Framework for Generalization and Improvement of Relational Data Warehouses

Goran Velinov[†] , Danilo Gligoroski[††] and Margita Kon Popovska[†],

[†]Institute of Informatics, Faculty of Sciences and Mathematics
St. Cyril and Methodius University, Skopje, Macedonia
[††]Centre for Quantifiable Quality of Service in Communication Systems,
Norwegian University of Science and Technology, Trondheim, Norway

Summary
In this paper we present the system of relational data warehouse optimization. The system is based on generalized definition of solution space which includes lot of factors relevant for performance of the relational data warehouses. Novelties are hybrid Greedy - Genetic algorithms applied to generalized solution space and specific techniques for improvement of the efficiency of optimization process. Experimental results shows that our algorithms outperforms traditional Genetic algorithms in the speed of finding optimal (or near optimal) solutions.

Key words:
Relational Data Warehouses; Greedy; Genetic Algorithm.

1. Introduction

This paper represents a collection and extension of three papers that we have recently published [13], [14], [15]. The performance of the system of relational data warehouses depends of several factors and the problem of its optimization is very complex. The main elements of a system for data warehouse optimization are: definition of solution space, objective function and choice of optimization method. The solution space includes factors relevant for data warehouse performance as view and index selection and view fragmentation i.e. partitioning. In some existing approaches the solution space for problem of optimization of data warehouses is simplified to great extent, and the selection of views or indexes is studied without considering other factors, see [1], [2], [6], [11] and [16]. These approaches are important for theoretical research, but are not applicable in practice.

In some papers [9], [13] view selection problem is generalized by including proper set of indexes for each view and selection of views and indexes is done together. If the selection of views and indexes is performed separately and the set of indexes is added to the optimal set of views, then the common set might not have optimal performances. In [3] the problem of optimization of horizontal scheme partitioning was defined. Optimization problem of data warehouses as combination of materialized views, indexes and horizontal data partitioning was introduced in [4] and the approach of vertical fragmentation was introduced in [9]. In [14] a model which includes selections of views and indexes and complete vertical fragmentation was introduced.

The optimization is usually considered with certain constraints which divide all solutions (whole solution space) to two groups of solutions: feasible and infeasible. There are two types of constraints: real system and logical constraints. System constraints are well studied in existing research prototypes [1], [2], [8], [9], [13] and [16]. They can be disk space or maintenance cost constraint. In [1], [2] and [9] the linear cost model was used under disk space constraint and formally system constraints were embedded in the penalty function. In [8], [13] and [16] the objective function involved query processing cost under views maintenance (refreshment) cost constraint. In [16] the constraints were incorporated into the algorithm through a stochastic ranking procedure. In [14] logical constraints ware considered and effects of solutions which violated logical constraints to the optimization process were analyzed.

Some types of evolutionary (genetic) algorithms as optimization method were used in [3], [11], [13] and greedy algorithm with its variants and some heuristic searching techniques were used in [5], [8]. In [13] greedy and genetic algorithms were compared for optimization problems in data warehousing. It was shown experimentally that for large solution space greedy algorithms have poor performances compared to genetic algorithms.

We present two novel hybrid algorithms named GGLA (Greedy-Genetic Linear Algorithm) and GGBA (Greedy-Genetic Binary Algorithm). Both of them are combination of Greedy and Genetic algorithm. The genetic part of the algorithms is based on SRGA (Stochastic Ranking Genetic (evolutionary) Algorithm) introduced in [16]. The algorithms are applied to generalized optimization problem of data warehouse performance, based on our multi-dimensional model that includes complete vertical fragmentation. The model provides definition of all possible aggregate views and their data dependencies. Further, the solution space of the

Manuscript received March 5, 2008
Manuscript revised March 20, 2008

optimization problem includes bitmapped indexes. For comparison, in [16] the SRGA algorithm is successfully applied to solution space that consists of 16 to 256 views, while in our work the algorithms are successfully applied to the solution space (views, fragmented views and indexes) up to 1110 objects.

Furthermore, the system is tested for complex workload i.e. queries with projection, selection, join and grouping operations and with complex selection predicates. Our optimization model is named - SFI since it includes selection of views (S), their vertical fragmentation (F) and their indexing (I). In [14] we have shown benefits of using SFI compared to approaches without fragmentation (SI), without indexes (SF) and without fragmentation and indexes (S). In [15] SRGA is adopted and applied to SFI model. To show the efficiency of our novel algorithms we have compared them with SRGA. We have made many experiments which verified dramatic improvements (up to 280%) of the optimization process. The conclusion is that our optimization algorithms are much more effective and powerful then that developed in [16].

The paper is organized as follows: in Section 2 we describe definition of the solution space, , in Section 3 we give practical example and in Section 4 we define objective function of the optimization system. In Section 5 we present our novel algorithms (GGLA, GGBA) that successfully solve the aforementioned optimization problem, in Section 6 experimental evidence of the efficiency of the algorithms is shown, applied to generalized solution space and finally, in Section 7 we give conclusions and discuss our future work.

