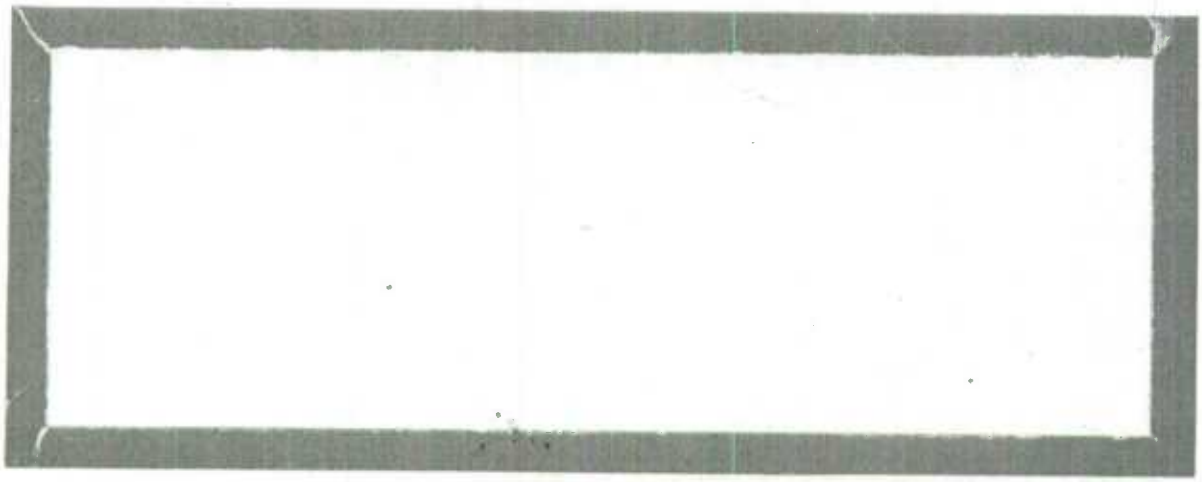
Statistics Canada | Statistique Canada



Methodology Branch

Business Survey Methods Division

Direction de la méthodologie

Division des méthodes d'enquêtes entreprises

Canada

WORKING PAPER NO. BSMD-89-001E

METHODOLOGY BRANCH

CAHIER DE TRAVAIL NO. BSMD-89-001E

DIRECTION DE LA MÉTHODOLOGIE

USE OF CHERNIKOVA'S ALGORITHM IN THE GENERALIZED EDIT AND IMPUTATION SYSTEM

by

I. Schiopu-Kratina and J.G. Kovar

January 1989

Use of Chernikova's algorithm in the Generalized Edit and Imputation System

I. Schiopu-Kratina, J.G. Kovar

Résumé

Ce document décrit une partie de la méthodologie du "Système généralisé de vérification et d'imputation" de Statistique Canada. Cette méthodologie est basée sur un algorithme, développé par Chernikova et généralisé par Rubin, qui trouve les solutions d'un système d'inégalités linéaires dont la cardinalité est minimale. L'application de ces résultats à la vérification et l'imputation des données est due à G. Sande.

CONTENTS

Use of Chernikova's algorithm in GEIS

0. Introduction

The purpose of this report is to present that part of the methodology of the Generalized Edit and Imputation System (GEIS) presently being developed at Statistics Canada which uses Chernikova's algorithm. This system will be used by several surveys, thus saving on development costs. It uses methods from linear programming to perform edit and imputation for numerical data. The applicability of such methods to edit and imputation in statistical business surveys was revealed by G. Sande (see [18], [19]). Based on his work, a system was built to deal with numerical edit and imputation (NEIS).

Chernikova's algorithim finds the extremal points associated with a system of linear inequalities with nonegative variables (see [2], [3] and [18]). The algorithm is used in GEIS for part of edit analysis (extremal points and implied edits) and in error localization. The methodology of the edit analysis is essentially based on the use of the simplex algorithm and will be described elsewhere (see [15]). The search for the closest donor to a record requiring partial imputation is done using k-d tree and is presented in [5].

For the sake of completeness, a brief overview of the entire system is given here. A numerical set of edits can often be represented as a linear system of inequalities, where the values of the variables are generally reported by the respondents. The general form of edits is specified by the subject matter specialists. Since the values reported in business surveys are non negative, it is assumed that all edits are in the form of linear inequalities and that all variables to be reported are nonegative. Various techniques employed in linear programming can be used to analyse the system of edits to check for inconsistencies, redundancies and hidden equalities. Bounds on all variables can be obtained at this stage. Details regarding the edit analysis can be found in [13]. The linear system of inequalities representing the edits defines the acceptance region associated with this system. If bounded, the acceptance region can be described with the help of the extremal points. These can be obtained using Chernikova's algorithm as presented in 1.1. For the purpose of edit analysis, new (implied) edits are generated by taking linear combinations with positive coefficients of subsets of the original set of edits. These edits contain fewer variables than the original edits and are thus easier to handle.

A major component of NEIS and GEIS is error localization, the module which identifies, for each record, the minimum number of fields that should be changed so that the corrected record passes the edits. This module, (see [10]) is placed in GEIS after the edit analysis modules and before the donor imputation modules. Conceptually, it provides the link between edit and imputation, as it flags for imputation the blank fields as well as the invalid values. Ideally, on each record the values for all fields should be reported and the record should pass all edits. Often the record is incomplete; that is, some of the fields on the record are blank. Even for a complete record, some of the values could be erroneously reported so that the record does not pass the edits. These values should then be removed and the corresponding fields, as well as the blank fields, should be flagged for imputation. Performing these tasks manually for each record is time consuming. It can also lead to errors, as the system of edits can be very complex and the number of fields very large. Corrections to some of the reported values, when based solely on the experience and the intuition of the editor may turn out to be wrong as the corrected record may not pass the edits. This is due to the fact that all variables are related through a complex system which is difficult to keep track of intuitively. A systematic way of error localization is provided by NEIS and GEIS.

Once the fields which require correction have been identified, the record, now labelled a recipient or candidate record, is ready for imputation. For the purpose of donor imputation, the values on each field on all records are rescaled so that they all belong to the interval (0,1). A k-d tree is constructed using a class of accepted records (see [5]). The m closest records to the recipient record are found in the space of matching variables using the L^∞ distance. From these potential donors, the closest suitable donor for the recipient record is selected and all values on the fields to be imputed are transfered from this donor record to the recipient record. A potential donor is a suitable donor for a recipient record if the recipient record passes the edits once the pertinent fields have been imputed from the donor record. The concept of matching fields, described in section 2.3 of this report, is intimately related to the concept of a suitable donor.

This report presents the methodology and gives a brief description of the following modules used in GEIS: the module for finding the extremal points, the module for generating the implied edits and the module used for error localization. Potential uses

of Chernikova's algorithm in finding the matching fields are also presented (see section 2.3).

The report is organized as follows. Chapter 1 gives an overall presentation of Chernikova's algorithm. The actual algorithm, as presented by Rubin [17], can be found in the Appendix. In Chapter 2, the more theoretical aspects regarding the application of Chernikova's algorithm to edit and imputation are discussed. Chapter 3 gives a general description of the algorithms used in the modules mentioned above. This report is addressed to methodologists who wish to gain some familiarity with the application of Chernikova's work in GEIS. The reader who aims for a general understanding of the methodology need not read Chapter 3 which contains implementation details. On the other hand, the reader who is more interested in the implementation, may start by reading Chapter 3 and use Chapters 1 and 2 for references. For this reason, we attempted to render each section of Chapter 3 self-contained. Even so, not all details which were used in programming have been included in Chapter 3. The interested reader should consult [8-14] in the references for this purpose.

1. Description of the algorithm and Rubin's cardinality constrained vertex generation.

1.0 Overview

In [2-3], Chernikova described an algorithm for finding all nonegative solutions of a system of linear inequalities. The general form of such a system is:

$$A\,x \leq b \tag{1.1 a}$$

$$x \geq 0 \tag{1.1 b}$$

where A is the matrix of coefficients with m rows and n columns, x is a vector in R^n, $x = (x_1, \ldots x_n)$ and b is a column matrix with m rows. The notation $x \geq 0$ means that all coordinates of x are nonegative, i.e. $x_i \geq 0$ $i = 1, \ldots n$. The system (1.1) may have no solution. It may also have one solution or infinitely many solutions. In the latter case, all points x which satisfy (1.1) belong to a polyhedron in R^n. Conversely, all points inside or on the facets of the polyhedron are solutions of the system (1.1). The region enclosed by the polyhedron is called the acceptance region. This region can be described by its extremal points, and extremal directions. Chernikova's algorithm produces all this information.

In [17], Rubin showed that Chernikova's algorithm can be adapted so as to find all the solutions of a cardinality constrained linear program (see 2.2.3). In his problem, a linear function of x is maximized, subject to (1.1) and a cardinality constraint. The cardinality constraint requires that the number of nonzero coordinates of the solution be less than or equal to a prespecified value. Notice that the cardinality function defined in 2.2.3 is not linear. Therefore the classical simplex method could not be applied to solve a cardinality constrained linear program.

In what follows, we are only concerned with solutions to (1.1) subject to the cardinality constraint. Rubin's application of Chernikova's algorithm relies mainly on two facts:

1) as the algorithm proceeds, the cardinality of the points which generate the extremal points does not decrease (see Lemma 2 of [17]).

