G. DIACONESCU
M. MAGDA
C. IONESCU
Department of Economic Cybernetics
Academy of Economic Studies

An Algorithm for Dynamic Determination of the Retrieval Thesaurus in a Statistical Data BASE

An important facility in the MUSCEL software support is presented, based upon the total character inversion of the files, an ordered cycling facility of the repetitive queries. The E.V.E. (Existing Values Exploration) algorithm, that allows this facility, can be also used for rapid sortings, for calculating maxima and minima, and it generally allows the dynamic generation of the retrieval thesaurus from the master file.

This facility is compared to other existing methods; to the end, a detailed description of the algorithm is given, as well as implementation indications.

1. INTRODUCTION

A series of software supports for data bases, especially for information data bases (STAIRS, ADABAS, MISTRAL), uses, as basic technique, the master file transformed into a *file triplet* (A — the master file or object, B — the inverted file and C — the retrieval thesaurus), or in a *file doublet* (master file and inverted file).

Especially in the big retrieval systems (e.g. STAIRS and MISTRAL), the retrieval thesaurus is separately present and it consists in the collection of the distinct A-V pairs from the master file, necessary for retrieval. In these cases, the master file is a little structured, and it generally contains texts wherefrom KEYWORDS are extracted, which make up the retrieval thesaurus.

With files structured by fields, the retrieval thesaurus has not a distinct existence, being dispersed and multiplied within the master file and possibly present in the keys of the inverted records (ADABAS).

With regard to these methods, the inversion by characters (presented for the first time in [1] and used in the MUSCEL software support, for statistic data bases) also introduces a new treatment of the thesaurus. The necessity of being able to treat distinctly the retrieval thesaurus in a structured files system is required by the stricter control of retrieving within an interactive context. With the inversion by characters (SA—V inversion or subatributte inversion), *the distinct A-V pairs collection is not separately stored.* Nevertheless, as will be seen further on, the thesaurus may be easily dynamically determined, during the retrievals, for the current necessities of the exploitation requirements.

In the case of statistical data basis for the teaching staff in the Ministry of Education (MUSCEL system), to which the character inversion is applied, the retrieval thesaurus is particularly rich, containing about 20,000 distinct names, 10,000 localities, 20,000 education institutions, several thousands of teaching disciplines etc.

Nevertheless, due to the method of inversion by characters, the number of inverted records (approx. 5000) is very small, though the file is totally inverted.

2. INTERACTIVE UTILIZATION OF THE RETRIEVAL THESAURUS

The retrieval language associated to the use of inverted files, includes, as an essential element, the query (list, number of records or distribution in the master file) under the form of a logical expression wherein the terms are A-V pairs, and the operators are AND, OR, NO. There often arises the necessity that the demand be resumed under nearly identical form, being looped according to the distinct values of one (some) attribute(s), as in the following example:

NUMBER OF PERSONS WITH SEX-FEMALE AND FUNCTION = A
NUMBER OF PERSONS WITH SEX-FEMALE AND FUNCTION = B
⋮
NUMBER OF PERSONS WITH SEX-FEMALE AND FUNCTION = Z

The reformulation by the user, from the display terminal, of each question, certainly renders clumsy the utilization of such repetitive requests in a conversational context. A more inconvenient circumstance is however the fact that, when formulating the request, the existing values (retrieval thesaurus) of the attribute according to which the loop is done, must be known. In the case of one thesaurus per attribute, having many values, this request is prohibitive, though various solutions may be imagined, e.g. the separate storage of the thesaurus and the displaying on request on the display etc.

But a basic solution is the automatic determination, from the SA—V inverted file (inverted by characters), of all successive values of the attribute and the completion in each step of the query of the kind:

(1) NUMBER OF PERSONS WITH-SEX FEMALE with demand AND (A-V), where A-V generically designate a current pair attribute-value, e.g. PROFESSION = A, PROFESSION = B, etc.

This means essentialy *the looping* of queries according to PROFESSION variable, which can be expressed at the level of the demand language through the following formulation:

(2) NUMBER OF PERSONS WITH SEX-FEMALE, *BY* PROFESSION

The E.V.E. algorithm, which will be further presented, ensures the *quick* generation of the attribute values existing in the file (or of a concatenation of attributes) and allows the implementation of BY statement. The algorithm was defined, for the first time, in [2]. In this paper, the problem of the algorithm implementation in the framework of the MUSCEL software is pointed out.

