International Journal of Computer and Communication Technology

Volume 1 | Issue 1 Article 7

January 2010

Greedy Selection of Materialized Views

T.V. Vijay Kumar
School of Computer and Systems Sciences Jawaharlal Nehru University New Delhi-110067,
tvvijaykumar@hotmail.com

Aloke Ghoshal
School of Computer and Systems Sciences Jawaharlal Nehru University New Delhi-110067,
alokghoshal@gmail.com

Recommended Citation
Kumar, T.V. Vijay and Ghoshal, Aloke (2010) "Greedy Selection of Materialized Views," *International Journal of Computer and Communication Technology*: Vol. 1 : Iss. 1 , Article 7.
DOI: 10.47893/IJCCT.2010.1006
Available at: https://www.interscience.in/ijcct/vol1/iss1/7

Greedy Selection of Materialized Views

T.V. Vijay Kumar*

School of Computer and Systems Sciences
Jawaharlal Nehru University
New Delhi-110067
Email: tvvijaykumar@hotmail.com
*Corresponding author

Aloke Ghoshal

School of Computer and Systems Sciences
Jawaharlal Nehru University
New Delhi-110067
Email: alokghoshal@gmail.com

Abstract: Greedy based approach for view selection at each step selects a beneficial view that fits within the space available for view materialization. Most of these approaches are focused around the HRU algorithm, which uses a multidimensional lattice framework to determine a good set of views to materialize. The HRU algorithm exhibits high run time complexity as the number of possible views is exponential with respect to the number of dimensions. The PGA algorithm provides a scalable solution to this problem by selecting views for materialization in polynomial time relative to the number of dimensions. This paper compares the HRU and the PGA algorithm. It was experimentally deduced that the PGA algorithm, in comparison with the HRU algorithm, achieves an improved execution time with lowered memory and CPU usages. The HRU algorithm has an edge over the PGA algorithm on the quality of the views selected for materialization.

Keywords: Materialized Views; View Selection; Greedy Algorithms

Reference to this paper should be as follows: Kumar,.T.V., and Ghoshal, A. "Greedy Selection of Materialized Views", *Int. J.CCT*, Vol-1. , No.-1, pp.47-58.

Biographical notes: Dr. T.V. Vijay Kumar is presently an Assistant Professor at the School of Computer and Systems Sciences, Jawaharlal Nehru University, New Delhi. He has done his Ph.D. in the area of Databases from Jawaharlal Nehru University, New Delhi after completing his M.Sc. and M.Phil. in Operational Research, from University of Delhi, Delhi. His research interests are Databases, Data Warehousing, Data Mining and Software Engineering.

Aloke Ghoshal has done his M.Tech in Computer Science and Technology from Jawaharlal Nehru University, New Delhi. He did his BE (Mechanical) from Delhi College of Engineering, Delhi. His areas of interest are Data Intensive Computing, Voice Based Applications, Object Oriented Technologies and Software Frameworks.

1 Introduction

Materialized views can significantly improve the response time for decision support queries [16] [20]. Its potential can only be realized when appropriate set of views is materialized. This issue of selecting the most appropriate set of views to materialize is referred to as view selection [3][14]. The election is defined in [3] 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". The aim of view selection is to select views that would improve query performance of the system [3]. The number of possible views, from which the views are to be selected, is exponential in the number of dimensions [15].

Consequently for higher dimensional data sets, all possible views cannot be stored due to space constraint. Therefore, there is a need to select optimal subset of views with respect to query response time. However, optimal views selection is an NP-Complete problem [6][8]. An alternative way to select views to materialize is by pruning the search space either empirically or heuristically [20]. The empirical selection makes use of the past querying patterns for selecting views to materialize. Heuristically the views are selected based on certain assumptions about parameters like size, cost, benefit, space etc. Several heuristics based approaches are discussed in literature [1][2][5][6][7][8][9][10][11][12][13] [15][17][18], most of which are greedy based. The greedy based approach, at each step, selects a view that has the maximum benefit per unit space and that fits within the available space for materialization. Several greedy based algorithms exist [2, 6, 7, 8, 15, 18], most of which uses multidimensional lattice to select views for materialization. Among these algorithms, the algorithm in [8], which hereafter in this paper is referred to as HRU algorithm, uses the dependencies among views in a multidimensional lattice to select the top-T views for materialization? The HRU algorithm is shown to have a solution lower bound of 63% of the optimal solution [8]. The algorithm in [6] extends the HRU algorithm by presenting greedy based algorithms that consider variable likelihoods of querying views within a space constraint. In [7], polynomial time greedy algorithms for selection of views to materialize using AND graphs and OR graphs, and an exponential time algorithm for AND-OR graphs, is presented. The greedy based algorithm in [18] makes use of the number of dependent relationships, and the frequency of updates to the base relations, to compute the cost of materializing a view. These dependent relationships can utilize the same materialized view, if there has not been any interim update, leading to cost savings. In [2], an adapted greedy algorithm is presented that makes use of a hybrid approach for selecting views for materialization. The hybrid approach selects those views for materialization that offer cost benefit, while the remaining views are virtual and are computed on the fly.

