# A Query Answering Greedy Algorithm for Selecting Materialized Views

T.V. Vijay Kumar and Mohammad Haider

School of Computer and Systems Sciences,
Jawaharlal Nehru University,
New Delhi-110067,
India

**Abstract.** Materialized views aim to improve the response time of analytical queries posed on a data warehouse. This entails that they contain information that provides answers to most future queries. The selection of such information from the data warehouse is referred to as view selection. View selection deals with selection of appropriate sets of views to improve the query response time. Several view selection algorithms exist in literature, most of them being greedy based. The greedy algorithm HRUA, which selects top-k views from a multidimensional lattice, is considered the most fundamental greedy based algorithm. It selects views having the highest benefit, computed in terms of size, for materialization. Though the views selected using HRUA are beneficial with respect to size, they may not account for a large number of future queries and may hence become an unnecessary overhead. This problem is addressed by the Query Answering Greedy Algorithm (QAGA) proposed in this paper. QAGA uses both the size of the view, and the frequency of previously posed queries answered by each view, to compute the profits of all views in each iteration. Thereafter it selects, from among them, the most profitable view for materialization. QAGA is able to select views which are beneficial with respect to size and have a greater likelihood of answering future queries. Further, experimental results show that QAGA, as compared to HRUA, is able to select views capable of answering greater number of queries. Though HRUA incurs a lower total cost of evaluating all the views, QAGA has a lower total cost of answering all the queries leading to an improvement in the average query response time. This in turn facilitates decision making.

**Keywords:** Data Warehouse**,** Materialized View Selection, Greedy Algorithm, Query Response Time.

## 1 Introduction

Large amounts of data are continuously being generated by disparate data sources spread across the globe. This data needs to be accessed and exploited by organizations in order to have an edge over their competitors. One way to access this data is by gathering data from disparate data sources, integrating and storing it in a repository and then posing queries against the repository. This approach, referred to as the eager or in-advance approach, is followed in the case of a data warehouse[23], which stores subject-oriented, integrated, time-variant and non-volatile data to support decision making [9]. The queries posed for decision making are usually exploratory and analytical in nature. These queries, being long and complex, consume a lot of time when processed against a large data warehouse. As a result the query response time is high. Although effective solutions for data representation suitable for analytical queries exists, the response time issue still needs to be addressed adequately[16]. To some extent, this problem has been addressed using traditional query optimization[4, 6] and indexing techniques[13]. However, with queries becoming more complex, increasing response times may become unacceptable for decision making[11]. An alternate way to improve the query response time is by using materialized views[14]. Materialized views contain pre-computed and summarized information primarily to improve response time for analytical queries. This necessitates that these materialized views contain relevant and required information for efficient query processing. The selection of such materialized views is referred to as the view selection problem [5, 18] in literature.

View selection is formally defined in [5] as "Given a database schema R, storage space B, and a workload of queries Q, choose a set of views V over R to materialize, whose combined size is at most B". Since the number of possible views is exponential in the number of dimensions, all possible views cannot be materialized as they would violate the space constraint. There is a need to select an optimal subset of views, from among all possible views, which is shown to be NP-Complete[8]. Alternatively, views can be selected empirically or heuristically[17]. In the former[10], the views are selected based on past query patterns and factors like frequency and data volume, whereas in the latter, the views are selected by pruning the search space based on heuristics like greedy[21], evolutionary[24] etc. This paper focuses on the greedy based selection of materialized views, which is discussed next.

Several greedy based algorithms have been proposed in the literature[1, 2, 3, 7, 8, 12, 15, 16, 19, 20, 22], most of which are focused around the algorithm in [8], which hereafter in this paper will be referred to as HRUA. HRUA selects the top-k beneficial views from a multidimensional lattice. It is based on a linear cost model, where cost, defined in terms of the size of the view, is used to compute the benefit of each view. HRUA, in each iteration, computes the benefit of all the non-selected views and selects the one with highest benefit, with respect to size, for materialization. It may be possible that the selected views though beneficial with respect to size, may not be so beneficial with respect to answering large number of future queries, thereby becoming an unnecessary space overhead. One way to address this problem is by considering the frequency of queries answered by each view, along with its size, to compute the benefit of the view. This would lead to selection of views that are not only beneficial with respect to size but can also provide answers to most future queries.

