Efficient Top-k Query Processing In Highly Distributed Environment

Seminar on non traditional data management

Vinay Setty

Saarland University — Department of Computer Science

20/01/2009

Contents

Find a hotel in Manhattan, cheap and near to the beach

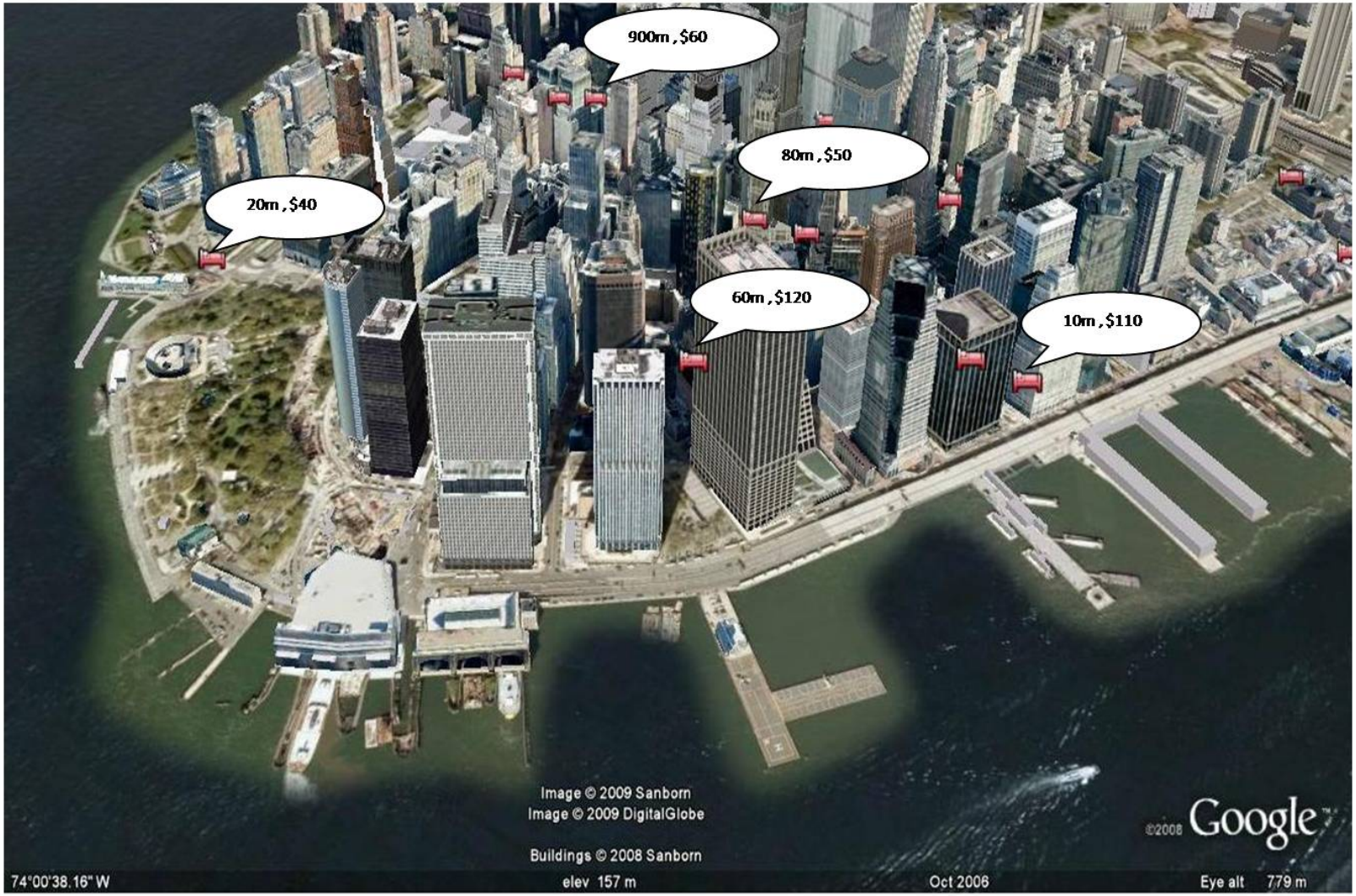


Figure: Skyline of Manhattan

Find a hotel in Manhattan, cheap and near to the beach

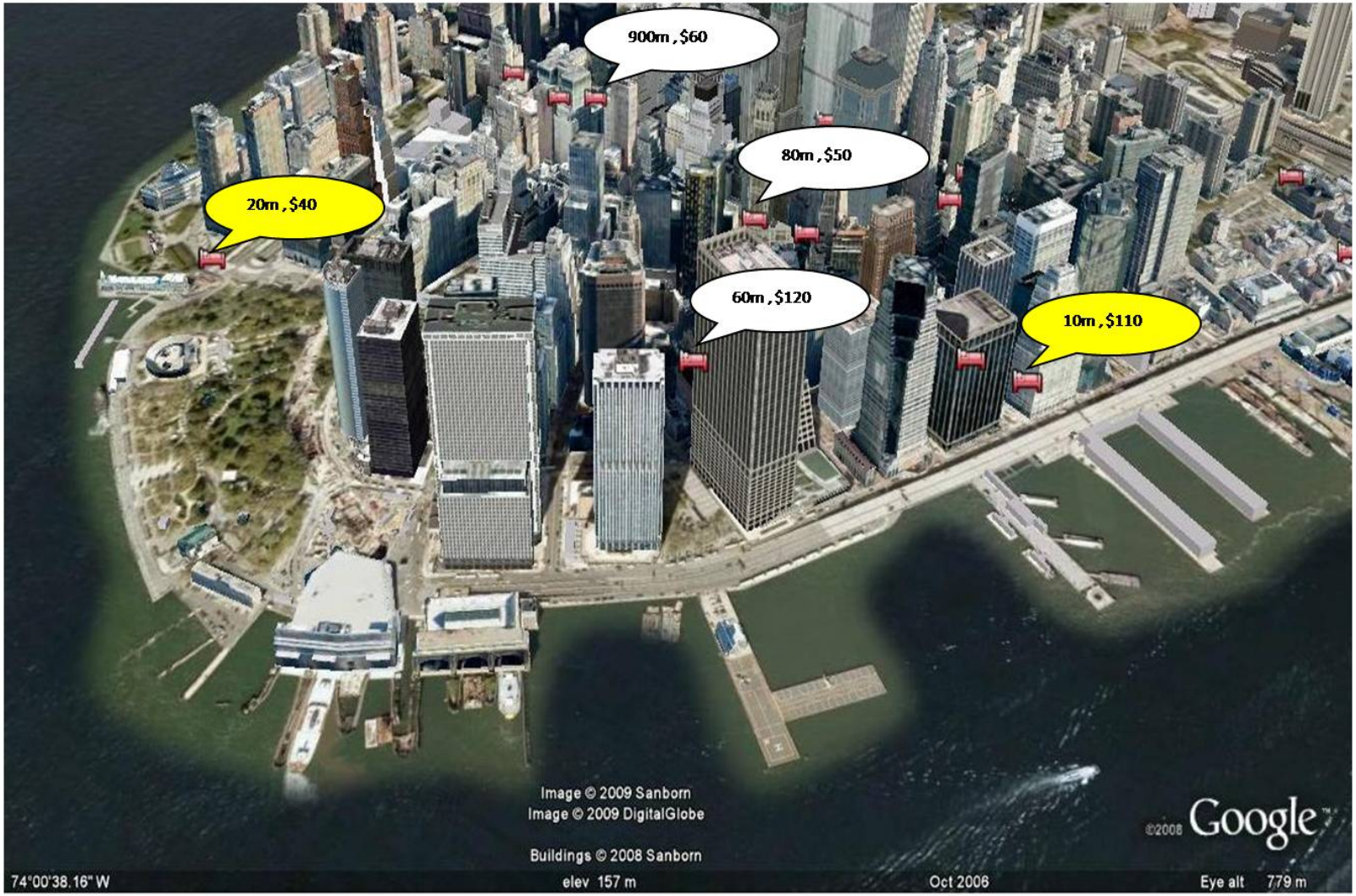


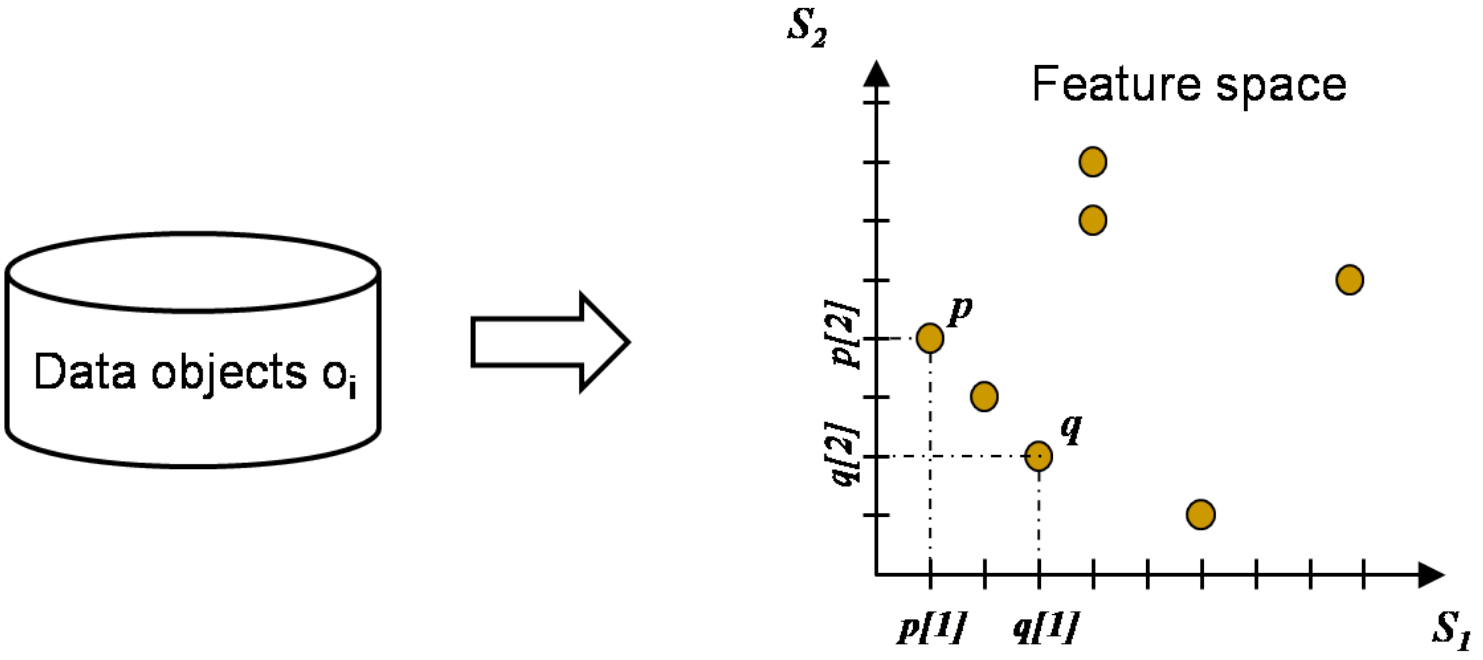
Figure: Skyline of Manhattan

Top-k Queries

- Exact best k results based on user-defined aggregate function
- Example:
  User query: top-k hotels with aggregate function distance + price
- Many centralized solutions are available
- Challenge is to support highly distributed environments like P2P
- Existing solutions for P2P are for vertical data distribution

Data representation for Top-k Queries

- Given set of data objects O with $|O| = n$
- The feature space is defined by the d scoring functions s_j
- Feature space is d-dimensional
- Each $o_i \in O$ is represented as a point p
- $p = \{p[1], ..., p[d]\}$ where $p[j] = s_j(o_i)$ $(1 \leq j \leq d)$

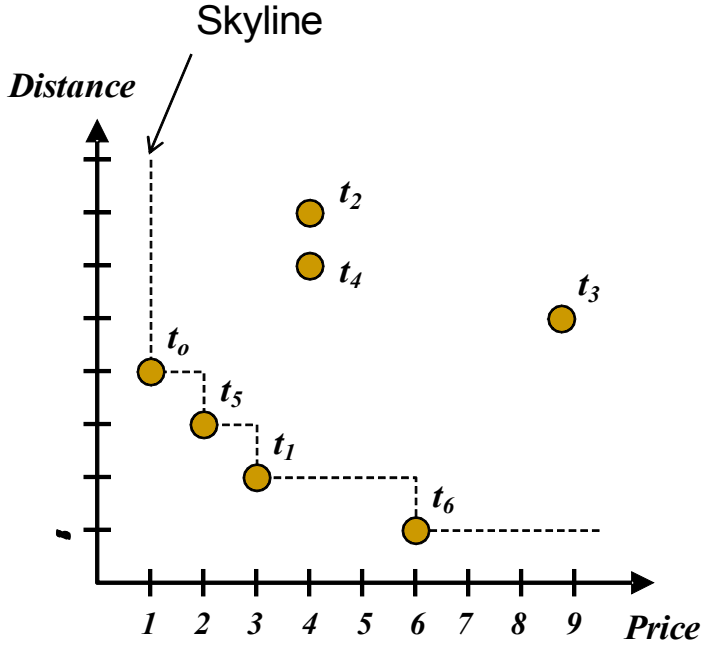


Aggregation Function

- User-defined increasingly monotone aggregate function f
- Increasingly monotone: $\forall i\ p[i] \leq p\prime[i] \Rightarrow$
  $f(p) = f(p[1], ..., p[d]) \leq f(p\prime[1], ..., p\prime[d]) = f(p\prime)$
- Example: Weighted sum over all the features (used in *SPEERTO*)
