

To the Graduate Council:

I am submitting herewith a dissertation written by Anand S. Kunnathur entitled "Optimal Imputation of Erroneous Data." I have examined the final copy of this dissertation for form and content and recommend that it be accepted in partial fulfillment of the requirements for the degree of Doctor of Philosophy, with a major in Management Science.

Robert S. Garfinkel
Robert S. Garfinkel, Major Professor

We have read this dissertation
and recommend its acceptance:

Kenneth C. Bellert
John W. Philpot
Robert B. Woodruff, Marketing

Accepted for the Council:

L. Evans Bell
Vice Chancellor
Graduate Studies and Research

OPTIMAL IMPUTATION OF ERRONEOUS DATA

A Dissertation

Presented for the

Doctor of Philosophy

Degree

The University of Tennessee, Knoxville

Anand S. Kunnathur

December 1982

3065044

Dedicated to my parents

ACKNOWLEDGEMENTS

I will forever be in the debt of Professor Robert S. Garfinkel, chairman of my Dissertation Committee, for his constant encouragement, great cooperation, wise counsel, understanding and acts of kindness too numerous to specify. He has generously devoted a substantial portion of his valuable time in the past two years in guiding me through this dissertation. The patience displayed by him in carefully reading through a long succession of drafts of the chapters of this dissertation only serves to increase my debt to him. Every statement in this work carries his impression. My appreciation and gratitude for his help can never be adequately expressed.

I am thankful to the other members of my dissertation committee, Professors Kenneth Gilbert, John Philpot, and Robert Woodruff for their help. I am also thankful to Dr. Gunar E. Liepins for his many acts of kindness and cooperation and for introducing me to the problems in data validation. I wish to thank my friend Dr. P. S. Sundararaghavan for his constant encouragement and support. I also wish to thank Dr. Brian Greenberg for his help.

I am especially grateful to my brother Mr. K. S. L. Narasimhan and to my sister-in-law, Mrs. Ranjini Narasimhan without whose emotional support, encouragement and affection this work could not have been completed. I am thankful to the School of Business Administration, the University of Wisconsin-Milwaukee, for graciously permitting me to use their facilities.

Last but not the least I wish to express my sincere and deep appreciation to Ms. Nancy Taylor for her patience, understanding, and cheerful cooperation in typing successive drafts of this dissertation.

ABSTRACT

The initial theoretical development in the area of Error Localization for purposes of data imputation in validating records which fail specified consistency conditions is due to Fellegi and Holt. Garfinkel and Liepins proposed two different procedures for solving the Minimum Weighted Fields to Impute problem (MWFI) associated with failing data records which contain categorical data only. The formal development of theory leading to these procedures and a comparative computational analysis of these two procedures forms a part of this work. Recognizing that the Minimum Fields to Impute problem with Continuous data records (MWFIC) is a special case of the general fixed charge problem and is closely related to the Cardinality Constraint Linear Programming problem (CCLP) the use of an extreme point enumeration algorithm such as one due Chernikova for solving MWFIC was suggested by Sande. Motivated by Rubin's adaptation of the Chernikova algorithm for solving CCLP's two extreme point (implicit) enumeration algorithms are proposed in this work for solving MWFIC and MFIC and the theory leading to the specification of these algorithms is carefully developed.

A cutting plane algorithm motivated by Garfinkel's algorithm solving for MWFI and similar in spirit to it is proposed for solving MWFIC. The theory leading to the specification of this algorithm and computational experience with it are also presented. For

purposes of solving larger MWFIC problems than can be solved by any of the existing algorithms a heuristic algorithm is proposed. A comparative analysis of the computational performance of the heuristic versus the cutting plane algorithm for solving MWFIC is also presented.

TABLE OF CONTENTS

CHAPTER	PAGE
1. INTRODUCTION	1
2. MINIMUM WEIGHTED FIELDS TO IMPUTE PROBLEM WITH CODED DATA	8
3. ALGORITHMS FOR THE MWFI PROBLEM WITH CODED DATA	37
4. ANALYSIS AND COMPARISON OF THE COMPUTATIONAL PERFORMANCE OF SOME ALGORITHMS FOR SOLVING MWFI	65
5. ERROR LOCALIZATION FOR CONTINUOUS DATA	83
6. A CUTTING PLANE ALGORITHM FOR THE MWFI PROBLEM WITH CONTINUOUS DATA	129
7. A HEURISTIC ALGORITHM FOR SOLVING MWFIC	149
8. EXTENSIONS AND FUTURE RESEARCH	167
LIST OF REFERENCES	172
VITA	176

LIST OF TABLES

TABLE	PAGE
2.1. A Fields that Enter Matrix	14
3.1. Implied Edits Generated Using Algorithm 3.1 in Example 3.4	56
3.2. Step 3 of Algorithm 3.2 in Example 3.5	61
4.1. A Comparison of the Computational Performance of Algorithms 3.1 and 3.2	77
5.1. Initial Tableau in Example 5.2	106
5.2. The Tableau after 1 Iteration in Example 5.2	107
5.3. The Tableau after 2 Iterations in Example 5.2 . . .	108
5.4. Initial Setup in Example 5.3	116
5.5. The Tableau after 1 Iteration in Example 5.3	117
5.6. The Tableau after 2 Iterations in Example 5.3 . . .	118
5.7. The Tableau after 3 Iterations in Example 5.3 . . .	119
5.8. The Tableau after 4 Iterations in Example 5.3 . . .	120
5.9. Final Tableau in Example 5.3	121
5.10. The Initial Tableau in Example 5.4	124
5.11. The Tableau after 1 Iteration in Example 5.4	125
5.12. The Tableau after 2 Iterations in Example 5.4 . . .	125
5.13. The Tableau after 3 Iterations in Example 5.4 . . .	126
5.14. The Final Tableau in Example 5.4	127
6.1. Initial Setup in Example 6.4	140
6.2. Computational Performance of the Cutting Plane Algorithm for MWFIC	147

TABLE	PAGE
7.1. Initial Simplex Tableau in Example 7.1	155
7.2. Simplex Tableau after Application of Step 1 of Heuristic 7.1	156
7.3. Comparison of the Computational Performance of Heuristic 7.1 and Algorithm 6.1	161

LIST OF FIGURES

FIGURE	PAGE
3.1. Example of an FCF	50
3.2. FCF on 6 Fields	55
4.1. The Matrix Q on 4 Rows with the Maximum Number of Undominated Columns	71
4.2. Records Generated in Step 3 of Example 3.5	73
5.1. The Constraint Set of Example 5.1	91
5.2. A Convex Cone	95

CHAPTER 1

INTRODUCTION

The fact that erroneous information is prevalent in large data sets is stressed by Terry [35] and by Pritzker, et al [29]. In these papers and others it is often estimated that the error rate on data gathered from various sources is typically between 1 and 10 percent, though at times the error rate is significantly higher. Thus, the magnitude of incorrect data can be staggering in, for instance, a population and housing census which processes tens or even hundreds of millions of records.

