dramatic reduction in CPU cost
- Early Skyline
 - Eliminate points „on-the-fly“
 - Saves I/O and CPU costs
 - (in rare cases: additional CPU costs)

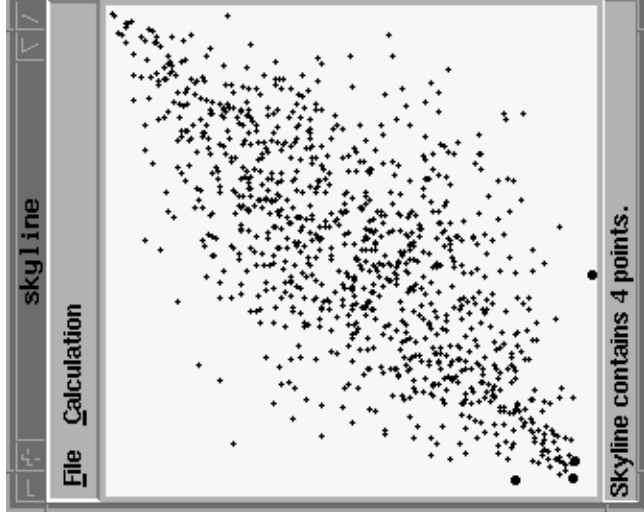
Performance Experiments

- Three data sets
 1. **Independent:** Coordinates of points are generated randomly using a uniform distribution
 2. **Correlated:** Points which are good in one dimension are likely to be good in other dimensions, too.
 3. **Anti-correlated:** Points which are good in one dimension are likely to be bad in other dimensions.
- DB with 100.000 and 1 mio. points
- Vary size of memory, vary # of dimensions

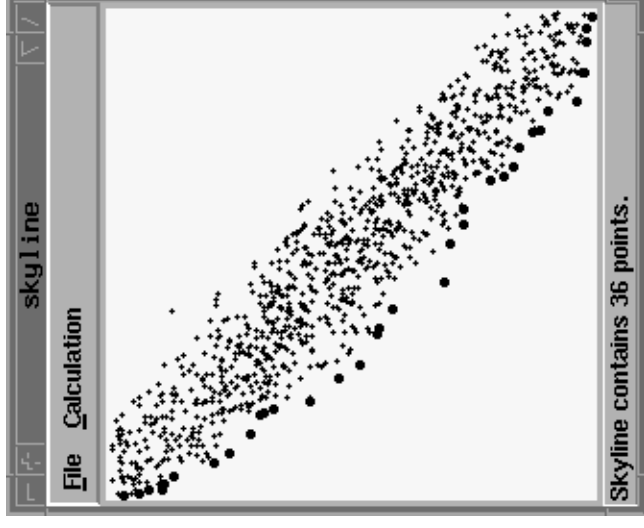
Data Sets



Independent
(Skyline is large)



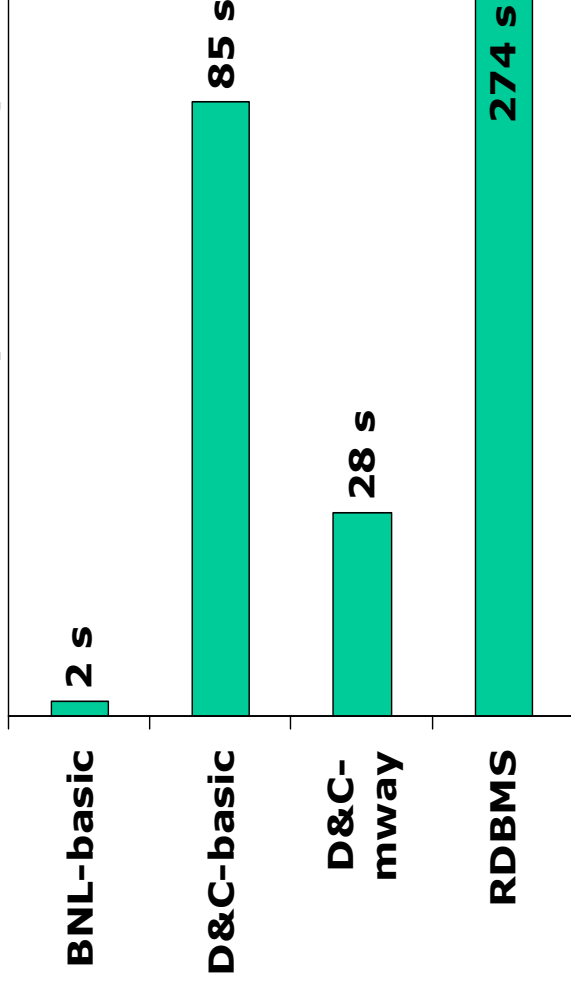
Correlated
(Skyline is small)