As an example, consider a four dimensional lattice shown in Fig. 1. The size of the view in million (M) rows, and the query frequency of each view, is given alongside the view. Suppose top-4 views are to be selected.

HRUA considers the size of the views in the multidimensional lattice to select top-4 views for materialization. These selections are shown in Fig. 2.

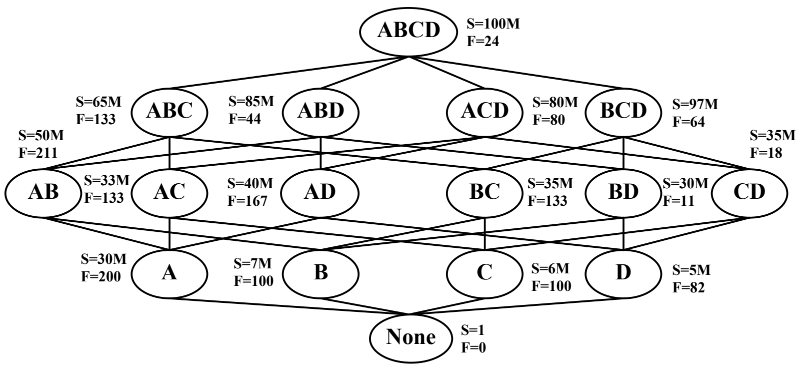

**Fig. 1.** 4-Dimensional Lattice with size and query frequency of each view

| Views | Benefit | | | |
|---|---|---|---|---|
| | 1st Iteration | 2nd Iteration | 3rd Iteration | 4th Iteration |
| ABC | 280 M | | | |
| ABD | 120 M | 60 M | 30 M | 30 M |
| ACD | 160 M | 80 M | 60 M | 60 M |
| BCD | 24 M | 12 M | 06 M | 06 M |
| AB | 200 M | 60 M | 30 M | 15 M |
| AC | 268 M | 128 M | 96 M | |
| AD | 240 M | 170 M | 85 M | 60 M |
| BC | 260 M | 120 M | 60 M | 30 M |
| BD | 280 M | 210 M | | |
| CD | 260 M | 190 M | 95 M | 65 M |
| A | 140 M | 70 M | 35 M | 03 M |
| B | 186 M | 116 M | 46 M | 46 M |
| C | 188 M | 118 M | 83 M | 51 M |
| D | 190 M | 155 M | 50 M | 50 M |

(a)

| Views | QF | QAV | C | TC |
|---|---|---|---|---|
| ABCD | 24 | ABCD | 100M | 2400M |
| ABC | 133 | ABC | 65M | 8645M |
| ABD | 44 | ABCD | 100M | 4400M |
| ACD | 80 | ABCD | 100M | 8000M |
| BCD | 64 | ABCD | 100M | 6400M |
| AB | 211 | ABC | 65M | 13715M |
| AC | 133 | AC | 33M | 4389M |
| AD | 167 | ABCD | 100M | 16700M |
| BC | 133 | ABC | 65M | 8645M |
| BD | 11 | BD | 30M | 330M |
| CD | 18 | CD | 35M | 630M |
| A | 200 | AC | 33M | 6600M |
| B | 100 | BD | 30M | 3000M |
| C | 100 | AC | 33M | 3300M |
| D | 82 | BD | 30M | 2460M |
| **TQ** | 1500 | **TAC** | | 89614M |

(b)

**Fig. 2.** Selection of top-4 views using HRUA

TQ= Total Queries, QF=Query Frequency of view, QAV=Query Answering View, C=Cost of view, TC=Total Cost of query answering by view, TAC=Total Answering Cost of all views.