2) if a solution to (1.1) with the cardinality constraint exists, then at least one vertex will satisfy the same constraints.

We will prove 2) in the case the polyhedron defined by the system (1.1) is bounded. The general case can be obtained by "regularization" (see p. 556 of [17]).

Firstly, if $f(x)$ represents the cardinality of x then:

$$(1.2) \quad f\left(\sum_{i=1}^{k} \alpha_i x_i\right) \geq \sum_{i=1}^{k} \alpha_i f(x_i),$$

where $\alpha_i \geq 0$, $i=1, \ldots k$ and $\sum_{i=1}^{k} \alpha_i = 1$

The proof of (1.2) uses the fact that $x_i \geq 0$, $i = 1, \ldots k$.

Secondly, since the polyhedron described by (1.1) is a convex body, every interior point x can be written as a convex combination of vertices, $x = \sum_{i=1}^{k} \alpha_i x_i$. If $f(x) < f(x_i)$ $1 \leq i \leq k$, i.e. if $f(x)$ attains a minimum at x, then $\sum_{i=1}^{k} \alpha_i f(x_i) > \sum_{i=1}^{k} \alpha_i f(x) = f(x) = f\left(\sum_{i=1}^{k} \alpha_i x_i\right)$ which contradicts (1.2).

Consequently, it suffices to find the extremal points which satisfy the cardinality restriction for a global solution to Rubin's cardinality constrained linear program.

A drawback of Chernikova's method for finding extremal points is the large number of transformations that have to be performed at times to arrive at the final matrix. This may require extended space for storing intermediate matrices. However, as Rubin has shown (see [17]), the performance of the method can be considerably improved when solving a cardinality constrained linear program, due to properties 1) - 2) stated above.

The importance of property 1) is that it allows the reduction of the number of transformations in Chernikova's algorithm. As we shall see in 2.2, many of the columns containing vectors which exceed the bound on cardinality can be discarded, as they will never produce the desired solution. The growth of the

matrix used in Chernikova's algorithm can therefore be curtailed. Consequently, there is less danger that the space required for storing intermediate matrices will be exceeded.

The reader is cautioned that in the original work of Chernikova (see [2-3]) the algorithm uses the transpose of the matrix of coefficients and the processing is done columnwise. In other papers ([17-19]) the algorithm proceeds rowwise. In this report we follow the latter procedure.

1.1 Description of Chernikova's algorithm

1.1.1 The homogeneous case. An example.

The algorithm was initially developed for finding extremal rays (i.e. the edges) of a cone defined by the system of homogeneous inequalities:

$$\begin{aligned} A\,x &\geq 0 \\ x &\geq 0 \end{aligned} \tag{1.3}$$

Generally speaking, the algorithm consists of transforming an augmented matrix which contains all the coefficients of the system (1.1) until all entries of the augmented matrix are non-negative. The rows corresponding to the matrix of coefficients in the augmented matrix are processed one at a time, resulting each time in a transformation of the augmented matrix. In the transformed matrix, all rows that have already been processed have non-negative entries. A transformation consists of copying columns which have the desired entries on the row being processed (non-negative entries for rows corresponding to inequalities and zero entries for rows corresponding to equalities) and linearly combining other columns. The coefficients of these linear combinations are always positive.

A complete description of the algorithm can be found in [17]. For convenience, a copy of Rubin's description can be found in the Appendix. We present here a simple example which can also be visualized in the three dimensional space. Some steps of the algorithm are illustrated by means of this example. Further examples can be found in [2], [3] and [17].

In what follows, the words tableau and matrix are synonymous.

Example 1 Find the extremal rays of the cone defined by the linear inequalities (1.3):

$$\text{1.3 a)} \qquad \begin{aligned} x + y - z &\geq 0 \\ 2x - y &\geq 0 \\ x - 2y - z &\geq 0 \end{aligned}$$

$$\text{1.3 b)} \qquad \begin{aligned} x &\geq 0 \\ y &\geq 0 \\ z &\geq 0 \end{aligned}$$

The vector of unknowns is (x,y,z). The matrix of coefficients for the system 1.3a) is

$$A = \begin{bmatrix} 1 & 1 & -1 \\ 2 & -1 & 0 \\ 1 & -2 & -1 \end{bmatrix}$$

An augmented matrix (tableau) Y corresponding to the system (1.3) is formed by placing the 3 x 3 identity matrix L at the bottom of the matrix A.

$$Y = \left[\begin{array}{ccc} 1 & 1 & -1 \\ 2 & -1 & 0 \\ 1 & -2 & -1 \\ \hline 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{array}\right]$$

The first row is first processed to produce the extremal vectors associated with 1.3b) and the first inequality in 1.3a).

The first two columns are copied, since the corresponding entries are positive and the first row corresponds to an inequality. This is a consequence of the fact that the unit vectors (1,0,0) and (0,1,0), placed on the corresponding columns in the lower submatrix, are already

extremal vectors of the cone represented by 1.3b) and the first inequality in 1.3a). Then the first and the second column are each added to the third in order to create zeroes on the third and fourth column of the new tableau. The new tableau is:

$$Y_1 = \left[\begin{array}{cccc} 1 & 1 & 0 & 0 \\ 2 & -1 & 2 & -1 \\ 1 & -2 & 0 & -3 \\ \hline 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \end{array}\right]$$

The first row has now been processed and all the vectors in the lower matrix L_1 are extremal vectors associated with the extremal rays of the cone: $x + y - z \geq 0$ and 1.3b). Indeed, the first two vectors are along two coordinate axes, whereas (1,0,1) and (0,1,1) are on the plane $x + y - z = 0$. The knowledge of the extremal rays of the cone is sufficient to define the acceptance region. Notice that the zero entries on the first row of Y_1 indicate the fact that the corresponding column vectors in L_1 are actually on the plane $x + y - z = 0$. Notice also that the vectors in L_1 are either unit vectors which satisfy the first inequality in 1.3a) or linear combinations of unit vectors which satisfy the first inequality in 1.3a). In this first step, all combinations of columns which have opposite signs on the first row Y are calculated. When combining such columns, zeroes are created in the first row of the tableau Y_1. Geometrically, this means that two of the unit vectors are combined if they are on different sides of the plane $x + y - z = 0$. The coefficients are chosen so that the resulting vector is actually on that plane. Notice that the constraints imposed by the second and third inequality in 1.3a) have not been used so far.

Now we process the second row of the tableau Y_1. The new tableau Y_2 is:

$$Y_2 = \left[\begin{array}{cccc} 1 & 0 & 3 & 0 \\ 2 & 2 & 0 & 0 \\ 1 & 0 & -3 & -6 \\ \hline 1 & 1 & 1 & 1 \\ 0 & 0 & 2 & 2 \\ 0 & 1 & 0 & 3 \end{array}\right]$$

Now all vectors in L_2 are extremal vectors of the cone corresponding to 1.3b) and the first two equations in 1.3a). The first and third columns of Y_1 which were copied to Y_2 correspond to extremal vectors obtained at step 1 which also satisfy the second inequality in 1.3a) (see Rule 1 of [2]). This can be seen in Y_1, since the first and the third columns have positive entries on the second row so they are on the "positive" side of the plane $2x - y = 0$. The first and second, and the third and fourth columns of Y_1 are combined to produce the last two columns of Y_2. Other combinations were not performed as required by step 3a of the Appendix. Notice that the constraint imposed by the last inequality in 1.3a) has not been used so far.

We now process the third row. The first two columns are copied. The first column of Y_2 is multiplied by 3 and added to the third column to produce the third column of Y_3. The last column and the first column of Y_2 are not combined, again by 3a of the Appendix. The algorithm terminates with this step and Y_3 is:

$$Y_3 = \left[\begin{array}{ccc} 1 & 0 & 6 \\ 2 & 2 & 6 \\ 1 & 0 & 0 \\ \hline 1 & 1 & 4 \\ 0 & 0 & 2 \\ 0 & 1 & 0 \end{array}\right]$$

The top matrix has non-negative entries. The system (1.3) has therefore a nontrivial solution (i.e. different from (0,0,0) which is always a solution of (1.3)). The three extremal rays all start at the origin and pass through the following points in R^3: (1,0,0) (1,0,1) and (2,1,0). The region enclosed by these vectors is the acceptance region associated with the system (1.3).

1.1.2 The general case

The general form of the system of linear inequalities is (1.1). The system (1.1), when consistent, represents a polyhedron in R^n. If the polyhedron is bounded, its vertices, or extremal points, define the acceptance region associated with the system (1.1). In order to solve

this system using Chernikova's algorithm, the system (1.1) is transformed into a system of type (1.3) as follows. An equivalent form of (1.1) is:

$$-A\,x + b \geq 0$$

$$x \geq 0$$

This can be reduced to a homogeneous system in $n + 1$ variables $(x_1, \ldots x_n, \xi)$, $\xi \geq 0$:

$$-A\,x + b\,\xi \geq 0$$

$$\begin{bmatrix} x \\ \xi \end{bmatrix} \geq 0 \tag{1.4}$$

Now (1.4) represents a cone with the vertex at the origin. Every solution of (1.4) with $\xi = 1$ gives a solution of (1.1) and vice versa. It is easy to see that the extremal points of (1.1) are the extremal vectors of (1.4) for which the last coordinate $\xi = 1$. All the extremal points of (1.1) are found by using Chernikova's algorithm applied to the system (1.4) and retaining those extremal vectors for which the last coordinate is non-zero. This is so because, if (x,ξ), $\xi \neq 0$ is a solution to (1.4), then $(x \cdot \xi^{-1}, 1)$ is also a solution. When the polyhedron defined by (1.1) is unbounded, some of the extremal rays of (1.4) are of the form $(x,0)$ (see [18], p. 12). Then the coordinates of x represent the extremal directions associated with the polyhedron.