Anti-correlated
(Skyline is very large)

“Correlated”

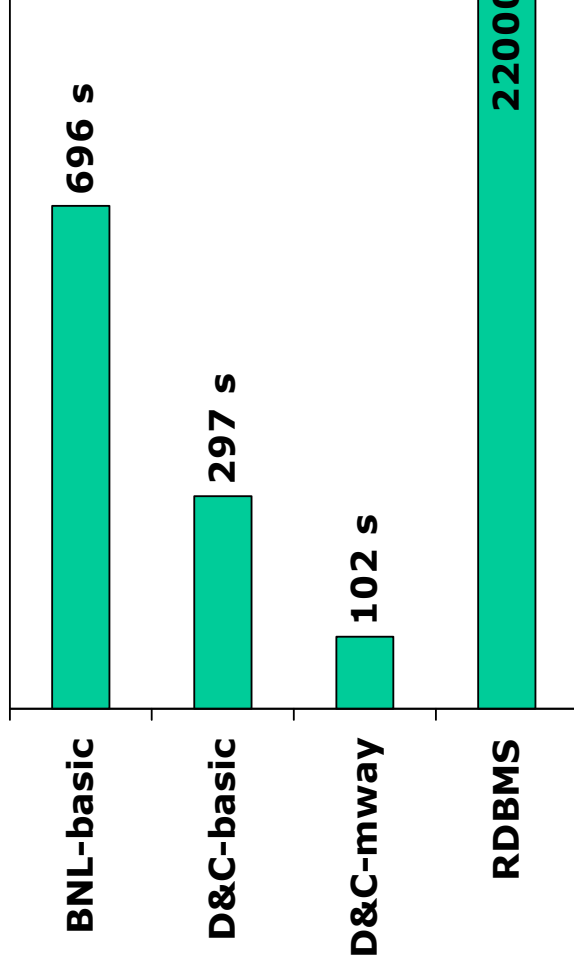
8 dimensions, Skyline: 121 points



- BNL is winner for “small” Skylines (best case)

“Anti-correlated”

8 dimensions, Skyline: 55691 points



- Extended D&C is winner for large Skylines (worst case)

Summary (so far)

- Extend RDBMS with special Skyline algos
- BNL algorithm if Skyline is expected to be small
- Extended D&C in other cases
- **However, algorithms are not interactive**
 - Possibly, first results returned after 100 secs
 - User has no control

Online Algorithms: Requirements

- Immediately return the first results
 - Give response time guarantees for the first x points
- Progressive processing
 - More results, the longer user waits
 - Algorithm terminates with whole Skyline
- Fairness; User interaction
 - User gives hints *while* the results arrive
- Correct – never return wrong results (Flackern)
- Universal; easy to integrate into DB products

Online Skyline Algorithm

[Kossmann, Ramsak, Rost 2002]