- Aggregated score, $score(o_i) = \sum_{j=1}^{d} w_j s_j(o_j)$, w_j : weight of s_j

Skyline

- Given a d-dimensional feature space D and a set of objects O
- A point $p \in O$ with $p = \{p[1], ..., p[d]\}$ dominates a point $q \in O$, if on each dimension $d_i \in D, p[i] \leq q[i]$ and at least one dimension with $p[i] < q[i]$
- The **"Skyline"** $SKY \subseteq O$ is set of points that are not dominated by any other points



Property of Skyline

Observation:

The top-1 object for any increasingly monotone function f belongs to the skyline set

Proof:

Assume $q \in top-1$ and $q \notin SKY$
f is increasingly monotone $\Rightarrow \forall d_i \in D, \exists p \in SKY$ *and* $p[i] \leq q[i]$
and at least one $d_j \in D, p[j] < q[j]$
This is a contradiction to our assumption that q is top-1

$K - Skyband$

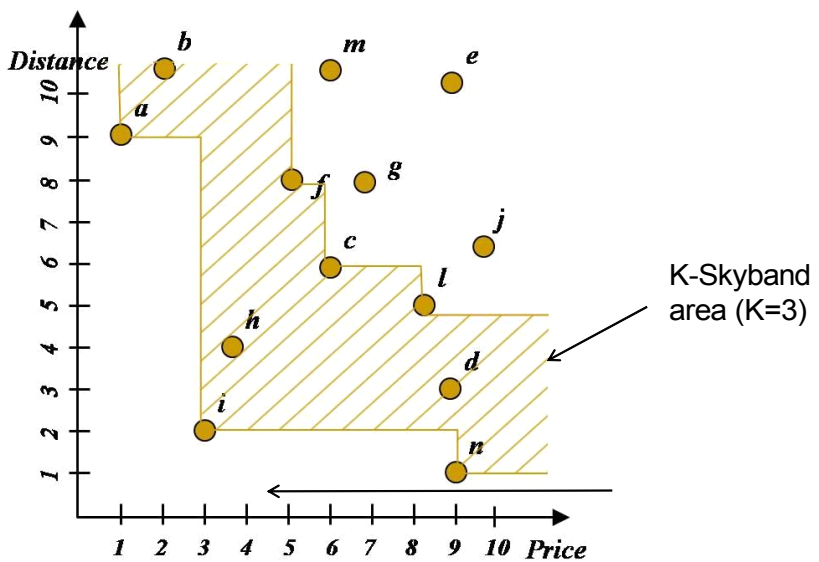
- To answer Top-k query with $k \leq K$ Skyline is not sufficient

Definition:

$K - Skyband$ is the set of points which are dominated by at most K-1 points.

- Special case: $1 - Skyband$ is *Skyline*

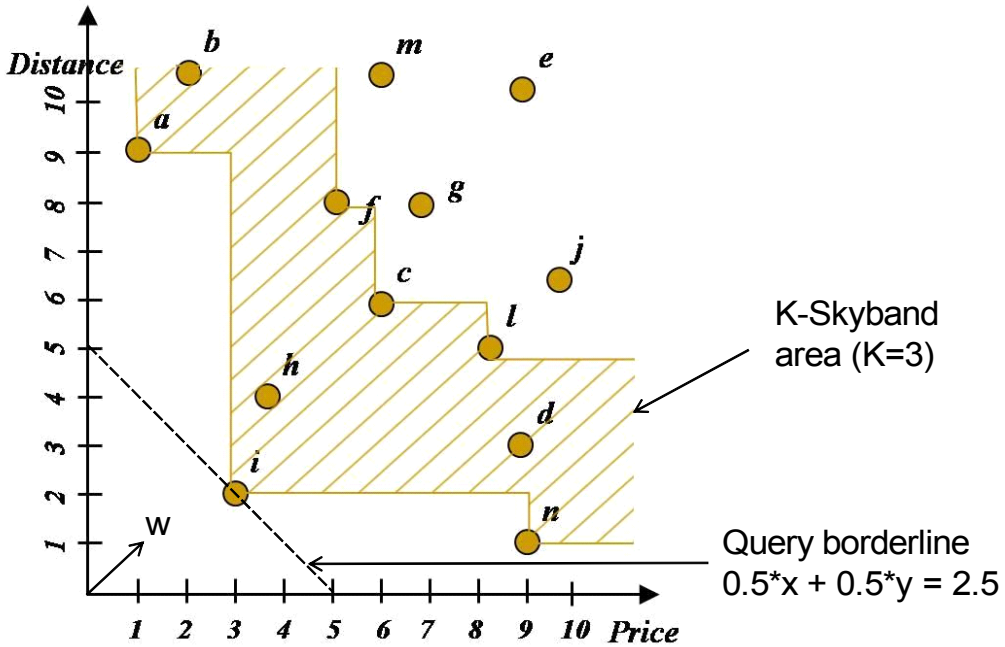
Example: $3 - Skyband$



Top-k Skyline Query

- Given a linear top-K query defined by a vector w
- Only direction of w matters, we can assume $\sum_{i=1}^{d} w_i = 1$
- In d-dimensional space, *query boderline* is d-1 dimensional hyperplane
- Query processing is sweeping *query boderline* in feature space