HRUA selects ABC, BD, AC and CD as the top-4 views. HRUA is able to reduce the total cost of evaluating all these views, referred to as Total View Evaluation Cost (TVEC), from 1600M to 949M, yielding a total benefit of 651M. If we consider the query frequency associated with each view then the number of queries answered, referred to as Total Queries Answered (TQA), by views selected using HRUA is 1121 from among 1500 queries. That is, the remaining 379 queries would need to be answered by the root view ABCD, which is assumed to be materialized as queries on it cannot be answered by any other view in the lattice. Considering the query frequency of each view and the views selected by HRUA, the total cost of answering all the queries, referred to as the Total Answering Cost(TAC), is computed as shown in Fig. 2(b). HRUA requires a TAC of 89614M to process 1500 queries. This value if reduced would result in an improvement in the average query response time. One way to reduce this value is by selecting views that are also capable of answering a greater number of queries so that only few queries would require to be answered by referring to the root view.

Though this may result in an increase in the TVEC value, an acceptable trade-off needs to be achieved between the TVEC and the TQA so that the selected views are not only beneficial with respect to size but are also able to account for large numbers of queries. A Query Answering Greedy Algorithm (QAGA) has been proposed in this paper that attempts to address this problem by selecting top-k views based on both the size of the view and the frequency of previously posed queries answered by the view, referred to as the query frequency of the view. QAGA is focused around HRUA but considers, in each iteration, query frequency and size of each view to compute their profit and then greedily selects the most profitable view for materialization.

The paper is organized as follows: The proposed algorithm QAGA is discussed in section 2 and examples based on it are given in section 3. Section 4 experimentally compares QAGA and HRUA. The conclusion is given in section 5.

## 2 QAGA

View selection aims to select such views that improve the query response time. As discussed above, HRUA considers only the size of the view and may therefore select views that may be unable to provide answers to large numbers of queries resulting in poor query response times. This problem is addressed by the proposed algorithm QAGA, which considers the query frequency of each view, along with its size, to select top-k views for materialization. The algorithm QAGA is given in Fig. 3. QAGA takes the lattice of views, along with the size and query frequency of each view, as input and produces the top-k views as output.

```
Input:   lattice of views L along with size and query frequency of each view
Output: top-k views
Method:
            L: set of views
            RootView: root view in the lattice
            Size(V):  size of view V in the lattice
            QFreq(V): query frequency of view V in the lattice
            NMA(V): nearest materialized ancestor of view V in the lattice.
            Desc(V): set of all descendent views of the view V in the lattice
            MV: set of materialized views.
            For V ∈ L
                        NMA(V) = RootView
            End
            Repeat
              MaxProfit = 0
              For each view V∈ (L – {RootView} ∪ MV)
                ProfitableVew = V
                Profit(V) = 0
                For  each view W ∈ Desc(V) and  (Size(NMA(W)) – Size(V)) > 0
                  Profit(V) = Profit(V) +  QFreq(NMA(W))*Size(NMA(W)) – QFreq(V)*Size(V)
                End
                If  MaxProfit < Profit(V)
                  MaxProfit = Profit(V)
                  ProfitableVew = V
                End
              End
              MV = MV ∪ {ProfitableVew}
              For W ∈ Desc(ProfitableVew)
                If Size(NMA(W)) > Size(ProfitableVew)
                  NMA(W) = ProfitableVew
                End
              End
            Until |MV| < k
            Return MV
```

**Fig. 3.** Algorithm QAGA

QAGA, in each iteration, computes the profit of each view as a product of the number of its descendents and the difference in the product of query frequency and size of its smallest nearest materialized ancestor and the view itself. The profit of a view, i.e. Profit(V), is given as:

$$\text{Profit}(V) = \sum \{ (\text{QFreq}(\text{NMA}(W)) \times \text{Size}(\text{NMA}(W)) - \text{QFreq}(V) \times \text{Size}(V)) \mid V \in A(W) \wedge (\text{SizeNMA}(W) - \text{Size}(V)) > 0 \}$$

where

Size(V)=Size of view V, SizeNMA(V)=Size of Nearest Materialized Ancestor of view V, QFreq(V)=Query frequency of view V, QFreq(NMA(V))=Query frequency of Nearest Materialized Ancestor of view V, A(W)=Ancestor of view W.

