Subspace Similarity Search: Efficient k-NN Queries in Arbitrary Subspaces

Thomas Bernecker, Tobias Emrich, Franz Graf, Hans-Peter Kriegel,
Peer Kröger, Matthias Renz, Erich Schubert, and Arthur Zimek

Institut für Informatik, Ludwig-Maximilians-Universität München
{bernecker,emrich,graf,kriegel,kroeger,renz,schube,zimek}@dbs.ifi.lmu.de
http://www.dbs.ifi.lmu.de

Abstract. There are abundant scenarios for applications of similarity search in databases where the similarity of objects is defined for a subset of attributes, i.e., in a subspace, only. While much research has been done in efficient support of single column similarity queries or of similarity queries in the full space, scarcely any support of similarity search in subspaces has been provided so far. The three existing approaches are variations of the sequential scan. Here, we propose the first index-based solution to subspace similarity search in arbitrary subspaces.

1 Introduction

In the last decade, similarity search in high-dimensional data has gained special interest. Several studies for research on data structures [1,2,3,4,5] showed that the suitability of the sequential scan [6] compared to methods using partitioning or clustering based data structures is dependent of the characteristics of the data distributions. However, this key message has been neglected in many research contributions [7,8,9,10,11,12]. Thus, it still appears to be well worth noting that nearest neighbor search is meaningful if and only if the nearest neighbor of the arbitrary query object is sufficiently different from its farthest neighbor. This is in general the case whenever a data set exhibits a natural structure in clusters or groupings of subsets of data.

While much effort has been spent on studying possibilities to facilitate efficient similarity search in high-dimensional data, scarcely ever the question arose how to support similarity search when the similarity of objects is based on a subset of attributes only. Aside from fundamentally studying the behavior of data structures in such settings, this is a practically highly relevant question. It could be interesting for any user to search, e.g., in a database of images represented by color-, shape-, and texture-descriptions, for objects similar to a certain image where the similarity is related to the shape of the motifs only but not to their color or even the color of the background. An online-store might like to propose similar objects to a customer where similarity can be based on different subsets of features. While in such scenarios, meaningful subspaces can be suggested beforehand [13,14], in other scenarios, possibly any subspace could be interesting.

M. Gertz and B. Ludäscher (Eds.): SSDBM 2010, LNCS 6187, pp. 555–564, 2010.

For example, for different queries, different regions of interest in a picture may be relevant. Since there are 2^D possible subspaces of a D-dimensional data set, it is practically impossible to provide data structures for each of these possible subspaces in order to facilitate efficient similarity search. Another application where efficient support of subspace similarity queries is required are many subspace clustering algorithms [15] that rely on searching for clusters in a potentially large number of subspaces (starting with all 1D subspaces, many combinations of 1D subspaces to 2D subspaces etc.). If efficient support of subspace range queries or subspace nearest neighbor queries were available, virtually all subspace cluster approaches could be accelerated considerably. Note that this problem is essentially different from the feature selection problem [15, 16].

In this paper, we facilitate efficient *subspace similarity search* in large and potentially high-dimensional data sets where the user or the application can define an interesting subspace for each query independently (that is, similarity is defined ad hoc based on an arbitrary subset of attributes only). To this end, we extend our preliminary approach addressed in [17] with more thorough experimental evaluation and we propose a top-down indexing method to support subspace similarity search.

In the remainder, we formally define the problem of subspace similarity search in Section 2. We discuss related work and the algorithmic sources of inspiration to our new solution in Section 3. We propose an index-based top-down approach as an adaptation of the R-tree in Section 4 and, additionally, give a general and theoretical comparison of this approach with [17]. An experimental evaluation of all methods is presented in Section 5. Section 6 concludes the paper.