Example: Top-k query in 2-dimensional feature space



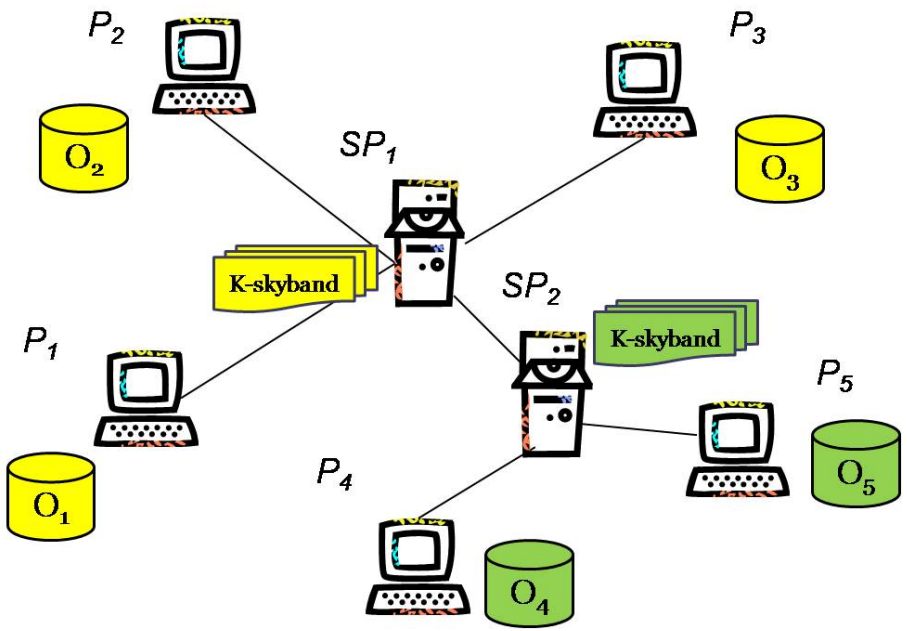
SPEERTO: System Overview

- Top-k queries in P2P networks
- Unstructured P2P network of N_p peers
- Special peers called "super-peers"
  $SP_i(1 \leq i \leq N_{sp})$ *and* $N_{sp} << N_p$
- Each super-peer maintains DEG_p links to simple peers
- Also, initially a super-peer is connected to
  $DEG_{sp}(DEG_{sp} < DEG_p)$ other super-peers
- Later, at query time each super-peer can open connection to any other super-peer

SPEERTO: System Overview

- Each peer P_i holds n_i d-dimensional points, denoted as set $O_i (1 \leq i \leq N_p)$
- Total data set $O = \bigcup O_i$ with $|O| = n = \sum_{i=1}^{N_p} n_i$
- K-skyband from N_p simple peers are merged and stored at each super-peer SP_i

Example:



Skyline based routing

- *Skyline* is used as preprocessing step
- Each peer computes its K-skyband
- Each super-peer merges all the K-skybands from its simple peers to get $KSKY_i$
- This merged data serves as routing table for query routing
- User query can be posed at any peer
- Associated super-peer handles the query routing
- Top-k queries with ($k \leq K$) are accurately answered

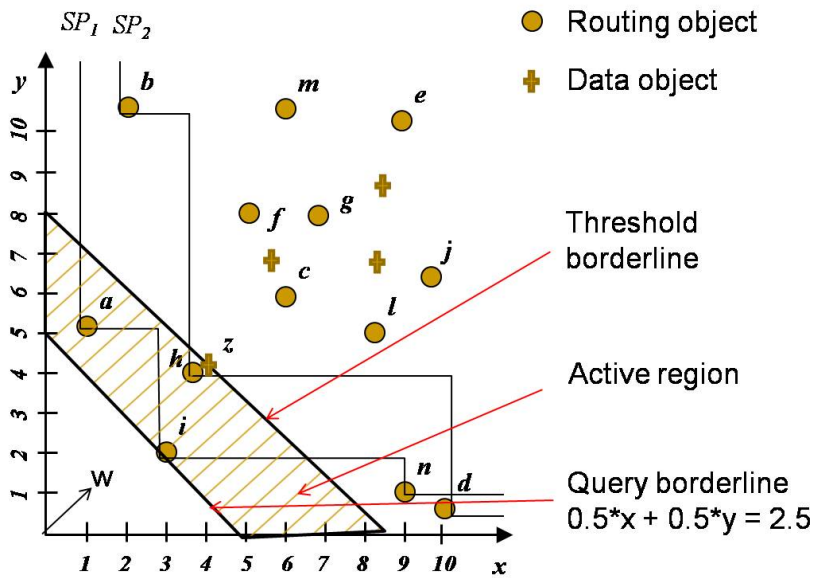
Solution Approach

- Naïve Solutions
  - Broadcast the Merged Skyband of a super-peer to other super-peers
  - Flood the query to all super-peers and get the Top-k results
- *SPEERTO* Approach
  - Broadcast only part of merged data: Skyline
  - Query only those peers which contain Top-k values

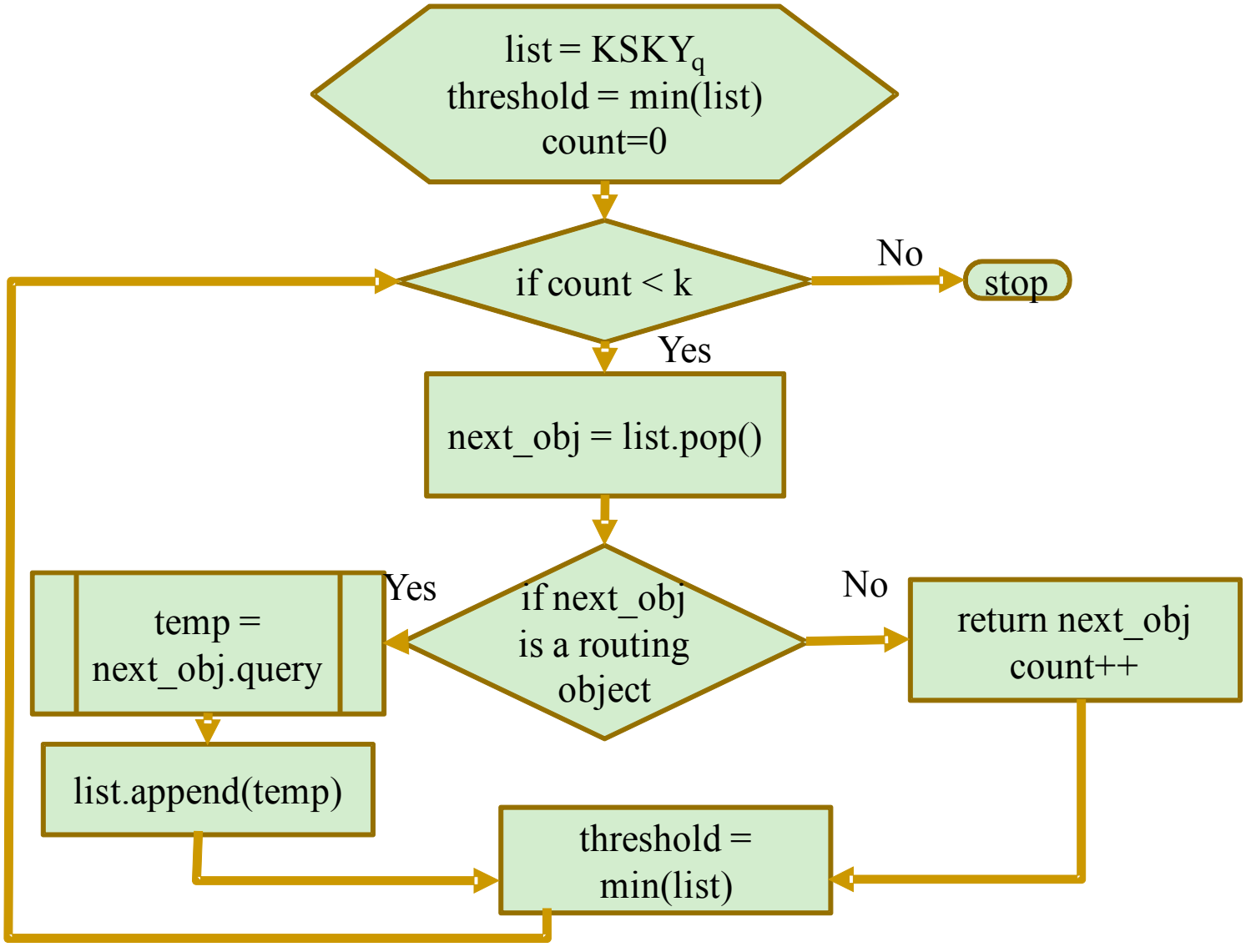
Threshold-Based Top-k

- Each super-peer SP_i assembles N_{sp} sets of skyline points $SKY_i, (1 \leq i \leq N_{sp})$, also called as *Routing Objects*
- A threshold value is defined as the score of the k-th point
- In 2-dimensional space, query borderline is sweeped to get top-k results

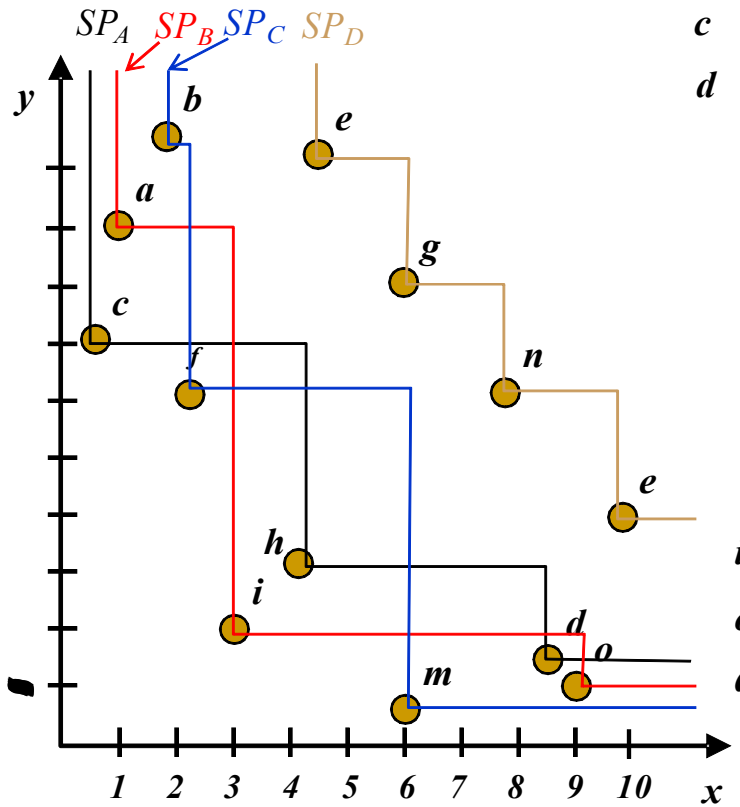
Example: A top-4 query with weights $w = (0.5, 0.5)$