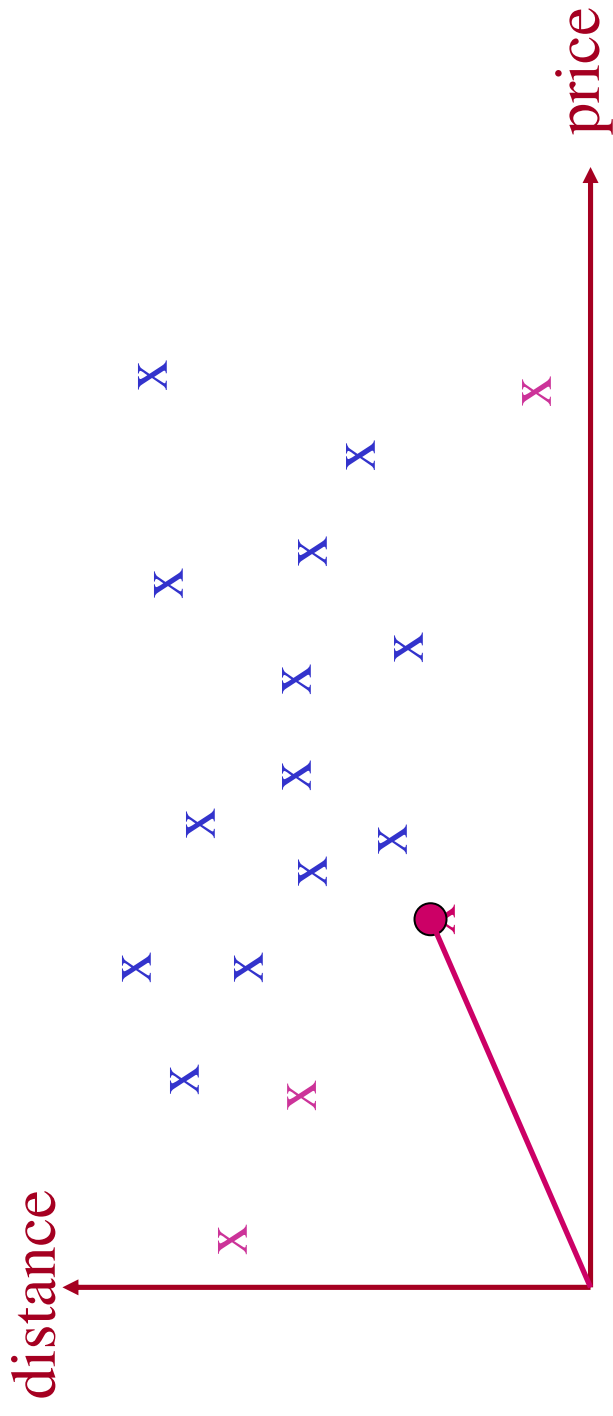
- Divide & Conquer Algorithmus
 - Search for *Nearest Neighbor* (e.g. with R* tree)
 - Partition Data space into *Bounding Boxes*
 - Search recursively for *Nearest Neighbors* in *Bounding Boxes*

Online Skyline Algorithm

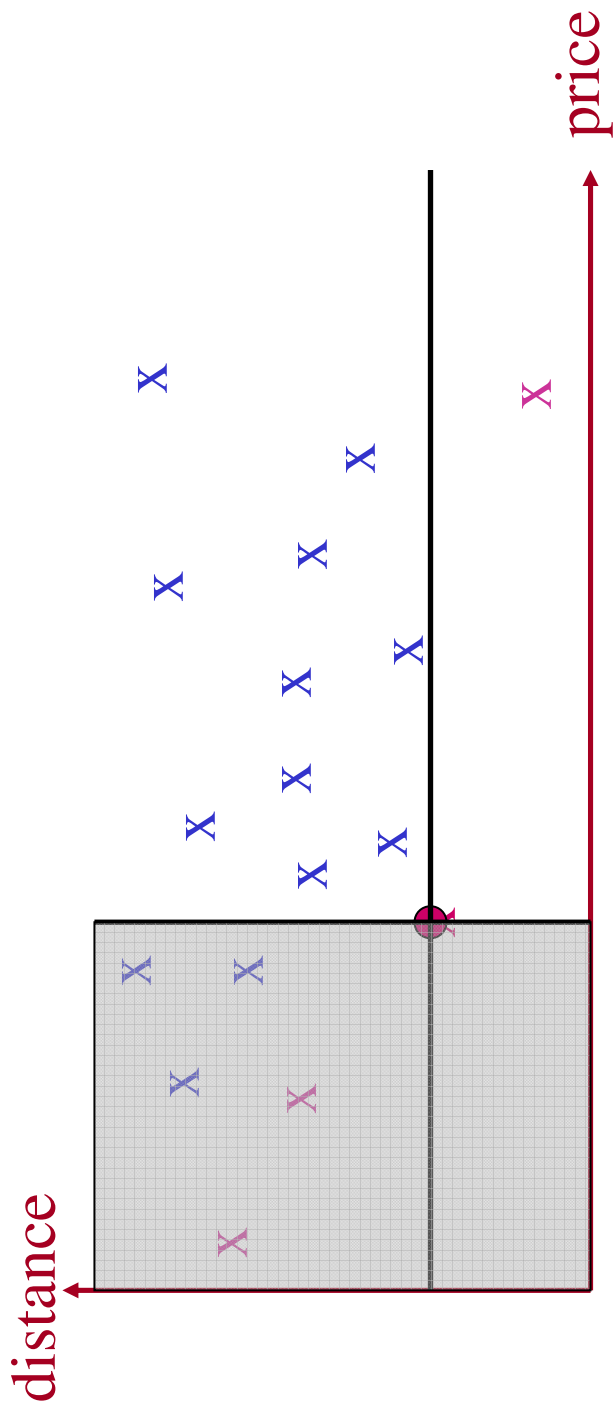
[Kossmann, Ramsak, Rost 2002]