2 Subspace Similarity Search

A common restriction for the small number of approaches tackling subspace similarity search (see Section 3) is that L_p-norms are assumed as distance measures. Hence we will also rely on this restriction in the problem definition. In the following, we assume that $\mathcal{DB}$ is a database of N objects in a D-dimensional space $\mathbb{R}^D$ and the distance between points in $\mathcal{DB}$ is measured by a distance function $dist : \mathbb{R}^D \times \mathbb{R}^D \rightarrow \mathbb{R}_0^+$ which is one of the L_p-norms ($p \in [1, \infty)$). In order to perform subspace similarity search, a d-dimensional query subspace will be represented by a D-dimensional bit vector S of weights, where d weights are 1 and the remaining $D - d$ weights are 0. Formally:

Definition 1 (Subspace). *A* subspace S *of the D-dimensional data space is represented by a vector $S = (S_1, \ldots, S_D) \in \{0, 1\}^D$, where $S_i = 1$, if the ith attribute is an element of the subspace, and $S_i = 0$, otherwise. The number d of 1 entries in S, i.e., $d = \sum_{i=1}^{D} S_i$ is called the* dimensionality *of S.*

For example, in a 3D data space, the 2D subspace representing the projection on the first and third axis is represented by $S = (1, 0, 1)$.

A distance measure for a subspace S can then be figured as weighted L_p-norm where the weights can either be 1 (if this particular attribute is relevant to the query) or 0 (if this particular attribute is irrelevant to the query), formally:

Definition 2 (Subspace Distance). *The distance in a subspace S between two points $x, y \in \mathcal{DB}$ is given by $dist_S(x,y) = \sqrt[p]{\sum_{i=1}^{d} S_i \, |x_i - y_i|^p}$, where x_i, y_i, and S_i denote the values of the* ith *component of the vectors x, y, and S.*

Thus, a subspace k-nearest neighbor (k-NN) query can be formalized as:

Definition 3 (Subspace k-NN Query). *Given a query object q and a d-dimensional ($d \leq D$) query subspace represented by a corresponding vector S of weights, a subspace k-NN query retrieves the set $NN(k,S,q)$ that contains k objects from $\mathcal{DB}$ for which the following condition holds: $\forall o \in NN(k,S,q), \forall o' \in \mathcal{DB} \setminus NN(k,S,q) : dist_S(o,q) \leq dist_S(o',q)$.*

Some of the rare existing approaches for subspace similarity search focus on ε-range queries. This is a considerable lack because choosing the number k of results that should be returned by a query is usually much more intuitive than selecting some query radius ε. Furthermore, the value of ε needs to be adjusted to the subspace dimensionality in order to produce meaningful results. This is a non-trivial task since recall and precision of an ε-sphere become highly sensitive to even small changes of ε depending on the dimensionality of the data space. In addition, many applications like data mining algorithms that further process the results of subspace similarity queries require to control their cardinality [15].

3 Related Work

Established index structures (such as [18,19,20,21]) are designed and optimized for the complete data space where all attributes contribute to partitioning, clustering etc. For these data structures, the space of queries facilitated by the index structure must be fixed prior to the construction of the index structure. While the results of research on such index structures designed for one single query space are abundant [22], so far there are some variations of the sequential scan addressing the problem of *subspace similarity search*, implicitly or explicitly.

The *Partial VA-File* [23] as an adaptation of the VA-file [6] is the first approach addressing the problem of subspace similarity search *explicitly*. The basic idea of this approach is to split the original VA-file into D *partial* VA-files, where D is the data dimensionality, i.e. we get one file for each dimension containing the approximation of the original full-dimensional VA-file in that dimension. Based on this information, upper and lower bounds of the true distance between data objects and the query are derived. Subspace similarity queries are processed by scanning only the relevant files in the order of relevance, i.e. the files are ranked by the selectivity of the query in the corresponding dimension. As long as there are still candidates that cannot be pruned or reported using the upper and lower distance bounds, the next ranked file is read to improve the distance approximations or (if all partial VA-files have been scanned) the exact information of the candidates accessed to refine the exact distance.

Another approach to the problem is proposed in [24], although only ε-similarity range queries are supported. The idea of this multi-pivot-based method is to

derive lower and upper bounds for distances based on the average minimal and maximal impact of a possible range of d dimensions, $d \in [d_{\min}, d_{\max}]$. The bounds are computed in a preprocessing step for a couple of pivot points. To optimize the selection of pivot points, also a distribution of possible values for ε is required. The lower and upper bounds w.r.t. all pivot points are annotated to each database object. Essentially, this approach allows to sequentially scan the database reading only the information on lower and upper bounds and to refine the retrieved candidates in a postprocessing step.