2. Definition of Solution Space

In this section we will define generalized solution space of the optimization process. The solution space is based on relational data warehouse scheme which includes definition of dimension relations, all possible aggregate views, their different states of normalization, named variants of views and relational dimensions as well as all possible indexes.

Let us define dimension D as set of relations $R_D = \{R_1,..., R_k\}$, where each dimension relation R_i is characterized by basic set of attributes $BA_i = PA_i \cup DA_i \cup FA_i$, where $PA_i \neq \phi$ is the set of primary key attributes (identifying attributes), $DA_i \neq \phi$ is the set of descriptive attributes and FA_i is the set of foreign key attributes. AA_i is the set of additional attributes of relation R_i, defined as union of basic sets of its dependent relations. $HD = \{RD_1,..., RD_m\}$ is the set of dependencies between dimension relations of dimension D, named dimension hierarchy. Each RD_i is characterized by two dimension relations R_j, R_p, and it is presented by $RD_i : R_j \rightarrow R_p$ i.e. $RD_i(R_j, R_p)$, and also $PA_j \subseteq FA_p$.

A data cube $DC = (DC_D, M)$ is pair of dimension set $DC_D = \{D_1,..., D_n\}$ and measure attributes set M. An implementation scheme SC of data cube DC is defined as pair $SC = (DC, AV)$, where $AV = \{V_1,...,V_p\}$ is set of aggregate views. Each V_l is characterized by set of measure attributes $M_l \subseteq M$ and by basic set of group by attributes BGA_l. Aggregate views can contain different subsets of the set of measure attributes, which provides vertical fragmentation of measure attributes. Between the appropriate dimension relations and the aggregate views there exist 1:M relationships i.e. primary key attributes of the dimension relation at the same time are foreign key attributes of the aggregate views. Group by attributes of any aggregate view consists of primary key attributes of at most one relation of each dimension. An extended set of group by attributes EGA_l of aggregate view V_l is defined as union of basic sets attributes of its dimension relations and additional set of group by attributes AGA_l is defined as $AGA_l = EGA_l \setminus BGA_l$. The set RSC is set of all dimension relations of implementation scheme SC.

The aggregate view V^i derived from aggregate view V by adding (possible empty) subset of union of basic sets attributes of its dimension relations is variant of original view V. Intuitively, the view variants are created by denormalization of original scheme. Our idea is to consider all possible states of normalization of data cube scheme, from fully normalized state in 3rd NF, trough the process of denormalization, to fully denormalized state (whole scheme in one view). The variant V^0 of view V, which is characterized by basic set of group by attributes $BGA^0 = BGA \cup \phi$ is named zero variant of view V.

The important issue is how to select a variant of each dimensional relation, a set of views for materialization (each view presented with its proper variant) and a set of their appropriate indexes in order to minimize the total query processing time of the queries with a certain constraint.

The next step in definition of solution space is determining the indexing strategy. According to the theory of indexing, novel bitmapped indexes are very suitable for processing of data cube based queries. Thus, in this work for aggregate views, one-attribute bitmap indexes are considered. For each attribute of the extended set of group by attributes, one index can be created. Note that the number of elements of the *EGA* and sequence of all indexes are disposed in advance. The set of all possible

indexes in implementation scheme SC is SI . The view length L_{V_l} of aggregate view V_l is defined as $L_{V_l} = 1 + k + m + n$, where k is the number of elements of additional set of group by attributes of the view AGA_l , m is the number of elements of set of measure attributes and n is the number of indexes of the view i.e. number of elements of extended set of group by attributes EGA_l . The parameters L_V are necessary to calculate number of bits (genes) for presentation of the solutions in our algorithms.

A data warehouse responses to large number of aggregate, data cube based ad hock queries (workload) i.e. to dynamic and in principle unpredictable queries. However, lot of them can be determined a priori and can be formalized. To evaluate the real performances of the system we define the workload i.e. the set of predefined generalized project-select-join queries based on aggregation views of the data cube scheme. The query definition is extended by using all possible subsets of grouping and selection attributes and by using complex selection expressions. Data operators i.e. predicates in selection expressions are also important for query processing time i.e. different data operators returns different numbers of tuples. Therefore, in this paper, in our query definition a set of different data operators is included.

Query Q in implementation scheme $SC = (DC, AV)$ is tuple $Q = (M_Q, GA_Q, P_Q, F_Q)$, with $M_Q \subseteq M$ as set of measure attributes, $GA_Q \subseteq EGA_p$ as set of group by attributes, P_Q as selection expression and F_Q as expected frequency of the query. P_Q as selection expression of query Q is in form: $p_1 AND p_2 AND ... AND p_n$, where $p_i : A_i \theta Value_i$, A_i is selection attribute, $\theta \in \{=, <, >, \leq, \geq\}$ is data operator, $Value_i \in Dom\{A_i\}$. The set SA_Q defined as $SA_Q = \{A_1, ..., A_n\}$ is set of selection attributes (where clause attributes) and $SA_Q \subseteq EGA_p$ (EGA_p is extended set of attributes of the primary view). Data operators are divided in two groups: equality and inequality data operators. Thus, inequality operators usually return more then one tuple and after the selection by a certain set of attributes it is reasonable to group the data by the same set of attributes. Consequently, the two sets obtained by grouping and selection attributes are not disjoint. Set of all possible queries in implementation scheme SC is SQ .