- Divide & Conquer Algorithmus
 - Search for *Nearest Neighbor* (e.g. with R* tree)
 - Partition Data space into *Bounding Boxes*
 - Search recursively for *Nearest Neighbors* in *Bounding Boxes*
- Correctness - 2 Observations
 - *Every Nearest Neighbor* is in the Skyline
 - *Every Nearest Neighbor* in a *Bounding Box* is in the Skyline

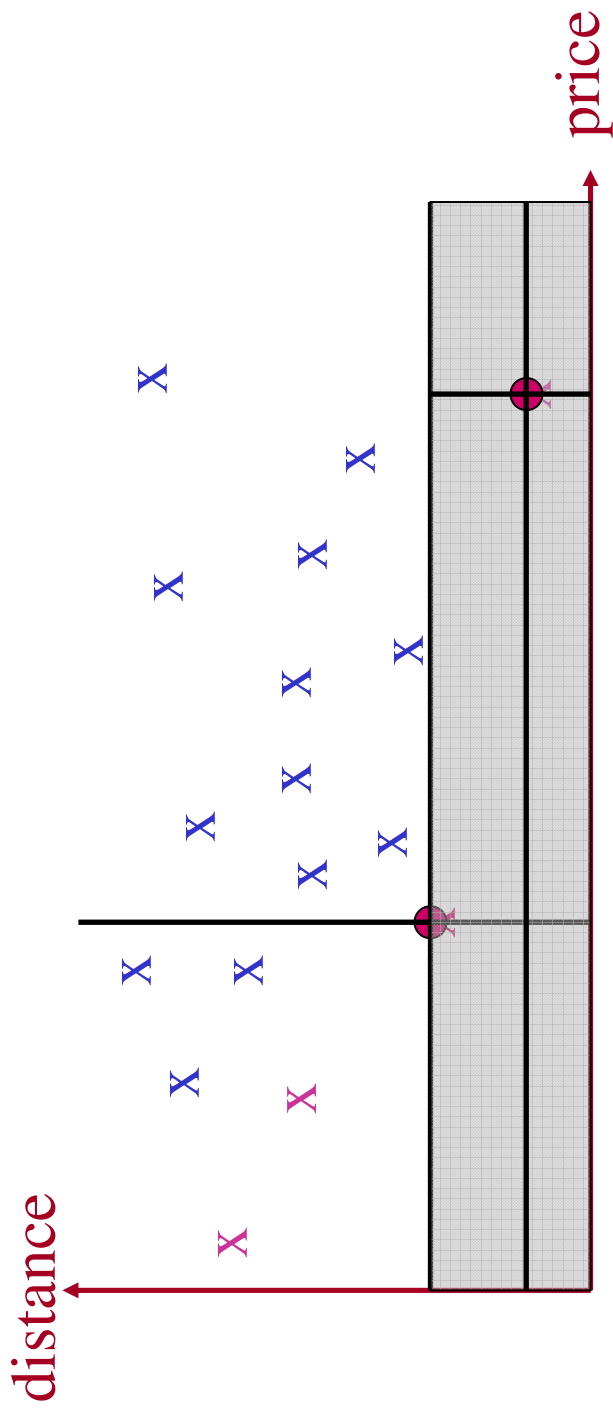
NN Algorithm



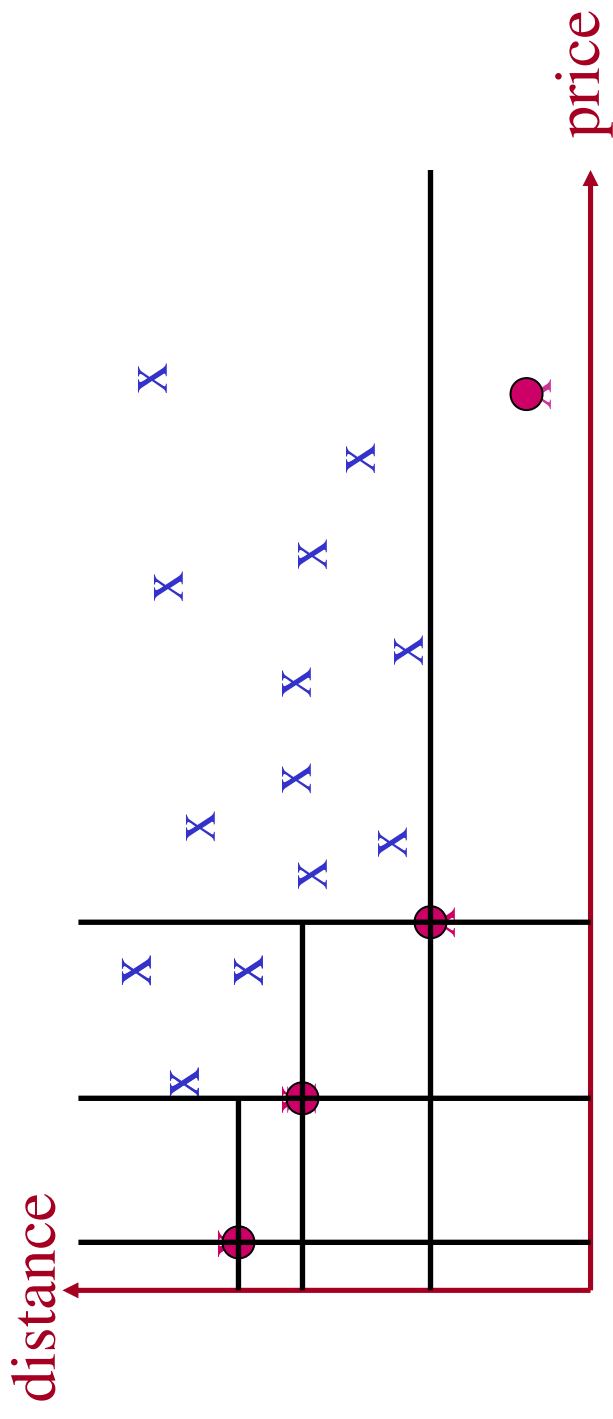
NN Algorithm



NN Algorithm



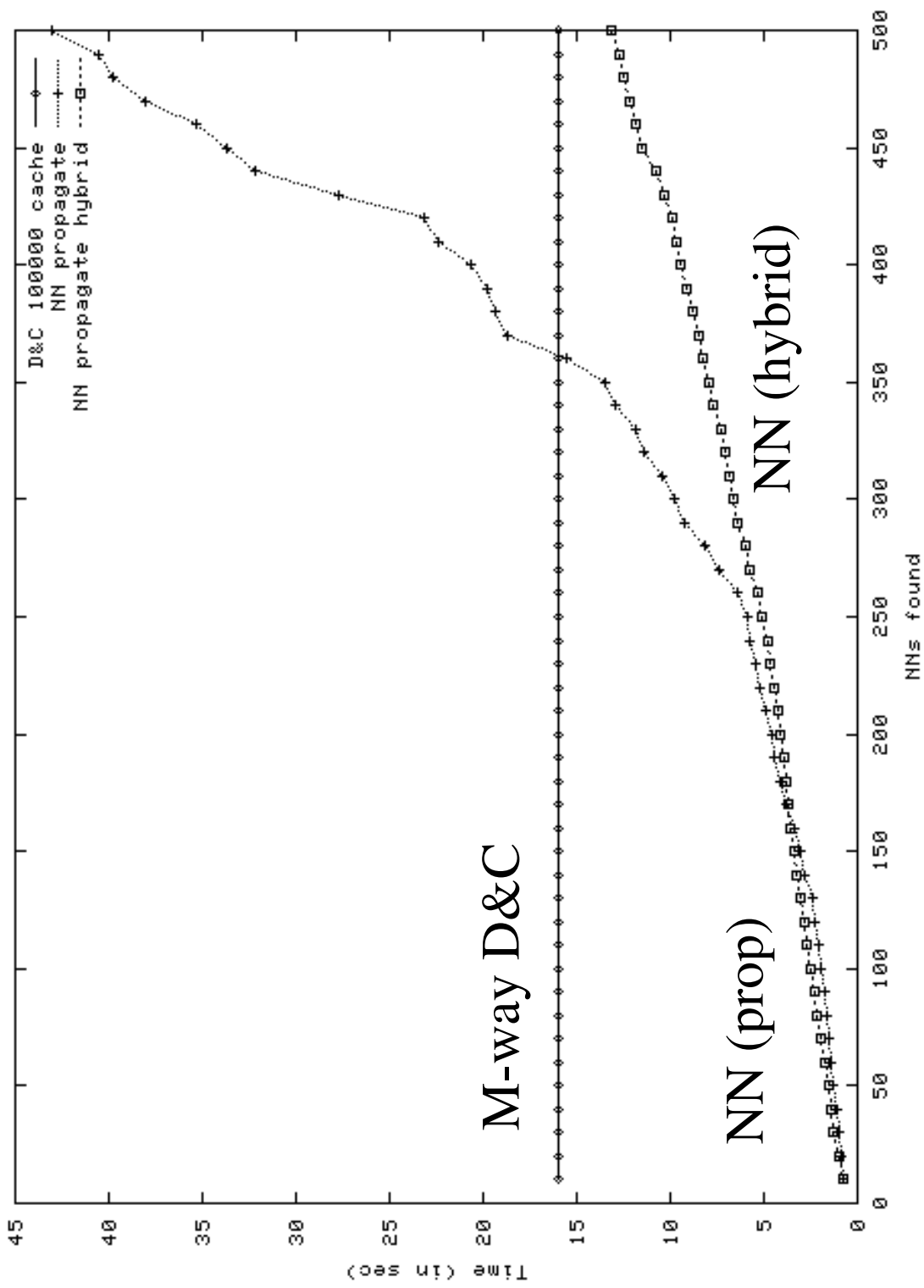
NN Algorithm



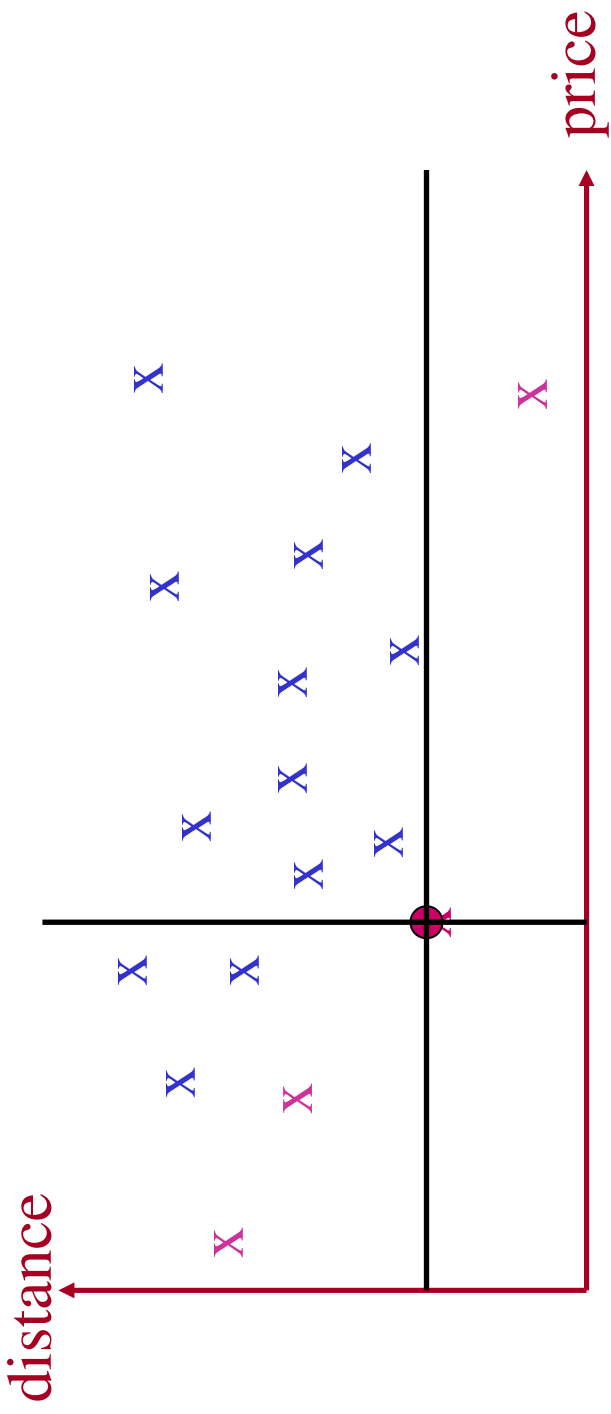
Implementation

- NN Search with R^* tree, UB tree, ...
 - Bounding Boxes can be considered in index lookup
 - Additional predicates can also be considered
 - NN is efficient, off-the-shelf DB operation
- For $d > 2$, bounding boxes overlap
 - Duplicate elimination
 - Merging of Bounding Boxes
 - Propagation of NNs
- Algorithm can be used for location-based service
 - Parameterized search in R^* tree