The profit takes into consideration the query frequency along with the size of the view, as considering only the size may result in selecting a view that may reduce TVEC without being able to answer an acceptably large number of queries. This may be due the fact that the query answering ability of the view is not considered for computing its benefit. QAGA considers this fact to compute the profit of each view, in each iteration, and then selects from amongst them the most profitable view for materialization. Examples solved using QAGA are illustrated in the next section.

## 3 Examples

Let us consider the selection of top-4 views, using QAGA from the lattice shown in Fig. 1. The greedy selection of top-4 views using QAGA is given in Fig. 4.

| Views | Profit | | | |
|---|---|---|---|---|
| | 1st Iteration | 2nd Iteration | 3rd Iteration | 4th Iteration |
| ABC | 49960 M | | | |
| ABD | 10720 M | 5360 M | 2680 M | 2680 M |
| ACD | 32000 M | 16000 M | 12000 M | 12000 M |
| BCD | 30464 M | 15232 M | 7616 M | 7616 M |
| AB | 32600 M | 7620 M | 3810 M | 1905 M |
| AC | 7956 M | 17024 M | 12768 M | |
| AD | 17120 M | 12490 M | 6245 M | 4280 M |
| BC | 9020 M | 15960 M | 7980 M | 3990 M |
| BD | 8280 M | 20770 M | | |
| CD | 7080 M | 19570 M | 9785 M | 1770 M |
| A | 7200 M | 5290 M | 2645 M | 1611 M |
| B | 3400 M | 15890 M | 740 M | 740 M |
| C | 3600 M | 16090 M | 8315 M | 4059 M |
| D | 3980 M | 10225 M | 160 M | 160 M |

(a)

| Views | QF | QAV | C | TC |
|---|---|---|---|---|
| ABCD | 24 | ABCD | 100M | 2400M |
| ABC | 133 | ABC | 65M | 8645M |
| ABD | 44 | ABCD | 100M | 4400M |
| ACD | 80 | ACD | 80M | 6400M |
| BCD | 64 | ABCD | 100M | 6400M |
| AB | 211 | ABC | 65M | 13715M |
| AC | 133 | AC | 33M | 4389M |
| AD | 167 | ACD | 80M | 13360M |
| BC | 133 | ABC | 65M | 8645M |
| BD | 11 | BD | 30M | 330M |
| CD | 18 | ACD | 80M | 1440M |
| A | 200 | AC | 33M | 6600M |
| B | 100 | BD | 30M | 3000M |
| C | 100 | AC | 33M | 3300M |
| D | 82 | BD | 30M | 2460M |
| **TQ** | 1500 | **TAC** | | 85484M |

(b)

**Fig. 4.** Selection of top-4 views using QAGA

QAGA selects ABC, BD, AC and ACD as the top-4 views. These views are able to reduce the TVEC from 1600M to 954M, which is slightly more than the TVEC of 949M achieved by views selected using HRUA. Moreover, the views selected using QAGA have a TQA of 1368, which is significantly more than the TQA of 1125 for views selected using HRUA. As a result, views selected using QAGA has a TAC for 1500 queries as 85484M, which is less than 89614M due to views selected using HRUA. This would result in an improvement in the average query response time.

Thus, it can be said that QAGA, in comparison to HRUA, is able to select views that account for a comparatively larger number of queries, resulting in improved average query response time, against a slight increase in the total cost of evaluating all the views. It, accordingly achieves an acceptable trade-off between TVEC and TQA.

QAGA need not always select views with higher TVEC values. As an example, consider the four dimensional lattice shown in Fig. 5.