To make the ideas in the next paragraphs more precise, the following notation is introduced. Let y be a record which consists of entries y_j in each of N fields. An edit E is simply a set of possible values for y which causes the record to be suspect. Thus, if $y \in E$ y is said to fail the edit E . In terms of the previous paragraph, the data were deemed erroneous because they failed certain edits.

I. Options for Handling Erroneous Data

Given that a record has failed one or more edits, there remains the question of what to do with it. The options include:

- a) Ask the respondent to revise the record;
- b) Keep the record;
- c) Discard the record;
- d) Revise the record.

The first option is the most desirable if it makes sense economically. For obvious reasons, that is often not the case.

In case the record is only known to be an outlier, option (b) may be reasonable if one also notes the existence of the suspicious record and keeps track of it for some future statistical analysis. In case the record is known to be in error, it is clearly more difficult to justify this option.

Option (c) is attractive on the surface, but two arguments may render it unacceptable. In some cases, individual records are irreplaceable and in other cases, too large a proportion of the data would be discarded. This is often the case, for instance, in the collection of energy data from various countries. Another objection to this option is that discarding of erroneous records may have the effect of biasing the distribution of the data.

If, for any of the reasons given above, options (a) - (c) are unacceptable, then erroneous records must be changed. That is, new values must be imputed to some of the fields. Many different ways to change erroneous records are outlined in Sande [32]. A number of methods impute these new values directly, whereas others first localize the error; that is, they first determine which fields are most likely to be in error, and leave the imputation to another stage. Both methods depend on the assumed underlying error model.

There are a number of factors which affect either error localization or imputation. In this section we will discuss the following: the form of the data, the type of edits, the form of the edits, and the overall objective, postponing consideration of the error model to Chapter 2.

a. Form of the Data and Data Records

Here we distinguish between continuous, discrete, and categorical data. By continuous data is meant data which can theoretically accept any value in some given interval, say $[l,u]$. For instance the kwh of electricity used by an industrial concern in a given month can be considered continuous data.

By discrete data is meant an acceptable response in a field that is any one of a discrete set of entries. Family size would be an example of discrete data. Categorical data are a special case of discrete data. For categorical data, the coded values are merely identification codes and are not to be interpreted in terms of the usual Euclidean distance. Relationship to head of household is one such example.

The nomenclature used here is that a data record is a vector, the coordinates of which are called fields. Thus, a data set is defined to be continuous, discrete, or categorical according to whether the data in each of the fields are continuous, discrete, or categorical, respectively. Data sets which have discrete data in some fields and continuous data in others are called mixed data.

This distinction in the form of the data is important because each data form dictates a distinct theoretical foundation which requires dramatically different techniques to implement the solution. The theoretical differences between the non-categorical and categorical data arise because of the meaningfulness or lack thereof (respectively) of the usual Euclidean norm. For example, in the case of categorical data, it is plausible to postulate that any errors are independent of the apparent magnitude of error. The theoretical differences between discrete

Euclidean data and continuous data are not so profound or apparent. Clearly, the first must be represented by a discrete distribution while the second may be represented by a continuous distribution. The full consequences of this latter difference are not well understood, though it is known that they will at least partially manifest themselves in the error model. In any case, the practical consequences are that the discrete cases lead naturally to optimization techniques like integer programming while the continuous case leads to linear or non-linear programming (as the case may be).

b. Type of the Edits

A distinction is made between two types of edits, namely deterministic and stochastic. Let $\bar{x}$ be the event that y is not a valid record, then the edit E is deterministic if

$$P(\bar{x} | y \in E) = 1$$

and stochastic otherwise. Clearly the different types of edits lead to different decision processes. For instance, if a respondent indicates an age of eight and claims parentage of 23 children, we would all agree that the probability that the record is in error is one. On the other hand, the event "parentage of 23 children" may not be grounds for rejection of the field (or record), since the event has certainly occurred in the past. It would, however, have a low enough probability to be considered an outlier and to warrant the record being "flagged" for further investigation. Throughout this work edits will be assumed to be deterministic.

c. Form of the Edits

With respect to the form of the edits, two types of edits are considered: algebraic and general edits. An algebraic edit, usually

associated with continuous or with non-categorical discrete data, is of the form

$$y \in E \Leftrightarrow f(y) \neq 0 \quad (1.1)$$

where f is some algebraic function. For instance if the first $N-1$ fields correspond to coal shipments to various regions, and if the N th field is total coal shipment, an edit would be

$$y \in E \Leftrightarrow \sum_{j=1}^{N-1} y_j - y_N \neq 0 \quad (1.2)$$

By a general edit, usually associated with categorical data is meant an edit which cannot (naturally) be put into the form (1.1). ("Naturally" because it is always possible, with some manipulation and possibly expansion of the number of fields to make any edit algebraic.) It often happens, however, that this conversion renders the problem very cumbersome to solve. An example of a general edit might be:

$$\text{Age} \leq 12 \Rightarrow \text{Marital Status} = \text{Single}.$$

Within the class of algebraic edits we may wish to distinguish further between linear and nonlinear edits. A linear edit (an example is (1.2)) is of the form

$$y \in E \Leftrightarrow \sum_{j=1}^N a_j y_j - b \neq 0 \quad (1.3)$$

where the a_j 's and b are given. An algebraic edit is nonlinear if f in (1.1) is nonlinear. In this work only linear algebraic edits are considered when dealing with continuous data.