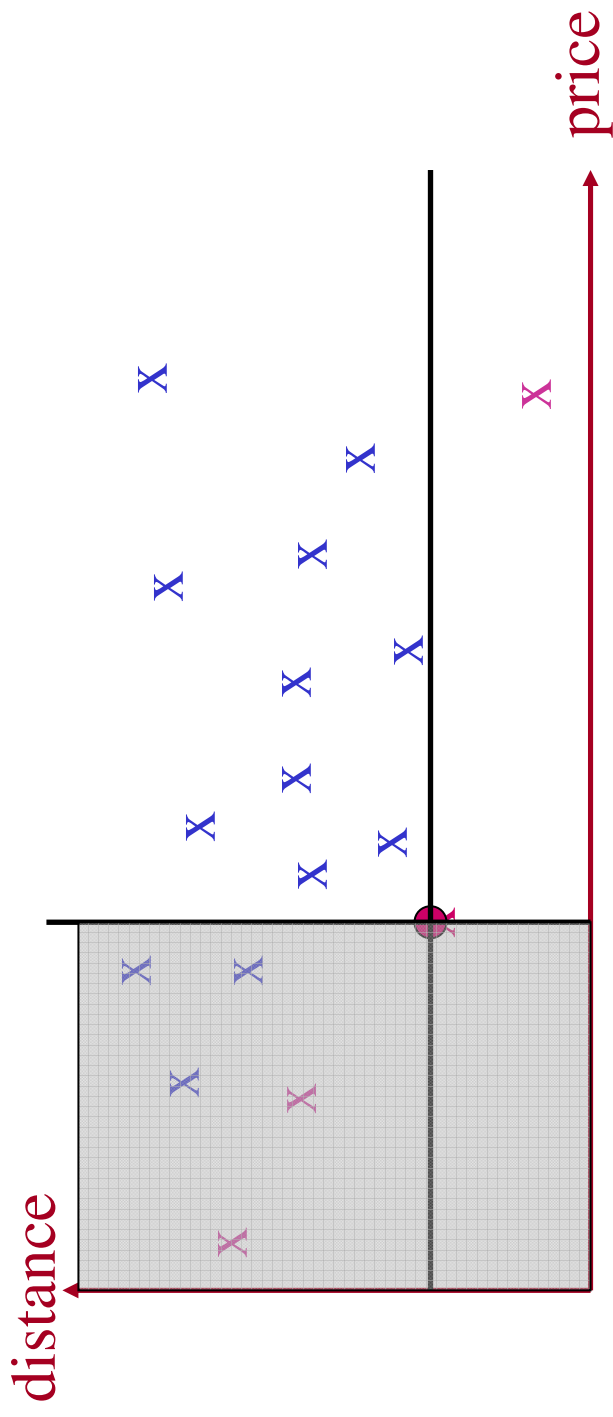
5 dimensions, anti-correlated



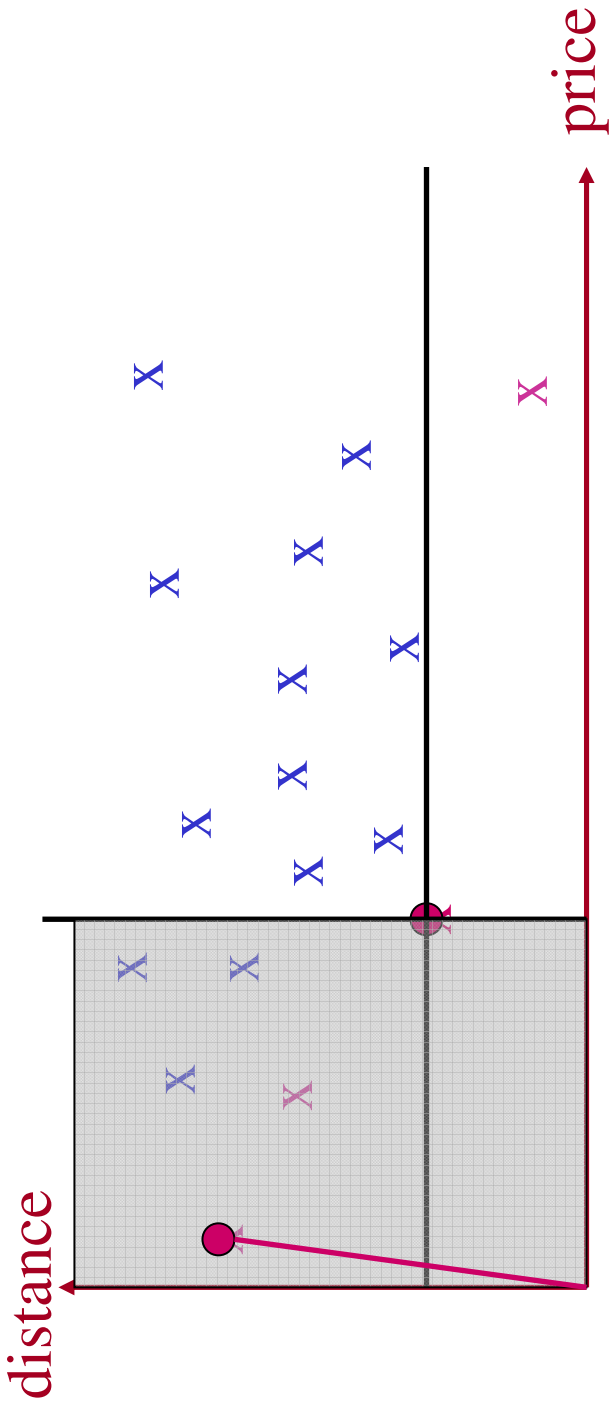
User Control



User Control



User Control



Related Work

- Maximum Vector Problem [Kung et al. 1975]
 - Only good if DB fits into main memory
 - Good in worst case, poor in good cases
- Progressive Skyline Computation [Tan et al. 2001]
 - Extended BNL Algorithm
- Konvexe Hülle [Yuval 1975]
 - Unknown how well this works for large DBs
- Mehrzieloptimierung (z.B. [Papadimitriou 2001])
 - Approximate Pareto-Kurve
- Extended NN Algorithm [Papadias et al. 2003]

Summary

- Skyline has many applications (Decision Support, Visualisation, ...)
- New „Batch“ and „Online“ Algorithms
- Special algorithms for Skyline + Join + Top N
- Future Work: Skyline-Ticker, high Update rates
 - Continuously display good restaurants in car
 - Continuously give good car offers