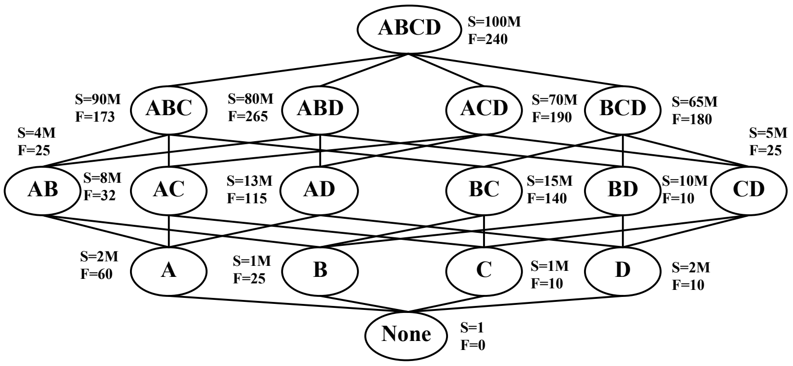

**Fig. 5.** 4-Dimensional Lattice with size and query frequency of each view

Selection of top-4 views using HRUA and QAGA is given in Fig. 6 and Fig. 7 respectively. The views AB, CD, BCD and AC, selected by QAGA, have a lower TVEC value, i.e. 718M, in comparison to 734M for the views BCD, AC, AB and AD selected using HRUA. Further, the TQA value due to QAGA is 632, which is more than the TQA of 517 attained by HRUA. Also, the TAC value for views selected using QAGA to process all the 1500 queries is 112276M, which is less than 120671M of views selected using HRUA. Thus it can be said that QAGA can select views that not only have a better TQA but also a better TVEC when compared with views selected using HRUA. The higher value of TQA due to views selected using QAGA has resulted in a comparatively lower TAC value.

| Views | Benefit | | | |
|---|---|---|---|---|
| | 1st Iteration | 2nd Iteration | 3rd Iteration | 4th Iteration |
| ABC | 80 M | 40 M | 30 M | 20 M |
| ABD | 160 M | 80 M | 60 M | 40 M |
| ACD | 240 M | 180 M | 90 M | 90 M |
| BCD | 280 M | 210 M | 105 M | |
| AB | 384 M | | | |
| AC | 368 M | 184 M | 92 M | 92 M |
| AD | 348 M | 174 M | 87 M | 87 M |
| BC | 340 M | 170 M | 85 M | 50 M |
| BD | 360 M | 180 M | 90 M | 55 M |
| CD | 380 M | 285 M | | |
| A | 196 M | 04 M | 04 M | 04 M |
| B | 198 M | 06 M | 06 M | 06 M |
| C | 198 M | 102 M | 07 M | 07 M |
| D | 196 M | 100 M | 05 M | 05 M |

**(a)**

| Views | QF | QAV | C | TC |
|---|---|---|---|---|
| ABCD | 240 | ABCD | 100M | 24000M |
| ABC | 173 | ABCD | 100M | 17300M |
| ABD | 265 | ABCD | 100M | 26500M |
| ACD | 190 | ABCD | 100M | 19000M |
| BCD | 180 | BCD | 65M | 11700M |
| AB | 25 | AB | 4M | 100M |
| AC | 32 | AC | 8M | 256M |
| AD | 115 | ABCD | 100M | 11500M |
| BC | 140 | BCD | 65M | 9100M |
| BD | 10 | BCD | 65M | 650M |
| CD | 25 | CD | 5M | 125M |
| A | 60 | AB | 4M | 240M |
| B | 25 | AB | 4M | 100M |
| C | 10 | CD | 5M | 50M |
| D | 10 | CD | 5M | 50M |
| **TQ** | 1500 | **TAC** | | 120671M |

