Közlemények 30/1984 37-51

SOME REMARKS ON STATISTICAL DATA PROCESSING

J. DEMETROVICS, P. KERÉKFY, A. KRÁMLI, M. RUDA

Computer and Automation Institute
Hungarian Academy of Sciences

1. INTRODUCTION

The so-called "software problem" or "software crisis" is the most important matter at issue in computer science. Several papers are devoted to discuss different aspects of the crisis (see e.g. [9]). There is a lot of contributions both in the theory and the practice that aims to resolve parts of the problems. These efforts can be classified into three major categories: very high level languages (VHLL's), logic-based and knowledge-based systems. The VHLL's continue the historical evolutionary trend of software development by developing new programming languages in which the program can be described at a higher level of abstraction. Statistical data processing system requires numerous new concepts and methods [2]. These tasks give rise to difficulties mainly in large and complicated statistical investigations. With our work started some years ago we wished to obtain results just in this field.

Our first attempts in this field were concluded from the Hungarian Hospital Morbidity Study [3] producing a simple statistical information system. We intended to answer very quickly questions about a large mass of data. One of our tools was collecting a large variety of statistical data in advance (producing the so-called "table files"). A very fast information retrieval can be realized using the collected data, its speed does not depend on the size of the sample.

Another basic technical tool was program generating [19,21]. In this way we built up system SIS77 (Statistical Information System 1977) for the Hungarian Hospital Morbidity Study [3] that acted very well in COMECON cooperation [15].

System GENERA [12] was developed for the extension and wide-ranging applications of the method used in SIS77. It gives

assistance to a program generator technique that makes the programming and usage of optimal-performance procedures possible.

GENERA processes directives imbedded in a host language, so it is quite flexible and can easily be extended or modified. Any supplementary program can be written in the host language. The generator procedures are independent modules written in high-level languages so they are easy to survey and correct. Error detection is supported by the simple and standardized structures of generated program fragments. On the contrary, traditional advanced programming tools (high level languages, program packages) are usually closed, the user cannot modify or supplement them. Some additions are allowed (as e.g. in BMDP) but they does not touch the inner structure of the system. It must be mentioned as an advantage of these advanced software tools that the user need not have much knowledge on the computer background. But it can make the usage mysterious. The user becomes "alienated" from the system, from the compute science embodied in it and from the applied program and the results. Consequently the user may be unable to give preliminary estimations on the computer resources needed, it provides means for maneuvers that cannot be taken in or controlled, and the user may not be able to interpret the results correctly. On the other hand, the user unfamiliar with the system can not necessarily use it properly even if the rules are simple when he does not know the mathematical, software and organizational background. Observing an error in the system, it cannot be located and corrected by simple means.

Utilizing GENERA the user organizes his work in the host language (a high-level language) and it can be expected that he has a good insight into the program. Besides, the directives of a system managed by GENERA (such as SIS79) release the user from the most cumbersome work in programming, moreover, its macro processor makes the production of parameter--controlled programs possible. Compared to the traditional methods, the generator procedure rather than the running

program receives the parameters. This is important to efficiency. However it should be noted that GENERA is not a macro generator in the traditional meaning of the word.

Coming back to the matter of statistical data processing, we shall touch upon some other problems.

While processing large and complicated data sets, beyond the problem of selecting proper software tools interesting mathematical (optimization) problems arise. They can emerge while designing the codes used, sampling, designing the processes and data storage.

Data checking, transformation and, in general, analysis of functions providing control, transformation or selection of the sample form another group of important questions. These tasks require (in the case of a large and complicated system) modelling of strange functions and convenient description of large code tables.

FORTRAN is used very often to write programs for statistical data processing, some well-known systems (such as BMDP or SPSS) utilize it. Present implementations of GENERA have FORTRAN as an optional host language (beside PL/1). Data description and I/O procedures of FORTRAN are sometimes inconvenient and slow. This fact inspired us to work out some procedures for input--output, data description and storage in systems SIS77 and SIS79.

2. HUNGARIAN HOSPITAL MODBIDITY STUDY AND SYSTEM SIS79/GENERA

In Hungary, representative hospital morbidity studies have been in progress (including each hospital and department) since 1972. Data of the inpatients are collected yearly with a sampling rate from 10 to 50 percent. It amounts to information on 200 to 600 thousand patients per year.