The objective of error localization, in the most broad based of terms, is to reproduce the true data (from the erroneous record) as

closely as possible at an affordable cost. More in-depth consideration of the objective is postponed until Chapter 2. The objective in the present work is to identify, given a failing record, a minimal number (a set of fields with minimum associated cost) of fields that can be changed to make the record pass all edits specified. This problem referred to as MWFI for categorical data is in the class of NP-Hard problems, see Garey and Johnson [11] for a definition of the term. This follows from the fact that the "satisfiability" problem, see [12], is a special case of MWFI. The MWFI problem for continuous data is a special case of the fixed charge problem, see [26], and is also a computationally difficult problem to solve. The computational performance of algorithms documented in Chapters 4, 6 and 7 must be evaluated from this perspective.

This work was motivated by the pioneering work of Fellgi and Holt [9], Garfinkel [12], and Liepins [21] in the area of error localization. Chapters 2 and 3 are partly expository documenting in detail the working of the algorithms proposed in Garfinkel [12] and in Liepins [21]. Chapter 4 contains a computational analysis of the performance of the algorithms proposed in [12] and [21]. The Chernikova algorithm for extreme point enumeration of polyhedral sets as a means of solving error localization problems with continuous data was first proposed in Sande [32]. The theory leading to the Chernikova Algorithm and to the specification of the VS-1 and VS-2 algorithms is presented in Chapter 5. Motivated by Garfinkel's row generation procedure for error localization with categorical data [12] an algorithm, Algorithm 6.1, for solving the continuous data problem is presented in Chapter 6. Computational

analysis of Algorithm 6.1 indicated that the algorithm is unlikely to perform well on large problems, motivating the development of Heuristic 7.1 described in Chapter 7. Heuristic 7.1 was found to perform well on test problems both with respect to execution time and accuracy of solutions. Areas of future research interest are presented in Chapter 8.

CHAPTER 2

MINIMUM WEIGHTED FIELDS TO IMPUTE PROBLEM WITH CODED DATA

Data are generated in surveys, say, in basically one of the following forms: continuous, discrete, coded or nominal. Variables such as height, weight, etc. are naturally expressible in continuous form. Population figures, number of units of a particular item in inventory, etc. are expressible in discrete form. Data such as marital status, sex, etc. can be numerically coded; e.g. a married person could be coded as being of type 1, a divorced person of type 2, etc.

The problems of data imputation in the continuous and discrete cases will be addressed in later chapters. The current chapter deals with coded data only, although some of the definitions and results are valid for these other forms of data as well.

A [data] record $y = (y_1, y_2, \dots, y_N)$ is an N -vector, where the component y_i , $i=1, 2, \dots, N$ is a specific value of data item (field) i . For example, with $N=2$, the record $(3, 1)$ could correspond to the specification ('married', 'male') where the code 3 in field 1 is a code for marital status and the code 1 in field 2 indicates the sex of the person.

Let N be the number of fields in each record. Let A_i be the set of possible code values which may be recorded in field i and let n_i be the finite number of such values possible in field i . The

data space D is then defined as the Cartesian product of possible values in fields $1, 2, \dots, N$,

$$D = A_1 \times A_2 \times \dots \times A_N .$$

Example 2.1 With $N=2$, field 1 corresponding to age and field 2 corresponding to marital status we could have:

$A_1 = \{1, 2\}$, where code 1 corresponds to an age of less than 18 years and code 2 to an age of 18 years or older.

$A_2 = \{1, 2, 3\}$, where code 1 corresponds to the marital status 'single,' code 2 to the status 'married,' and code 3 to 'divorced.'

The data space, D , is then the set of points in 2-dimensional space characterized by the coordinates (value in field 1,

value in field 2): $(1, 1)$, $(1, 2)$, $(1, 3)$, $(2, 1)$, $(2, 2)$ and $(2, 3)$.

Note that only the coordinates $(1, 1)$, $(2, 1)$, $(2, 2)$ and $(2, 3)$ are valid combinations, under the assumption that a person has to be at least 18 years old to be legally married. Note also that the coordinates mentioned are simply possible records.

The records $(1, 2)$ and $(1, 3)$ are not acceptable because of the assumption that the legal age for marriage is 18 years or older. This leads us to the definition of an edit.

An edit E is a specification of a set of possible "failing" values for a record y . More precisely an edit E is a subset of the data space, D , so that if a record $y \in E$ then y fails E .

Example 2.2 Using the situation in Example 2.1, an edit specification E could be as follows:

$E \equiv$ if age < 18 years, marital status = 'married' or 'divorced,'
the record is failing.

A record $y = (1,2)$ is then a failing record; i.e. $y \in E$. It should be noted that a record $y = (y_1, y_2, \dots, y_N)$ contains exactly one specification of a value for each field i , while an edit E may have more than one value specified for each field, as in Example 2.2.

If x denotes the event that a record, y , fails, then E is a deterministic (stochastic) edit if $P(x|y \in E) = 1$ ($P(x|y \in E) < 1$); where $P(\alpha|\beta)$ denotes (conditional) probability. To illustrate a stochastic edit, if a respondent indicates, in a survey, an age of 15 and authorship of 10 books in physics, the record need not necessarily be in error, although it very probably is. In this work, we shall assume that all edits are deterministic.

Example 2.3 Consider the situation in Example 2.1. Let the edit specifications be the following:

E_1 : (1,2) i.e. if age < 18 and the person indicates 'married,'
the record fails.

E_2 : (1,3) i.e. if age < 18 and the person indicates 'divorced,'
the record fails.

An edit system $\bar{E}$ can be constructed from E_1 and E_2 by noting that $y \in E_1$ or E_2 means y fails; i.e.

$$\bar{E} = E_1 \cup E_2$$

specifies the edit system E such that $y \in \bar{E} \Leftrightarrow y \in E_1$ or $y \in E_2$.

It has been stated that an edit is merely a specification of a subset of the data space D . Thus an edit should be expressible as a Cartesian product of the subsets of entries in each of the fields. Before formalizing this notion let us look at an example.

Example 2.4 Suppose, in a demographic survey, the edit specification $\bar{E}$ is: "If a person's age is ≤ 18 years or he(she) is an elementary school student, then his(her) marital status should be single and number of children should be zero."

The fields for which information is sought are: 1. Age
2. Education 3. Marital Status 4. Number of Children 5. Sex.
Let A_i denote the set of allowable code values in field i . Here,
 $A_1 = \{1,2\}$, $A_2 = \{1,2\}$, $A_3 = \{1,2,3\}$, $A_4 = \{1,2,3\}$, $A_5 = \{1,2\}$
where age ≤ 18 years is coded as 1 and age > 18 as 2.