Most of the greedy based algorithms are focused around the HRU algorithm. The HRU algorithm exhibits a high run time complexity [8] primarily because the number of possible views that it needs to evaluate is exponential in the number of dimensions [15]. A scalable solution to this problem was presented as Polynomial Greedy Algorithm (PGA) [15], which selects views for materialization in polynomial time. The PGA algorithm first nominates a set of views and then performs a greedy based selection over these nominated views. It has been given in [15] that PGA is able to select top-T views for materialization in O (K^2d^2) time as against O ($K2^{2d}$) time taken by HRU, for a d-dimensional data set[15]. This indicates that the PGA algorithm is much more scalable than the HRU algorithm for higher dimensional data sets. This paper compares the HRU and the PGA algorithm on parameters like views selected, execution time, benefit value, memory usage and CPU usage. This paper is an extended version of [4].

The paper is organized as follows: multidimensional lattice, used by HRU and PGA algorithm, is discussed in section 2 followed by the algorithms themselves in sections 3 and 4 respectively. Section 5 gives an example-based comparison of the HRU and the PGA algorithm followed by their experiment-based comparison in section 6. Section 7 is the conclusion.

2 Multidimensional Lattice

Multidimensional lattice [8] [14] 18] consists of nodes, depicting the possible views that can be materialized, and edges representing dependencies between these views. The higher-level views represent the base fact table computed from an aggregation of the dependent views at the lower level. Partial ordering among views is used to construct a lattice, where all the views depend, directly or indirectly, on the root node of the lattice.

View X is said to be dependent on view Y, if queries on view X can be answered using view Y. Direct dependencies among views get captured within the lattice by defining an edge between the corresponding nodes, i.e. $X \preceq Y$ (i.e. X is dependent on Y). While indirect dependencies get captured transitively, i.e. if $X \preceq Y$ and $Y \preceq Z$, then it implies that $X \preceq Z$. A node in a lattice is referred to as the

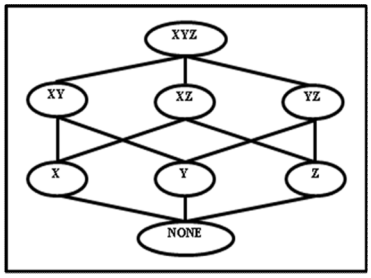
ancestor node of all nodes that appear at a level lower than it in the lattice and are dependent on it (directly or indirectly).

As an example consider a lattice, shown in Figure 1, constructed for a data set comprising three dimensions X, Y and Z. There will thus be a total of 2^3 nodes, i.e. 8 possible nodes (views) in the lattice.

The Lattice in Figure 1 shows that all views directly or indirectly depend on the top view, i.e. XYZ. View NONE, which has no dimension associated with it, has no dependent view. All nodes have edges connecting them to their direct ancestors e.g. view X and Z have edges connected to view XZ.

The HRU and the PGA algorithms, which use the multi-dimensional lattice to select views for materialization, are discussed in the following sections.

Figure 1 Lattice Structure for a 3-Dimensional Data



3 The HRU Algorithm

The HRU algorithm greedily selects the top-T beneficial views from a multidimensional lattice. The algorithm assumes a linear relationship between the cost of answering a user query and the size of the view that can provide an answer to it. This cost, which is the number of rows in the view, is then used to select the most beneficial view for materialization. The method based on HRU algorithm is given in Figure 2.

The HRU algorithm requires size of the views to be known upfront using which it computes the benefit of the view. The benefit of a view is defined as the function of the number of its dependents and its size difference with nearest materialized ancestral view. The root view of the lattice is initially assumed to be materialized and benefits of all other views in the lattice are computed with respect to it. The view having maximum benefit is then selected for materialization. The benefit values of views, which are not yet selected for materialization, may change and are, thus, re-computed with respect to their nearest materialized ancestral view. In this way the algorithm continues to select the most beneficial views for materialization till a predefined number of views are selected.