Ratio of equality operator ER in selection expression of query Q is defined as k/n , where $n > 0$ is number of data operators i.e. number of elements of the set of selection attributes SA_Q and k is number of equality operators. The parameter ER is necessary to estimate number of tuples returned in each step of query execution i.e. to calculate query execution cost. This is important to develop realistic query evaluator. Query Q is computable from (can be answered by) aggregate view V in implementation scheme $SC = (DC, AV)$ if: 1. $M_Q \subseteq M_V$ and 2. $GA_Q \cup SA_Q \subseteq EGA_V$, where M_V , EGA_V is set of measures and extended set of group by attributes of V , respectively. The previous definition is necessary to determinate the set of views from which each query Q can be answered. Thus, we define ratio of usability UR of aggregate view V as k/n , where $n > 0$ is number of queries i.e. number of elements of the set SQ and k is number of queries which can be answered by V . The UR parameter is necessary to estimate the frequency of usability i.e. "importance" of each view. It will be used in procedure of greedy selection of the views.

3. Practical Example

For better presentation of our formal model a practical example of sale database system is considered. Three dimensional data cube *SALE* with $SALE_D = \{I, S, D\}$ and *M*={S_*quantity*, *S_amount*, *S_price*} is considered. Dimension relations organized in dimension hierarchies are shown in Figure 1.

Simplified scheme of the data cube, with all dimension relations and only with two views (the primary view and one supporting view of the scheme) formally is described by:

Item (I_id, It_id, I_name);
Item_type(It_id, It_name)
Supplier (S_id, C_id, S_name);
City (C_id, Co_id, C_name);
Country (Co_id, Co_name);
Date (D_id, W_id, M_id, D_date);
Week (W_id, D_week);
Month (M_id, Y_id, D_month);
Year (Y_id, D_year);
Sale_ISD (I_id, S_id, D_id, S_quantity, S_amount, S_price);
Sale_ItS (It_id, S_id, S_quantity, S_price);

and graphically is shown in Figure 2.

Set of relations of dimension *I* is *RI*={*Item*,*Item_type*}. Relation Item is characterized by *PA*={*I_id*}, *DA*={*I_name*}, *FA*={*It_id*}, *BA*={*I_id, I_name, It_id*}, *EA*={*I_id, I_name, It_id, It_name*} and *AA*={*It_name*}. Number of variants of Item relations is 2^1=2. Dimension hierarchy of dimension *I* is defined as H_I={It-I}, where *It-I: Item_type* $\rightarrow$ *Item* i.e. *It-I*(*Item_type, Item*). According to the dimension hierarchies, the number of aggregate views of scheme of the data cube *SALE* is 60.

Primary aggregate view is *Sale_ISD* and is characterized by set of measure attributes M_{Sale_ISD}={*S_quantity, S_amount, S_price*} and by BGA_{Sale_ISD}={*I_id, S_id, D_id*} as basic set of group by attributes. Extended set of group by attributes is EGA_{Sale_ISD}={*I_id, I_name, It_id, It_name, S_id, S_name, C_id, C_name, Co_id, Co_name, D_id, D_date, W_id, D_week, M_id, D_month, Y_id, D_year*} i.e. all attributes. Number of variants of aggregate view *Sale_ISD* is 2^{15}=32768. Supporting aggregate view *Sale_ItS* is characterized by set of measure attributes M_{Sale_ItS}={*S_quantity, S_price*} and by BGA_{Sale_ItS}={*It_id, S_id*} as basic set of group by attributes. Extended set of group by attributes is EGA_{Sale_ItS}={*It_id, It_name, S_id, S_name, C_id, C_name, Co_id, Co_name*}. Number of variants of *Sale_ItS* is 2^6=64.

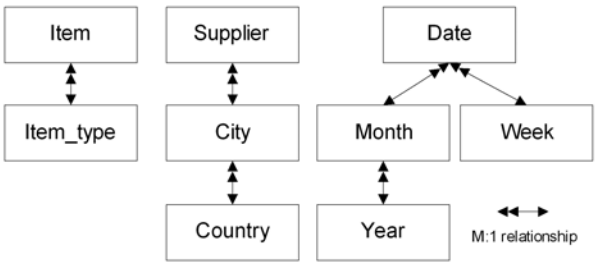

Fig. 1 Dimension hierarchies for *I, S* and *D* dimension

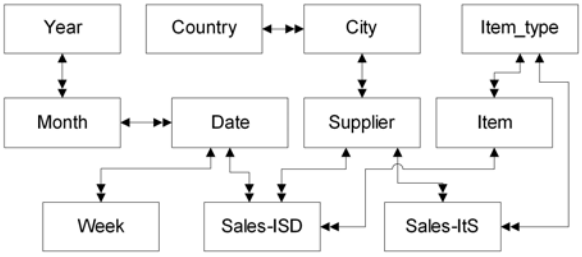

Fig. 2 Simplified scheme of data cube *SALE*

Note that *Sale_ItS* is computable from *Sale_ISD*. Actually, if *Sale_ISD* is given by its zero variant $Sale\_ISD^0$, this means that the view *Sale_ItS* can be created by the following SQL statement:

```
CREATE MATERIALIZED VIEW Sale_ItS AS
SELECT I.It_id, F.S_id,
    SUM(F.S_quantity) S_quantity,
    AVG(F.S_price) S_price
FROM Sales_ISD F, Item I
WHERE I.I_id=F.I_id
GROUP BY I.It_id, F.S_id;
```