A student at elementary school level and below is coded 1 and education higher than that coded 2. The marital status codes are:
1 $\rightarrow$ never married, 2 $\rightarrow$ married, 3 $\rightarrow$ divorced. The number of children codes are: 1 $\rightarrow$ no children, 2 $\rightarrow$ one to three children, 3 $\rightarrow$ more than three children. The sex codes are: 1 $\rightarrow$ female, 2 $\rightarrow$ male.

The data space is:

$$D = \{1,2\} \times \{1,2\} \times \{1,2,3\} \times \{0,1,2\} \times \{1,2\} .$$

The edit specification is a subset of D . It may be easier to identify this subset by decomposing the edit specification as follows:

$\bar{E} : [\text{Age} \leq 18 \text{ or } (\text{elementary school})] \text{ and not } [(\text{no children}) \text{ or } (\text{single})] \equiv F$

where F denotes that $y \in \bar{E}$ is a failing record.

$\bar{E}$ can be broken down into

$E_x : [\text{Age} \leq 18 \text{ or } (\text{elementary school})] \text{ and not } [\text{single}]$

$E_y : [\text{Age} \leq 18 \text{ or } (\text{elementary school})] \text{ and } [\text{children}]$

Clearly $\bar{E} = E_x \cup E_y$. E_x and E_y can be further broken down into

$E_1 : [\text{Age} \leq 18] \text{ and } [\text{not single}]$

$E_2 : [\text{Age} \leq 18] \text{ and } [\text{number of children} > 0]$

$E_3 : [\text{elementary school}] \text{ and } [\text{not single}]$

$E_4 : [\text{elementary school}] \text{ and } [\text{number of children} > 0]$

and $\bar{E} = E_x \cup E_y = E_1 \cup E_2 \cup E_3 \cup E_4$. (2.1)

It is easier to see the correspondence between E_i , $i=1,2,3,4$, and points in D than the correspondence between $\bar{E}$ and the same points in D . In fact, E_1 , E_2 , E_3 , and E_4 can be written in Cartesian product form as follows:

$$E_1 = \{1\} \times A_2 \times \{2,3\} \times A_4 \times A_5 \quad (2.2)$$

$$E_2 = \{1\} \times A_2 \times A_3 \times \{2,3\} \times A_5 \quad (2.3)$$

$$E_3 = A_1 \times \{1\} \times \{2,3\} \times A_4 \times A_5 \quad (2.4)$$

$$E_4 = A_1 \times \{1\} \times A_3 \times \{2,3\} \times A_5 \quad (2.5)$$

where, for instance, (2.2) states that no matter what the entries are in fields 2, 4, and 5, if field 1 has code 1 and field 3 has codes 2 or 3 in a record, then the record is invalid.

Formally, given any compound statement of an edit, the decomposition process of Example 2.4 can be performed to restate the

edit as a union of subsets of the data space such as in (2.1) to (2.5). Given sets A and B , $A \subset B$ indicates that A is a proper subset of B while $A \subseteq B$ indicates A to be a subset of B . In general, an edit E_p may be expressed as:

$$E_p = A_{p1} \times A_{p2} \times \dots \times A_{pN} \quad (2.6)$$

where $A_{pi} \subseteq A_i$, $i=1,2,\dots,N$.

Edits in the form (2.6) are said to be in normal form [9]. Edits will be assumed to be in normal form for the rest of this chapter. Note that edits in normal form must have at least one (permissible) entry in each field.

In (2.6), if $A_{pk} \subset A_k$, $k \in I_N \equiv \{1,2,\dots,N\}$, then field k is said to enter E_p . Thus, in Example 2.4, fields 2 and 3 enter E_3 . Further, in (2.6), if $A_{pt} = A_t$, $t \in I_N$, then the field t is said to be uninvolved in E_p . In Example 2.4, fields 1, 4 and 5 are uninvolved in edit E_3 .

I. The Complete Set of Edits

For a given data validation situation let $\bar{E}$ be the set of r edits so that,

$$\bar{E} = \bigcup_{j \in J_r} E_j, \quad J_r = \{1,2,\dots,r\}. \quad (2.7)$$

A record y fails if and only if $y \in \bar{E}$. Under the assumption that edits are deterministic, if $y \in \bar{E}$, then y needs to be corrected. Note that for y to belong to $\bar{E}$, it suffices for y to fail any one of the E_j , $j \in J_r$, while for $y \notin \bar{E}$ the record must pass all E_j , $j \in J_r$.

If $y \in \bar{E}$, it is a simple matter to identify $j_r \subseteq J_r$ such that $y \in E_j$, $j \in J_r$ and $y \notin E_j$, $j \notin J_r$. However, it is not as easy to identify the fields where entries can be changed to remove the edit failures.

The problem of identification of such fields is known as the "Fields to Impute" problem. If there is a passing record at all then, clearly, any failing record, y , can be made to pass all edits by changing (if necessary) all fields of y . Our goal, however, is to change the data as little as possible while producing a passing record. It is intuitively clear that at least one entering field for each edit failed by the record must be changed. The problem of identification of a 'smallest' set of fields with the property that it contains at least one entering field for each failed edit is a Set Covering Problem (SCP). See Garfinkel [13] for a discussion of SCP's. The following definitions are needed for an SCP formulation of the Fields to Impute problem. Consider the binary matrix of Table 2.1 whose columns are fields and whose rows correspond to edits.

Table 2.1
A Fields that Enter Matrix

Edit/Fields	1	2	3		N
E_1	1	1	1		1
E_2	0	1	1		0
$\vdots$					
E_M	1	1	0		1

$P \equiv$

The matrix in Table 2.1 is defined as follows:

$$P_{ij} = \begin{cases} 1 & \text{if field } j \text{ enters edit } i \\ 0 & \text{otherwise .} \end{cases} \quad (2.8)$$

Matrices with rows corresponding to edits, columns corresponding to fields and entries defined by (2.8) will be known as Fields that Enter Matrices (FEM's).

Let $I_m = \{1, 2, \dots, m\}$ be the index set of edits and $Q = \{Q_j\}$ be a collection of sets such that $Q_j \subseteq I_m$, $j \in I_N$. Then $S \subseteq I_N$ is said to be a cover of Q if $\bigcup_{j \in S} Q_j = I_m$. If, in addition no subset of S is a cover then S is said to be a prime cover.