Processing of a rather large sample (600 thousand records, 60 to 80 million bytes) was to be accomplished on a comparatively small machine. The requirements were rather complicated

and subject to modifications from time to time. At first, the machine used was CDC-3300 having 64 K words of memory with two 8 Mbyte disc units and two or three tape units available. The machine was overloaded so we could run small jobs (some minutes of CPU time) only. Consequently, jobs utilizing the total sample were to be run rarely. It became necessary to examine questions concerning the strategy of data processing. On the other hand, the data set was to be divided and compressed. The running time of the job (as well as other resources: memory, disc, tape) was to be minimized.

Later we got access to higher-capacity machines [15] (a HwB 66/60 or two HwB 66/20 with 100 Mbyte discs and 256 K words of memory). The problem of capacity became less important. But taking into account the requirements of conversational processing and the aspiration to faster turn-around in bacth processing (and the expenses as well) optimization of storage and time were expedient.

Let us outline the basic methods utilized to achieve shorter run-time in statistical investigations. First, we created statistical tables from frequencies and cumulated values instead of the original data. The tables were obtained in some seconds, practically independently of the sample size [15]. Let us note here that the hospital morbidity study required descriptive statistics mainly: tables contained frequencies, cumulated values and some simple rates (e.g. morbidity rate, etc.) and basic characteristics of the distribution (such as mean, standard deviation, range). More complicated statistical analyses do not make rise to new situations concerning fast processing of a large amount of data. Usually, mathematical statistics need frequencies and cumulated values (sums, quadratic sums, sums of products) (see statistical literature [11]). Then these values can be processed e.g. by SPSS programs.

Another method applied was a general technique in programming. The programs of the system are generated in each case depending on the parameters of the task. This technique (based on earlier experiences) was consistently applied in system SIS77 developed on HwB 66's [21]. In this improved version

(in SIS79/GENERA) this technique was developed further [13]. The task of generating was placed under control of a general purpose system (GENERA) improving integrity and efficiency of the system. System GENERA and the possibilities provided by statistical system SIS79 will be dealt with in later sections.

In the Hungarian Hospital Morbidity Study, statistical systems SIS77 and SIS79 provide quick access to data and detailed analysis even for individual researchers. Even in the case of large mass of data and complicated conditions the system needs modest resources only. A COMECON-project on juvenile hypertension coordinated by the National Cardiology Institute that was successfully accomplished by systems SIS77 and SIS79.

In statistical tasks (especially in large and complicated systems) the method of sequential processing is suitable. Sequential processing is similar to sequential sampling [23] known from mathematical statistics. It produces a more and more widening sphere of information depending on the information obtained before. But compared to sequential sampling it does not mean an increasing amount of information of the same kind; in this case the kind of information is subject to change as well. Users (doctors, economists, etc.) first receive simple, easy-to-survey data (tables, graphs, descriptive statistics). The more and more detailed questions are based on the information obtained earlier and can optionally be answered on the base of a widening population. (That is a wider subset of the data set.) In this way needless information is not to be gathered, simple relations are enlightened immediately and the user gets an overall picture of the sample investigated. This method provides means for obtaining more valuable information from the data available.

Determination of code values for data is another interesting and important problem in data processing. It may require mathematical statistical investigations as well as representative sampling. One of the problems in the hospital morbidity studies was producting a reliable identifier for patients. A comparatively short, easy-to-code identifier was required with

negligible probability of accidental coincidence (incorrect identification). The task of selecting the representative sample was a problem of similar complexity. Sampling based on the birthday of inpatients proved to be quite uniform [3]. Usually, representative sampling from individuals of multiple occurences is a complex matter requiring complicated mathematical investigations [10].

Multiple hospitalized persons and inpatients having multiple diagnoses require a file organization different from usual statistical data bases. The elements to be examined arenot the original records (hospitalized cases). New basic elements (one multiple hospitalized person or one diagnosis) are to be constructed. Problems of this kind are directly connected to data bases (to relational data models [1,4] especially).

3. PROGRAM GENERATING

System GENERA is a system to build generator programs having subsystems. Subsystems can have a set of parameters, they are given value by unified and flexible methods. A generator system based on GENERA has a predefined host language (or a set of host languages such as FORTRAN or PL/1). Text to be processed consists of host language statements and GENERA directives. Former ones become statements of generated program without any modification. On the other hand, the appropriate text generated by the designated subsystem replaces the directive.