1.1.3 General comments on Chernikova's algorithm

With Example 1 in mind we now make some general comments on Chernikova's algorithm.

The original tableau contains the matrix of coefficients A on the top and the identity matrix on the bottom. The role of the lower matrix is two-fold; it keeps track of the transformations successively applied to the matrix of coefficients A and it represents the inequalities $x_i \geq 0$, $i = 1, \ldots n$ in the search for the extremal vectors of the cone (1.3).

The idea is to transform the tableau Y in an optimum way until all the entries in the upper part are non-negative. If this cannot be done, then either the cone is degenerate at the origin, when dealing with system (1.3) or there is no solution, when dealing with the system (1.1) (the two differ when at least one of the components of b is nonzero). We consider now the system (1.1). The successive transformations applied to it can be represented as matrices which multiply Y on the right hand side. Let T be the multiplication of all matrices involved in applying Chernikova's algorithm to its completion. Assume there is a solution of the system (1.1). Then $A\,T \geq 0$, $T \geq 0$. Since the same transformations are applied to the bottom matrix, we have $I \cdot T = T$, where I is the identity matrix, so the matrix T appears at the end of the algorithm in the bottom part of Y. Therefore, T provides a solution to the system (1.1). The rules of combining and discarding columns of the original matrix are such that T gives only the extremal solutions which define the acceptance region in (1.1).

This procedure brings to mind classical algorithms for finding inverses of matrices. For details, see Section 3 of [7]. Example 1 in 1.1.1 allows for a geometric interpretation of the algorithm. In [1-3] as well as in [7], algebraic methods are used. To make the connection between the two, the following observation may be helpful. Consider the second inequality in 1.3 a). In the upper matrix of Y, it is represented by the second row, which also represents the plane whose equation in R^3 is: $2x-y=0$. A vector (x_0,y_0,z_0) is on this plane if its coordinate satisfy this equation or, equivalently, if the scalar product of $(2,-1,0)$ and (x_0,y_0,z_0) is zero. All intermediate tableaux in Chernikova's algorithm have the property that the entry on ith row and jth column of the upper matrix is, in fact, the scalar product of the corresponding column vector in the lower matrix and the corresponding row vector in the <u>original</u> tableau. Indeed, the zero entry on the second row and third column of the tableau Y is the scalar product of $(2,-1,0)$ and $(0,0,1)$. The fact that it is zero corresponds to the fact that $(0,0,1)$ is on the plane $2x-y=0$. This property is preserved as the algorithm proceeds due to properties of the scalar product as well as the nature of the transformations performed in algorithm (p. 155 of [2]). For example,

the entry 1 in the last row of the upper matrix of Y_2 is the scalar product of (1,0,0) and (1,-2,-1); the latter vector represents the third inequality of 1.3 a). Therefore, when zero entries are created in the upper tableaux, the corresponding column vectors are on the planes defining the original inequalities represented by the corresponding rows. Positive entries correspond to column vectors which are on the positive side of the plane represented by the corresponding rows.

We proceed now to a brief overview of the mathematical foundation of Chernikova's algorithm. This can be essentially found in [1] and consists of a general recursive procedure of finding extremal rays of a cone defined by several inequalities. When all variables are nonegative (i.e. 1.1 b) holds), the vectors corresponding to the coordinate axes are extremal vectors associated with 1.3 b). When the first row of Y is processed, the recursive procedure is applied to construct, from these vectors, the extremal vectors associated with the cone 1.3 b) and the first inequality of 1.3 a). According to Rule 1 of [2], the vectors (1,0,0) and (0,1,0) are retained because their scalar product with (1,1,-1) is positive (indeed, it is equal to 1 in both cases). Then the first and the third as well and the second and the third columns are combined to create two extremal rays of the new cone, namely (1,0,1) and (0,1,1). At the first step, all columns with opposite signs are combined as described, since 3 of the algorithm (see Appendix) allows for it. Rule 3 of the algorithm, as described by Rubin (see Appendix) is a device used for checking the dimensionality of a linear subspace associated with rows already processed (see Satz (Proposition) 3 of [1]). Note that the maximal linear subspace (see [1]) associated with 1.1b) is the vector 0 of R^n.

So far we have not discussed the case of equalities in (1.1). An equality in the system (1.1) can be represented by two inequalities, and they can be treated as such. However, the algorithm can be simplified in the presence of equalities. Firstly, when each row representing an equality is processed, only the columns which contain zeroes on that row will be copied. This corresponds to the fact that all the extremal points of the polyhedron will have to lie on the plane described by the equality. In

deciding which columns should be combined, the rows containing processed equalities need not be checked, as they contain only zeroes. If equalities are stored at the top of the upper matrix and are processed first, or if all rows in the upper part of the matrix A correspond to equalities, the algorithm can be simplified even further. However, it has to be emphasized that the general description of the algorithm covers all particular situations.

2. Application to edit and imputation (G. Sande)

2.0 Overview

Chernikova's algorithm is used in GEIS in three places: to produce the extremal points of the acceptance region of the system (1.1) (see [8]), to generate the implied edits (see [11]) and for error localization (see [10]). We also present in section 2.3 a possible use of this algorithm for finding the matching variables. This procedure was not implemented because it involves finding the extremal points of a potentially large system.

The concept of implied edits was introduced in [6]. Essentially, variables are eliminated from the original set of edits, to generate new edits containing fewer variables than the original ones. The implied edits may be easier to understand than the original set of edits. For example, bounds for each of the variables may be provided by the new set of edits. The implied edits can provide, to some degree, a check of the consistency of the original system of edits. Inconsistency might be easier to detect in the new edits which involve fewer fields. Technical details concerning implied edits are given in section 2.1.

A uniform and systematic method for error localization is provided by NEIS and GEIS. Records with missing values and complete records which have not passed the edits are treated alike. The blank fields are assigned a value which ensures that the record, unless corrected, will fail the edits (say, a negative value so that 1.1 b) fails). To each value of the now completed record a positive and a negative correction is attached. The corrections now become the new variables, subject to constraints imposed by the system (1.1). In other words, the condition is that the corrected record passes the edits. We now have a system of linear

inequalities and we wish to find correction vectors for which the number of nonzero corrections is a minimum. This places the problem within the scope of Rubin's cardinality constrained program with a twist. The cardinality constraint is not given ahead of time; rather it is defined along the way, as extremal points are generated. The cardinality of the first extremal point found using Chernikova's algorithm serves as a first constraint and so forth. Indeed, the minimum cardinality attained at a vertex cannot exceed this constraint. The technical details regarding error localization can be found in 2.2.

Let us consider a record x of which i fields require imputation, $1 \leq i < n$, whereas the remaining n-i fields are accepted as reported by the respondent. The requirement that the record passes the edits leads, upon replacement of the n-i values of fields into the original system of edits, to a new system of linear inequalities in i variables. We call this system the reduced system. It is important then to choose a donor record in such a way as to ensure that the imputed values fall within the acceptance region of the reduced system. The imputed record will then pass the edits. If the donor record is chosen at random, there is no guarantee that the transferred values fall within the reduced acceptance region. On the other hand, if all reported fields are indiscriminately used for matching, variables which do not actually determine the reduced acceptance region may prevail in the calculation of the L^∞ distance and exclude suitable donors from the list of potential donors. It follows that only the variables that actually determine the acceptance region should be considered. These are called the matching variables and should be used in the search for a suitable donor from the set of all accepted records. It has to be noted that, in some instances, according to this approach no variable is required for matching. In such a situation the donor record may be chosen at random from the list of potential donors. The technical details regarding the matching variables can be found in 2.3.

2.1 The implied edits

2.1.1 Definition

As mentioned earlier, the concept of implied edits was introduced by Fellegi and Holt in [6]. Their article deals mostly with qualitative data but it also touches upon numerical edits. Essentially, a new implied edit is an edit which is obtained from the original or the previously implied edits by eliminating one or more variables. Successive applications of this procedure leads to the creation of all implied edits. The original as well as the newly created edits form a complete set of edits. The complete set of edits is used in error localization (see p.2). Fellegi and Holt also propose the use of implied edits for detecting inconsistencies in the original set of edits. The use of implied edits for error localization or detecting inconsistencies for numerical data is inefficient, especially when a large number of edits or a large number of variables is involved. In NEIS as well as GEIS, error localization is based instead on Chernikova's algorithm as used by Rubin in [17], which leads to a more efficient algorithm (see 2.2). Inconsistencies are detected in the analysis of edits using the revised simplex algorithm as described in [13].