We are now in a position to formulate an SCP for identification of a set of fields which may be changed in attempting to make a failing record pass all edits.

If A is the FEM corresponding to the edits $E_i \in \bar{E}$, $i=1, \dots, m$, failed by a record y , and if

$$w_j = \begin{cases} 1 & \text{if field } j \text{ is to be changed} \\ 0 & \text{otherwise} \end{cases}, \quad j \in I_N$$

then the SCP corresponding to the record y is:

$$\begin{aligned} \text{Min} \quad & \sum_{j \in I_N} w_j \\ & \sum_{j \in I_N} a_{ij} w_j \geq 1, \quad i = 1, \dots, m. \end{aligned} \quad (2.9)$$

Example 2.4a demonstrates that the solution to the SCP formulation (2.9) may not always lead to a passing record.

Example 2.4a Consider the edit system

$$E_1 = \{1\} \times \{2, 3\} \times A_3$$

$$E_2 = A_1 \times \{1, 2\} \times \{2\}$$

$$A_1 = A_3 = \{1, 2\}, \quad A_2 = \{1, 2, 3\}$$

A record $y = (1,2,2)$ fails both E_1 and E_2 . The SCP is:

$$\begin{aligned} \text{Min} \quad & w_1 + w_2 + w_3 \\ & w_1 + w_2 \geq 1 \\ & w_2 + w_3 \geq 1 \end{aligned}$$

with a solution of $w_2 = 1$; i.e. change field 2. But there is no way to change field 2 of y so that a passing record results.

The following is a development of theory to use the solution to the SCP formulation (2.9) for solving the Fields to Impute problem.

Definition Let $\bar{E}$ be the explicitly specified system of edits in a given data validation situation and let E_k be a subset of the data space such that:

$$E_k = \bigotimes_{i \in I_N} A_{ki}$$

i.e., $E_k = A_{k1} \times A_{k2} \times \dots \times A_{kN}$. E_k is said to be an implied edit of the edit system $\bar{E}$ if:

- (i) $A_{ki} \neq \phi$, $i = 1, 2, \dots, N$.
- (ii) if $y \in E_k$, then $y \in \bar{E}$.

It will be shown that inclusion of implied edits in the set $\bar{E}$, against which records are tested, will result in the solution of (2.9) solving the Fields to Impute problem.

Example 2.5 Consider the edit system (2.1) of Example 2.4 and the following edits defined on the same data space,

$$E_5 = A_1 \times \phi \times A_3 \times \{2,3\} \times \{1\}$$

$$E_6 = \{1\} \times A_2 \times \{2,3\} \times \{2,3\} \times A_5$$

$$E_7 = \{2\} \times \{2\} \times A_3 \times \{2,3\} \times A_5.$$

E_5 is not a valid edit because E_5 requires that the entry in field 2 be 'null' which is not a permissible entry for field 2. E_6 , on the other hand, is a valid edit, by definition. Further, if $y \in E_6$ then y does fail one or more of the edits in $\bar{E}$. Thus E_6 is a valid implied edit. E_7 is also a valid edit, by definition; i.e. $E_7 \subset D$, the data space. However, E_7 is not implied by the edit system $\bar{E}$ in that a record $y = (2,2,1,2,1)$ which fails E_7 does not fail any edit in $\bar{E}$.

Lemma 2.1 specified a procedure for the generation of implied edits given an initial system of edits.

Lemma 2.1 [9]

If E_k are edits for all $k \in S$, where S is any index set, i.e.

$$E_k = \bigotimes_{j \in I_N} A_{kj}, \quad k \in S$$

then for an arbitrary choice of $i, i \in I_N$, the expression

$$E_{\star} = \bigotimes_{j \in I_N} A_{\star j} \quad (2.10)$$

is an implied edit if,

$$(i) \quad A_{\star j} \neq \phi, \quad \text{for all } j \in I_N$$

$$(ii) \quad A_{\star j} = \bigcap_{r \in S} A_{rj}, \quad j \in I_N; \quad j \neq i \quad (2.11)$$

$$\text{and} \quad A_{\star i} = \bigcup_{r \in S} A_{ri} \quad (2.12)$$

where i is known as the generating field.

Proof: (i) implies that $E_{\star}$ is a valid edit. If $y = \bigotimes_{i \in I_N} y_i \in E_{\star}$, then (2.11) ensures that $y_k \in A_{rk}$, for all $r \in S, k \neq i$. Also (2.12) ensures that at least one $E_r, r \in S$ has $y_i \in A_{ri}$. Thus, $E_{\star}$ is an implied edit.

QED.

The edit E_* of (2.10) will be denoted by $E(i, \{E_k | k \in S\})$. Lemma 2.1 provides a procedure to generate implied edits. In fact, F-H also show that all implied edits, needed to make the SCP formulation valid for solving the Fields to Impute problem, can be generated by this construction. It follows from Lemma 2.1 that if $\bar{E}_I$ is a set of implied edits generated using some subset(s) of $\bar{E}$, the explicitly specified set of edits, then E_p is an implied edit of $\bar{E}$ in the sense that $y \in E_p \Rightarrow y \in \bar{E}$.

Example 2.6 Consider a situation with three data fields and

$A_1 = \{1,2,3,4\}$, $A_2 = \{1,2,3,4\}$, and $A_3 = \{1,2,3,4,5,6\}$. Let E_1, E_2, E_3 be the three explicitly specified edits; i.e.

$$\bar{E} = E_1 \cup E_2 \cup E_3$$

where

$$E_1 = \{1,2\} \times \{1,2,3\} \times \{1,2,5,6\}$$

$$E_2 = \{3,4\} \times \{2,3,4\} \times \{2,3,4\}$$

and $E_3 = \{1,3\} \times \{1,4\} \times A_3$.

Let E_* be the implied edit generated by using field 1 as the generating field on edits E_1 and E_2 .

Then $A_{*1} = \{1,2\} \cup \{3,4\} = A_1$

$$A_{*2} = \{1,2,3\} \quad \{2,3,4\} = \{2,3\}$$

$$A_{*3} = \{1,2,5,6\} \quad \{2,3,4\} = \{2\}$$

So $E_* = A_1 \times \{2,3\} \times \{2\}$.