The example in *Fig. 1* illustrates a source file of a generator system. Function of directives is not be explained here, for details see the following sections of this paper and [12] describing the generator system

3.1. STRUCTURE OF A SYSTEM BUILT UP ON G E N E R A

A system based on GENERA integrates any number of generator procedures to make a precompiler. These procedures form subsystems of the generator program and are called into execution by entering a directive onto the source file. Detecting

a directive control is passed to the main entry point of the subsystem to read in parameters. Then the subsystem is executed. Having completed its function, subsystems return control to the main program to continue processing of the source file.

OPTION is a subsystem of program control (see example on *Figure 1*). It can be executed as the first step of a GENERA run and initializes some global variables of system to achieve a non-standard handling of source lines. The user controls the structure and contents of output information (generated program and listings) by OPTION.

A preprocessor subsystem (PREP) is contained in batch oriented versions of GENERA. This is a subsystem that cannot be called in by the user directly, and is always executed prior to any other functions of GENERA. Each line of input is examined, lines containing directives or parameters are checked. Statistics of the recognized directives are collected, and the unrecognized ones are reported. Then the parameters are tested if they meet the rules defined for the subsystem. Having found an error, the run is terminated abnormally at the end of preprocessor phase. The preprocessor performs transformations on parameter descriptions to provide an interface between a user-oriented description scheme and the program requirements. It can make both the programming of subsystems and definition of parameters easier.

As GENERA processes a number of input files (primary input containing host language program, directives and parameters; secondary input file containing specially structured data for certain subsystems; job generator (JOBGEN) input file describing non-standard job-setup) an Initial File Conversion Subsystem (IFCS) accompanies the system. IFCS builds up the input files from a single input file (MIXEDIN) and it can include some additional features (such as selecting given disc files or tapes as parts of input file) depending on the possibiliites provided by the operating system.

4. STATISTICAL INFORMATION SYSTEM SIS79/GENERA

System SIS77 and SIS79 have been mentioned before. The most important procedures of SIS79 will be presented here.

4.1. DATA TRANSMISSION AND CONVERSION

In a system to handle a great mass of data, efficiency of input-output operations is important. We developed a pair of I/O statements (#LECTOR, #SCRIPTOR) to perform these operations They are given the record structure (name, length and type of each field), and a program-fragment is generated to read or write the annotated variables.

The example in *Figure 1* shows an input directive #LECTOR. The meaning of the set $PARAM goes without saying. The set $DESCR gives format for reading record named PATIENT (COBOL--style level numbers and FORTRAN format items are used).

Procedures to generate I/O operations are needed in some systems because high-level languages analyze format specifications in run-time. Formats are usually not changed while running the program, so run-time evaluation is not needed. However, compilers do not translate format items to machine code.

Our input procedure generates a set of host-language statements to read the record 'as-is' (without any conversion) and to select and convert values of variables using efficient character-handling routines. Hence, FORMAT items are evaluated in compile-time instead of run-time. A large amount of processor time can be saved if there are I/O statements frequently used in the program. The method is especially useful in FORTRAN programs.

4.2. COMPRESSED BINARY STORAGE

Data storage can be a problem of great importance in some statistical systems. Let us see the following example. A large amount of data is to be stored on mass storage devices. It is known that data set contains numbers of small values. These numbers can be described by one or two decimal digits but they are freqently used and character form requires a conversion to

be performed each time the data are read or written. On the other hand, data stored in binary form can be read or written without any conversion but in this case each number requires a full word of storage. (It is right for word oriented machines only.) We should find a method that is efficient in both means. That is, it should provide a fast conversion and data should not occupy superflouos storage space.

The compressed binary representation used in our system reduces the storage space required while processor time used is not increased significantly. It is achieved by compressing length of binary form to the number of bits required to contain the greatest allowable value of variable. The compressed binary read and write procedures generate a program-fragment performing I/O operationand compression or decompression.