For numerical edits given in the form of linear inequalities, the definition amounts to successively combining pairs of inequalities so that at least one variable is eliminated at each stage. The coefficients (weights) used for these combinations are always positive. The problem with this definition for numerical edits is that several applications of the procedure may lead to the creation of an edit which is not new. For this reason, the definition adopted for numerical edits which is given below is more restrictive.

Definition A linear inequality associated with a system of edits expressed as m linear inequalities with n variables is an implied edit for that system if it is a linear combination with positive coefficients of $k > 1$ such edits and contains at most $n - k + 1$ variables with nonzero coefficients.

The new edits contain fewer variables than the original ones and so provide the user with some idea about the form of the acceptance region associated with the original system of edits (1.1). The implied edits provide an additional diagnostic tool for the user. Even if the analysis of the original edits indicates that all is well, examination of the implied edits may identify problems with the edit specifications. This is the main purpose for deriving them in GEIS.

2.1.2 The minimum set of implied edits.

The definition of the implied edits has led to an algorithm which will be explained by means of an example.

Example 1 Assume that in the system of edits 1.1a) we combine the first three edits to eliminate two variables, x_1 and x_5. Consider the vector of weights $w = (w_1, w_2, w_3, 0, \ldots, 0)$, where the first three components are nonzero. If we look at the transpose of the matrix of coefficients, A^T, then combining the first three edits amounts to multiplying this matrix on the right by the column vector of weights w. Notice that the columns of the matrix A^T are the original edits. We now group constant terms to find the coefficients of the variables to be eliminated. The coefficient of x_1 is $a_{11}w_1 + a_{21}w_2 + a_{31}w_3$. Likewise, the coefficient of x_5 is $a_{15}w_1 + a_{25}w_2 + a_{35}w_3$ and the condition is that both expressions are equal to zero. Let us denote the first row of the matrix A^T by r_1 and its fifth row by r_5. Then, with the column of weights w as above, we are led to a homogeneous system of equations which is, in matrix form:

$$\begin{aligned} r_1 w &= 0 \\ r_5 w &= 0 \\ w &\geq 0 \end{aligned} \qquad (2.0)$$

Notice that no condition is imposed on $r_i w$, $1 \leq i \leq n$, $i \neq 1,5$.

We are interested in the vectors of weights which satisfy this system and have at most three nonzero components. When the system (2.0) admits infinitely many such solutions, the extremal vectors of weights could be

obtained using Chernikova's algorithm for such systems (see [2]). Notice that the implied edits are obtained by multiplying the transpose of the matrix of coefficients on the right by the appropriate vector of weights. They appear as newly created columns in the tableau used in the algorithm.

When producing all implied edits we are, in fact, looking at all possible systems of type (2.0) involving edits in 1.1) a) and 1.1 b). There could be infinitely many vectors of weights with k nonzero components which generate edits with at most $n - k + 1$ variables, $k > 1$. Many of these edits are redundant. In some cases, it is easy to eliminate redundant edits or identify identical implied edits. Other cases are more subtle (see [16], Appendix I). We would like to produce a minimum set of such weights that generate implied edits which, in some sense, represent all implied edits. We are therefore led to considering the entire transposed matrix associated with the system (1.1) and applying a modified version of Chernikova's algorithm to it. Only the vectors of weights which produce implied edits (new columns in the course of processing the matrix) as defined above will be retained in the lower matrix used in Chernikova's algorithm.

The initial matrix is then the transpose of the matrix of coefficients associated with the system of inequalities (1.1). For consistency, the original system is written in the form:

$$1.1\ a)_1 \qquad A_1 x - b_1 \leq 0$$

$$1.1\ b)_1 \qquad -x \leq 0$$

An edit represented by an equality in (1.1) appears in 1.1 $a)_1$ as two inequalities. If there are m edits in (1.1) of which e are equalities, $0 \leq e \leq m$, then there are $m + e + n$ edits altogether in the system above since there are n variables and therefore n edits in 1.1 $b)_1$. These edits form the columns of the upper work matrix at the onset of Chernikova's algorithm. Since there are n variables and the system is non-homogeneous, the upper matrix has $n + 1$ rows. An additional row will be used to indicate, for each column, whether the edit corresponds

to an equality or not. The lower matrix contains at the onset the unit vectors in R^{m+e+n}. It will eventually contain all the "extremal" weights generating implied edits as defined above.

The problem of finding the implied edits is somewhat dual to the problem of finding the extremal points described in 1.1.2. The actual weights, which appear in the lower submatrix of the work matrix, are irrelevant. It is the implied edits, which appear in the upper submatrix, that are of interest. In this setting, no column is discarded, as each newly created column is an implied edit. The purpose of the lower matrix is two-fold:

Firstly, each vector of weights keeps track of the number of combinations that led to the creation of the edit represented by the column above. If $f(w)$ is the cardinality of the vector w as defined in 1.0, then $f(w) = k$ represents the number of edits that were combined to generate the edit on the column above. For example, at the onset of the algorithm, the cardinality of each unit vector is 1 and the number of edits that "generated" the original edits is 1. If two different columns are combined while processing the first row, the cardinality of the new weight is 2 and so forth. When two columns are combined in the course of processing the rth row, the number z of zero coefficients created in the first r rows of the new edit is counted and compared to k. Only if $z \geq k - 1$, is the new edit retained. Note that the row containing the coefficients b_1 (see 1.1 a)$_1$ above) is combined as it is used in the calculation of the new edits. It is, however, not used in the calculation of z as we are only interested in eliminating variables.

Secondly, the lower submatrix represents the last constraint in the system (2.0) above and is therefore used in Chernikova's algorithm in restricting the number of possible combinations that generate the "extremal" weights and therefore the number of "representative" implied edits.

On the other hand, the upper matrix could be viewed as an augmented matrix in which all systems of type (2.0) are embedded and so it imposes no restrictions on the combinations of columns generating the implied edits (see [2]).

In summary, the purpose of this algorithm is to generate, edits applicable to subsets of variables, by linearly combining a minimum number of edits. The coefficients of these linear combinations (weights) are positive. The implied edits are stored in the upper submatrix of the matrix used in Chernikova's algorithm.

2.2 Error Localization

2.2.1 The problem

As mentioned in 2.0., the purpose of error localization is to identify, for each record, the minimum number of fields which should be modified so that the corrected record passes the original set of edits. Records with missing values also fall into this category, as the missing fields could be given a negative value, say -1, to ensure that the record will not pass 1.1b) of the original system.

Let x represent the n values of the n fields of a record (missing values have been set equal to -1). Since we do not know which field is correct, every field is corrected and the modified record must now satisfy the system (1.1). More precisely, let $y = (y_1, \ldots, y_n)$ be the vector of positive corrections and $z = (z_1, \ldots, z_n)$, the vector of negative corrections, each with non-negative components. We can look at a vector of corrections c as a vector having 2n components, $c = (y_1, \ldots, y_n; z_1, \ldots z_n)$. The corrected record is $x + y - z$. The reason for using positive and negative corrections is that we want to place ourselves in the context of chapter 1 which requires that all variables be positive; yet for some fields of the record the reported values are too large and a negative correction is required.

Example 1 Consider the record $x = (x_1, x_2)$. Assume that x_2 is correct and that the value of x_1 has to be decreased by 2 units. Then $c_1 = (1, 0; 3, 0)$ will provide the necessary corrections as $x_1 + 1 - 3 = x_1 - 2$. On the other hand, so does $c_2 = (0, 0; 2, 0)$. There are, therefore, infinitely many correction vectors that can be successfully used.

Definition 2.1 A vector of corrections $c = (y_1, \ldots, y_n; z_1, \ldots, z_n)$ is in reduced form if $y_i \, z_i = 0$, for all $i=1, \ldots n$. In other words, for each coordinate i, either $y_i = 0$ or $z_i = 0$. This is equivalent to the complementary condition of [18], $\langle y, z \rangle = 0$ where $\langle \, , \, \rangle$ is the scalar product in R^n.

For details concerning the complementary condition see [18].

Example 2 In Example 1, c_1 is not in the reduced form, whereas c_2 is.

The requirement that the corrected record passes the edits yields:

$$\begin{aligned} A\,(x + y - z) &\leq b \\ x + y - z &\geq 0 \\ \begin{bmatrix} y \\ z \end{bmatrix} &\geq 0 \end{aligned} \tag{2.1}$$

We also require that the number of actual corrections to the record be a minimum. If $f(x)$ is the cardinality of x as defined in 1.0, then we wish $f(y-z)$ to be a minimum.

As noted in example 1, there are infinitely many correction vectors which provide the same actual correction of the record x. We would like to consider only vectors in reduced form, that is, which satisfy the complementary condition introduced in Definition 2.1.