However, in HRU algorithm, the benefit values of nearly all views of the lattice have to be evaluated. The number of possible views is exponential with respect to the number of dimensions [15]. Accordingly, for a higher dimensional data set, the number of views whose benefit values need to be evaluated will be high. This would result in a phenomenal increase in execution time thereby making the HRU algorithm infeasible for high dimensional data sets [15][19] [20]. A scalable solution to this problem was presented as PGA Algorithm in [15]. The PGA algorithm is discussed next.

Figure 2 Methods based on HRU algorithm

Input: Lattice L corresponding to dependencies among Views V along with their Sizes S
Output: Top-T views
Method:
1. For every view N from the lattice, compute:
   a. $NoDependants_N$:= COUNT (Number of views appearing at a lower level in the lattice and having $View_N$ as an ancestor)
   b. $BenefitValView_N := (Size_{ROOT} - Size_N) * NoDependants_L$
   c. $CommonView_{NM}$: = Identify the direct dependant views common with every other view M, from the same level in the lattice as N
   d. Prepare $OrderedListBeneficialViews_L$ =ORDER_DESC($BenefitValView_N$)
2. Count:=0; MatViewsList :=NULL;
   WHILE (Count < T) DO
   a. Select top view (V_{TOP}) from OrderedListBeneficialViews and Add V_{TOP} to MatViewsList
   b. // Update benefit value and no of dependants
   i. For every view V_K from the same level in the lattice as V_{TOP}, having a common node V_{COMMON} := $CommonView_{KTOP}$
      Adjust $NoDependants_K$:=$NoDependants_K$ - $NoDependants_{COMMON}$
   ii. For every view V_J from a higher level in the lattice than V_{TOP}, s.t. $V_{TOP} \rightarrow V_J$
      Adjust $NoDependants_J$:= $NoDependants_J$ - $NoDependants_{TOP}$
   iii. For every view V_M from a lower level in lattice than V_{TOP}, s.t. $V_M \rightarrow V_{TOP}$
      Adjust, $BenefitValView_M = (Size_{TOP} - Size_M) * NoDependants_M$
   c. Remove V_{TOP} from OrderedListBeneficialViews
   d. Count++
   e. Re-compute OrderedListBeneficialViews values
3. Return MatViewList

4 The PGA Algorithm

The PGA algorithm was proposed in [15] as a solution to the exponential run time complexity of the HRU algorithm. The PGA algorithm works as a two-stage process, nomination followed by selection. A small number of views, termed as promising views, get nominated first. This is followed by greedily selecting a beneficial view from among these nominated views. Since the nomination phase significantly reduces the number of views under consideration, the PGA algorithm is able to perform the selection of views in polynomial time in the number of dimensions [15]. The method based on PGA algorithm is given in Figure 3. In the PGA algorithm, nomination of views starts from the root view of the lattice. Its smallest sized child view, not yet nominated, is nominated first. Next the nomination moves down a level, with the nominated child considered as the parent, and the smallest child of it being nominated as the next nominated view. The nominations continue until the bottom of the lattice is reached. In the selection phase, the most beneficial view, from these nominated views, is selected for materialization. The selected view is then removed from the nominated list. In subsequent iterations, the nomination and selection of beneficial views are carried out similarly until a predefined number of views are selected.

An example-based comparison of HRU and PGA algorithms is given in the next section.