Threshold-based Top-k algorithm



Top-3 Query Example



SP_A

	X	Y	Score
h	4	3	3.5
c	0.5	7	3.75
d	8	1.5	4.75
	5	4	4.5
	2	8	5
	3	8	5.5
	7	4	5.5

SP_C

	X	Y	Score
f	2	6	4
m	6	0.5	3.25
b	1.5	10	5.25
	8	1	4.5
	5	6	5.5
	4	8	6
	3	10	6.5

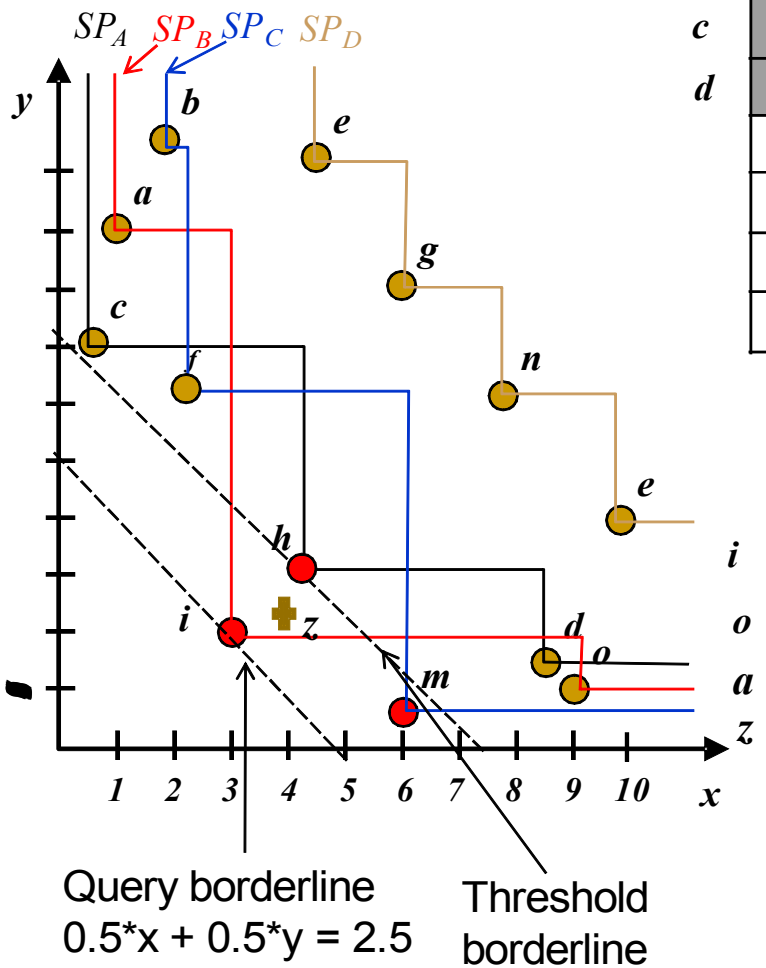
SP_B

	X	Y	score
i	3	2	2.5
o	9	1	5
a	1	9	5
	4	2.5	3.25
	7	3	5
	5	7	6
	4	9	6.5

SP_D

	X	Y	score
e	4.5	10	7.25
g	6	8	7
n	8	6	7
j	10	4	7
	7	10	8.5
	8	9	8.5
	9	8	8.5
	10	7	8.5

Top-3 Query Example



SP_A

	X	Y	Score
h	4	3	3.5
c	0.5	7	3.75
d	8	1.5	4.75
	5	4	4.5
	2	8	5
	3	8	5.5
	7	4	5.5

SP_C

	X	Y	Score
f	2	6	4
m	6	0.5	3.25
b	1.5	10	5.25
	8	1	4.5
	5	6	5.5
	4	8	6
	3	10	6.5

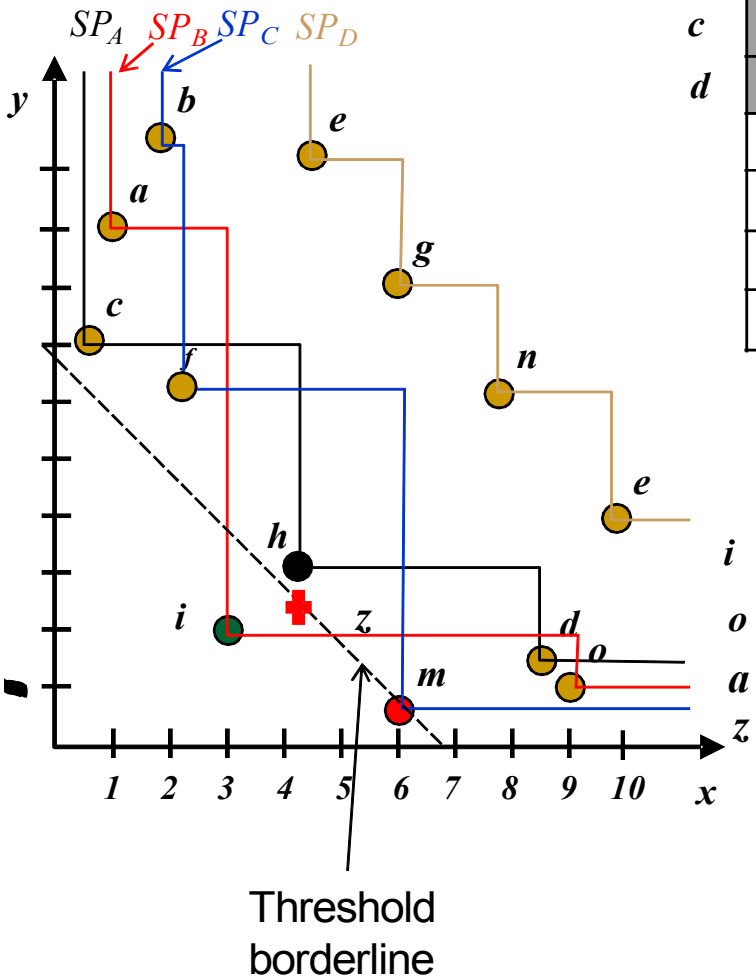
SP_B

	X	Y	score
i	3	2	2.5
o	9	1	5
a	1	9	5
z	4	2.5	3.25
	7	3	5
	5	7	6
	4	9	6.5

SP_D

	X	Y	score
e	4.5	10	7.25
g	6	8	7
n	8	6	7
j	10	4	7
	7	10	8.5
	8	9	8.5
	9	8	8.5
	10	7	8.5

Top-3 Query Example



SP_A

	X	Y	Score
h			
c	0.5	7	3.75
d	8	1.5	4.75
	5	4	4.5
	2	8	5
	3	8	5.5
	7	4	5.5

SP_C

	X	Y	Score
f	2	6	4
m	6	0.5	3.25
b	1.5	10	5.25
	8	1	4.5
	5	6	5.5
	4	8	6
	3	10	6.5

SP_B

	X	Y	score
i	3	2	2.5
o	9	1	5
a	1	9	5
z	4	2.5	3.25
	7	3	5
	5	7	6
	4	9	6.5

SP_D

	X	Y	score
e	4.5	10	7.25
g	6	8	7
n	8	6	7
j	10	4	7
	7	10	8.5
	8	9	8.5
	9	8	8.5
	10	7	8.5

Correctness and Optimality of *SPEERTO*

Correctness

- Use of skyline at each super-peer guarantees correct result

Number of queried super-peers

- A super-peer SP is queried only if next best object o in sorted list
- o is next best match so there is at least one object in SP which is part of Top-k result

Data transfer

- Only objects with score less than threshold value are transferred

SPEERTO Extensions

Parallel processing

- Linear processing of object list is blocking
- We can query more than one peers in an iteration
- Results in inaccuracy of result

Reduced Skyline cardinality

- $|SKY|$ is very high
- Solution is to find an approximation of the skyline
- Approximate skyline is robust to data updates

SPEERTO: Parallel Query processing

1. Compute mean score $m = \frac{t - score(o_1)}{N_r}$, t - Threshold value, $score(o_1)$- score of top-1 object, N_r- number of retrieved objects

SPEERTO: Parallel Query processing

1. Compute mean score $m = \frac{t - score(o_1)}{N_r}$, t - Threshold value, $score(o_1)$- score of top-1 object, N_r- number of retrieved objects
2. If next object o_2 is routing object then compute $\overline{N_{r2}} = \frac{score(o_2) - score(o_1)}{m}$

SPEERTO: Parallel Query processing

1. Compute mean score $m = \frac{t - score(o_1)}{N_r}$, t - Threshold value, $score(o_1)$- score of top-1 object, N_r- number of retrieved objects
2. If next object o_2 is routing object then compute $\overline{N_{r2}} = \frac{score(o_2) - score(o_1)}{m}$
3. else if o_2 is data object, then process super-peers found so far