Using (2.1), the problem can now be stated as follows: Find all possible correction vectors $c = (y;z)$ such that $f(y-z)$ is a minimum, subject to:

$$\begin{aligned} A(y - z) &\leq b - Ax \\ y - z &\geq -x \\ \begin{bmatrix} y \\ z \end{bmatrix} &\geq 0 \\ \langle y, z \rangle &= 0 \end{aligned} \tag{2.2}$$

Notice that here x is known and the variables are y and z. We can therefore restate the problem in R^{2n}, with appropriate matrices A_1 and b_1 (see 2.2.4):

Find min $f(y-z)$, where $(y;z)$ satisfies:

$$A_1 \begin{bmatrix} y \\ z \end{bmatrix} \leq b_1$$

$$\begin{bmatrix} y \\ z \end{bmatrix} \geq 0 \tag{2.3}$$

$$< y,z >= 0$$

The complementary condition is not linear in y and z and so it is not obvious that the system (2.3) can be solved as a cardinality constrained linear program. Neither are we dealing with the cardinality of the vector (y,z) in R^{2n}. We will show in 2.2.2 that, in the presence of the complementary condition, we minimize the cardinality of the vector (y,z) in R^{2n}. We also show that, solving (2.3) without the complementary condition leads to solutions which, in fact, satisfy it. As we shall see, exploiting the properties of the correction vectors which are in reduced form leads to the design of a more efficient algorithm.

If the original system of edits is consistent, there is always a solution to the system (2.2). In fact, in most cases, we encounter more than one solution, even in the presence of the complementary condition. We are interested in the patterns of corrections.

Example 3 Let c_1 be a vector of corrections in R^{2n} such that $y_1 \neq 0$, $z_2 \neq 0$ and all other coordinates are equal to zero. Consider another vector of corrections c_2, with $y_3 \neq 0$, $z_4 \neq 0$ and all other coordinates zero. These two vectors have different patterns and the same cardinality in R^{2n}, $f(c_1) = f(c_2) = 2$. If the first solution is retained, we decide to impute the first and the second field; if c_2 is retained, we impute the third and the fourth. The decision taken at this point will determine the path that the record will follow from here on in the GEIS.

It has to be noted that the actual coordinates of the correction vectors given by this method are ignored. Using these values would amount to making the minimum change in the corresponding variables in order for the record to pass the edits. The records so imputed will then always be placed on some facet of the polyhedron defining the acceptance region and that will change the distributions of the data variables in the

imputed dataset. We would therefore like to use more appropriate imputation techniques.

2.2.2 The complementary condition

Let f denote the cardinality function in R^n, n=1, 2... . To each vector $(y_1, \ldots, y_n; z_1, \ldots, z_n)$ in R^{2n} we associate a "reduced" vector $(y_1 - \alpha_1, \ldots y_n - \alpha_n; z_1 - \alpha_1, \ldots z_n - \alpha_n)$, where $\alpha_i = \min (y_i, z_i)$, i=1, ... n.

Example 4 The reduced vector corresponding to the vector (2, 5; 3, 1) is (0,4 ; 1,0).

Proposition 1 The cardinality of a vector in R^{2n} exceeds the cardinality of the associated reduced vector. Equality takes place if and only if the vector is in reduced form.

To justify this property, we look at Example 4. There f(2, 5 ; 3, 1) = 4 > 2 = f(0, 4; 1, 0).

Notice also that, if c is in reduced form, then f(c) = f(y - z), where y - z $\in R^n$.

Proposition 2 If c satisfies (2.3) without the complementary condition, then the reduced vector associated to c also satisfies (2.3) and, of course, the complementary condition. This follows from the form of (2.2) which is equivalent to (2.3).

From these properties, it follows that it is possible to solve (2.3) as a cardinality constrained linear program in R^{2n} (see chapter 1). Indeed, (2.3) is equivalent to solving the problem:

Find min $f(y\ ;\ z)$ subject to:

$$(2.4) \qquad A_1 \begin{bmatrix} y \\ z \end{bmatrix} \leq b_1$$

$$\begin{bmatrix} y \\ z \end{bmatrix} \geq 0$$

In 2.2.4 we show how to adapt Rubin's cardinality constrained vertex generation to solve this problem.

2.2.3 The cardinality function

In 2.2.1 we considered the problem of correcting a record by altering a minimum number of fields. The cardinality function used was defined, for each $x = (x_1, \ldots x_n)$ by $f(x) = \sum_{i=1}^{n} \delta(x_i)$

where $\delta(x_i) = \begin{cases} 0 \text{ if } x_i = 0 \\ 1 \text{ otherwise} \end{cases}$ $\qquad i = 1, \ldots, n.$

This function was applied to the correction vector $c = (y_1, \ldots, y_n\ ;\ z_1, \ldots, z_n)$ in R^{2n}. There are various types of "cardinality" functions which could be used, depending on the problem at hand. As long as such a function possesses the basic properties required in Rubin's cardinality constrained linear program (see 1.0 of this report, as well as p. 13 of [18]), its minimum, subject to the constraints imposed by a system of linear inequalities is attained at a vertex. If follows that one can solve cardinality constrained linear programs as described in 2.2.2 where the function to be minimized represents a generalized cardinality function.

Examples of generalized cardinality functions can be found on p. 14 of [18]. In GEIS we are concerned with generalized cardinality functions of the type $f(x) = \sum_{i=1}^{n} w_i\ \delta(x_i)$, where w_i is a positive coefficient (weight) associated with the i th field. If $c = (x_1, \ldots x_n;\ y_1, \ldots y_n)$ is a correction vector, then the same weight w_i is associated to both x_i and y_i , $i \leq n$. Note that f is now applied to a vector c in R^{2n}. The use of this kind of generalized function in GEIS reflects the fact that the values reported for certain fields are considered more reliable than others. For various patterns with the same number of fields, the patterns which require corrections to the least reliably reported fields are selected for imputation.

It is also possible to solve a cardinality constrained linear program applying the generalized cardinality function to the reduced vector of corrections in R^n, i.e. to y-z. The value of f(y-z) does not exceed the value of f(c). Equality occurs if and only if c is in reduced form and the global minimum is attained on such correction vectors. By Proposition 1, the algorithm as described in Chapter 3 is more efficient when f(c) is used, as more columns can be purged.

2.2.4 The solution

As mentioned in the previous sections, we are looking for min f(y ; z) subject to the constraints imposed by the system (2.4). As explained in 2.2.2 the complementary condition can be removed from the constraints of (2.3). The matrix A_1 is constructed using the matrix of coefficients A and b_1 is constructed using b and the uncorrected values of x. More precisely,

$$(2.5) \qquad A_1 = \begin{bmatrix} A & -A \\ -I^n & I^n \end{bmatrix} \quad b_1 = \begin{bmatrix} b - Ax \\ x \end{bmatrix}$$

In (2.5), I^n is the nxn identity matrix. The matrix A_1 has 2n columns and m + n rows. The column vector b_1 has m + n rows.

Chernikova's algorithm is applied to the system:

$$A_1 \begin{bmatrix} y \\ z \end{bmatrix} \leq b_1$$

$$\begin{bmatrix} y \\ z \end{bmatrix} \geq 0.$$

For simplicity, from here on we will use the term "cardinality" for "generalized cardinality". When the first extremal point is found, the cardinality of each of the points of the lower matrix of Chernikova's tableau is calculated (see 1.1) and compared to the cardinality of the extremal point. All the columns containing points with cardinality greater than the cardinality of the extremal point are deleted. These columns will never generate, upon subsequent applications of Chernikova's algorithm an extremal point of a lower cardinality then the

one just found. The cardinality of the first extremal point becomes thus the first constraint on the cardinality of all extremal points to be generated subsequently. This way, the growth of the tableau is curtailed and all solutions to the problem of finding the extremal point of minimum cardinality are still produced. Then, the next extremal point is found, and its cardinality is calculated. Now the minimum of the first constraint and the new cardinality becomes the new constraint etc. The process continues until all rows of the upper matrix in Chernikova's tableau are processed. Upon completion, the lower matrix contains all extremal points of minimum cardinality.

2.3 Matching fields

As mentioned in 2.0, the concept of matching variables is related to the concept of a suitable donor. The use of matching variables increases the chance of finding a suitable donor. If variables that are not matching are used in the search for a suitable donor for a recipient record or if some of the matching variables are omitted in this search, the selected donor may not be a suitable donor and the imputed record may not pass the edits.

It has to be noted that, for quantitative data, one might not find a suitable donor even if the search is done using the matching variables alone. If the matching variables of the recipient record and the potential donor coincide, then the potential donor is a suitable donor.

Consider a recipient record $x = (x_1, \ldots, x_n)$ for which the fields to be imputed have been appropriately identified. Let us assume that i fields, $0 < i < n$, were identified for imputation by the error localization procedure described in 2.2. The remaining $n - i$ reported fields are potentially matching fields. We denote by x^I the vector whose components are the fields to be imputed and by x^R the vector formed with the remaining $n - i$ components. The values of the components of x^R are the components of a known vector x^0. Let us consider the original system of edits (1.1) with the additional information about the values of x^R for the recipient record, that is:

$$
\begin{aligned}
Ax &\leq b \\
x &\geq 0 \qquad (2.6) \\
x^R &= x^0
\end{aligned}
$$

Since x is a concatenation of x^I and x^R, the system (2.6) has x^I as a vector of unknowns with i coordinates. For this reason we call (2.6) the reduced system associated with the recipient record x.