The solution to subspace similarity search we are proposing in this paper is based on the ad hoc combination of 1D index structures. The combination technique is algorithmically inspired by top-k queries on a number of different rankings of objects according to different criteria. Let us assume that we have a set of objects that are ranked according to m different score functions (e.g. different rankings for m different attributes). The objective of a top-k query is to retrieve the k objects with the highest combined (e.g. average) score. In our scenario, if we assume the objects are ranked for each dimension according to the distance to the query object, respectively, we can apply top-k methods to solve subspace k-NN queries with the rankings of the given subspace dimensions. For the top-k query problem, there basically exist two modes of access to the data given by the m rankings, the sequential access (SA) and the random access (RA) [25]. While the SA mode accesses the data in a sorted way by proceeding through one of the m rankings sequentially from the top, the RA mode has random access to the rank of a given object w.r.t. a given ranking.

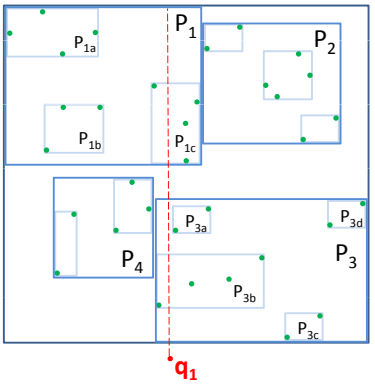
4 Index-Based Subspace Similarity Search – Top-Down

In this section, we propose the *projected R-tree*, a redefinition of the R-tree to answer subspace queries. Let us note, though, that our solution can be integrated into any hierarchical index structure and is not necessarily restricted to R-trees.

The idea of the top-down approach is to apply one index on the full-dimensional data space. The key issue is that for a subspace similarity query, the minimum distance between an index page P and the query object q in subspace S is properly defined because then, we can just use the best-first search algorithm without any changes. The minimum distance between an index page P and the query object q in subspace S can be computed as

$$mindist_S(q,P) = \sqrt[p]{\sum_{i=1}^{D} s_i \cdot \begin{cases} P_i^{\min} - q_i & \text{if} \quad P_i^{\min} > q_i \\ q_i - P_i^{\max} & \text{if} \quad P_i^{\max} < q_i \\ 0 & \text{else} \end{cases}}, \tag{1}$$

where $P_i^{\min}$ and $P_i^{\max}$ are the lower and upper bound of the page P in the ith dimension, respectively. It should again be noted that Eq. 1 is designed for the rectangular page region of R-trees. For the implementation in this paper we used an R*-tree [19] as underlying tree index.



(a) 1D subspace query on 2D space.

Iteration	APL	max1nnDist
0	(0.0, root)	∞
1	(0.0, P_1), (0.0, P_3), (0.7, P_4), (1.6, P_2)	∞
2	(0.0, P_{1c}), (0.0, P_3), (0.7, P_4), (1.6, P_2), (3.0, P_{1b}), (3.2, P_{1a})	∞
3	(0.0, P_3), (0.7, P_4), (1.6, P_2), (3.0, P_{1b}), (3.2, P_{1a})	0.8
4	(0.0, P_{3b}), (0.2, P_{3a}), (0.7, P_4), (1.6, P_2), (3.0, P_{1b}), (3.2, P_{1a})	0.8
5	(0.2, P_{3a}), (0.7, P_4), (1.6, P_2), (3.0, P_{1b}), (3.2, P_{1a})	0.5
6	(0.7, P_4), (1.6, P_2), (3.0, P_{1b}), (3.2, P_{1a})	0.2

(b) Processing of a sample query.

Fig. 1. Subspace query using a projected R-Tree

4.1 Query Processing

When a query q arrives, it is processed in a best-first manner. The algorithm maintains an active page list (APL) which contains pages of the index structure ordered ascending by their *mindist* to the query. Note that since a subspace query is processed, only the dimensions defined by the query are taken into account for the calculation of the *mindist*. The algorithm starts inserting the root of the index structure into the APL. In each iteration, the first page from the APL is processed. If it is a directory node, its children are inserted into the APL. If it is a leaf node, the distance of each point contained in the page to the query is computed. Each point may therefore update the *maxKnnDist*, which is the distance of the kth-nearest point found so far. The process stops if the *mindist* of the first entry of the APL is larger than the *maxKnnDist*. In this case none of the pages in the APL can contain an object being part of the k-nearest neighbors of the query. Figure 1 illustrates an example of a subspace query.