Consider the record $y' = \{2\} \times \{3\} \times \{6\}$. Note that $y' \in E_1$, but $y' \notin E_*$ even though E_1 participates in the generation of E_* .

However, the record

$$y^2 = \{1\} \times \{3\} \times \{2\}$$

with $y^2 \in E_*$ must fail E_1 and/or E_2 (the edits contributing to the generation of E_*) and we note that y^2 does fail E_1 .

Definition An edit E_p is said to dominate another edit E_q if and only if $E_q \subset E_p$. Thus, for $E_p = \bigotimes_{i \in I_N} A_{pi}$ and $E_q = \bigotimes_{i \in I_N} A_{qi}$, $E_q \subset E_p$ implies that $A_{qi} \subseteq A_{pi}$, for all $i \in I_N$ and $A_{qj} \subset Q_{pj}$ for some $j \in I_N$.

Definition [9] If all the sets A_{ri} in the statement of Lemma 2.1 are such that $A_{ri} \subset A_i$, but $A_{*i} = A_i$, then the edit in (2.12) is called an essentially new implied edit.

If i is the generating field for an implied edit E_* and if E_* is essentially new then field i is uninvolved in E_* . It is clear that if any of the edits, say E_q , involved in the generation of the implied edit E_* already had $A_{qi} = A_i$ in the generating field i , then $E_* \subseteq E_q$ and E_* is not essentially new. E_* of Example 2.6 is an essentially new implied edit. (Note that the set covering constraint corresponding to an essentially new implied edit has a zero in column i .)

Definition [9] The set of explicitly (initially) specified edits together with the set of all essentially new implied edits is said to constitute a complete set of edits and is denoted by Ω .

It will be shown that a complete set of edits is sufficient for solving the Fields to Impute problem (in the sense that the problem is equivalent to (2.9)).

Definition A set of edits $\bar{E}$ is said to be non-redundant if and only if for any $E_p, E_q \in \bar{E}$, $E_p \neq E_q$, E_p does not dominate E_q . It may be noted that the complete set of edits need not be a non-redundant set of edits.

Example 2.7 Let $A_1 = \{1,2\}$, $A_2 = \{1,2\}$, $A_3 = \{1,2,3,4\}$ and the original edits be:

$$E_1 = \{1\} \times \{1,2\} \times \{1,2\}$$

$$E_2 = \{1,2\} \times \{1\} \times \{3,4\}$$

$$E_3 = \{2\} \times \{1\} \times \{1,2\}.$$

The implied edits

$$E_{\star}^1 = E(3, \{E_1, E_2\}) = \{1\} \times \{1\} \times A_3$$

and
$$E_{\star}^2 = E(3, \{E_2, E_3\}) = \{2\} \times \{1\} \times A_3$$

are both essentially new. Similarly, $E(1, \{E_{\star}^1, E_{\star}^2\}) = E_{\star}^3 = A_1 \times \{1\} \times A_3$ is essentially new and dominates E_2 , E_3 , $E_{\star}^1$, and $E_{\star}^2$. The F-H complete set of edits would still have all three of $E_{\star}^1$, $E_{\star}^2$, and $E_{\star}^3$ besides E_2 and E_3 while the non-redundant complete set is $\{E_1, E_{\star}^3\}$.

The following example illustrates the fact that the implied edit generation procedure of Lemma 2.1 does not extend, in general, to simultaneous generation over more than one generating field.

Example 2.8 Let $A_1 = \{1,2\}$, $A_2 = \{1,2\}$, $A_3 = \{1,2,3,4\}$ and let the originally specified edit set be

$$E_1 = \{1,2\} \times \{1\} \times \{3,4\}$$

and
$$E_2 = \{2\} \times \{1\} \times \{1,2\}.$$

Simultaneous generation of fields 1 and 3, using E_1 and E_2 yields

$$E_3 = \{A_1\} \times \{1\} \times A_3.$$

However, E_3 is not an implied edit. Consider, for example, a record $y = \{1\} \times \{1\} \times \{1\}$. Clearly, $y \notin E_1 \cup E_2$, yet $y \in E_3$. A condition

for the validity of generation of implied edits over more than one field is examined in the next chapter.

Let $\Omega_k \subseteq \Omega$ be the set of edits which do not involve fields $k+1, \dots, N$, i.e. for $E_r \in \Omega_k$, $E_r = \bigotimes_{i \in I_N} A_{ri}$, where $A_{rj} = A_j$, $j = k+1, k+2, \dots, N$.

Theorem 2.1 [9]

If a_i^0 , $i = 1, 2, \dots, k-1$, are respectively some possible values for the first $k-1$ fields and if these values satisfy all the edits in Ω_{k-1} , then there exists some value a_k^0 such that a_i^0 , $i = 1, 2, \dots, k$ satisfy all edits in Ω_k .

Proof Assume the contrary. Then there exist a_i^0 , $i=1, 2, \dots, k-1$, such that the edits in Ω_{k-1} are satisfied but which have the property that for every possible value of field k , at least one of the edits in Ω_k is violated.

Identify one such failed edit in Ω_k corresponding to each possible value a_k of field k . There are then $|A_k|$, not necessarily distinct, edits possibly involved. Let R be the number of distinct edits involved and

$$E_r = \bigotimes_{i=1}^N A_{ri}, \quad (r = 1, 2, \dots, R) \quad (2.13)$$

where $a_i \in A_{ri}$, $i = 1, 2, \dots, k-1$
 $a_k \in A_{rk}$.

Note that $A_{rk} \subset A_k$, for otherwise there would be an edit in Ω_{k-1} which is failed by the record $(a_1^0, a_2^0, \dots, a_{k-1}^0)$, contradicting our original assumption.