The reduced system can be written in the standard form with all known values on the right hand side and the unknowns on the left hand side. This leads to the system (2.7):

$$A^I x^I \leq b^I \qquad (2.7a)$$

$$x^I \geq 0 \qquad (2.7b)$$

The reduced system defines a polyhedron in R^i which is embedded in the original polyhedron defined by the system of edits (1.1) in R^n. In (2.7), the components of x^0 are used to calculate the entries of the column matrix b^I. Each inequality in (2.7a) represents a "reduced" edit. There are at most m such inequalities because some of the inequalities of the system (2.6) may have been eliminated had they contained none of the variables to be imputed. Thus, we consider in (2.7) only the "active" edits as defined by P. Giles in [11].

It follows from the systems (1.1), (2.6) and (2.7) that a recipient record will pass the postimputation edits (system (1.1)) if and only if the components imputed for x^I satisfy the system (2.7). Any potential donor (i.e. in the acceptance region of (1.1)) is therefore a suitable donor (the recipient record passes the post imputation edits) if its corresponding components satisfy (2.7).

Definition The matching variables for the recipient record x are those components of x^R that are effectively involved in defining the acceptance region of the system (2.7).

We show now that, if the distance calculated on the matching variables alone between a recipient record and a potential donor is zero, then the potential donor is a suitable donor. Let us write a potential donor as $d = (d^I; x_0^M; d^{NM})$,

where x_0^M is the vector of matching components. Since d is a potential donor and the components of x_0^M alone determine the acceptance region of the reduced system (2.7), it follows that d^I is in the acceptance region of the reduced system. Then the partially imputed record $(d^I;\ x_0^M;\ x_0^{NM})$ belongs to the acceptance region of the original system (1.1), so d is, in fact, a suitable donor.

This reasoning breaks down if nonmatching variables are used. A zero distance between a potential donor d and the recipient, calculated on the nonmatching variables alone does not guarantee that d^I satisfies (2.7). In fact, d^I could belong to a larger acceptance region. If all variables x^R are indiscriminately used in matching, the nonmatching variables may prevail in the calculation of the distance and donors which are not suitable might be chosen.

It may happen that all the components of x^R are required for matching or none of them is. Each of these situations (as well as any intermediate situation) is a consequence of the overall shape and position with respect to the coordinate axes of the larger polyhedron determined by the original system of edits.

There are various ways of establishing the functional dependency of the acceptance region of the system (2.7) on (some) of the coordinates of x^R. One way is to "fit" the "reduced" polyhedron into the original, larger polyhedron determined by the system (1.1). Theoretically, this amounts to considering the systems (1.1) and (2.7) together. Then the coordinates of x^0 which prevent a "perfect fit" are those on which the acceptance region effectively depends. These coordinates will appear at the vertices of the "fitted" polyhedron. The main disadvantage of this method if Chernikova's algorithm is used is that there is a real possibility that the storage space will be exceeded before the algorithm terminates. The vertices which are not produced could contain matching variables which do not appear in other vertices and would therefore be lost.

A simpler way, used by NEIS and GEIS, consists of deriving the information needed for determining the matching variables from the reduced system alone. The redundant edits should be removed from the (2.7) form of the system. The coordinates of x^0 which are used in calculating the b^Is in the remaining constraints of the reduced system correspond to the matching fields. Notice that, according to this procedure, a matching field may or may not appear in a

redundant edit but it has to appear in a nonredundant edit belonging to group 2 or 3 below. On the other hand, a nonmatching field appears only in redundant edits (group 1).

Whatever procedure for removing redundancies is used, one should keep track of the coordates of x^0 involved in the nonredundant edits.

Any hyperplane corresponding to an inequality of the system (2.7), which is assumed consistent, belongs to only one of the three groups defined below:

1) The redundant hyperplanes lie entirely outside the acceptance region.
2) The nonredundant unrestrictive hyperplanes do not lie entirely outside the acceptance region but their removal would not change this region.
3) The nonredundant restrictive hyperplanes "tightly" define the acceptance region. If any such hyperplane is removed, the acceptance region will change.

It should be noted that equality edits are nonredundant, according to this classification.

It is clear that the nonredundant hyperplanes contain all the variables which determine how the reduced polyhedron fits into the original one. It is not clear, just by examining the reduced system, that the nonredundant unrestrictive hyperplanes could be safely deleted without loss of matching variable. For this reason, we retain all nonredundant hyperplanes (see [12]). The disadvantage of this method is that one may use more variables for matching than are really necessary.

Example 1: The original system of edits is:

$$x_i \leq 0, \quad i=1, \ldots 7.$$

$$10^{-5} x_1 - x_2 \leq 0$$

$$- x_1 + 10^{-5} x_3 \leq 0$$

$$-x_4 + x_5 + x_6 \leq 0$$

$$x_2 - x_4 + x_7 \leq 0$$

For a particular record, the values of x_2 and x_4 must be imputed. Therefore $x^I=(x_2, x_4)$ and $x^R=(x_1,x_3,x_5,x_6,x_7)$. The reported values are: $x_1=200,000$, $x_3=10^5$, $x_5=0.1$, $x_6=0.2$, $x_7=0.5$. The components of x^R are all potentially matching variables. Upon substitution in the original system, we obtain the standard form of the reduced system:

		Components of x^R involved:
$-x_i \leq 0$	$i=2,4$	none
$-x_2 \leq -2$		x_1
$-x_2 \leq -1$		x_3
$-x_4 \leq -0.3$		x_5, x_6
$x_2 - x_4 \leq -0.5$		x_7

Just by looking at the system, we can see that the third line above corresponds to a redundant inequality. Indeed, $-x_2 \leq -2$ is a more restrictive inequality than $-x_2 \leq -1$. Since x_3 only appears in the latter inequality, it follows that x_3 is <u>not</u> a matching variable. From the last inequality it follows that $-x_4 \leq -0.5 - x_2$ and because $x_2 \geq 0$, $-x_4 \leq -0.5$. This is a more restrictive inequality than $-x_4 \leq -0.3$, obtained for $x_5 = 0.1$, $x_6 = 0.2$. Since x_5, x_6 only appear in this inequality, they, too, are <u>not</u> matching. The matching variables in this example are x_1 and x_7.

If the system (2.7) is of small dimension (the number i of fields requiring imputation is small), Chernikova's algorithm could be applied to find the redundant inequalities in (2.7). As remarked in [4], the redundant rows can be read off the last matrix used in determining the extremal points of the reduced system. A row corresponding to an inequality with no zero entries corresponds to a redundant edit. In Example 1 of 1.1, the second inequality in 1.3 a) is redundant, as apparent in the tableau Y_3. If only one zero appears on such a row than the edit defines a hyperplane which contains only one of the vertices defining the acceptance region (see section 1.1).

As mentioned before, for the purpose of defining the matching variables all nonredundant edits would be retained. A method for removing redundancies was described in [13]. It is based on the use of the simplex algorithm. It is this

method that is recommended for use in GEIS. The advantage here is that no assumption needs to be made on the number of fields to be imputed.

3. Implementation

3.0 Storage considerations

The main problem encountered in implementing modules which require the use of Chernikova's algorithm is the extensive use of storage space. The final matrix which contains the extremal points has a determined number of columns. The intermediate matrices, however, may have a larger number of columns and the space required for their storage may exceed the available space. When a row corresponding to an inequality is processed, as many as $p + z + pq$ columns can be created in the new tableau, where p, z, q represent the positive, zero, respectively negative entries on that row. When equalities are processed, the maximum number of columns of the new tableau is $z + pq$. The minimum number of columns, corresponding to an inequality is $p + z$; to an equality it is z. Thus, processing some rows may lead to a decrease of the number of columns from the previous tableau. It follows that a judicious choice of the order of processing the rows can curtail the growth of the successive tableaux. Ideally, if the matrix containing the extremal points has more columns than the original matrix, than the number of columns of the intermediate tableaux should not exceed the final value. Since the number of columns of the final tableau is not known in advance, it is difficult to devise a strategy which minimizes the number of columns of the intermediate tableaux. We adopted a strategy which, at each step, controls the growth of the newly created tableau. More precisely, the selection of the row to be processed is done according to the following criterion. Let $m = z + pq$ be the maximum number of columns of the newly created matrix when the row being processed is an equality. This number is, for an inequality, $m = z + p + pq$. These values are calculated after each iteration (a selected row has been processed) for the rows to be processed. Then the minimum value of m is calculated, over all rows described above. One of the rows that has a minimum value for m is then processed. This strategy is used in some modules that make use of Chernikova's algorithm and it is based on remarks found in [17] and [18]. Although this strategy does not ensure an optimum overall order of processing the rows, it has proved quite effective, in several tests, not only in preventing the intermediate tableaux from exceeding the available space but also in considerably reducing the execution time.

In spite of the use of the strategy described above, we felt that the possibility of exceeding the available space still existed. This is true especially of the module which determines the extremal points (see [8]). A strategy was devised which, when space limitations are about to be exceeded, saves some of the extremal vectors providing thus a solution, albeit incomplete, to the problem. The vertices of higher cardinality (or generalized cardinality) are sacrificed. The user is advised that vertices with cardinality larger than a certain value have not been produced. The justification for the claim that all vertices of cardinality smaller than the specified value are produced can be found in the works of Rubin and G. Sande (see [17], [18]). Essentially, we are using the fact that the cardinality of the generated columns cannot decrease as the algorithm proceeds.