4.2 Discussion

The top-down approach is a relatively straightforward adaptation and can be regarded as complementary to the bottom-up approach discussed in [17]. In contrast to the bottom-up approach using one index per dimension, the top-down approach just needs one index applied to the full-dimensional data space. As a result, the top-down approach does not need to merge the partial results of the rankings performed for each dimension in the corresponding subspace. Relying on the full-dimensional indexing of a data set, the top-down approach can be expected to perform better than the bottom-up approach where the dimensionality of the query subspace is approaching the dimensionality of the data set, if the latter does not disqualify methods based on full-dimensional indexing. On the other hand, as the index used in the top-down approach organizes the data w.r.t. the full-dimensional space, the locality property of a similarity query which

might hold for the full-dimensional space does not necessarily hold for the subspace the query relates on. Generally, the more the dimensionality of the original data space differs from that of the query subspace, the smaller is the expected effectiveness of the index for a given subspace query. In summary, depending on the dimensionality of the query subspace, both indexing approaches qualify for subspace similarity search. While the bottom-up method is more appropriate for lower-dimensional subspace queries, the top-down approach should be used when the dimensionality of the query subspace approaches that of the data space.

5 Evaluation

In this section, we evaluate the proposed methods. In particular, Section 5.1 compares the different algorithms for subspace indexing on real-world data sets, whereas Section 5.2 focuses on the performance of the different heuristics for the bottom-up approach proposed by the authors in [17] on synthetic data sets having different characteristics (cf. Table 1).

Table 1. Data sets

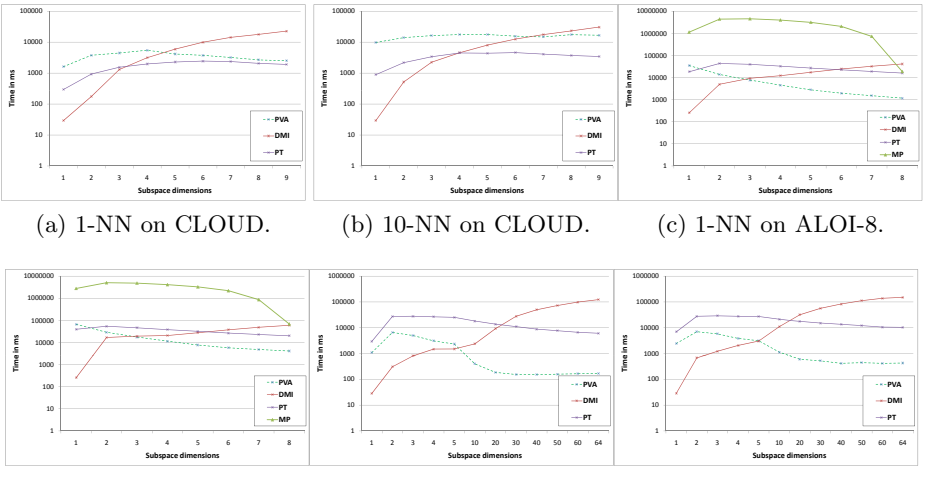
Data set	Type	Size	Dims
CLOUD	meteorological data	$> 1,000,000$	9
ALOI-8/ALOI-64[1]	color histograms	110,250	8/64
UNIFORM	synthetic, uniform	100,000	20
CLUSTERED	synthetic, multivariate Gaussian clusters	100,000	20

5.1 Evaluation of Methods for Subspace Indexing

In this section, we compare the approaches **DMI** (Dimension-Merge-Index [17]), **PT** (Projected R-Tree proposed in Section 4), **PVA** (Partial VA-File [6]) and **MP** (Multi-Pivot-Based algorithm [24]) for subspace indexing. Unless stated otherwise, we compare the different approaches on a data set with $k = 1$ and $k = 10$ with increasing subspace dimension displayed on the x-axis.

In order to compare the different approaches, we performed between $1,000$ and $10,000$ k-NN queries for each data set. For DMI, PT and MP, we measured all page accesses that could not be served by the LRU-cache. PVA does not only perform random page accesses for reading data, but also implements a heuristic that switches to a block read of pages if it turns out that multiple subsequent pages have to be read. Therefore, we measured block read pages and randomly accessed pages of PVA separately. In order to make the results of PVA comparable to the results of the other approaches, we combined the amount of block read pages with the amount of randomly accessed pages and calculated an estimated read time. To achieve this, we assumed a seek time of 8 ms and a transfer rate of 60 MB/s.