To create (compute) *Sale_ItS* it is necessary to join aggregate view *Sale_ISD* and dimension relation *Item*. But if *Sale_ISD* is given by its variant $Sale\_ISD^1$ characterized by $BGA^1_{Sale\_ISD}$ = {*I_id, It_id, It_name, S_id, S_name, D_id*} then the view *Sale_ItS* can be created by the following SQL statement:

```
CREATE MATERIALIZED VIEW Sale_ItS AS
SELECT F.It_id, F.S_id,
    SUM(F.S_quantity) S_quantity,
    AVG(F.S_price) S_price
FROM Sales_ISD F
GROUP BY F.It_id, F.S_id;
```

Note that maintenance cost of *Sale_ISD* is increased by adding the new attributes, but at the same time maintenance cost of *Sale_ItS* is decreased. Also note that view *Sale_ItS* is presented by its zero variant i.e. $BGA_{Sale\_ItS} = BGA^0_{Sale\_ItS} = \{It\_id, S\_id\}$.

If we assume that the view *Sales_ItS* is presented by its zero variant i.e.materialized by above SQL statement, then to process the following query, named *QSale* (sales of type items "Milk Products" by supplier names), it is necessary to join proper dimension relations to the materialized view:

```
SELECT F.It_id, F.S_id, S.S_name,
    SUM(F.S_quantity) S_quantity
FROM Sales_ItS F, Item_type It,
    Supplier S
WHERE It.It_name="Milk Products"
    AND F.It_id=It.It_id
    AND F.S_id=S.S_id
GROUP BY F.It_id, F.S_id, S.S_name;
```

Formally, the query *QSale* can be described by following sets: M_{QSale}={*S_quantity*} as set of measure attributes, GA_{QSale}={*It_id, S_id, S_name*} as set of grouping attributes, P_{QSale}:*It_name*="Milk Products"$ as selection expression and set of selection attributes SA_{QSale}={*It_name*}. Expected query frequency is F_{QSale}=1.

To reduce the processing time required for joining views and dimension relations, frequently accessed attributes can be stored into materialized views. Thus, another possible way to create view *Sales_ItS*, from variant of *Sales_ISD* with $BGA^2_{Sale\_ISD}$ = {*I_id, It_id, It_name, S_id, S_name, D_id*} i.e. another variant of *Sales_ItS* is:

```
CREATE MATERIALIZED VIEW Sale_ItS AS
SELECT F.It_id, F.It_name
    F.S_id, F.S_name,
    SUM(F.S_quantity) S_quantity,
    AVG(F.S_price) S_price
FROM Sales_ISD F
GROUP BY I.It_id, It.It_name,
    F.S_id, S.S_name;
```

The second variant *Sales_ItS*[1] of the view *Sales_ItS* is characterized by the set $BGA^1_{Sale\_ItS} = \{It\_id, It\_name\ S\_id, S\_name\}$ of grouping attributes. It is easy to note that the number of tuples is the same as the first variant of the view. To process previous query now it is not necessary to join dimension relations to fact relation and the SQL statement for the query is:

```
SELECT F.It_id, F.It_name,
    F.S_id, F.S_name, F.S_quantity
FROM Sales_ItS F
WHERE F.I_tname="Milk Products";
```

The advantage of using the second variant is decreasing the processing time of the queries, but the disadvantage is increasing the storage space i.e. maintenance cost. In [5] some frequently accessed dimension keys and attributes are stored in various materialized views. However, a serious problem is to consider set of all possible variants of the data cube views and to find the optimal one with largest benefit of query processing and minimal maintenance cost.

4. Definition of Objective Function

In this section we will propose a suitable evaluation function of the optimization process. Let SC_M is the state of the data cube scheme SC with the set $AV_M \subseteq AV$ of candidate views for materialization where each of them is presented by its variant and the set $SI_M \subseteq SI$ of candidate indexes. Let also all dimensional relations are presented by their appropriate variants. Then maintenance-cost constrained optimization problem is the following one: select a state SC_M of data cube scheme SC that minimizes

$$\tau(SC, SC_M, SQ) = \sum_{Q \in SQ} F_Q * P(Q, SC_M) \ ,$$

under the constraint $U(SC, SC_M) \le S$, where SQ is the set of predefined queries, F_Q is query frequency and $P(Q, SC_M)$ denotes the minimum processing cost of the query Q in the SC_M state of SC.