4.3. DATA TRANSFORMATION AND GRAPH REPRESENTATION OF FUNCTIONS

Data preparation tasks involving transformations (coding, analysis of functions) are included in this group. To perform these tasks we have to describe the transformation procedure itself. It does not cause any difficulties in the case of functional dependencies defined by simple formulas. On the other hand, code-tables can be extremely large, larger than the total amount of core memory available on the machine used. Description, control and storage of these tables can cause hardly resolvable problems. One of the installed generator systems based on GENERA, system SIS79, involves a certain storage method especially designed to be used in generated programs.

Using this method, functions or transformations defined by code-tables are described in the form of a hierarchical graph [13,15]. This graph is divided into levels corresponding to arguments of the function. A level contains one or more subtables controlling values of the variable belonging to given intervals. Being empty parts or identical segments included in the table, this method can provide a significant reduction of storage required for the table. Moreover, an efficient program can be generated to read and analyze the graph. While the necessary storage capacity is radically reduced (e.g. in a sys-

tem used by the Health Service, tables based on the international code system of diseases were reduced to 5 percent of size, approximately), compute time did not increase essentially as compared to the time required for the method using a unique large table of values. Reduction of storage and run-time contains several interesting problems of graph theory and finite projective geometry [16].

Figure 1 contains two consequtive GRAPH directives. The first, "AGE CODING", codes variable AGE to variable CDAGE using one code table (SACKNO=1, LEVELS=1). Maximal value allowed for AGE is 100 (UPPBOU=100). Variable NUMBER is used for signaling errors. Second procedure "CONTROL" checks variables CDAGE, SEX, MAINCD and SUBCD using a graph of 4 levels and 34 elementary tables.

The tables are filled up by a general procedure contained in systems SIS77 and SIS79. Several advantages are obtained using this subroutine: the method applied to fill up the tables is the most compact and comfortable one, appropriate security is provided by the syntax analysis of table descriptions and detailed error messages. This subroutine provides means for a quick and easy calculation of some multivariate functions as well. We demonstrate the method to construct a graph on *Figures 2-4* using a very simple function. Table generating statements (on *Fig. 4*) contains (left to right) command codes, table identifiers or table values, index values and optional comments. Negative values are pointers.

4.4. EVALUATION OF LOGICAL EXPRESSIONS

Performing a statistical analysis, sometimes, data base should be divided into parts meeting requirements of the subsystem to be used. Decision rules dividing the data base are usually described by logical expressions of high complexity. In system SIS79 a generator procedure is applied to provide a simple method for defining these rules and to generate a program fragment performing the selection. (On mathematical logical investigations concerning this topic a lecture was given by I. Ratkó in Salgótarján, Hungary at the Conference on Mathematical Logics in Theory of Programming.)

We performed interesting investigations in probability theory concerning the problem of selection [13] to find optimal strategies of file dividing.

4.5. TABLE-FILES, OUTPUT TABLES

Results obtained by statistical data processing do not contain the data of individual items but those of typifying ones. Thus the files consisting of these raw data must be transformed into that of statistical data (frequency characteristics, code values totals, quadratic sums, product sums, etc.). Consequently, in statistical information systems it is not advisable to apply the languages developed particularly for handling and querying processes of raw data items. We achieved that after a suitable preprocessing (creating 'table-files') a lot of different output tables can be obtained using a few seconds of CPU time (on HwB 66/60) independently of the size of the sample. It makes possible to perform statistical study of large sample in interactive mode, too.

5. REFERENCES

[1] E.G. Codd, "A Relational Model of Data for Large Shared Data Banks", Comm. ACM, Vol. 13., 1970, pp. 377-387.

[2] E.G. Coffman, P.J. Denning. Operating Systems Theory, Prentice-Hall, 1973.

[3] M. Csukás, L. Greff, A. Krámli, and M. Ruda, "An Approach to the Hospital Morbidity Data System Development in Hungary", Colloques IRIA, Tome 1, Informatique Medicale, 1975, pp. 381-390. (paper presented at the Symposium on Medical Data Processing, Tolouse, 1975)

[4] J. Demetrovics, "On the Equivalence of Candidate Keys with Sperner Systems", Acta Cybernetica, Vol. 4, No. 3, 1979, pp. 247-252.

[5] J. Demetrovics, E. Knuth and P. Radó, "Specification Meta Systems", Computer, May 1982, pp. 29-35.

[6] D.E. Denning, P.J. Denning, and M.D. Schwartz, "The Tracker: A Threat to Statistical Database Security", ACM Transactions on Database Systems, Vol. 4, No. 1. 1979, pp. 76-96.