Figure 3 Method based on PGA Algorithm

Input: Lattice L corresponding to dependencies among Views V along with their Sizes S
Output: Top-T views
Method:
Count :=0; MatViewsList:=NULL; W:={} // Set of nominated views;
1. WHILE (Count < T) DO
a. // Nominate views
R:= $Node_{ROOT}$
b. WHILE (More nodes to consider at a level below R)
i. FOR (Each Node from one level below R)
Identify smallest node $r_{SMALLEST}$ that has not been nominated previously
ii. W:= W U { $r_{SMALLEST}$ }
iii. R:= $r_{SMALLEST}$
c. // Select views
For every view N, from set of nominated views W, compute the following using lattice L:
i. $NoDependants_N$:= COUNT (Number of views appearing at a lower level in the lattice and having $View_N$ as an ancestor)
ii. $BenefitValView_N := (Size_{ROOT} - Size_N) * NoDependants_L$
iii. $CommonView_{NM}$: = Identify the direct dependant views common with every other view M,
from the same level in the lattice as N
iv. Prepare $OrderedListBeneficialViews_L$ = ORDER_DESC ($BenefitValView_N$)
d. Select top view (V_{TOP}) from OrderedListBeneficialViews and Add V_{TOP} to MatViewsList
W := W – {V_{TOP}}
e. // Update benefit value and no of dependants of non-nominated views
i. For every view V_K from the same level in the lattice as V_{TOP}, having a common node V_{COMMON} := $CommonView_{KTOP}$
Adjust $NoDependants_K$:=$NoDependants_K$ - $NoDependants_{COMMON}$
ii. For every view V_J from a higher level in the lattice than V_{TOP}, s.t. $V_{TOP} \rightarrow V_J$
Adjust $NoDependants_J$:= $NoDependants_J$ - $NoDependants_{TOP}$
iii. For every view V_M from a lower level in lattice than V_{TOP}, s.t. $V_M \rightarrow V_{TOP}$
Adjust, $BenefitValView_M = (Size_{TOP} - Size_M) * NoDependants_M$
iv. Remove V_{TOP} from OrderedListBeneficialViews
v. Count ++
2. Return MatViewList

5 An Example

Consider the two lattices, shown in Figure 4, for a 3-dimensional data set having a total of 8 nodes or views. The size of each view (in million (M) rows) is given alongside the node in the lattice. Suppose top-3 views are to be selected.

In the lattices of Figure 4, the HRU and PGA algorithm assumes that the root view XYZ is materialized. The total cost of evaluating all the views would be 40 M × 8 = 320 million rows.

The selection of top-3 views, from lattice in Figure 4(a), using HRU algorithm and PGA algorithm is given in Figure 5 and Figure 6 respectively. The HRU and PGA algorithm select same top-3 views XY, YZ and X. The total number of benefit value computations is 18 for HRU algorithm, whereas for PGA algorithm it is 12. This difference would increase as the number of dimensions increases. Thus, for higher dimensional data set, it would be more feasible to select views using PGA algorithm in comparison to HRU algorithm. Further, materializing views XY, YZ and X along with XYZ would decrease the total cost of evaluating all the views from 320 million rows to 196 million rows.

Figure 4 Lattice Structure for a 3-Dimensional Data

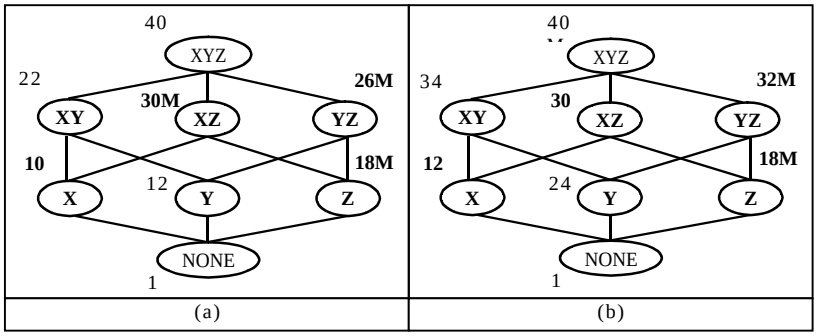


Figure 5 Selection of top-3 views using HRU algorithm

	V	$Size_V$	$SizeNMA_V$	D_V	B_V
1st Iteration	XY	**22M**	40M {XYZ}	4	72 M
	XZ	30M	40M {XYZ}	4	40 M
	YZ	**26M**	40M {XYZ}	4	56 M
	X	10M	40M {XYZ}	2	60 M
	Y	**12M**	40M {XYZ}	2	56 M
	Z	18M	40M {XYZ}	2	44 M
	NONE	1	40M {XYZ}	1	40 M - 1
	V	$Size_V$	$SizeNMA_V$	D_V	B_V
2nd Iteration	XZ	30M	40M {XYZ}	2	20 M
	YZ	**26M**	40M {XYZ}	2	28 M
	X	**10M**	22M {XY}	2	24 M
	Y	**12M**	22M {XY}	**2**	20 M
	Z	18M	40M {XYZ}	2	26 M
	NONE	1	22M {XY}	1	22 M - 1
	V	$Size_V$	$SizeNMA_V$	D_V	B_V
3rd Iteration	XZ	**30M**	40M {XYZ}	1	10 M
	X	**10M**	22M {XY}	**2**	24 M
	Y	12M	22M {XY}	2	20 M
	Z	18M	26M {YZ}	2	12 M
	NONE	1	14M {Z}	1	22 M - 1