Let $U(SC, SC_M)$ is total maintenance cost defined as:

$$U(SC, SC_M) = \sum_{R \in R_{SC}} G_R * m(R, SC_M) + \sum_{V \in AV_M} G_V * m(V, SC_M) + \\ + \sum_{I \in I_M} G_I * m(I, SC_M) \ ,$$

where G_R, G_V and G_I is update frequency of relations, views and indexes, respectively. Let $m(R, SC_M)$, $m(V, SC_M)$ and $m(I, SC_M)$ is the minimum cost of maintaining a relations, views and indexes, respectively in presence of state SC_M.

We note that $P(Q, SC_M)$ is objective function of the problem. To calculate values of the functions $P(Q, SC_M)$, $m(R, SC_M)$, $m(V, SC_M)$ and $m(I, SC_M)$ we developed algorithms based on common query execution (processing) theory, presented in [7] and also on some ideas from [1], [10], [11], as well as [12].

5. Novel Greedy - Genetic Algorithms

In this section we will present two novel hybrid algorithms for optimization of relational data warehouses.

First, we will define presentation of the object (dimensional relations, aggregate views and indexes) by an array of bits i.e. by parts which will be concatenated in a chromosome. Dimensional relations are presented (as special type of views for materialization) with their different variants. Number of bits needed to present each dimensional relation with all its variants is $k+1$, where k is number of elements of additional set of attributes. Thus, the first bit is used to present a dimensional relation and other n bits to present additional attributes. As we said, all dimensional relations must be materialized, thus each of them is presented by 1. An attribute of the additional set, if it is added to its dimensional relation is presented by 1 and if it is not added it is presented by 0. Aggregate views are presented in similar way i.e. for each view, one bit is used for its presentation (by 1 if it is selected or by 0 if it is not selected for materialization) and k bits are used to present its additional attributes i.e. their variants. The attributes of the additional set of aggregate views are presented in the same way as attributes of additional sets of dimensional relations. Finally, for each aggregate view the measure attributes are presented by n bits. A measure attribute, if it is added to the appropriate view, is presented by 1 and if it is not added, then it is presented by 0. Note that for each view at least one measure attribute must be added. The presentation of each view is followed by presentation of its possible indexes. Each index is presented exactly by one bit i.e. by 1 if it is selected or by 0 if it is not selected. The number and sequence of all indexes are disposed in advance.

If a view is not materialized all its variant, measure and index bits are irrelevant for evaluation of solution and we named those bits recessive bits. Note that the number of recessive bits is very large in regard of other bits. They are irrelevant for evaluation of solution, but they can be important for optimization process in the next generations.

Our algorithms are hybrid because they are combination of greedy and genetic algorithms. The Algorithm 1 is named ***GGLA - Greedy-Genetic Linear Algorithm***.

Input parameters of the algorithms are: *NC* - Number of Chromosomes (population size), *NG* - Number of Generations, *LC* - Length of Chromosome (number of bits needed to present whole solution space), *NF* - Number of Fragments of the solution space, which is equal to number of steps of greedy procedure, *SQ* - Set of Queries, *AV* - set of Aggregate Views. By *POP(i)* is presented i^{th} generation of the population.

***Algorithm 1**: GGLA(NC, NG, LC, NF, AV, SQ)*

```
Begin
    AV_B:=Ø;
    Choose_views(Round(LC/NF), CL, AV_B,AV_C, AV);
    Extend_chromosome(POP(1), AV_C, NC);
    Evaluate_population(POP(1), SQ);
    Evaluate_population_materialization(POP(1));
    j:=2;
    For i=2 to NG Do
        If Mod(i,Round(NG/NF))=1 Then
            Choose_views(Round(LC*j/NF),CL,AV_B,
            AV_C,AV);
            Extend_chromosome(POP(i), AV_B, NC);
            j:=j+1;
        End If;
        Perform_crossover(POP(i-1), NC);
        Perform_mutation(POP(i), NC);
        Evaluate_population(POP(i), SQ);
        Evaluate_population_materialization(POP(i));
        Perform_selection(POP(i), NC);
    End For;
End GGLA;
```

Parameters of the Subalgorithm 1 are: *NL* - New Length of chromosome (input), *CL* - Current Length of the chromosome (input/output), AV_B - set of chosen views (input/output), AV_C - set of chosen views in current step (output), *AV* - set of Aggregate Views (input).

***Subalgorithm 1**:Choose_views(NL, CL, AV_B, AV_C, AV)*

```
Begin
    AV_C:= Ø ;
    While NL>CL Do
        AV_B := AV_B ∪ V_i , where V_i has maximal
                UR_i in AV \ AV_B;
        AV_C := AV_C ∪ V_i ;
        CL:= CL+LV_i;
    End While;
End Choose_views;
```

The *GGLA* algorithm graphically is shown in Figure 3. Before the optimization process starts, we order all aggregate views by their ratio of usability (*UR*) i.e. their importance. The general idea is in certain steps (*NF*) by greedy procedure to choose the subsets of views with all their bits (Subalgorithm 1 i.e. procedure *Choose_views*) and to add them to already selected ones i.e. to concatenate their randomly generated bits to the already created chromosomes (*Extend_chromosome* procedure). By those procedures we roughly modulate solution space. After each step of greedy procedure, we perform fine optimization by using genetic algorithm (few generations on current solution space). The procedure *Evaluate_population* evaluates the quality of solutions and the procedure *Evaluate_population_materialization* evaluates maintenance solution cost. The procedures *Perform_crossover*, *Perform_mutation* and *Perform_selection* perform genetic algorithm operations - crossover, mutation and selection, respectively.