**(b)**

**Fig. 6.** Selection of top-4 views using HRUA

| Views | Profit | | | |
|---|---|---|---|---|
| | 1st Iteration | 2nd Iteration | 3rd Iteration | 4th Iteration |
| ABC | 67440 M | 33720 M | 16860 M | 8430 M |
| ABD | 72400 M | 11200 M | 5600 M | 5600 M |
| ACD | 85600 M | 42800 M | 32100 M | 21400 M |
| BCD | 98400 M | | | |
| AB | 95600 M | 71000 M | | |
| AC | 94976 M | 70376 M | 35188 M | |
| AD | 90020 M | 65420 M | 32710 M | 32710 M |
| BC | 87600 M | 38400 M | 19200 M | 9600 M |
| BD | 95600 M | 46400 M | 23200 M | 23200 M |
| CD | 95500 M | 46300 M | 34725 M | 23281 M |
| A | 47760 M | 35460 M | 40 M | 40 M |
| B | 47950 M | 23350 M | 150 M | 150 M |
| C | 47980 M | 23380 M | 11780 M | 336 M |
| D | 47960 M | 23360 M | 11760 M | 11760 M |

**(a)**

| Views | QF | QAV | C | TC |
|---|---|---|---|---|
| ABCD | 240 | ABCD | 100M | 24000M |
| ABC | 173 | ABCD | 100M | 17300M |
| ABD | 265 | ABCD | 100M | 26500M |
| ACD | 190 | ABCD | 100M | 19000M |
| BCD | 180 | BCD | 65M | 11700M |
| AB | 25 | AB | 4M | 100M |
| AC | 32 | AC | 8M | 256M |
| AD | 115 | AD | 13M | 1495M |
| BC | 140 | BCD | 65M | 9100M |
| BD | 10 | BCD | 65M | 650M |
| CD | 25 | BCD | 65M | 1625M |
| A | 60 | AB | 4M | 240M |
| B | 25 | AB | 4M | 100M |
| C | 10 | AC | 8M | 80M |
| D | 10 | AD | 13M | 130M |
| **TQ** | 1500 | **TAC** | | 112276M |

**(b)**

**Fig. 7.** Selection of top-4 views using QAGA

From the above, it can be inferred that the profit computation of a view in QAGA is fairly good as it is able to select views as good as those selected using HRUA.

In order to compare the performance of QAGA with HRUA, both the algorithms were implemented and run on data sets with varying dimensions. The experimental based comparisons of QAGA and HRUA are given next.

## 4 Experimental Results

The algorithm QAGA and HRUA were implemented using JDK 1.6 in a Windows-XP environment. The two algorithms were compared by conducting experiments on an Intel based 2 GHz PC having 1 GB RAM. The comparisons were carried out on parameters like TVEC, TQA and TAC. The tests were conducted by varying the number of dimensions of the data set from 5 to 12. The experiments were performed for selecting top-20 views for materialization.

First, graphs were plotted to compare QAGA and HRUA on TQA against the number of dimensions. The graph is shown in Fig. 8.

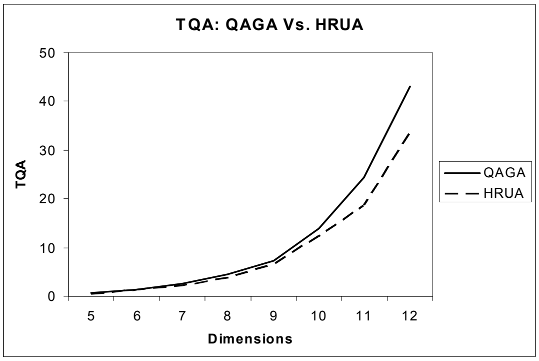

**Fig. 8.** TQA – QAGA Vs. HRUA

It is observed from the above that the increase in TQA value, with respect to number of dimensions, is more for QAGA vis-à-vis HRUA. For dimensions 11 and 12, the TQA value of QAGA is significantly higher than that of HRUA. This shows that the views selected using QAGA perform better than those selected using HRUA in terms of total queries answered by them.

In order to study the impact of better TQA, due to QAGA, on the TVEC, a graph for TVEC (in million rows) versus Dimensions is plotted and is shown in Fig. 9.