V- Views, $Size_V$ – Size of V, $SizeNMA_V$ – Size of the Nearest Materialized Ancestor View of V, D_V – Number of Dependents of V and B_V – Benefit of V

Figure 6 Selection of top-3 views using PGA algorithm

	V	$Size_{NV}$	$SizeNMA_{NV}$	D_{NV}	B_{NV}
1st Iteration	XY	**22M**	40M {XYZ}	4	72 M
	X	10M	40M {XYZ}	2	60 M
	NONE	1	40M {XYZ}	1	40 M - 1
	V	$Size_{NV}$	$SizeNMA_{NV}$	D_{NV}	B_{NV}
2nd Iteration	YZ	**26M**	40M {XYZ}	2	28 M
	X	**10M**	22M {XY}	2	24 M
	Y	**12M**	22M {XY}	**2**	20 M
	NONE	1	22M {X}	1	22 M - 1
	V	$Size_{NV}$	$SizeNMA_{NV}$	D_{NV}	B_{NV}
3rd Iteration	XZ	**30M**	40M {XYZ}	1	10M
	X	10M	22M {XY}	2	24 M
	Y	**12M**	22M {XY}	**2**	20 M
	Z	18M	26M {YZ}	2	16 M
	NONE	1	12M {X}	1	22 M - 1

NV- Nominated View, $Size_{NV}$ – Size of NV, $SizeNMA_{NV}$ – Size of the Nearest Materialized Ancestor View of NV, D_{NV} – Number of Dependents of NV and B_{NV} – Benefit of NV

HRU and PGA need not always select the same views. As an example consider the selection of top-3 views from the lattice in Figure 4(b). These are given in Figure 7 and Figure 8 respectively. The HRU algorithm select X, YZ and Z as top-3 views whereas PGA algorithm select X, YZ and XZ as top-3 views. Though the views selected by the two algorithms are not the same, PGA is able to select two views out of the three views selected by HRU algorithm. Further, the views X, YZ and Z selected by HRU algorithm is able to reduce the total cost of evaluating all the views from 320 million rows to 226 million rows earning a benefit of 94 million rows. On the other hand, the views X, YZ and XZ are able to reduce the total cost of evaluating all the views from 320 million rows to 228 million rows earning a benefit of 92 million rows. The views selected using HRU algorithm is able to achieve slightly more benefit in comparison to those selected using PGA algorithm.

It can be inferred from above that the algorithms show greater convergence, in terms of views selected for materialization, when the beneficial views are concentrated towards the top of the lattice. Further, it can be said that the PGA algorithm requires fewer benefit values computations, in comparison to HRU algorithm, to select views for materialization. However, HRU algorithm is able to select better quality views.

It has been given in [15] that PGA is much more scalable than HRU for higher dimensional data sets. To ascertain this, experiments were carried out to compare HRU and PGA algorithms on performance parameters. The experimental results comparing the HRU and the PGA algorithms are given next.

Figure 7 Selection of top-3 views using HRU algorithm

	V	$Size_V$	$SizeNMA_V$	D_V	B_V
1st Iteration	XY	**34M**	40M {XYZ}	4	24 M
	XZ	30M	40M {XYZ}	4	40 M
	YZ	**32M**	40M {XYZ}	4	32 M
	X	12M	40M {XYZ}	2	56 M
	Y	**24M**	40M {XYZ}	2	32 M
	Z	18M	40M {XYZ}	2	44 M
	NONE	1	40M {XYZ}	1	40 M - 1
	V	$Size_V$	$SizeNMA_V$	D_V	B_V
2nd Iteration	XY	**34M**	40M {XYZ}	2	12 M
	XZ	30M	40M {XYZ}	2	20 M
	YZ	**32M**	40M {XYZ}	3	24 M
	Y	**24M**	40M {XYZ}	**1**	16 M
	Z	18M	40M {XYZ}	1	22 M
	NONE	1	12M {X}	1	12 M - 1
	V	$Size_V$	$SizeNMA_V$	D_V	B_V
3rd Iteration	XY	**34M**	40M {XYZ}	1	6 M
	XZ	30M	40M {XYZ}	2	12 M
	Y	**24M**	32M {YZ}	**1**	8 M
	Z	18M	32M {YZ}	1	14 M
	NONE	1	12M {X}	1	12 M - 1