Besides looping of some queries (ordered looping) the algorithm is useful for carrying out the quick sorting according to a field. The algorithm allows a partial cycling (on a range of values from an attribute) if combined with the SA-V algorithm of retrieving the values according to intervals; this facility is extremely useful, taking into account the existing descriptive codes (see [3]), on the basis of statistical data.

The introduction of the BY statement on MUSCEL language, for retrieving in inverted files by characters must be made comlete with a possibility of statement *in advance* of the possible looping, so as to avoid longcycles in conversational. This is effected through an algorithm which will be dealt with in a future article, and which allows to evaluate quickly the number of cycles (ERN), before starting a statement BY. Algorithm ERN rapidly approximates *the number* of distinct values in an attribute or *a combination* of attributes (example of combination: NUMBER OF PERSONS WITH SEX-FEMALE, BY PROFESSION *DISTRICTS*, and allows the interactive utilization of statement BY with possible *reductions by intervals of the cycling*), depending on a statement previous to retrieval, which determines the number of cycles.

3. COMPARISON OF **BY** STATEMENT WITH OTHER METHODS.

We shall begin with MISTRAL system, where no support is provided for controlling the repetitive retrieval queries neither in ADABAS is that feature provided. As a matter of fact, in SOCRATE there is the possibility to carry out the loop *POUR TOUT attribute, query FIN*, but in this case the retrieval thesaurus of the attribute must be loaded distinctly from the master file, and the relations (ANNEAU, REFERE) must be defined, the relations must be loaded — and besides the cycling in SOCRATE is done according to the loading order, in most of cases irrelevant to the user.

In MUSCEL it is a question of ordered looping (the looping variable being either numerical attribute (field) or an alphanumerical one), this facility being nearer to DO from FORTRAN.

Taking again the analogy with the instructions control in SOCRATE, we may note that in MUSCEL language, the BY statement may be extended so as to control cycles in cycles to *several retrieval demands*, the syntax using BY as a *prefix* of demands.

```
BY      DISTRICTS
        LIST OF ...
        NUMBER OF ...
            BY PROFESSION
                NUMBER OF ...
            END
END
```

The semantics is the same as by the *post-fixed* utilization *of* BY, illustrated in (2), and which consists in the automatic association of the respective. AND. in the cycled retrieval query.

4. ALGORITHM DESCRIPTION

Introducing following notations:

I — index of distinct values in the master file, of an attribute in the increasing (sorted) order of values. For instance, for the NAME attribute, we shall have:

ALBEANU	i = 1
ALBESCU	i = 2
BUCURESCU	i = 3

etc.

J = index of the subattribute position in a value, in the order from L. H. to R. H. For simplifying, we shall consider *numerical* attributes, though the reasonings remain perfectly valid for alphanumerical attributes (fields) in general, taking into account the representation in 2 zero digits of any digit or letter (0, 1, 2, ..., 9, A, B, ..., Z). For the remuneration attribute, for instance, we shall have the thousands position denoted by j = 1 (subattribute 1), the hundreds position by j = 2 (subattribute 2), the tenths position by j = 3, and the units position by j = 4 (subattribute 4).

K = 0, 1, 2, ..., 9,— index of the current subattribute value (possible digits for each position).

Boolean vector. SA-V inverted record with bit pattern corresponding to the associated indexes. Example: R_j^k = inverted record of subattribute j, taking value k. As a matter of fact, this inverted record is a series of bits, 0, 1, 1, 0, 0, 0, ... etc.

R — followed by a value in round brackets is bit pattern of the answer for a value of attribute.

Example: (if it is a matter of remuneration).

R(4956) — is the answer in bit pattern at the query *remuneration = 4956.*

MIN — as lower index indicates the *attribute's values* in increasing order. Example: for the NAME attribute.

MIN_1 is ALBEANU

MIN_2 is ALBESCU

MIN_3 is BUCURESCU

MIN_i

MIN_m

MIN with higher and lower index indicates the subattributes values, within the existing attribute values sorted increasingly according to the following scheme:

$$MIN_1 = \min_1^1 \min_1^2 \text{ ——— } \min_1^n = MAX_m$$

$$\downarrow$$

attribute value 1 subattribute j

$$\downarrow$$

$$MIN_m = \min_m^1 \text{ ——— } \min_m^n = MAX_1$$

Example for NAME attribute MIN_1 = ALBEANU

MIN_2 = ALBESCU

MIN_3 = BUCURESCU

Here, for instance, $\min_1^1$ is A from ALBEANU, in the first position

$\min_1^2$ is L from ALBEANU

$\min_2^3$ is B from ALBESCU

$\min_2^5$ is S from ALBESCU, etc.

hence the lower index indicates which value of attribute, and the higher index indicates the subattribute.