SPEERTO: Parallel Query processing

1. Compute mean score $m = \frac{t - score(o_1)}{N_r}$, t - Threshold value, $score(o_1)$- score of top-1 object, N_r- number of retrieved objects
2. If next object o_2 is routing object then compute $\overline{N_{r2}} = \frac{score(o_2) - score(o_1)}{m}$
3. else if o_2 is data object, then process super-peers found so far
4. If next object o_3 is routing object then compute $\overline{N_{r3}} = \frac{score(o_3) - score(o_2)}{m}$

SPEERTO: Parallel Query processing

1. Compute mean score $m = \frac{t - score(o_1)}{N_r}$, t - Threshold value, $score(o_1)$- score of top-1 object, N_r- number of retrieved objects
2. If next object o_2 is routing object then compute $\overline{N_{r2}} = \frac{score(o_2) - score(o_1)}{m}$
3. else if o_2 is data object, then process super-peers found so far
4. If next object o_3 is routing object then compute $\overline{N_{r3}} = \frac{score(o_3) - score(o_2)}{m}$
5. repeat 1 to 4 until $\overline{N_{r2}} + \overline{N_{r3}} > (k - c)$

SPEERTO: Parallel Query processing

1. Compute mean score $m = \frac{t - score(o_1)}{N_r}$, t - Threshold value, $score(o_1)$- score of top-1 object, N_r- number of retrieved objects
2. If next object o_2 is routing object then compute $\overline{N_{r2}} = \frac{score(o_2) - score(o_1)}{m}$
3. else if o_2 is data object, then process super-peers found so far
4. If next object o_3 is routing object then compute $\overline{N_{r3}} = \frac{score(o_3) - score(o_2)}{m}$
5. repeat 1 to 4 until $\overline{N_{r2}} + \overline{N_{r3}} > (k - c)$
6. A new mean is computed $m\prime = \frac{m + m_1 + m_2}{3}$, $m_2 = \frac{t - score(o_2)}{N_{r2}}$ and $m_3 = \frac{t - score(o_1)}{N_{r3}}$

SPEERTO: Parallel Query processing

1. Compute mean score $m = \frac{t - score(o_1)}{N_r}$, t - Threshold value, $score(o_1)$- score of top-1 object, N_r- number of retrieved objects
2. If next object o_2 is routing object then compute $\overline{N_{r2}} = \frac{score(o_2) - score(o_1)}{m}$
3. else if o_2 is data object, then process super-peers found so far
4. If next object o_3 is routing object then compute $\overline{N_{r3}} = \frac{score(o_3) - score(o_2)}{m}$
5. repeat 1 to 4 until $\overline{N_{r2}} + \overline{N_{r3}} > (k - c)$
6. A new mean is computed $m\prime = \frac{m + m_1 + m_2}{3}$, $m_2 = \frac{t - score(o_2)}{N_{r2}}$ and $m_3 = \frac{t - score(o_1)}{N_{r3}}$
7. A new Threshold $t\prime$ is computed

SPEERTO: Parallel Query processing

1. Compute mean score $m = \frac{t - score(o_1)}{N_r}$, t - Threshold value, $score(o_1)$- score of top-1 object, N_r- number of retrieved objects
2. If next object o_2 is routing object then compute $\overline{N_{r2}} = \frac{score(o_2) - score(o_1)}{m}$
3. else if o_2 is data object, then process super-peers found so far
4. If next object o_3 is routing object then compute $\overline{N_{r3}} = \frac{score(o_3) - score(o_2)}{m}$
5. repeat 1 to 4 until $\overline{N_{r2}} + \overline{N_{r3}} > (k - c)$
6. A new mean is computed $m\prime = \frac{m + m_1 + m_2}{3}$, $m_2 = \frac{t - score(o_2)}{N_{r2}}$ and $m_3 = \frac{t - score(o_1)}{N_{r3}}$
7. A new Threshold $t\prime$ is computed
8. Repeat until k objects are found

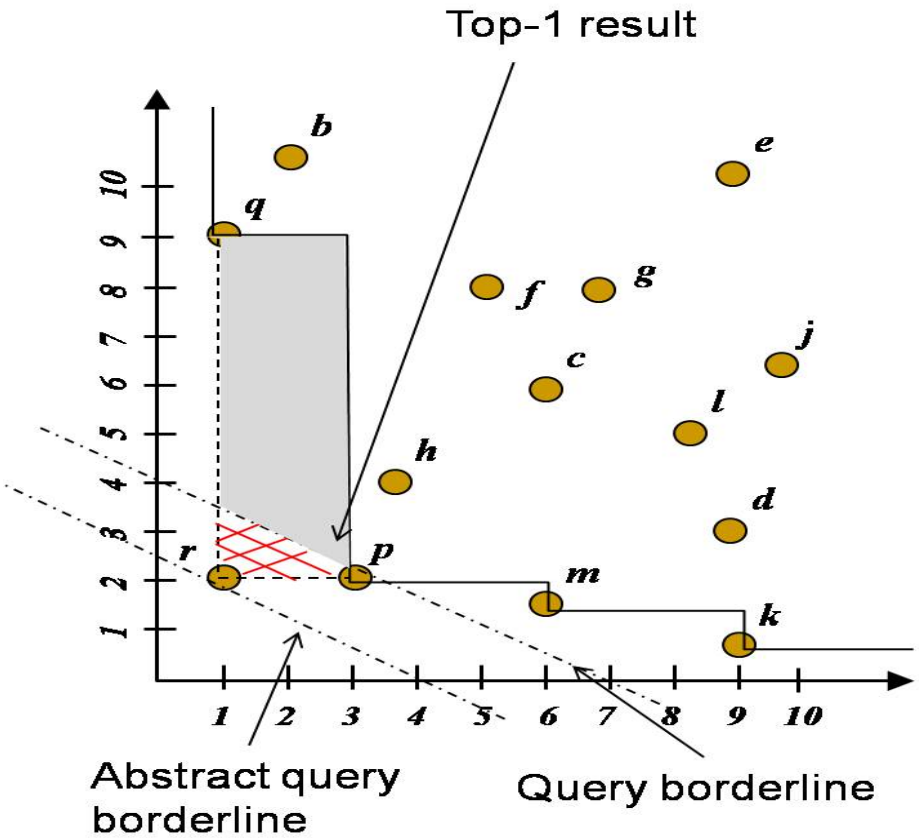
Reducing skyline cardinality

Given an upper limit U, abstract the skyline $aSKY$ with at most U points ($U < |SKY|$)

- each point $p \in SKY$ is either dominated by or equal to at least one point $q \in aSKY$
- $|aSKY| \leq U < |SKY|$
- It only slightly influences the routing power of the skyline
- There are many ways to find $aSKY$

How to Abstract The Skyline?

- Consider an example with q, p, m, k as skyline
- Given an upper limit $U = 3$
- Suppose we decide to replace q, p with one point r
- To ensure accuracy the super-peers will also be contacted based on abstraction



Heuristic to Abstract Skyline

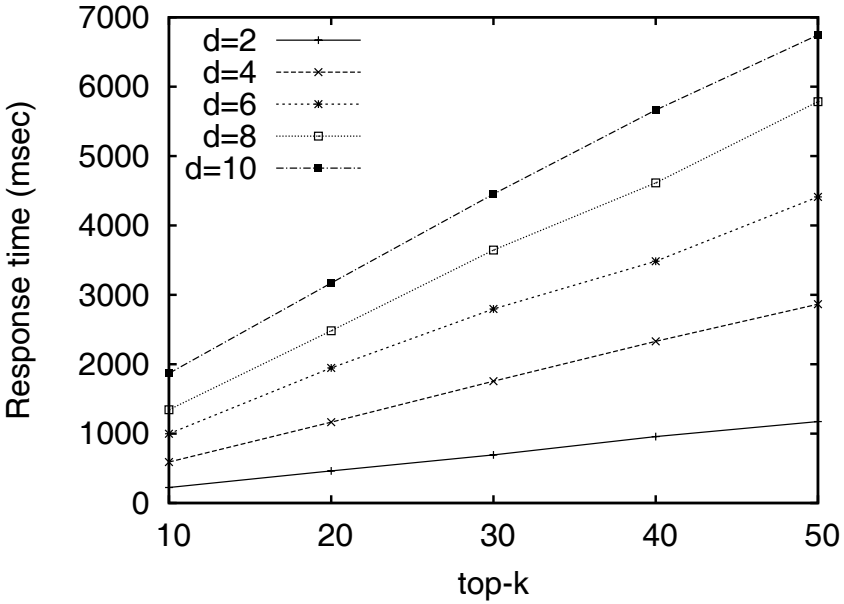
1. Choose a point p with largest entropy value $E(p) = argmax_{\forall t \in SKY_i}(\sum_{1 \leq i \leq d} ln(p[i] + 1))$
2. Choose another point q with minimum distance $dist = min_{1 \leq i \leq d}(|p[i] - q[i]|), \forall t \in SKY_i$
3. Then replace p and q with r with $r[i] = min(p[i], q[i]), (1 \leq i \leq d)$
4. Iterate and terminate when $U < |SKY|$