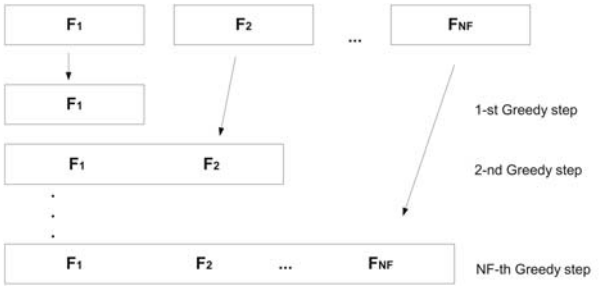

Fig. 3 *GGLA* - Greedy-Genetic Linear Algorithm.

The second algorithm is named ***GGBA - Greedy-Genetic Binary Algorithm***. All input parameters are the same as in Algorithm 1. The *GGBA* algorithm graphically is shown in Figure 4.

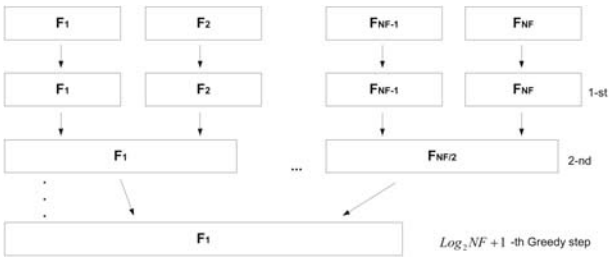

Fig. 4 *GGBA* - Greedy-Genetic Bynary Algorithm.

POP(i,AV_k) presents i^{th} generation of the population and consists of fragment of chromosomes represented by AV_k subset of views. *POP2(i)* presents i^{th} generation of the population and consists of whole chromosomes (created by concatenation of all fragments of *POP* population), represented by set of all aggregate views *AV*. The population *POP2* is necessary to evaluate maintenance solution cost. The variable *NS* is number of steps of the greedy procedure.

***Algorithm 2**: GGBA(NC, NG, LC, NF, AV, SQ)*

```
Begin
    Divide_views(NF, AV);
    For k=1 NF Do
        Create_population(POP(1, AV_k), NC);
        Evaluate_population(POP(1, AV_k), SQ);
        Concat_whole_chr(POP(1, AV_k), POP2(1));
    End For;
    Evaluate_population_materialization(POP2(1));
    NS:=log2NF+1;
    For i=2 to NG Do
        If Mod(i,Round(NG/NS))=1 Then
            For k=1 to NF/2 Do
                Concat_chr(POP(i, AV_k), AV_2*k-1, AV_2*k);
            End For;
            NF:=NF/2;
        End If;
        For k=1 to NF Do
            Perform_crossover(POP(i-1, AV_k), NC);
            Perform_mutation(POP(i, AV_k), NC);
            Evaluate_population((POP(i, AV_k), SQ);
            Concat_whole_chr(POP(i, AV_k), POP2(i));
        End For;
        Evaluate_population_materialization(POP2(i));
        For k=1 to NF Do
            Perform_selection((POP(i, AV_k), NC);
        End For;
    End For;
End GGBA;
```

In procedure *Concat_chr* we define a new set of aggregated views $AV_k := AV_{2*k-1} \cup AV_{2*k}$ and new population $POP(i,AV_k)$ which consists of chromosome fragments created by concatenation of chromosome fragments of $POP(i,AV_{2*k-1})$ and $POP(i,AV_{2*k})$ populations. The chromosome fragments of both populations are ordered from the best to the worst evaluated and concatenated fragments at the same position. We named this concatenation strategy *best-to-best*. In similar way, procedure *Concat_whole_chr* is creates *POP2(i)* as concatenation from populations of all subsets of views $POP(i,AV_k)$. Here we also use best-to-best concatenation strategy. All procedures with the exception of *Evaluate_population_materialization* can be parallelized for different fragments of chromosomes i.e. subsets AV_k of *AV*, which gives the total improvement of the performances of the algorithm.

6. Experimental Results

In this section using our experimental system we compare performances of the three optimization algorithms: *SRGA* - Stochastic Ranking Genetic Algorithm, *GGLA* - Greedy Genetic Linear Algorithm and *GGBA* - Greedy Genetic Binary Algorithm.