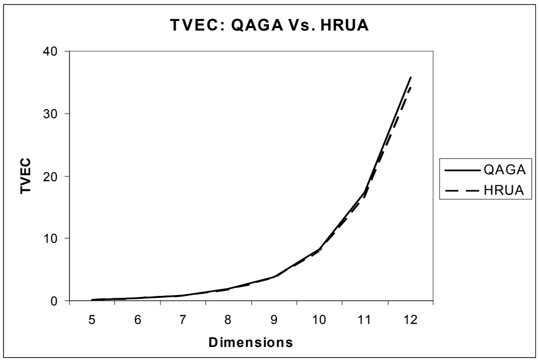

**Fig. 9.** TVEC - QAGA Vs. HRUA

It can be observed from the graph that, with increase in the number of dimensions, the increase in the TVEC value of HRUA is slightly lower than that of QAGA indicating that HRUA performs slightly better than QAGA with respect to total cost of evaluating all the views.

Thus it can be inferred from the above that QAGA, in comparison to HRUA, performs better in TQA with HRUA having a slight edge in TVEC.

Further, in order to compare QAGA and HRUA in terms of the total cost incurred for answering all the queries, a graph for Total Answering Cost (in million rows) Vs. Dimensions is plotted and is shown in Fig. 10.

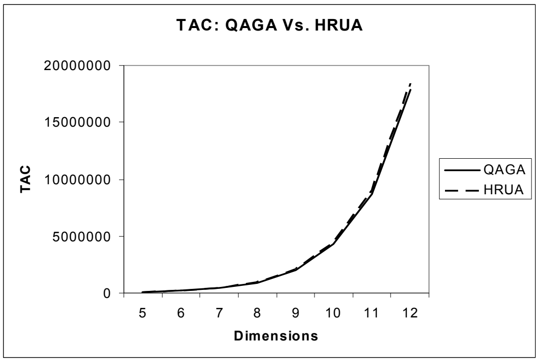

**Fig. 10.** TAC - QAGA Vs. HRUA

The graph shows that with an increase in the number of dimensions, the TAC due to views selected using QAGA is slightly lower than those selected using HRUA. Though HRUA has a slight edge over QAGA in terms of TVEC, the latter incurs less cost in terms of answering all the queries. As a result views selected using QAGA would result in a better average query response time.

It can be reasonably inferred from the above graphs that QAGA trades significant improvement in TQA with a slight drop in TVEC of views selected for materialization. This has lead to lesser TAC for QAGA, which in turn would result in better average query response time.

## 5 Conclusion

In this paper, an algorithm QAGA, that uses both the size and query frequency of a view to select the top-k profitable views from a multidimensional lattice, is proposed. In each iteration, QAGA computes the profit of each individual view, defined in terms of the size and query frequency of the view. This is followed by selecting, from amongst them, the most profitable view for materialization. Unlike most of the greedy based algorithms for view selection, QAGA selects views which are not only profitable with respect to size but are also capable of answering a large number of queries. Also, the views selected using QAGA not only have a higher TQA but also, generally, have a lower TVEC when compared with HRUA. Accordingly, QAGA has an edge over HRUA in the TAC value.

Furthermore, experimental based comparisons show that QAGA performs significantly better than HRUA vis-à-vis TQA value due to the views selected by the two algorithms. Though HRUA has a slight edge over QAGA in terms of the TVEC value, QAGA seems to perform better on the TAC value. This shows that QAGA is not only able to select views which provide answers to a greater number of queries but it is also able to provide answers to all the queries at comparatively lesser cost leading to an improvement in the average query response time. This in turn would facilitate decision making.

## References

1. Agrawal, S., Chaudhuri, S., Narasayya, V.: Automated Selection of Materialized Views and Indexes in SQL Databases. In: Proceedings of VLDB 2000, pp. 496–505. Morgan Kaufmann Publishers, San Francisco (2000)
2. Aouiche, K., Darmont, J.: Data mining-based materialized view and index selection in data warehouse. Journal of Intelligent Information Systems, 65–93 (2009)
3. Baralis, E., Paraboschi, S., Teniente, E.: Materialized View Selection in a Multidimensional Database. In: Proceedings of VLDB 1997, pp. 156–165. Morgan Kaufmann Publishers, San Francisco (1997)
4. Chaudhuri, S., Shim, K.: Including Groupby in Query Optimization. In: Proceedings of the International Conference on Very Large Database Systems (1994)
5. Chirkova, R., Halevy, A., Suciu, D.: A Formal Perspective on the View Selection Problem. The VLDB Journal 11(3), 216–237 (2002)