Experimental Setup

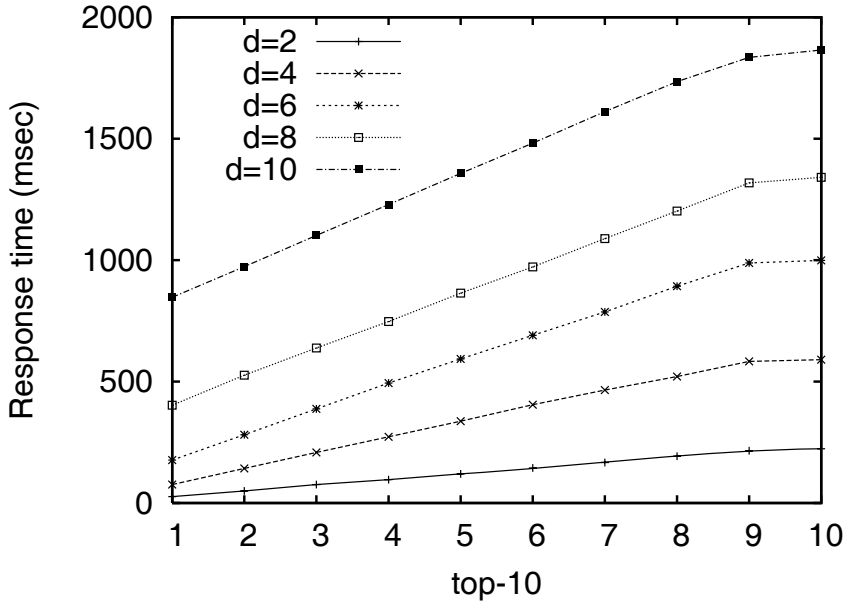
- Experiments were done on simulator running on single machine
- Data was horizontally partitioned, evenly among peers
- Uniform data: random points in a space $[0, L]^d$
- Clustered data:
  - Super-peer picks cluster centroids randomly
  - All associated peers obtain points based on Gaussian distribution
- Default values:
  $d = 4, K = 50, 10 \leq k \leq 50, n = 10^6, n_p = 2000, N_{sp} = 0.1N_p$

SPEERTO Performance - Response Time

- Response time increases with dimensionality
- Initial results are returned immediately



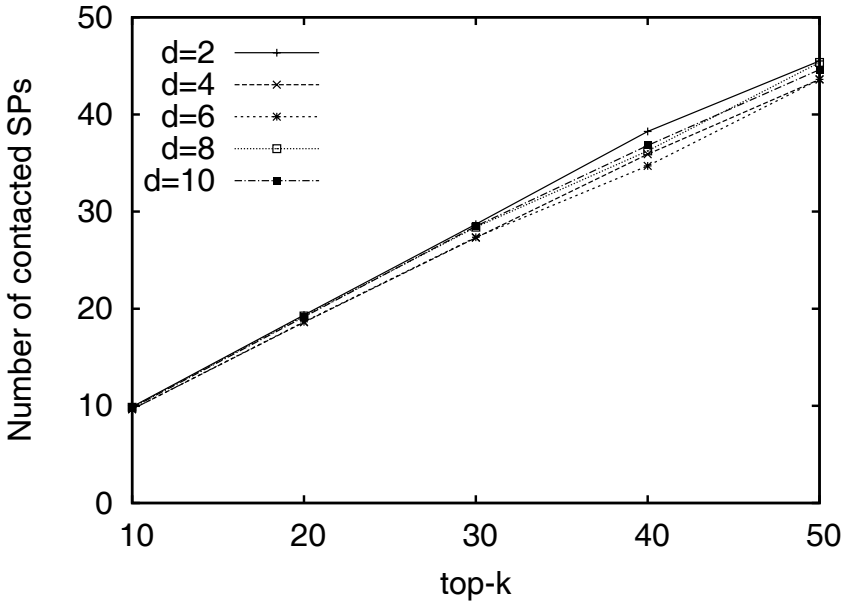
Response time with d



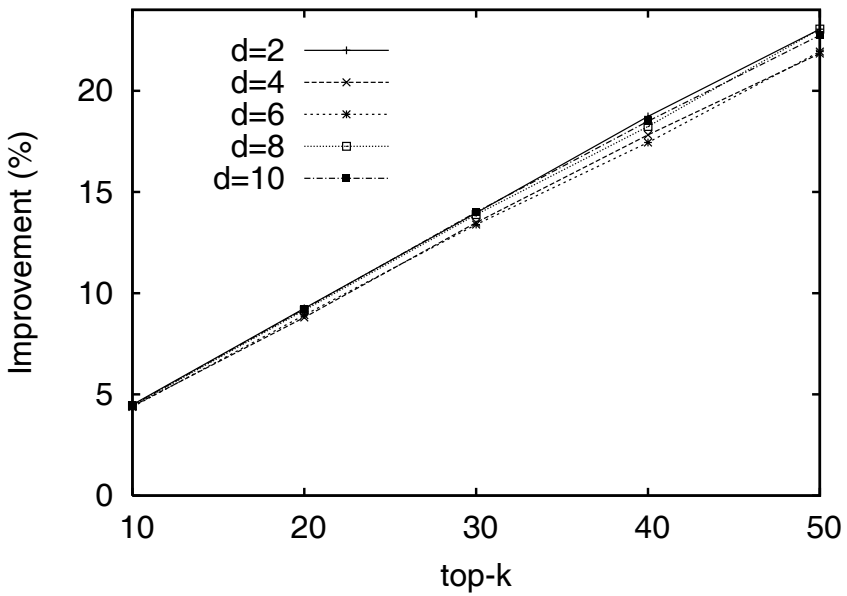
Response time for first 10 objects

Effectiveness of Threshold Based Algorithm

- Number of contacted super-peers increase slightly with d
- Gain in number of transferred objects is around 21.9 for k=50



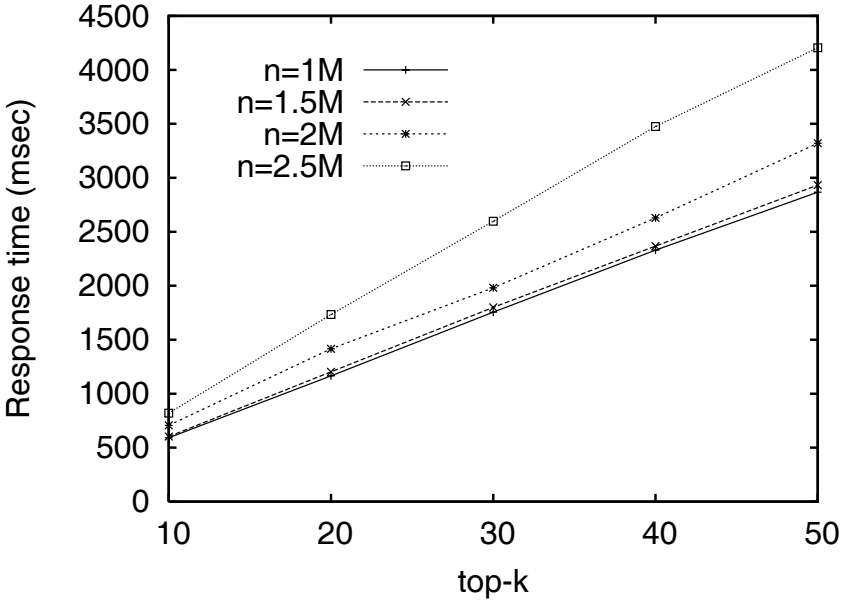
Number of contacted SPs



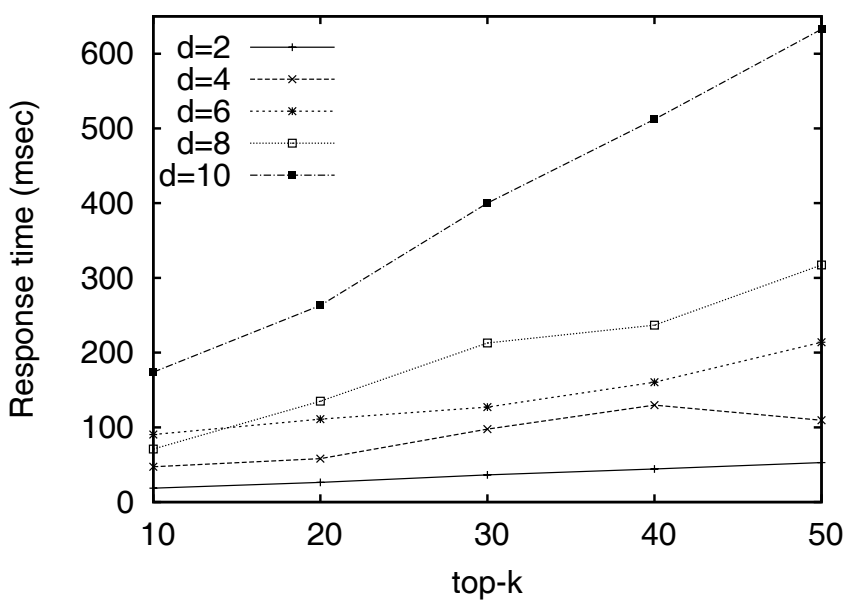
Improvement in number of objects transferred

Scalability With Data Cardinality and Response Rime With Clustered Data

- For n = 1M to 2.5M response time slightly increases
- *SPEERTO* performs better for clustered data