[7] W.J. Dixon, M.B. Brown (editors), BMDP Biomedical Computer Programs (P-series), Uniersity of California Press, Berkley, Los Angeles, London, 1979.

[8] M. Finkelstein, "A Compiler Optimization Technique", Computer Journal, Vol. 11, No. 1, 1968, pp. 22-25.

[9] J. Foisseau, R. Jacquart, M. Lemaitre, M. Lemoine, J.C. Vignat, and G. Zanon, "Program Development With or Without Coding", Software World, Vol. 12, No. 1. 1981, pp. 9-12.

[10] L. Greff, A. Krámli and J. Soltész, "The Modeling of the Sampling Procedure for the Hungarian Hospital Morbidity Studies", Modeling Health Care Systems (ed. E. Shingan, P. Aspden, P. Kitsul), IIASA, Laxenburg, Austria, 1979, pp. 172-177.

[11] M.G. Kendall, A. Stuart, The Advanced Theory of Statistics Vol. I-III, Griffing, London, 1958, 1961, 1966.

[12] P. Kerékfy, "GENERA - A Program Generator System", Progress in Cybernetics and Systems Research, Vol. 11., Hemisphere, Washington, 1980. (paper presented at the Fifth European Meeting on Cybernetics and Systems Research (EMCSR'80), Vienna, 1980.)

[13] P. Kerékfy, A. Krámli, and M. Ruda, "SIS79/GENERA Statistical Information System", Progress in Cybernetics and Systems Research, Vol. 11., Hemisphere, Washington, 1980. (paper presented at the Fifth European Meeting on Cybernetics, and Systems Research (EMCSR'80), Vienna, 1980.)

[14] E. Knuth, P. Radó, and A. Tóth, "Preliminary Description of SDLA", Tanulmányok - MTA Számitástechnikai es Automatizálási Kutató Intézete,105/1980.

[15] A. Krámli, M. Ruda, M. Csukás, and M. Galambos, "Large Sample Size Statistical Information System for HwB", Data Analysis and Informatics, ed. E. Diday, North-Holland, 1980, pp. 457-462. (paper presented at the Second International Symposium on Data Analysis and Informatics, Versailles, 1979.)

[16] A. Krámli, P. Lukács, and M. Ruda, "Probabilistic Approach to the Performance Evaluation of Computer Systems", Proceedings of the Third Hungarian Computer Science Conference, Vol. I, Invited papers, Budapest, 1981. pp. 51-64.

[17] N.H. Nie et al., SPSS Statistical Package for the Social Sciences (end edition), Mc Graw-Hill, 1975.

[18] J. Nievergell, "On the Automatic Simplification of Computer Programs", Comm. ACM, Vol. 8, No. 6, 1965, pp. 366-370.

[19] B. Perron (ed.) et al., IDMS Concepts and Facilities, Cullinane Corporation, 1977.

[20] M. Ruda, "Some Estimates in Connection with the Critical Path Method", Project Planning by Network Analysis, Proceedings of the Second International Congress (ed. H.J.M. Lombaers), North-Holland, Amsterdam, 1969, pp. 207-215.

[21] M. Ruda, "Statistical Information System with Health Service Application", MTA SZTAKI Tanulmányok, 87/1978, pp. 167-172. (paper presented at the Fourth Winterschool of Visegrád on the Theory of Operating System, Szentendre, Hungary, 1978.)

[22] M.D. Schwartz, D.E. Denning, P.J. Denning, "Linear Queries in Statistical Databases", ACM Transactions on Database SYstems, Vol. 4, No. 2, 1979, pp. 156-167.

[23] A. Wald, Sequential Analysis, Wiley, New York, 1947.

ÖSSZEFOGLALÁS

MEGJEGYZÉSEK A STATISZTIKAI ADATFELDOLGOZÁSSAL KAPCSOLATBAN

J. Demetrovics, P. Kerékfy, A. Krámli, M. Ruda

A dolgozat a statisztikai adatfeldolgozásban használt program-generáló eljárásokat ismerteti.

ОБ ОБРАБОТКЕ СТАТИСТИЧЕСКИХ ДАННЫХ

Я. Деметрович, П. Керекфи, А Краммли, М. Руда

Изучается система генерирующая программы для обработки статистических данных.