Figure 8 Selection of top-3 views using PGA algorithm

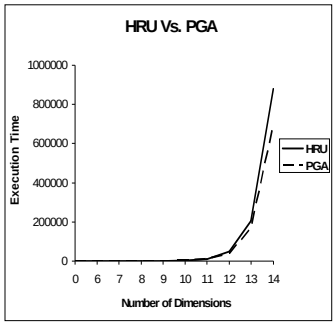
	V	$Size_{NV}$	$SizeNMA_{NV}$	D_{NV}	B_{NV}
1st Iteration	XZ	30M	40M {XYZ}	4	40 M
	X	12M	40M {XYZ}	2	56 M
	NONE	1	40M {XYZ}	1	40 M - 1
	V	$Size_{NV}$	$SizeNMA_{NV}$	D_{NV}	B_{NV}
2nd Iteration	XZ	30M	40M {XYZ}	2	20 M
	YZ	**32M**	40M {XYZ}	3	24 M
	Z	18M	40M {XYZ}	1	22 M
	NONE	1	12M {X}	1	12 M - 1
	V	$Size_{NV}$	$SizeNMA_{NV}$	D_{NV}	B_{NV}
3rd Iteration	XY	**34M**	40M {XYZ}	1	6 M
	XZ	30M	40M {XYZ}	2	20 M
	Y	**24M**	32M {YZ}	**1**	8 M
	Z	18M	32M {YZ}	1	14 M
	NONE	1	12M {X}	1	12 M - 1

6 Experimental Results

The HRU and PGA algorithms were implemented using Java 5 in a Windows-XP environment. The two algorithms were compared by conducting experiments on an Intel based 2 GHz PC having 512 MB RAM. The comparisons were carried out on parameters like execution time, benefit value, memory usage and CPU usage, for selecting top-d views, where d is the number of dimensions. The tests were conducted by varying the number of dimensions of the data set from 6 to 14.

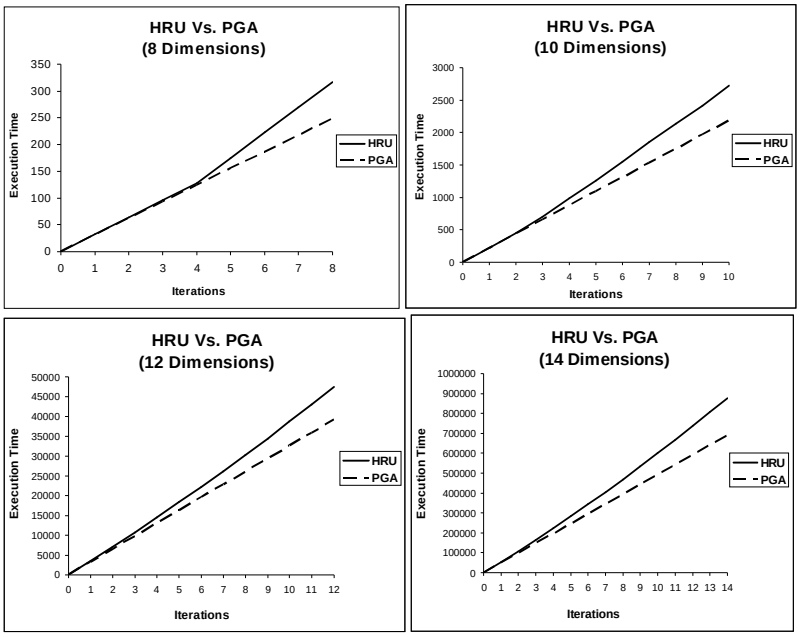
First, a graph was plotted to compare HRU and PGA algorithms on execution time (in milliseconds) as against the number of dimensions. The graph is shown in Figure 9.

Figure 9 **HRU vs. PGA: Execution Time vs. Dimensions**



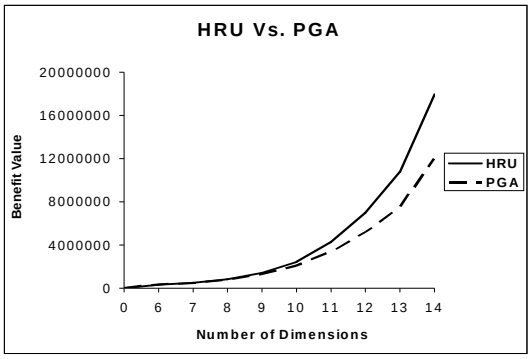
It is observed from the graph that the execution time for PGA algorithm is lower than that for HRU algorithm. As the number of dimensions increases, this difference becomes significant. Further, to better understand this difference, execution time versus iteration graphs were plotted for dimensions 8, 10, 12 and 14. These graphs are shown in Figure 10. In each of these graphs, it is observed that the execution time of PGA algorithm is lower than that of HRU algorithm. These observations establish the claim in [15] that the PGA algorithm has a better execution time than the HRU algorithm.