6. Gupta, A., Harinarayan, V., Quass, D.: Generalized Projections: A Powerful Approach to Aggregation. In: Proceedings of the International Conference of Very Large Database Systems (1995)
7. Gupta, H., Mumick, I.: Selection of Views to Materialize in a Data Warehouse. IEEE Transactions on Knowledge and Data Engineering 17(1), 24–43 (2005)
8. Harinarayan, V., Rajaraman, A., Ullman, J.: Implementing Data Cubes Efficiently. In: Proceedings of SIGMOD 1996, pp. 205–216. ACM Press, New York (1996)
9. Inmon, W.H.: Building the Data Warehouse, 3rd edn. Wiley Dreamtech, Chichester (2003)
10. Lehner, R., Ruf, T., Teschke, M.: Improving Query Response Time in Scientific Databases Using Data Aggregation. In: Proceedings of 7th International Conference and Workshop on Databases and Expert System Applications, pp. 9–13 (September 1996)
11. Mohania, M., Samtani, S., Roddick, J., Kambayashi, Y.: Advances and Research Directions in Data Warehousing Technology. Australian Journal of Information Systems (1998)
12. Nadeau, T.P., Teorey, T.J.: Achieving scalability in OLAP materialized view selection. In: Proceedings of DOLAP 2002, pp. 28–34. ACM Press, New York (2002)
13. O'Neil, P., Graefe, G.: Multi-Table joins through Bitmapped Join Indices. SIGMOD Record 24(3), 8–11 (1995)
14. Roussopoulos, N.: Materialized Views and Data Warehouse. In: 4th Workshop KRDB 1997, Athens, Greece (August 1997)
15. Serna-Encinas, M.T., Hoya-Montano, J.A.: Algorithm for selection of materialized views: based on a costs model. In: Proceeding of Eighth International Conference on Current Trends in Computer Science, pp. 18–24 (2007)
16. Shah, A., Ramachandran, K., Raghavan, V.: A Hybrid Approach for Data Warehouse View Selection. International Journal of Data Warehousing and Mining 2(2), 1–37 (2006)
17. Teschke, M., Ulbrich, A.: Using Materialized Views to Speed Up Data Warehousing, Technical Report, IMMD 6, Universität Erlangen-Nümberg (1997)
18. Theodoratos, D., Bouzeghoub, M.: A general framework for the view selection problem for data warehouse design and evolution. In: Proceedings of DOLAP, pp. 1–8 (2000)
19. Valluri, S.R., Vadapalli, S., Karlapalem, K.: View Relevance Driven Materialized View Selection in Data Warehousing Environment. In: Proceedings of CRPITS 2002, Darlinghurst, Australia, pp. 187–196. Australian Computer Society (2002)
20. Vijay Kumar, T.V., Ghoshal, A.: A Reduced Lattice Greedy Algorithm for Selecting Materialized Views. In: Communications in Computer and Information Science (CCIS), vol. 31, pp. 6–18. Springer, Heidelberg (2009)
21. Vijay Kumar, T.V., Ghoshal, A.: Greedy Selection of Materialized Views. International Journal of Computer and Communication Technology (IJCCT), 47–58 (2009)
22. Vijay Kumar, T.V., Haider, M., Kumar, S.: Proposing Candidate Views for Materialization. In: Communications in Computer and Information Science (CCIS), vol. 54, pp. 89–98. Springer, Heidelberg (2010)
23. Widom, J.: Research Problems in Data Warehousing. In: Proceedings of ICIKM, pp. 25–30 (1995)
24. Zhang, C., Yao, X., Yang, J.: Evolving Materialized views in Data Warehouse. In: Proceedings of the Congress on Evolutionary Computation, vol. 2, pp. 823–829 (1999)