Consider the following edit, E_* , implied by the edits (2.13), and formed by using Lemma 2.1 from the edits E_r , $r = 1, 2, \dots, R$ with field k as the generating field, namely

$$E_* = \bigotimes_{i=1}^{k-1} \{ \bigcap_{r=1}^R A_{ri} \} \times \{ \bigcup_{r=1}^R A_{rk} \} \bigotimes_{j=k+1}^N A_j. \quad (2.14)$$

Since $a_i \in A_{ri}$, $i = 1, 2, \dots, k-1$, for all $r = 1, 2, \dots, R$, the intersections $\bigcap_{r=1}^R A_{ri}$ are not empty. Hence E_* is a valid edit. Also according to our assumption, every possible value a_k of field k is included in one of the sets A_{rk} ; hence,

$$\bigcup_{r=1}^R A_{rk} = A_k.$$

Thus (2.14) reduces to the form

$$E_* = \bigotimes_{i=1}^{k-1} \{ \bigcap_{r=1}^R A_{ri} \} \times A_k \bigotimes_{j=k+1}^N A_j. \quad (2.15)$$

Since $E_* \in \Omega_{k-1}$, any record with $(a_1^0, a_2^0, \dots, a_{k-1}^0)$ as the entries in the fields $1, 2, \dots, k-1$ passes E_* . Thus it must follow that there is at least one value of field k , say, a_k^0 , which, together with a_i^0 , $i = 1, 2, \dots, k-1$, satisfies all edits in Ω_k .

QED.

Note that it was implicitly assumed that E_* is a member of the set of edits Ω_{k-1} . Note also that E_* is an essentially new implied edit. Thus the validity of the theorem depends on the set of edits being complete with respect to essentially new implied edits, but not necessarily all possible edits.

Example 2.9 In Example 2.6, with E_1 and E_2 as the only two explicit edits, $E_\star = A_1 \times \{2,3\} \times \{2\}$ together with E_1 and E_2 constitutes a complete set of edits. With $k=3$ in Theorem 2.1, choose the record y' with $a_2^0 = 1$ and $a_3^0 = 4$ and note that $\Omega_2 = E_\star$. Note also that $y' \notin \Omega_2$. Then a_1^0 can be chosen as $\{1\}$ to have $\{a_1^0, a_2^0, a_3^0\}$ satisfying all edits in the complete set $\Omega_3 = E_1 \cup E_2 \cup E_\star$.

Corollary 2.1.1

If a record has N fields, and if the values a_i^0 , in fields $i=1,2,\dots,k-1$, are such that all edits in Ω_{k-1} are satisfied, then there exist values a_i^0 , $i=k,k+1,\dots,N$, such that the values a_i^0 , $i=1,2,\dots,N$ satisfy all edits.

Proof Follows immediately by repeated application of Theorem 2.1.

Corollary 2.1.2

Suppose that a record has N fields having the values a_i , $i=1,2,\dots,N$ and S is a subset of these fields having the property that at least one of the fields in S enters each edit in Ω failed by the given record. Then values a_j^0 , $j \in S$ exist such that the changed record consisting of the values a_i^0 , $i \notin S$ together with a_j^0 , $j \in S$ satisfies all edits.

Before formally proving Corollary 2.1.2, it will be useful to get a feeling for its contents. Corollary 2.1.2 is a formalization of the observation made in Example 2.4a, namely, the solution to the SCP (2.9) will not always lead to a passing record. Corollary 2.1.2 demonstrates that a passing record can indeed be found using the solution

to the associated SCP if the edits, against which the record is being tested are a complete set. Example 2.10 illustrates the content of Corollary 2.1.2.

Example 2.10 Let $A_1 = \{1,2,3,4\}$, $A_2 = \{1,2,3\}$, $A_3 = \{1,2\}$ and let $\bar{E} = E_1 \cup E_2$, where

$$E_1 = \{1,2\} \times \{2,3\} \times A_3$$

$$E_2 = \{2,3,4\} \times \{1,2\} \times \{1\}$$

Let the record be $y = \{2\} \times \{2\} \times \{1\}$. E_1 and E_2 are both failed by y . Also fields 1 and 2 enter each of the failed edits E_1 and E_2 , while field 3 enters E_2 only. The SCP formulation for this situation is

$$\begin{aligned} \text{Min} \quad & w_1 + w_2 + w_3 \\ & w_1 + w_2 \geq 1 \\ & w_1 + w_2 + w_3 \geq 1 \\ & w_1, w_2, w_3 \geq 0 \end{aligned}$$

Possible solutions are: $w_1 = 1$; $w_2 = 1$. However, changing y_1 to 1, 3 or 4 (keeping fields 2 and 3 unchanged) still does not result in a passing record; nor does changing y_2 to 1 or 3.

However, the essentially new implied edits

$$E_{\star}^1 = E(1, \{E_1, E_2\}) = A_1 \times \{2\} \times \{1\}$$

and
$$E_{\star}^2 = E(2, \{E_1, E_2\}) = \{2\} \times A_2 \times \{1\}$$

together with E_1 and E_2 form a complete set of edits. On testing y against this complete set of edits we obtain the following SCP:

$$\begin{array}{ll}
 \text{Min} & w_1 + w_2 + w_3 \\
 & w_1 + w_2 \geq 1 \\
 & w_1 + w_2 + w_3 \geq 1 \\
 & w_2 + w_3 \geq 1 \\
 & w_1 + w_3 \geq 1 \\
 & w_1, w_2, w_3 \geq 0 .
 \end{array}$$

No single field enters all the failed edits. Field combinations (1,2), (2,3), and (1,3) are covers of the failed edits in the sense of the set S in the corollary. By changing, for instance, both fields 1 and 2, simultaneously, a passing record such as $y^* = \{3\} \times \{3\} \times \{1\}$ can be constructed. It is easily verified that changing the field combinations (2,3) and (1,3) in similar fashion also leads to passing records.

It is clear that in any given data validation situation with a set $\bar{E}$ of consistent edits, the set I_N of fields is one specification of the set S of Corollary 2.1.2. Changing all fields in a record to make it pass the edits in $\bar{E}$ is clearly not desirable and in fact will, usually, not be necessary.

It is clear, from the formal statement of the covering problem, (2.9), and Example 2.10 that, in general, there may be more than one cover of the edits available in any given situation. Further, if we associate with the fields a cost of altering values in the fields, based on considerations of reliability of entries in fields, which is non-uniform, then a particular cover or covers may prove to be better, in the sense of lower associated cost, than others. For instance, in Example 2.10, with the complete set under consideration,

with costs of 1,2,4 associated with fields 1, 2 and 3 respectively the covers (1,2,3), (1,2), (1,3), (2,3) cost, respectively 7, 3, 5 and 6 resulting in (1,2) being the 'best' cover. The problem of identifying the fields in a set such as S of Corollary 2.1.2, with associated minimum cost, will be designated the Minimum Weighted Fields to Impute problem (MWFI). We shall examine the MWFI problem in greater detail in subsequent sections and chapters.