For the efficiency of genetic algorithm several input parameters are important. In [13] we have already described some experiments with wide range of their values. In this work we fixed the parameters to their optimal values, obtained in [13]: population size *PS*=20, probability of mutation *PC*=0.5 and *PG*=0.05. Exactly 192 experiments were performed. For all experiments termination condition of optimization process was 64 generations. All algorithms were applied to generalized solution space based of our model which includes view selection with vertical view fragmentation and selection of indexes (SFI). For *GGLA* and *GGBA* we experimented with different values of the parameter *NF* - Number of Fragments. For *GGLA* the number of fragments is equal to the number of steps of the greedy procedure, while for *GGBA* number of steps of the greedy procedure is given by log_2NF+1.

The comparison of the optimization process execution time of all three algorithms, for different number of fragments *NF*, is shown in Figure 5. The optimization process is considered with four different values of *NF* parameter, from 4 to 32. Improvements of *GGLA* and *GGBA* over *SRGA* are evident when we increase the values of the parameter. The value of the parameter is limited by the number of generation of the optimization process (in our case 64). However, usually the number of generations increases by the increasing of the complexity of the problem i.e. its solution space size.

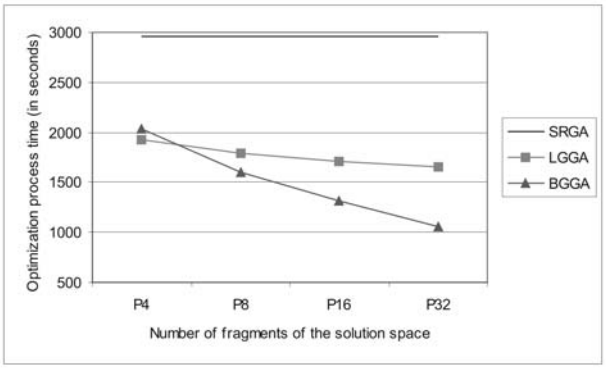

Fig. 5 Optimization process time according number of fragments of the solution space

Next important parameter of optimization process is the size of the solution space i.e. the length of the chromosome for the genetic algorithms. A comparison of different algorithms according to the solution space size is shown in Figure 6. For better presentation of the algorithms performances on the same chart, we scaled values of optimization process execution time. On y-axis the benefit of optimization process execution time

relatively to the worst algorithm is shown. Evidently, in all cases GGBA algorithm has the highest improvement of the optimization time. Note that improvements of both GGLA and GGBA algorithms rise by increasing the solution space size, which is another important feature: the scalability of our algorithms and their appropriateness for practical implementation.

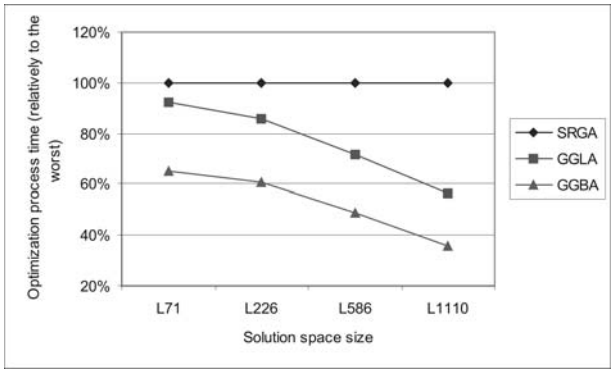

Fig. 6 Optimization process time (relatively to the worst) according solution space size

In proposed algorithms we use a priori knowledge about usability of aggregate views and implement that knowledge as a heuristics in the greedy procedure for the views choice. Using greedy algorithm in the first step enables the genetic material decrease i.e. to decrease the number of calculations in the genetic part of the GGLA algorithm and possibility to parallelize some segments of genetic part in the GGBA algorithm. Finally, we note that values of given solutions by using GGLA and GGBA are in the range of 98%-101% of solutions given by using SRGA.

6. Conclusions and Future Work

The performance of the system of relational data warehouses depends of several factors and its optimization is rather complex problem, thus making a perfect system still a challengeable endeavor. Therefore, in this paper we have focused on the generalizations of the problem and improvement of the efficiency of optimization process.. We fully analyzed the problem by including a lot of factors relevant for the optimization of the system i.e. views selection, vertical view fragmentation and index selection. Then we have designed two novel algorithms *GGLA* and *GGBA* and applied them on the generalized solution space. We have achieved significant performance improvements in the optimization process compared to the stochastic ranking genetic algorithm and we have verified those improvements by performing large set of experiments.

In the future we plan to extend our multidimensional model by including horizontal partitioning, definition of a clustering strategy and to define an optimization process by applying the algorithms on the extended solution space. There is also possibility to parallelize the algorithms on different levels such as: evaluation of the chromosomes within populations, parallelization of the populations or parallel optimization of the different data cubes. In this work static algorithms were considered. Our plan is to develop dynamic algorithms for data warehouse design optimization.

References

[1] K. Aouiche, J. Darmont, O. Boussaid, F. Bentayeb, Automatic Selection of Bitmap Join Indexes in Data Warehouses, *Proc. 7th International Conference on Data Warehousing and Knowledge Discovery, DAWAK'05*, Copenhagen, Denmark, pp.64-73, August 2005.