[1] Amsterdam Library of Object Images.



(a) 1-NN on CLOUD. (b) 10-NN on CLOUD. (c) 1-NN on ALOI-8.

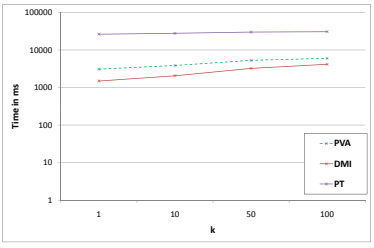
(d) 10-NN on ALOI-8. (e) 1-NN on ALOI-64. (f) 10-NN on ALOI-64.

Fig. 2. Queries with different subspace dimensions. Due to the long runtime and due to the high amount of disc accesses of MP, we only executed tests on ALOI-8 and omitted MP from the remaining experiments.

In Figure 2, we compare the proposed methods on real-world data sets. For CLOUD (cf. Figures 2a and 2b) it can be seen clearly that DMI is superior or equal to the other approaches up to a subspace size of 4 dimensions. For ALOI-8, DMI is better or equal for a subspace size of up to 3 dimensions. In ALOI-64, DMI outperforms PVA and PT up to 4 dimensions until it reaches a break even point with PVA at a subspace size of 5 dimensions. Regarding the dimensionality of the data set and the subspace dimensions where DMI is better or equal to one of the other methods (3 on ALOI-8, 4 on CLOUD (9D) and 5 on ALOI-64), we can state that DMI - such as PVA - performs better on data sets with higher dimensionality, depending on the parameter k (exemplarily shown in Figure 3a for ALOI-64). The obtained results confirm the discussion from Section 4.2. In all tested settings DMI performs best as long as the dimensionality of the subspace query is moderate. When the dimension increases, PT becomes superior to DMI. PVA is a scan-based approach and well suited if a data set is hard to index (e.g. very high-dimensional). CLOUD seems to be well indexable by the R*-Tree, therefore PT performs better than PVA. The ALOI data sets in contrast are rather hard to index (in particular ALOI-64 having a very high dimensionality).

5.2 Evaluation of Heuristics

The proposed heuristics for the bottom-up approach (cf. [17]) address different problems of the data distribution. To accurately show their behavior we tested the heuristics Round-Robin (RR), Global-MinDist (GMD) and MinScore (MS)



(a) k-NN queries on ALOI-64 (4 subspace dimensions, increasing k).

(b) Heuristics on data sets with different characteristics.

Fig. 3

on synthetic data sets with different characteristics. We performed 1,000 10NN queries on a 3D subspace and measured the page accesses needed by each dimension using our bottom-up approach and averaged the outcomes. The results are illustrated in Figure 3b. On UNIFORM and CLUSTERED the more sophisticated heuristics (GMD and MS) are superior to the naïve RR method, since they try to find a better dimension instead of more or less randomly picking one. If the dimensions are scaled randomly, the GMD heuristics favors the dimension with the minimal scale factor. However, this dimension does only increase the minimum distance of all other objects by a small value. Therefore it can stop the filter step very late, which results in many unnecessary page accesses.

6 Conclusions

In this paper, we proposed and studied new, index-based solutions for supporting k-NN queries in arbitrary subspaces of a multi-dimensional feature space. Therefore, we studied two different approaches. One of the main problems we addressed is how to schedule the rankings from the various dimensions in order to get good distance approximations of the objects for an early pruning of candidates. The evaluation shows that our solutions perform superior to the most recent competitors. As future work, we plan to study further heuristics based on our results and to perform a broad evaluation to study the impact of different data characteristics on all existing approaches for subspace similarity search.

Acknowledgements

This research has been supported in part by the THESEUS program. They are funded by the German Federal Ministry of Economics and Technology under the grant number 01MQ07020. The responsibility for this publication is with the authors.