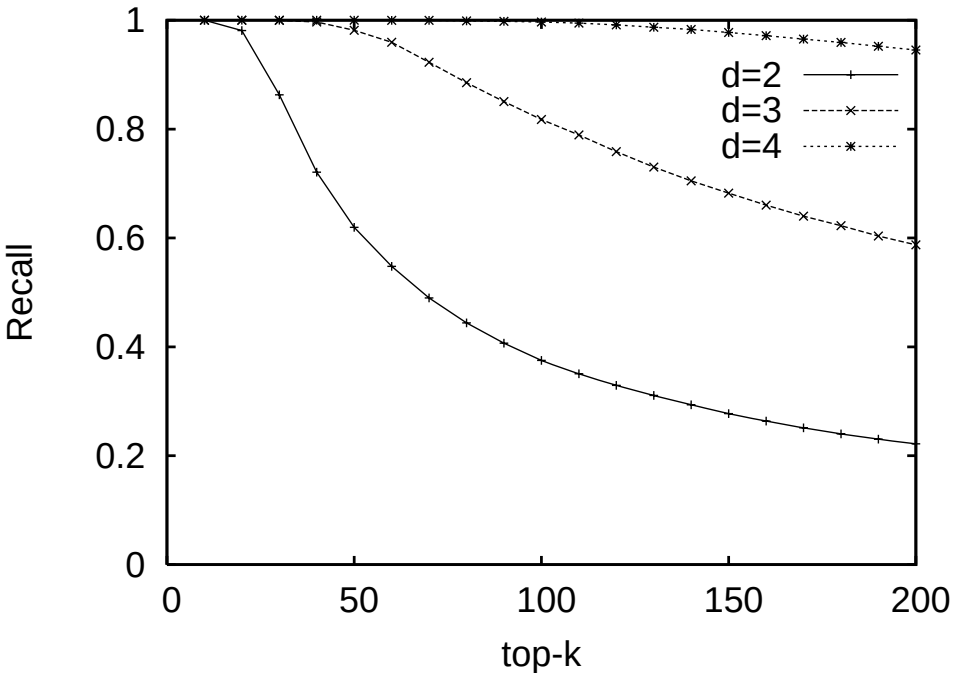
Scalability with cardinality



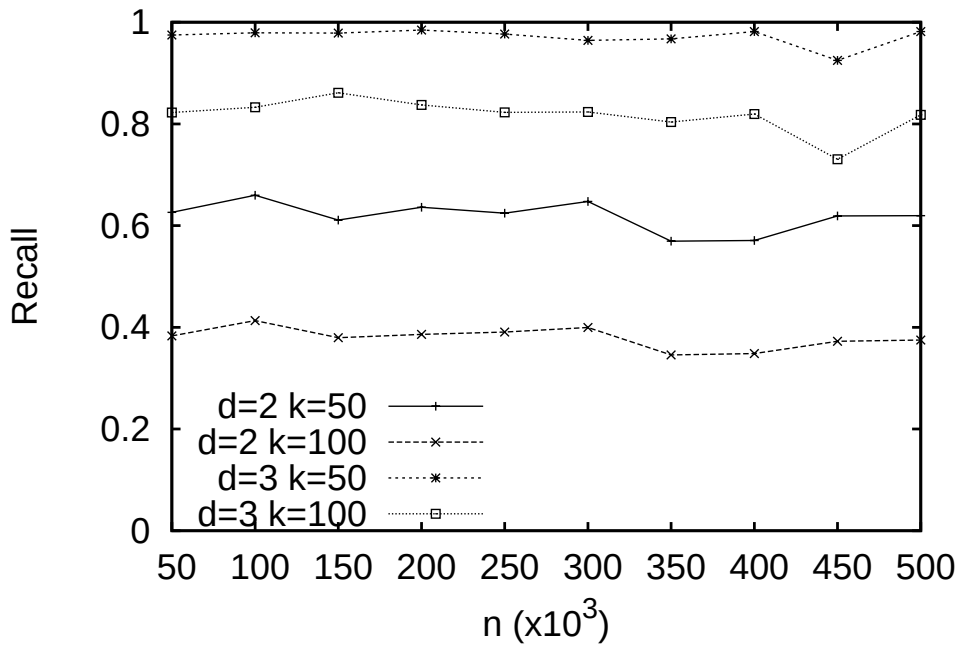
Clustered dataset

Top-k with $k > K$

- Relative recall: $\frac{|ReturnedTop-k \in TrueTop-k|}{|TrueTop-k|}$
- with $K = 10$, $k = 100$ and $d = 2$ relative recall is around 40%
- Scalable for varying data even with $k > K$



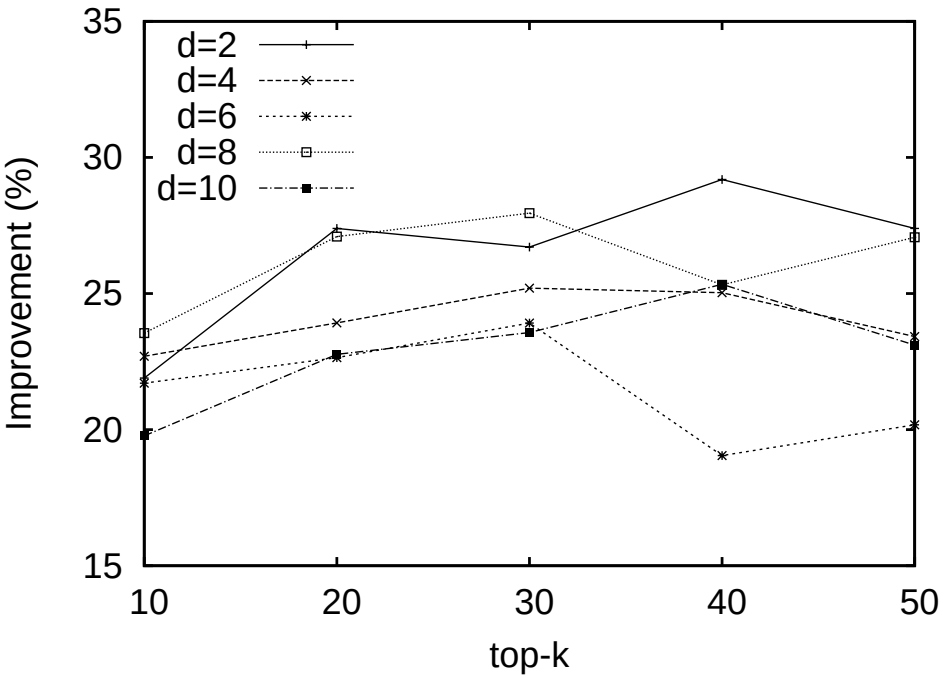
Recall for uniform data



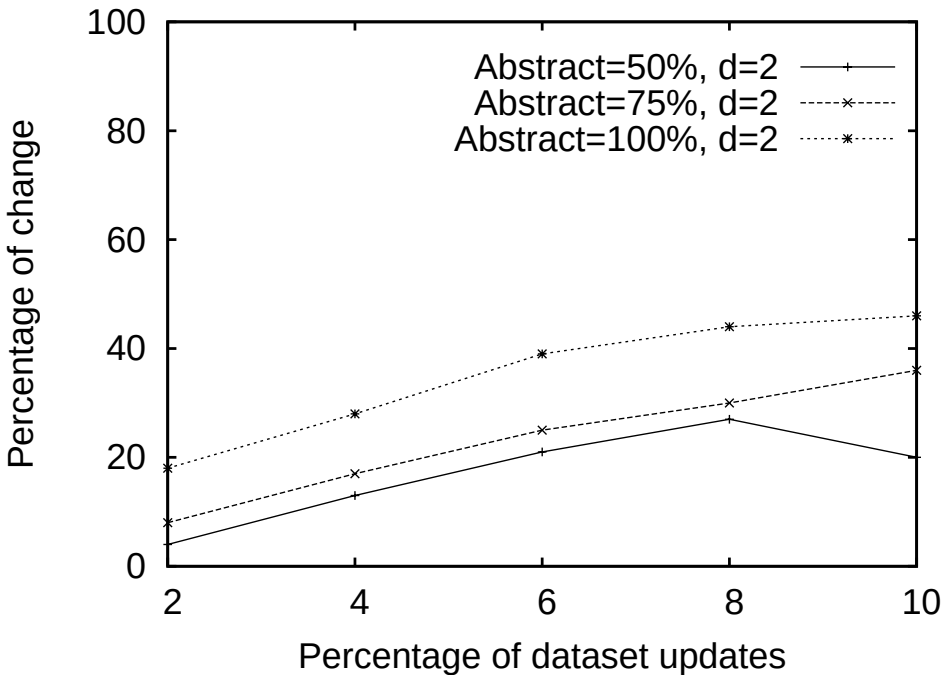
Recall for varying data

Performance With Extensions

- Parallel variant: Response time is improved by 20%, with only small increase in objects transferred
- Abstract variant: with d=2, 2% of dataset update and 50% abstraction, only 4% super-peers need to update skyline



Improvement in response time with parallel version



Data updates for $d = 2$

Conclusions

- A novel approach for answering top-k queries in a P2P network based on super-peer architecture
- A threshold-based algorithm which forwards the top-k query requests among super-peers efficiently
- A variant of *SPEERTO* that queries in parallel
- An extension that restricts the cardinality of the skyline
- *SPEERTO* scales well to bigger systems
- *SPEERTO* performs considerably good even with $k > K$

Thank You!