3.1 The extremal points

The input for this module is the matrix of edits as described in step 1 below. As mentioned in sections 2.0 and 3.0, Chernikova's algorithm is used to find the extremal points of the acceptance region associated with the system (1.1). The output is the set of extremal points (or a subset of it, if space limitations have been exceeded). It can be read off the columns of the final matrix (see step 8 below). A complete description of the algorithm can be found in [17] or in the Appendix. We give here the main steps of the algorithm.

1. The input file of edits is of the form:

$$Ax \leq b$$

where A is an mxn matrix of coefficients, x is an nx1 column of variables and b is an mx1 column of constraints. That is, m is the number of edits and n is the number of fields which appear in this group of edits. The edits are assumed to be in the form produced by the module ED-ANAL(1) (see [13]).

2. A matrix Y contains the matrix -A followed by the column matrix b. The lower matrix contains, at the onset, the identity matrix in R^{n+1}. Upon termination of the algorithm, it will contain the extremal points.

3. At the onset of the algorithm, an integer value c_{max} may be specified. This represents the maximum cardinality (number of nonzero components) of the extremal points to be produced by the algorithm. The initial value is specified by the user. A value greater than or equal to n corresponds to the case when all extremal points are required. A value for c_{max} strictly smaller than n corresponds to the situation when only extremal points of cardinality smaller than or equal to c_{max} could be generated. The value of c_{max} can be updated. This happens if, while processing a row, the required number of columns of the matrix Y being created exceeds the space limitations.

4. The row of the matrix Y are processed, one at a time. The efficiency of the algorithm increases if the choice of the row to be processed (the pivotal row) is done according to the following rule:

   Consider a particular row and let p represent the number of positive entries, q the number of negative entries and z the number of zero entries on that row. If the row corresponds to an equality edit, define $m = z + pq$. Otherwise define $m = z + p + pq$. The value of m is calculated at the onset of the algorithm as well as after each iteration, for all rows that need processing. That is, the value of m is calculated for rows corresponding to inequalities which contain at least one negative entry and for rows corresponding to equalities which contain at least one nonzero entry. The minimum value of m is calculated over all rows described below. We denote this value by m_0. The new pivotal row is selected among the rows for which $m = m_0$. Notice that, according to this rule, rows corresponding to equalities tend to be processed earlier.

5. At the end of each iteration, all columns for which the last entry is nonzero are scaled so that this last entry becomes 1. These columns will contain the extremal points defining the acceptance region associated with the system of edits (1.1). The components of these vertices can be found on the first n rows of the lower matrix L used in the algorithm. The last entry on columns corresponding to vertices is then 1. If $c_{max} < n$, the cardinality of every vector in the lower matrix is calculated. This cardinality is at most equal to n, since the last entry is not used in counting the number of nonzero components of the vectors. All columns containing vectors with cardinality

greater than c_{max} are deleted. Since the cardinality of the vectors increases as the algorithm proceeds, these columns cound not produce vertices of cardinality lower than c_{max}.

6. When processing a row, a new matrix Y is created. If, while creating this matrix, the number of columns is about to exceed the space limitations, the value of c_{max} is updated for the purpose of purging some of the columns of the newly created matrix while retaining all columns that may produce extremal points of cardinality smaller than or equal to c_{max}. This is done as follows:

   The previously created matrix Y is recalled. The cardinality of each of the vectors with the last coordinate equal to 1 is calculated. The maximum cardinality is taken over all such vectors and is denoted c^*. Then $c_{max} = c^* - 1$. Note that the partially created matrix is not used and that the algorithm can return to this step repeatedly without creating a new matrix. However, since c^* is an integer, the process will terminate eventually. A message will be printed to the effect that no extremal points of cardinality larger than or equal to c^* will be produced.

7. The algorithm terminates unsuccessfully if, at any stage, a row containing only nonpositive elements is found. In this situation, no solution to the original system of edits exists, when at least one component of the vector b is non-zero. When all components of b are zero, then the only solution is the zero solution. In either case, no extremal point is generated by the algorithm and a message to that effect is printed. This situation should not occur after the application of the module ED-ANAL(1).

8. The algorithm terminates successfully when all the rows have been processed and the situation described in step 7 above has not occurred. The extremal points are read off the columns of the final matrix that have the last entry equal to 1. Even when the algorithm terminates successfully, there may be no extremal points of the specified cardinality, if $c_{max} < n$.

3.2 The implied edits

In this module, the input is the transpose of the matrix of coefficients A along with a submatrix corresponding to equalities in the original system of edits. Part of Chernikova's algorithm is used in producing the implied edits associated with the system (1.1). The output is a set of implied edits. A complete description of the algorithm can be found in [11]. We give here the main steps of the algorithm.

1. The input file is of the form:

$$Ax \leq b$$

where A is an mxn matrix of coefficients, x is an nx1 column of variables and b is an mx1 column of constants. That is, m is the number of edits and n is the number of fields which appear in this group of edits. The edits are assumed to be in the form produced by the module ED-ANAL(1) (see [13]). Each row of the matrix A corresponds to an equality or an inequality and should be labeled as such. In this module, equalities are treated as double inequalities and so a submatrix E of A containing all rows that correspond to equalities is created. Let e be the number of equalities corresponding to the system (1.1). Notice that $0 \leq e \leq m$.

2. A matrix Y is created to be used in the algorithm. In this modification of Chernikova's algorithm, no column is deleted as the algorithm progresses, unless it corresponds to an existing edit (original or previously accepted as implied). The upper part of the matrix is formed by adjoining to the transpose of the matrix A the negative of the identity matrix in R^n followed by the negative of the transpose of the matrix E. The lower part is formed with the identity matrix in R^{m+n+e} in the first m + m + e rows, followed by a row containing the coefficients b and then by an indicator row. There are thus m + n + e columns in the matrix Y and 2n + m + e + 2 rows. The second last row of the matrix Y contains the transpose of the vector of coefficients b followed by a string of n zeroes and then by the negative of the transpose of b corresponding to equalities in the original system.

Notice that in this set-up the original edits are placed columnwise and the use of the Chernikova's algorithm amounts to combining edits in which at least one of the variables has opposite signs. The identity matrix which appears in the upper part of the matrix Y as well as the string of zeroes in the lower part represent the "positivity" edits; that is, the edits which require that all variables be positive. Edits corresponding to equalities are represented as double inequalities; they are represented in the matrix A once and then by the matrix E with an opposite sign. The last row contains, at the onset, indicators for the original edits. A zero entry in this row indicates that the column above represents an equality edit whereas a nonzero entry corresponds to an inequalitiy. This holds true throughout the processing of the matrix Y.

3. At the onset of the algorithm, the user may specify c_{max}, the maximum number of new edits that he or she may want to see. The default value for c_{max} should be set high enough to ensure that all implied edits are produced.

4. Chernikova's algorithm is modified so as to produce all implied edits (see section 1.1 as well as 3.1 for comparison). All rows of the upper matrix are processed, one at the time and in their natural order. Neither the original edits nor the accepted new edits are deleted from the matrix Y as the algorithm proceeds. Therefore the number of columns of the original matrix Y cannot decrease. Also, the entries of the final matrix Y need not be nonegative.

5. While processing rows, new columns are being created. Not all are, however, retained. A newly created edit may be accepted only if it effectively eliminates variables. This is accomplished by comparing the number of zero coefficients created on the column representing the new edit and the number of combinations that generated the edit. It is then checked if the created edit is essentially new.

6. Edits representing the same equality with opposite signs are unduplicated. At the end of the algorithm, all original and new edits will have been generated and they are stored in the final matrix Y. If, at any time during processing the number of columns of the intermediate matrices exceeds the available

space or the value of c_{max} is exceeded, the processing stops. A message to this effect is then printed.

3.3 Error localization

The input consists of the system of linear inequalities 1.1, a record x requiring partial imputation and a vector of weights. A positive and negative correction is attached to each field of this record. Rubin's cardinality constrained linear program is applied to an augmented matrix with a generalized cardinality function (see 2.2.4 and step 2 below) in order to minimize the weighted number of corrections. The output is the set of all solutions to this problem with unique patterns (step 8 below). A complete description of the algorithm can be found in [10]. We give here the main steps of the algorithm.

1. The input file is of the form:

   $$Ax \leq b$$

   where A is an mxn matrix of coefficients, x is an nx1 column vector of data and b is an mx1 column of constants. That is, m is the number of edits and n is the number of fields which appear in this group of edits. The edits are assumed to be in the form used by the module ED-ANAL(1) (see [13]).

2. A matrix is created to be used by the algorithm. At the onset, the top part of the upper matrix U contains the negative of the matrix of coefficients, - A, followed by A and then by the column vector b - Ax. The lower part of the matrix U contains the identity matrix in R^n followed by its negative and then by the column vector x. The upper matrix U has then 2n + 1 columns and m + n rows. The lower part L of the matrix Y contains the identity matrix in R^{2n+1}. Thus the work matrix Y has m +3n + 1 rows and 2n + 1 columns at the onset of the algorithm. The number of columns changes as the algorithm proceeds. Upon the completion of the algorithm, the lower matrix L contains the solution to the cardinality constrained linear program. As explained in 2.2.2, all solutions satisfy the complementary condition.