Figure 10 HRU vs. PGA: Execution Time Vs. Iterations for Dimensions – 8, 10, 12 & 14



Next, a graph was plotted to observe the benefit value (in number of rows) of the views selected versus the number of dimensions. This graph is shown in Figure 11.

Figure 11 HRU vs. PGA: Benefit Value vs. Dimensions



The graph shows that the benefit value of the views selected by the PGA algorithm is lower than that of the views selected by the HRU algorithm. This is because of the different heuristic followed by the PGA algorithm. This difference can also be seen in the graphs for benefit value versus iterations for dimensions 8, 10, 12 and 14, shown in Figure 12.

Next the two algorithms were compared on runtime memory usage (in MB). The corresponding graph in Figure 13 shows that the HRU algorithm has a higher memory requirement than the PGA algorithm. This difference becomes significant for higher dimensions, where large number of views is required to be evaluated.

Figure 12 HRU vs. PGA: Benefit Value vs. Iterations for Dimensions – 8, 10, 12 & 14

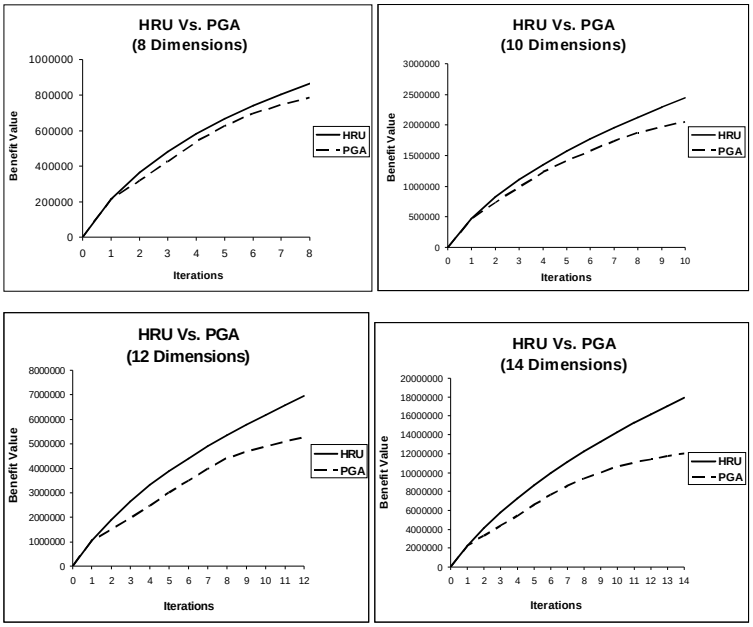
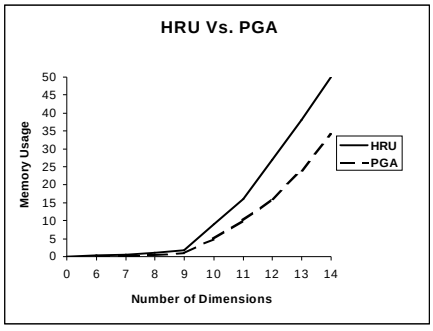
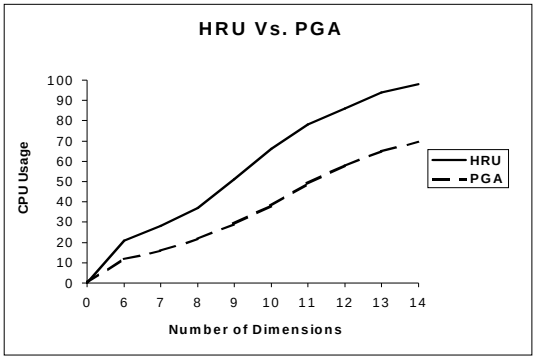


Figure 13 HRU Vs. PGA: Memory Usage Vs. Dimensions



Finally the two algorithms were compared on CPU usage (in percentage). The corresponding graph is shown in Figure 14. This graph shows that HRU algorithm has higher percentage of CPU usage. This may be as a result of higher number of evaluations performed by the HRU algorithm for selecting beneficial views for materialization.