References

- Akrivi Vlachou, Christos Doulkeridis, Kjetil N, Michalis Vazirgiannis: On efficient top-k query processing in highly distributed environments. SIGMOD 2008, pp. 753-764
- S. B orzs onyi, D. Kossmann, and K. Stocker. The skyline operator. In Proceedings of IEEE Int. Conf. on Data Engineering (ICDE), pages 421-430, 2001.
- D. Papadias, Y. Tao, G. Fu, and B. Seeger. Progressive skyline computation in database systems. ACM Transactions on Database Systems, 30(1):41-82, 2005

Backup slides

Skyline based routing with Updates and Churn

Updates

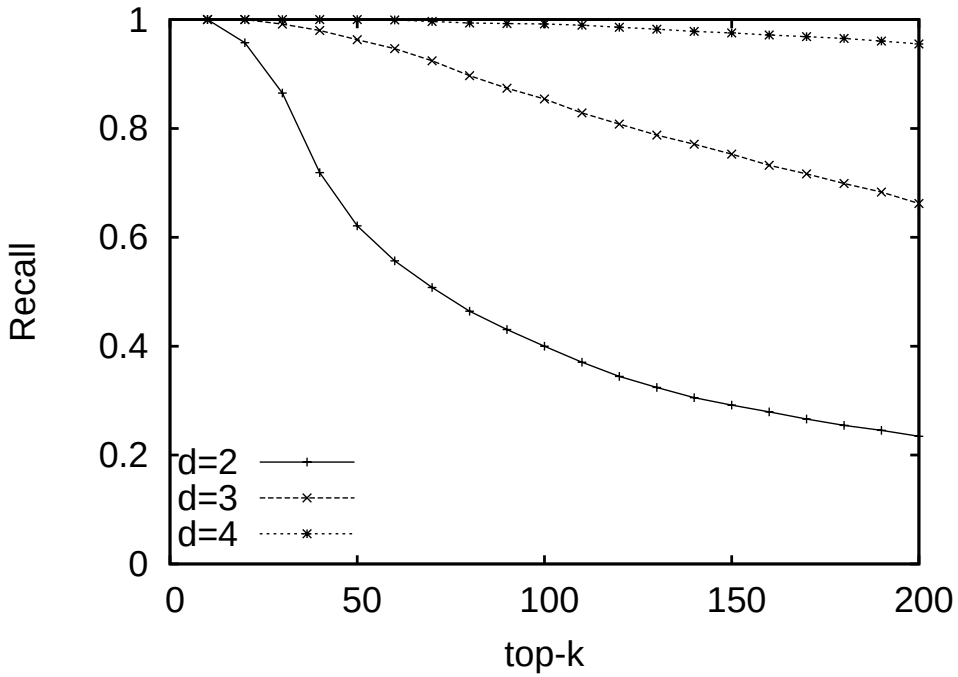
- Periodic updates of skyline suffices
- Broadcast the skyline updates, when either the skyline has significantly changed or the validity time has expired.
- K-skyband update at each super-peer is done more frequently

Churn

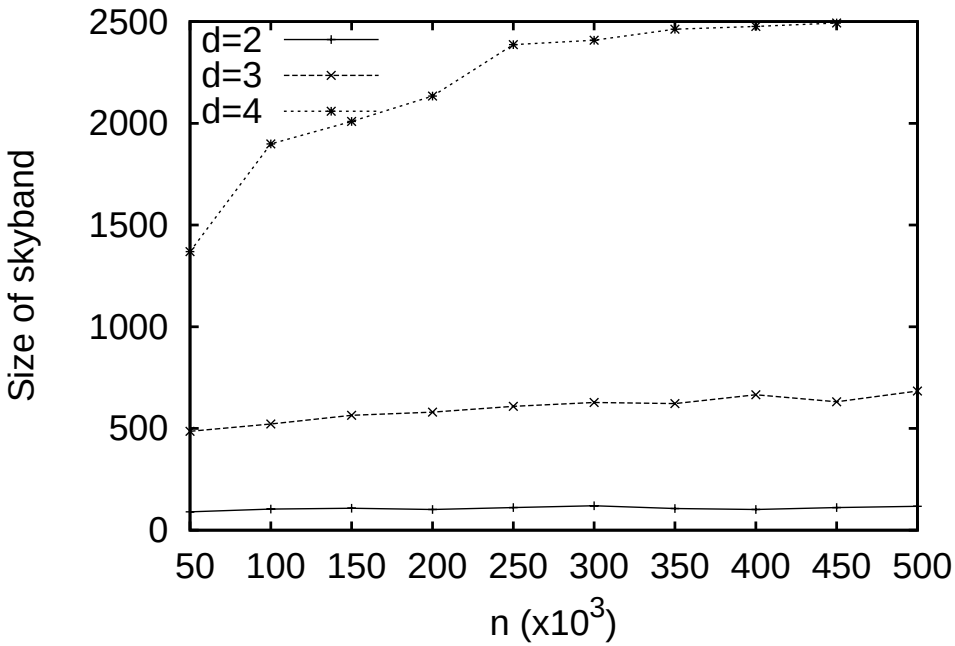
- The skyline entry of the departed super-peer is removed at the querying super-peer
- When a super-peer joins the network, its skyline is broadcast
- Churn of simple peers is handled by recomputing the super-peer K-skyband

Top-k with $k > K$: More results

- Clustered datasets result in similar recall values as uniform data
- By varying cardinality the skyband size is hardly inluenced



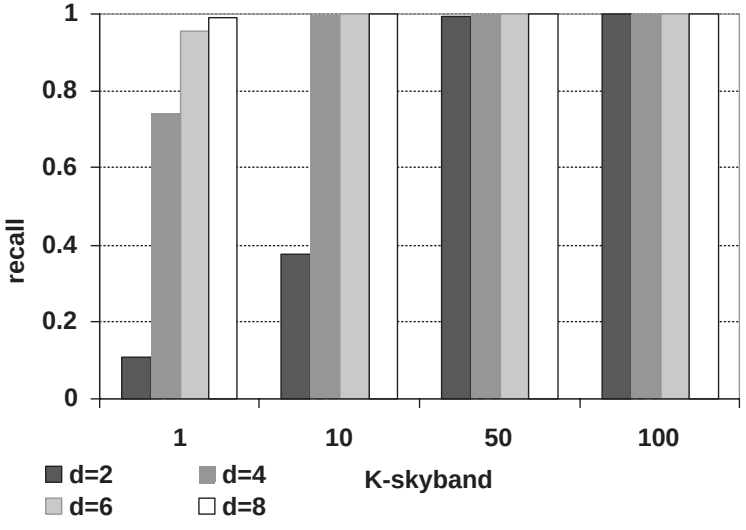
Recall for clustered data



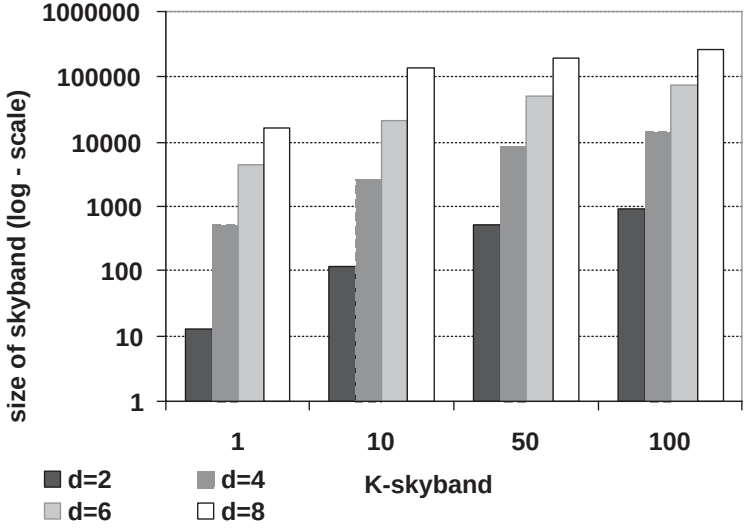
Size for varying n

Top-k with $k > K$: More results

- Recall values increase with dimensionality because the size of the K-skyband also increases
- For d = 2 recall is low since the skyline consists of only 13 points



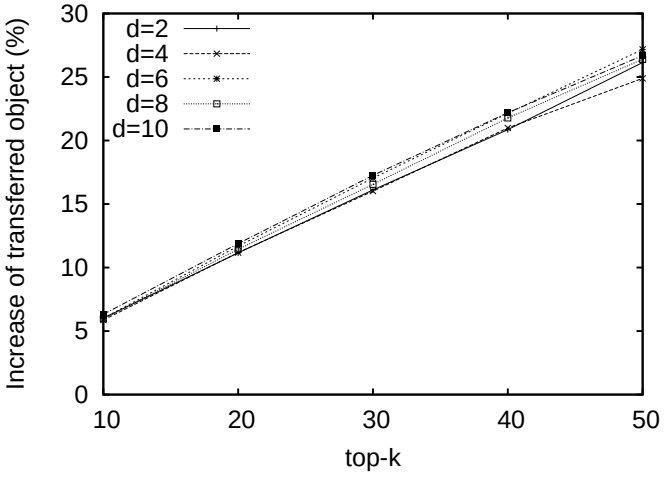
Recall for varying K



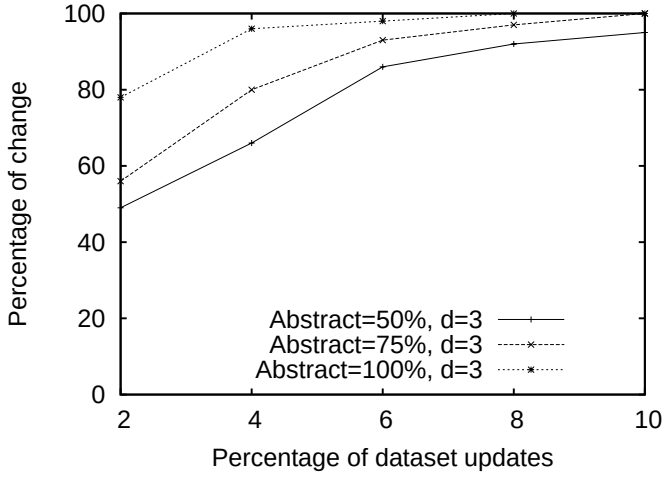
Size for varying K

Abstract Skyline Performance

- In abstract variant The number of transferred objects increases rapidly, since the threshold is not used
- For d = 3 the number of modified super-peers increases, but again the gain of abstraction is significant



Variant with abstract skyline



Data updates for d = 3