[2] K. Aouiche, P. Jouve, J. Darmont, Clustering-Based Materialized View Selection in Data Warehouses, *Proc. 10th East-European Conference on Advances in Databases and Information Systems, ADBIS'06*, Thessaloniki, Greece, pp.81-95, September 2006.

[3] L. Bellatreche, K. Boukhalfa, An Evolutionary Approach to Schema Partitioning Selection in a Data Warehouse, *Proc. 7th International Conference on Data Warehousing and Knowledge Discovery, DAWAK'05*, Copenhagen, Denmark, pp.115-125, August 2005.

[4] L. Bellatreche, M. Schneider, H. Lorinquer, M. Mohania, Bringing Together Partitioning, Materialized Views and Indexes to Optimize Performance of Relational Data Warehouses, *Proc. 6th International Conference on Data Warehousing and Knowledge Discovery DAWAK'04*, Zaragoza, Spain, pp.15-25, 2004.

[5] G.K.Y. Chan, Q. Li, L. Feng, Optimized Design of Materialized Views in a Real-Life Data Warehousing Environment, *International Journal of Information Technology*, vol. 7, no. 1, pp.30-54, 2001.

[6] R. Chirkova, Y.A. Halevy, D. Suciu, A formal perspective on the view selection problem, *Proc. 27th International Conference on Very Large Data Bases VLDB'02*, Hong Kong, China, pp.216-237, August 2002.

[7] R. Elmasri, S.B. Navathe, Fundamentals of Database Systems, *Fourth Edition, Addison-Wesley Publishing Company Inc.*, 2003.

[8] H. Gupta, S. Mumich, Selection of Views to Materialize Under a Maintenance Cost Constraint, *Proc. 7th International Conference on Database Theory, ICDT'99*, Jerusalem, Israel, pp.453-470, January, 1999.

[9] M. Golfarelli, V. Maniezzo, S. Rizzi, Materialization of fragmented views in multidimensional databases, *Data & Knowledge Engineering*, Volume 49, Issue 3, pp.325-351, June 2004.
[10] M. Golfarelli, S. Rizzi, E. Saltarelli, Index Selection Techniques In Data Warehouse Systems, *Proc. International Workshop on Design and Management of Data Warehouses DMDW'02*, Toronto, Canada, pp.33-42, 2002.
[11] J. Kratica, I. Ljubic, D. Tosic, A Genetic Algorithm for the Index Selection Problem, *Proc. Applications of Evolutionary Computing: EvoWorkshops 2003*, Essex, UK, pp.280-290, April 2003.
[12] A. Tsois, N. Karayannidis, T. Sellis, D. Theodoratos, Cost-based optimization of aggregation star queries on hierarchically clustered data warehouses, *Proc. International Workshop on Design and Management of Data Warehouses DMDW'02*, Toronto, Canada, pp.62-71, 2002.
[13] G. Velinov, M. Kon-Popovska, Solving View and Index Selection Problem *Using Genetic Algorithm*, *Proc. Second Balkan Conference in Informatics*, *BCI'05*, Ohrid, Macedonia, pp.180-192, November 2005.
[14] G. Velinov, M. Kon-Popovska, D. Gligoroski, Optimization of Relational Data Warehouses, *Proc. 4th European Conference on Intelligent Systems and Technologies*, *ECIT2006*, Iasi, Romania, September 2006.
[15] G. Velinov, D. Gligoroski, M. Kon-Popovska, Hybrid Greedy and Genetic Algorithms for Optimization of Relational Data Warehouses, *Proc. of the 25th IASTED International Multi-Conference: Artificial intelligence and applications*, Innsbruck, Austria, pp.470-475, February 2007.
[16] J.X. Yu, X. Yao, C. Choi, G. Gou, Materialized Views Selection as Constrained Evolutionary Optimization, *IEEE Transactions on Systems, Man and Cybernetics, Part C: Applications and Reviews*, Volume 33, No. 4, pp.458-468, November 2003.

Goran Velinov received the B.Sc. and M.Sc. degrees in Computer Science from Institute of Informatics, St. Cyril and Methodius University, Skopje, Macedonia, where he is currently pursuing the PhD degree. He is a research and teaching assistant at the Institute of Informatics. His recent research focuses on data warehouse, on line analytical processing, index and materialized view selection. He has also interests in optimization problems and genetic algorithms.

Danilo Gligoroski received the PhD degree in Computer Science from Institute of Informatics, Faculty of Natural Sciences and Mathematics, at University of Skopje – Macedonia in 1997. His research interests are Cryptography, Computer Security, Discrete algorithms, Information Theory, Coding and Optimization Algorithms. Currently he is visiting professor at Q2S – Centre for Quantifiable Quality of Service in Communication Systems at Norwegian University of Science and Technology - Trondheim, Norway.

Margita Kon-Popovska received the B.Sc. in Applied Mathematics, M.Sc. in Operations Research and the PhD in Informatics and Management Sciences from the University of Ljubljana, Slovenia. She is professor at the Institute of Informatics at the Ss Cyril and Methodius University in Skopje, Macedonia. Her research interests are information system design and processes, data base design, heuristics optimisation and gridification. She is mentor of several PhD students working in these areas.