Figure 14 HRU Vs. PGA: CPU Usage Vs. Dimensions



From the above graphs, it can be reasonably inferred that the PGA algorithm, when compared with the HRU algorithm, is found to achieve improved execution times with lower memory and CPU usages. The HRU algorithm is able to select better quality views than PGA algorithm.

7 Conclusion

In this paper, the greedy based selection of materialized views is discussed with emphasis on HRU and PGA algorithms. It is observed that, with increase in the number of dimensions, the number of views requiring evaluation of benefit values increases more significantly in case of HRU algorithm as compared to the PGA algorithm, where top-T views would be selected from a relatively smaller set of nominated views. Further, it is established through experiments that the PGA algorithm, in comparison to the HRU algorithm, performs well on parameters like execution time, memory and CPU usage. The HRU algorithm has an edge over the PGA algorithm on the quality of the views selected for materialization.

References

[1] Agrawal,S., Chaudhuri, S. and Narasayya, V.: Automated Selection of Materialized Views and Indexes for SQL Databases, 26th VLDB Conference, Cairo, Egypt; 2000

[2] Chan, G.K.Y. and Li, Q. and Feng, L.: Optimized Design of Materialized Views in a Real-Life Data Warehousing Environment, International Journal of Information Technology, 7 (1). pp. 30-54; 2001

[3] Chirkova, R., Halevy, A. Y., and Suciu,D.: A formal perspective on the view selection problem, 27th VLDB Conference, Italy; 2001

[4] Ghoshal, A. and Vijay Kumar, T.V.: Greedy Algorithms for Materialized Views Selection, In the proceedings of the International Conference on Data Management (ICDM-2009), Ghaziabad February 10-11, 2009, pp. 101-113

[5] Gupta, H.: Selection of Views to Materialize in a Data Warehouse, ICDT '97, Delphi, Greece, Springer, pp. 98-112; January 1997

[6] Gupta, H., Harinarayan, V., Rajaraman, A., and Ullman, J.D.: Index Selection for OLAP, 13th ICDE Conference, pp. 208-219; April 1997

[7] Gupta, H. and Mumick, I. S.: Selection of Views to Materialize in a Data Warehouse, IEEE; 2005

[8] Harinarayan, V., Rajaraman, A. and Ullman, J.: Implementing Data Cubes Efficiently, In proceedings of ACM SIGMOD International Conference on Management of Data, 1996

[9] Horng, J.T., Chang, Y.J., Liu, B.J. and Kao, C.Y.: Materialized View Selection Using Genetic Algorithms in a Data Warehouse System, IEEE CEC; July 1999

[10] Labrinidis, A. and Roussopoulos, N.: Web View Materialization, ACM SIGMOD Conference, Dallas, Texas, USA; May 2000

[11] Lawrence, M.: Multiobjective Genetic Algorithms for Materialized View Selection in OLAP Data Warehouses, ACM GECCO'06, US; July 2006

[12] Lin, Z., Yang, D., Song, G. and Wang, T.: User-oriented Materialized View Selection, 7th IEEE International Conference on Computer and Information Technology; 2007

[13] Loureiro, J and Belo, O.: An Evolutionary Approach to the Selection and Allocation of Distributed Cubes, IEEE IDEAS'06; 2006

[14] Mohania, M., Samtani, S., Roddick, J, and Kambayshi, Y.: Advances and Research Directions in Data Warehousing Technology, Australian Journal of Information Systems, Vol 7, No 1; 1999

[15] Nadeua, T.P., Teorey, T.J.: Achieving Scalability in OLAP Materialized View Selection, DOLAP '02, pp. 28–34, ACM; 2002
[16] Roussopoulos, N.: Materialized Views and Data Warehouse, 4th Workshop KRDB-97, Athens, Greece; August 1997
[17] Saidi, S., Slimani, Y. and Arour, K.: WebView Selection from User Access Patterns, PIKM'07, Lisboa, Portugal; November 2007
[18] Serna-Encinas, M.T. and 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, pages 18-24, 2007
[19] Shah, B., Ramachandaran, K and Raghavan, V.: A Hybrid Approach for Data Warehouse View Selection, Intlernational Journal of Data Warehousing and Mining; 2006
[20] Teschke, M. and Ulbrich, A.: Using Materialized Views to Speed Up Data Warehousing, Technical Report, IMMD 6, Universität Erlangen-Nümberg, 1997.