Using these notations, the EVE algorithm is given by the following 5 formulas, which show the determination of the attribute values in increasing order, digit by digit (letter by letter) (generally-subattribute value by subattribute value), from L H to R H.

$$\min_i^j = \text{the smallest K for which } (R_j^k \cdot PRODUCT_i^{j-1} - SUM_{i-1}) \neq 0 \quad (3)$$

where $\varnothing$ is the Boolean vector 0, neutral to addition and subtraction, having all components zero, and the subtraction is logical subtraction (A. AND. NOT. B (A—B)), the multiplication is logical multiplication (*AND* between Boolean vectors), and the addition is logical *OR* (non-exclusive), between the Boolean vectors.

The external cycle is according to i from 1 until reaching the maximal value, and the internal cycle according to j, from 1 up to m, total number of attributes.

It was supposed:

(4) $SUM_0 = (0\,0\,0\,...\,0)$ Boolean vector $\varnothing$, neutral to addition and subtraction, having all components zero
(5) $SUM_i = SUM_{i-1} + R(MIN_i)$
(6) $PRODUCT_i^0 = (1\,1\,1\,1\,...\,1)$ Boolean vector 1, neutral to multiplication, having all components 1.
(7) $Product_i^j = PRODUCT_i^{j-1} \cdot R(min_i^{j-1})$.

We suppose that the value maximum was determined before starting the algorithm. It is obvious that, for determining the maximum, the expression "smallestk" from (3) is replaced by "biggest k".

The proof is based upon:

— the determination of the minimum among several values;

— the successive exclusion of the found out minima (see logical subtraction) and taking again of minimum determining algorithm.

For determining the (first) minimum MIN_1, it is obvious to seek firstly after the minimal L.H. digit (the minimal value of the subattribute having the biggest weight).

Hence the seeking for:
min_1^1 is the smallest k for which $R_1^k \neq \varnothing$

$$k = 0, 1, ..., 9.$$

Further on, the second digit min_1^2 must be looked for, on condition that the first one be the already found one.

Example: $MIN_1 = 235$ we have $min_1^1 = 2$

$MIN_2 = 249$

$MIN_3 = 314$ but 1 from the tens is not good for min_1^2 as the hundreds digit is not 2.

Hence the formula:

$$min_1^2 = \text{the smallest k for which } R_1^k \cdot R(min_1^1) \neq \varnothing$$

$$k = 0, 1, ... \%.$$

Further on, through induction, at index j there appears the recurrent *product*. The passing to the second value MIN_2 requires to take again the algorithm, but with the exclusion of the already found out value MIN_1. Hence the logical subtraction. Going on, to index *i* appears the recurrent *sum*, through induction.

REFERENCES

1. DIACONESCU G.: *SA—V Algorithm for Retrieval in a Statistical Data Base.* Economic Computation and Economic Cybernetics Studies and Research, no. 1, 1975.
2. DIACONESCU G.: *The Design of a Statistical Data Base.* Thesis for Doctor's Degree.
3. DIACONESCU G.: *The Retrieval of Information into a SA—V Organized Data Base.* The Third World Congress of Cybernetics and Systems, Editura Tehnică and Springer-Verlag, 1977.

UN ALGORITHME POUR LA DETERMINATION DYNAMIQUE DU STOCK DE MOTS-CLES DANS UNE BASE DE DONNES STATISTIQUES

On présente une facilité importante dans le support software MUSCEL, basé sur l'inversion totale des fichiers, par caractères, facilité de cyclage ordonné des demandes à répétition L'algorithme E.V.E. (Exploration des Valeurs Existantes) qui permet cette facilité, peut être utilisé au sortage rapide, au calcul des maximums et des minimums et en général permet l'engendrement dynamique du stock de mots-clés dans le fichier de base.

La facilité est comparée à d'autres méthodes et dans la partie finale de l'article on fait la description en détail de l'algorithme et on donne des indications pour la mise en pratique.

АЛГОРИТМ ДЛЯ ДИНАМИЧЕСКОГО ОПРЕДЕЛЕНИЯ СОКРОВИЩНИЦЫ ПЕРЕНАХОЖДЕНИЯ В БАЗЕ СТАТИСТИЧЕСКИХ ДАННЫХ

В работе представлен метод упорядоченного циклирования повторных заявок, основанный на полном расположений в обратном порядке файлов по признакам. Предложенный алгоритм может быть использован и для быстрых сортировок, и для определения максимума и минимума и, вообще, создаёт возможность динамической генерации сокровишницы перенахождения из основного файла.