3. At the onset of the algorithm the value of c_{max} may be specified. This represents the maximum generalized cardinality of the extremal points to be generated by the algorithm. Notice that this value need not be an integer, as weights could be used in the definition of the generalized cardinality (see 2.2.3). The value of c_{max} could be updated in the course of finding the extremal points if, in the course of processing one row, the number of columns exceeded the available space.

4. The rows of the matrix Y are processed, one at the time, using Chernikova's algorithm. Although a strategy for curtailing the growth of the matrix is built in the algorithm for solving a cardinality constrained linear program, it is still important to render the algorithm as efficient as possible. For that reason and following suggestions found in [18], the following overall strategy was adopted for the order of processing the rows:

   a) The rows for which the corresponding entry of the vector $b - Ax$ or x is negative are processed first. These rows correspond to edits which the record x has failed (see step 1 for the matrix form of the edits). We call them failed edits.

   b) Within each group of rows (failed edits, edits corresponding to equalities, etc.), the strategy for choosing the row to be processed next as described in step 4 of 3.1 is used.

   In order to use the purge of columns built in this algorithm, it is essential that the algorithm quickly produces an extremal point. When processing failed edits, due to the form of the matrix Y, the column with a nonzero last entry is combined with other columns generating more columns with a nonzero last entry. It is only such columns that eventually contain extremal points (see 1.1.2). Rows corresponding to equalities are also processed early within the group of passed edits (that is edits that are not failed edits). This is justified by the fact that an equality edit is more restrictive than an inequality edit and all extremal points are placed on all the hyperplanes representing equality edits.

5. At the end of each iteration, all columns for which the last entry is nonzero are scaled so that this entry becomes 1. These columns will contain the patterns of corrections which are solutions to the cardinality constrained linear program. They are stored in the lower submatrix of the matrix Y as vectors of corrections (y;z;1), where y stores the n components of positive corrections and z represents the n components of the negative correction. The generalized cardinality of each vector y - z is calculated, for each column of Y. Notice that this generalized cardinality is applied to a vector in R^n and that the last entry is not used in calculating it. It is then checked if an extremal point has been generated. An extremal point corresponds to a vector of corrections for which the last entry is 1 and all entries on the corresponding column are nonegative. Furthermore, there should be zero entries on the rows corresponding to equality edits. If no extremal point is found, all columns with generalized cardinality larger than c_{max} are deleted. At the onset of the algorithm, c_{min} is assigned a "large" value (e.g. $c_{min} = c_{max}$). If an extremal point is found, the value of c_{min} is calculated. The minimum of all generalized cardinalities associated to extremal vectors is found. Then the minimum of this value and c_{max} is calculated and that defines the new value of c_{min}. All columns with cardinality strictly larger than c_{min} are deleted. Columns with cardinality equal to c_{min} are treated as follows:

   a) If they do not correspond to extremal vectors, they are retained. This is so because their corresponding columns may eventually contain extremal points of minimum cardinality.

   b) If they correspond to extremal points, they are retained only if they represent a pattern that has not been retained so far.

6. A new matrix A is created each time a row is processed. If, while creating this matrix, the number of columns is about to exceed the space limitations, the value of c_{max} is updated for the purpose of purging some of the columns of the newly created matrix while retaining all columns that may produce extremal points of generalized cardinality smaller than or equal to c_{max}. This is done as follows: The previously created matrix is recalled. The maximum generalized cardinality is calculated over all columns and is

denoted c^*. All columns with generalized cardinality equal to c^* are deleted. Then $c_{max} = c^* - eps$, where eps is sufficiently small, i.e. less than the smallest weight specified. A message is printed to the effect that only solutions with generalized cardinality less than c^* will be found. Note that the partly created matrix is not used and that the algorithm can return to this step repeatedly without creating a new matrix. However, since c^* can only take a finite number of values, the process will eventually end.

7. The algorithm terminates unsuccessfully if, at any stage, a row containing only negative entries is found. No solution exists or the only possible solution is the zero solution. In either case, the likely cause of termination can be traced to the original system of edits.

8. The algorithm terminates successfully if all rows have been processed and step 7 above has not been taken. The solutions are read off the columns of the final matrix which have the last entry equal to 1. Each solution represents a unique pattern of possible corrections, all requiring a minimum change in the record x in the sense given by the generalized cardinality. Each pattern indicates which fields should be corrected. These fields are represented by nonzero entries in the lower part of the matrix Y corresponding to the solution. Recall that the very last entry does not represent a field; it is used, in fact, to identify the solutions. It has to be pointed out that, even when the algorithm terminates successfully, there may be no solution provided to the user. This happens when, due to space limitations, step 6 above is taken.

Appendix

Chernikova's algorithm as described by Rubin in [17].

The algorithm is as follows:

0.0 If any row of U has all components negative, then $w = 0$ is the only point in C.

0.1 If all the elements of U are nonnegative, then the columns of L are the edges of C, i.e. the ray $(\ell_j) = \{\omega = \lambda \ell_j,\ \lambda \geq 0\}$ is an edge of C; here ℓ_j denotes the jth column of L.

1. Choose the first row of U, say row r, with at least one negative element.

2. Let $R = \{j \mid y_{rj} \geq 0\}$. Let $v = |R|$, i.e., the number of elements of R. Then the first v columns of the new matrix, $\bar{Y}$ are all the y_j for $j \in R$, where y_j denotes the jth column of Y.

2' If Y has only two columns and $y_{r1} y_{r2} <$, adjoin the column $|y_{r2}| y_1 + |y_{r1}| y_2$ to the $\bar{Y}$ matrix. Go to step 4.

3. Let $S = \{(s, t) \mid y_{rs} y_{rt} < 0,\ s < t\}$, i.e., the set of all (unordered) pairs of columns of Y whose elements in row r have opposite signs. Let I_o be the index set of all nonnegative rows of Y. For each $(s, t) \in S$, find all $i \in I_o$ such that $y_{is} = y_{it} = 0$. Call this set $I_1(s,t)$. We now use some of the elements of S to create additional columns for $\bar{Y}$:

   a. If $I_1(s, t) \neq \phi$ (the empty set), then y_s and y_t do not contribute another column to the new matrix.

   b. If $I_1(s, t) = \phi$, check to see if there is a u not equal to either s or t such that $y_{iu} = 0$ for all $i \in I_1(s, t)$. If such a u exists, then y_s and y_t do not contribute another column to the new matrix. If no such u exists, then choose $\alpha_1, \alpha_2 > 0$ to satisfy $\alpha_1 y_{rs} + \alpha_2 y_{rs} = 0$. (One such choice is $\alpha_1 = |y_{rt}|$, $\alpha_1 = |y_{rs}|$.) Adjoin the column $\alpha_1 y_s + \alpha_2 y_t$ to the new matrix.

4. When all pairs in S have been examined, and the additional columns (if any) have been added, we say that row r has been "processed". Now let Y denote the matrix $\bar{Y}$ produced in processing row r and return to step 0.0.

References

1. Burger, E.: Uber homogene lineare Ungleichungssysteme. Z. angew. Math. Mech. 36, 135-159 (1956).
2. Chernikova, N.V.: Algorithm for finding a general formula for the non-negative solutions of a system of linear equations. USSR Computational Mathematics and Mathematical Physics, 4, 151-158 (1964).
3. Chernikova, N.V.: Algorithm for finding a general formula for the non-negative solutions of a system of linear inequalities. USSR Computational Mathematics and Mathematical Physics 5, 228-233 (1965).
4. Dixon, D.: Memoranda to N. Cox. ISDD, Statistics Canada, 1987.
5. Estevao, V.: Donor imputation specifications. RGS, Statistics Canada, September 1988.
6. Fellegi, I.P. and Holt, D.: A systematic approach to edit and imputation. Journal of the American Statistical Association 71, 17-35 (1976).
7. Galperin, A.M.: The general solution of a finite system of linear inequalities. Mathematics of Operations Research 1, 185-196 (1976).
8. GEIS: Specifications for extremal points generation, Module ED-ANAL(2), BSMD, Statistics Canada, December 1987.
9. GEIS: Specifications for donor imputation. BSMD, Statistics Canada, March 1988.
10. GEIS: Specifications for error localization, Module ERROR_LOC(1). BSMD, Statistics Canada, updated April 1988.
11. GEIS: Specifications for implied edits, Module ED-ANAL(3). BSMD, Statistics Canada, updated April 1988.
12. GEIS: Specifications for matching fields, Module MATCH_FIELDS. BSMD, Statistics Canada, May 1988.
13. Giles, P.: Module specifications - GEIS, Module ED_ANAL(1). BSMD, Statistics Canada, January 1987.
14. Giles, P: Module specifications - GEIS, Module IMPUTE(1). BSMD, Statistics Canada, January 1987.
15. Giles, P.: Analysis of edits in a generalized edit and imputation system. Statistics Canada technical report, Draft, November 1988.
16. Giles, P.: A model for generalized edit and imputation of survey data. Can. J. Statist., to appear.
17. Rubin, D.S.: Vertex generation and cardinality constrained linear programs. Operations Research 23, 555-565 (1975).
18. Sande, G.: An algorithm for the fields to impute problems of numerical and coded data. Statistics Canada technical report, 1978.
19. Sande, G.: Numerical edit and imputation. Proceedings of the 42nd Session of the International Statistical Institute, Manila, Phillipines, 1979.