References

1. Beyer, K., Goldstein, J., Ramakrishnan, R., Shaft, U.: When is "nearest neighbor" meaningful? In: Beeri, C., Bruneman, P. (eds.) ICDT 1999. LNCS, vol. 1540, pp. 217–235. Springer, Heidelberg (1998)
2. Bennett, K.P., Fayyad, U., Geiger, D.: Density-based indexing for approximate nearest-neighbor queries. In: Proc. KDD (1999)
3. Hinneburg, A., Aggarwal, C.C., Keim, D.A.: What is the nearest neighbor in high dimensional spaces? In: Proc. VLDB (2000)
4. Aggarwal, C.C., Hinneburg, A., Keim, D.: On the surprising behavior of distance metrics in high dimensional space. In: Van den Bussche, J., Vianu, V. (eds.) ICDT 2001. LNCS, vol. 1973, p. 420. Springer, Heidelberg (2000)
5. Francois, D., Wertz, V., Verleysen, M.: The concentration of fractional distances. IEEE TKDE 19(7), 873–886 (2007)
6. Weber, R., Schek, H.J., Blott, S.: A quantitative analysis and performance study for similarity-search methods in high-dimensional spaces. In: Proc. VLDB (1998)
7. Aggarwal, C.C.: Re-designing distance functions and distance-based applications for high dimensional data. SIGMOD Record 30(1), 13–18 (2001)
8. Katayama, N., Satoh, S.: Distinctiveness-sensitive nearest-neighbor search for efficient similarity retrieval of multimedia information. In: Proc. ICDE (2001)
9. Berchtold, S., Böhm, C., Jagadish, H.V., Kriegel, H.P., Sander, J.: Independent Quantization: An index compression technique for high-dimensional data spaces. In: Berchtold, S., Böhm, C., Jagadish, H.V., Kriegel, H.P., Sander, J. (eds.) Proc. ICDE (2000)
10. Jin, H., Ooi, B.C., Shen, H.T., Yu, C., Zhou, A.Y.: An adaptive and efficient dimensionality reduction algorithm for high-dimensional indexing. In: Proc. ICDE (2003)
11. Dittrich, J., Blunschi, L., Salles, M.A.V.: Dwarfs in the rearview mirror: How big are they really? In: Proc. VLDB (2008)
12. Aggarwal, C.C., Yu, P.S.: On high dimensional indexing of uncertain data. In: Proc. ICDE (2008)
13. Koskela, M., Laaksonen, J., Oja, E.: Use of image subset features in image retrieval with self-organizing maps. In: Enser, P.G.B., Kompatsiaris, Y., O'Connor, N.E., Smeaton, A., Smeulders, A.W.M. (eds.) CIVR 2004. LNCS, vol. 3115, pp. 508–516. Springer, Heidelberg (2004)
14. He, X.: Incremental semi-supervised subspace learning for image retrieval. In: Proc. ACM MM (2005)
15. Kriegel, H.P., Kröger, P., Zimek, A.: Clustering high dimensional data: A survey on subspace clustering, pattern-based clustering, and correlation clustering. ACM TKDD 3(1), 1–58 (2009)
16. Guyon, I., Elisseeff, A.: An introduction to variable and feature selection. J. Mach. Learn. Res. 3, 1157–1182 (2003)
17. Bernecker, T., Emrich, T., Graf, F., Kriegel, H.P., Kröger, P., Renz, M., Schubert, E., Zimek, A.: Subspace similarity search using the ideas of ranking and top-k retrieval. In: Proc. ICDE Workshop DBRank (2010)
18. Guttman, A.: R-Trees: A dynamic index structure for spatial searching. In: Proc. SIGMOD (1984)
19. Beckmann, N., Kriegel, H.P., Schneider, R., Seeger, B.: The R*-Tree: An efficient and robust access method for points and rectangles. In: Proc. SIGMOD, pp. 322–331 (1990)

20. Berchtold, S., Keim, D.A., Kriegel, H.P.: The X-Tree: An index structure for high-dimensional data. Proc. VLDB (1996)
21. Katayama, N., Satoh, S.: The SR-tree: An index structure for high-dimensional nearest neighbor queries. In: Proc. SIGMOD (1997)
22. Samet, H.: Foundations of Multidimensional and Metric Data Structures. Morgan Kaufmann, San Francisco (2006)
23. Kriegel, H.P., Kröger, P., Schubert, M., Zhu, Z.: Efficient query processing in arbitrary subspaces using vector approximations. In: Proc. SSDBM (2006)
24. Lian, X., Chen, L.: Similarity search in arbitrary subspaces under Lp-norm. In: Proc. ICDE (2008)
25. Fagin, R., Lotem, A., Naor, M.: Optimal aggregation algorithms for middleware. JCSS 66(4), 614–656 (2003)