Представленный метод сравнивается со существующими методами и, в конце, даётся подробное описание алгоритма с указаниями по его применению.

7.VI.1978

ERATUM

Table 2, page 44 and table 3, page 46 in no. 2 / 1978 will be filled in as follows :

Results obtained by the SUMT method with the DFP algorithm with Kaplan penalty

Table 2

r_k	MODEL	M_1 94 constraints, 30 variables		M_2 64 constraints, 30 variables	
$r_1 = 1$	IT	= 10		IT	= 8
	NF	= 51		NF	= 34
	$F(x_k^*, r_k)$	= 36.02		$F(x_k^*, r_k)$	= 22.50
	$\|\|\nabla F(x_k^*, r_k)\|\|$	= 1.831		$\|\|\nabla F(x_k^*, r_k)\|\|$	= 0.48×10^{-6}
	$f(x_k^*)$	= 0.178		$f(x_k^*)$	= 0.0254
$r_2 = 4$	IT	= 10		IT	= 50
	NF	= 64		NF	= 310
	$F(x_k^*, r_k)$	= 2.178		$F(x_k^*, r_k)$	= 2.339
	$\|\|\nabla F(x_k^*, r_k)\|\|$	= 11.68		$\|\|\nabla F(x_k^*, r_k)\|\|$	= 1.325
	$f(x_k^*)$	= 1.065		$f(x_k^*)$	= 0.995
$r_3 = 16$	IT	= 10		IT	= 50
	NF	= 49		NF	= 292
	$F(x_k^*, r_k)$	= 29.09		$F(x_k^*, r_k)$	= 1.148
	$\|\|\nabla F(x_k^*, r_k)\|\|$	= 60.53		$\|\|\nabla F(x_k^*, r_k)\|\|$	= 1.516
	$f(x_k^*)$	= 0.959		$f(x_k^*)$	= 1.147
$r_4 = 24$	IT	= 10		IT	= 9
	NF	= 61		NF	= 56
	$F(x_k^*, r_k)$	= 15.16		$F(x_k^*, r_k)$	= 1.145
	$\|\|\nabla F(x_k^*, r_k)\|\|$	= 17.18		$\|\|\nabla F(x_k^*, r_k)\|\|$	= 30.36
	$f(x_k^*)$	= 8.940		$f(x_k^*)$	= 1.145

Table 3

Results obtained by the SUMT method with the DFP algorithm with Nguyen – Strodiot exponential penalty

r_k	MODEL	M_1 94 constraints 30 variables	M_{12} 94 constraints 31 variables	M_2 64 constraints 30 variables	M_{22} 64 constraints 31 variables
$r_1 = 1$	IT	= 10	= 9	= 7	= 6
	NF	= 52	= 41	= 29	= 39
	$F(x_k^*, r_k)$	= -6.329	= -61.96	= -44.65	= -41.61
	$\|\|\nabla F(x_k^*, r_k)\|\|$	= 1.032	= 2.25	= 0.16×10^{-6}	= 4.163
	$f(x_k^*)$	= 0.034	= 0.242	= 0.006	= 0.557
$r_2 = 4$	IT	= 10	= 5	= 9	= 9
	NF	= 56	= 27	= 49	= 47
	$F(x_k^*, r_k)$	= -20.64	= -3.13	= -2.049	= -2.95
	$\|\|\nabla F(x_k^*, r_k)\|\|$	= 0.196	= 5.98	= 0.17×10^{-2}	= 0.27
	$f(x_k^*)$	= 0.254	= 0.65	= 0.548	= 0.646
$r_3 = 16$	IT	= 10	= 6	= 19	= 5
	NF	= 54	= 44	= 91	= 41
	$F(x_k^*, r_k)$	= -34.75	= 2.935	= 1.058	= 2.648
	$\|\|\nabla F(x_k^*, r_k)\|\|$	= 13.512	= 11.77	= 0.17×10^{-2}	= 4.456
	$f(x_k^*)$	= 0.899	= 2.76	= 1.059	= 2.78
$r_4 = 24$	IT	= 10	= 13	= 43	
	NF	= 46	= 59	= 230	
	$F(x_k^*, r_k)$	= 7.40	= 4.521	= 1.113	
	$\|\|\nabla F(x_k^*, r_k)\|\|$	= 43.20	= 4.70	= 0.28×10^{-2}	
	$f(x_k^*)$	= 0.990	= 4.68	= 1.110	