Proof of Corollary 2.1.2 Without loss of generality, suppose that S consists of the fields $k, k+1, \dots, N$. Thus at least one of the fields $k, k+1, \dots, N$ enters each failed edit. Then, there are no failed edits involving only the fields $1, 2, \dots, k-1$, so that the values a_i , $i=1, 2, \dots, k-1$ satisfy all edits in Ω_{k-1} . Thus, from Corollary 2.1.1, there exist values a_i^0 , $i=1, k+1, \dots, N$, such that a_i , $i=1, 2, \dots, k-1$ together with a_i^0 , $i=k, k+1, \dots, N$ satisfy all the edits.

QED.

The generation of a complete set of edits, needed to solve the MWFI problem, involves the identification of all essentially new implied edits. Lemma 2.1 provides us with a method for generating essentially new implied edits; however, it does not guarantee that repeated application of that method will, in fact, generate all essentially new implied edits. Theorem 2.2 fills this gap.

Theorem 2.2

If $E_p = \bigotimes_{i \in I_N} A_{pi}$ is an edit which is logically implied by the explicit edits, then it can be generated by the edit generation procedure of Lemma 2.1.

Proof Pg. 34 [9]

Note If the original specification of the edits $\bar{E}$ is in itself inconsistent (unsatisfiable), but not obviously so, the inconsistency of $\bar{E}$ will be revealed during the process of generating a complete set of edits. An implied edit E_r , say, of the form $E_r = \bigotimes_{\substack{i \in I_N \\ i \neq k}} A_i \times A_{rk}$ will be generated with $A_{rk} \subseteq A_k$, which is inconsistent. This is because E_r states that entries $j \in A_{rk}$ in field k are invalid no matter what the entries in the other fields are, and yet $A_{rk} \subseteq A_k$ is a permissible set of values.

Felligi and Holt [9] proposed a procedure for constructing a complete set of edits from a given explicit set of edits based on the content of Theorem 2.2.

The Felligi-Holt Algorithm [9]

Construct all essentially new implied edits by "combining all edits in pairs, triples, and higher combinations using each field as the generating field in turn until all combinations have been exhausted." Realizing that this algorithm would be very slow in all but the simplest of cases, they proposed the following three rules to speed up the algorithm.

- Rule 1: For any set of edits to produce an essentially new edit they must have an entering field in common.
- Rule 2: No essentially new edit will be produced from an implied edit and a subset of the edits from which it was itself generated (by generating on the same field).

Rule 3: A combination of edits using a particular field i as the generating field need not be considered if some subset of the proposed combination using the same field i has already resulted in an essentially new implied edit.

Rule 1 is an immediate and direct consequence of the definition of an essentially new implied edit.

Proof of Rule 2 Let $E(k, E_q) = E_I = \bigotimes_{i \in I_N} A_{Ii}$ be an implied edit and let $\bar{E}_p \subseteq \bar{E}_q$. We have then $A_{Ij} \subseteq A_{pj}$, $j \neq k$, $j \in I_N$ and $A_{pk} \subseteq A_{Ik}$. Then $E_x = E(k, \bar{E}_p \cup E_I)$ has $A_{xj} \subseteq A_{Ij}$, $j \neq k$, $j \in I_N$, and $A_{xk} = A_{Ik}$. Thus $E_x \subseteq E_I$ and cannot be an essentially new implied edit.

Proof of Rule 3 Follows from the proof of Rule 2 with E_I designated as the essentially new implied edit. Example 2.11 illustrates the F-H procedure.

Example 2.11 Consider edits E_1 and E_2 as the original set of edits where $E_1 = \{1,2\} \times \{1,2,3\} \times \{1,2,5,6\}$, $E_2 = \{3,4\} \times \{2,3,4\} \times \{2,3,4\}$, and $A_1 = A_2 = \{1,2,3,4\}$, $A_3 = \{1,2,3,4,5,6\}$. The Fellgi-Holt procedure, even with the provisions of Rule 1, 2 and 3 incorporated, would appear to require generation of all edits using fields 1, 2, and 3 in various combinations and permutations. It will be shown in Chapter 3, however, that implied edits that are not essentially new are not needed to generate a set of edits sufficient for the solution of the MWFI problem. A step by step application of the F-H procedure to the edits E_1 and E_2 follows.

Step 1: 1 as the generating field:

$$E_3 = E(1, \{E_1, E_2\}) = A_1 \times \{2, 3\} \times \{2\}$$

Notice that E_3 is essentially new. No more new implied edits can be generated using 1 again as the generating field at this point (Rule 3).

2 as the generating field:

$$E_4 = E(2, \{E_1, E_3\}) = \{1, 2\} \times \{1, 2, 3\} \times \{2\}.$$

$$E_5 = E(2, \{E_2, E_3\}) = \{3, 4\} \times \{2, 3, 4\} \times \{2\}.$$

No more new implied edits can be generated on 2 at this stage (Rule 2).

3 as the generating field:

The edit set available for generation of implied edits is

$\{E_1, E_2, E_3, E_4, E_5\}.$

$$E_6 = E(3, \{E_1, E_3\}) = \{1, 2\} \times \{2, 3\} \times \{1, 2, 5, 6\}.$$

$$E_7 = E(3, \{E_2, E_3\}) = \{3, 4\} \times \{2, 3\} \times \{2, 3, 4\}.$$

No other new edits are produced at this stage.

Now let us start with 2 as the generating field on the system of edits $\{E_1, E_2, E_3, E_4, E_5, E_6, E_7\}.$

Step 2: 2 as the generating field:

$$E_8 = E(2, \{E_3, E_6\}) = \{1, 2\} \times \{2, 3\} \times \{2\}.$$

$$E_9 = E(2, \{E_3, E_7\}) = \{3, 4\} \times \{2, 3\} \times \{2\}.$$

No more new edits can be generated here.

1 as the generating field:

The edit set is now $\{E_1, E_2, E_3, E_4, E_5, E_6, E_7, E_8, E_9\}.$

However, $E(1, \{E_1, E_4\})$, $E(1, \{E_1, E_6\})$, and $E(1, \{E_6, E_7\})$ result in

E_4 , E_6 and E_3 respectively. Using Rules 1 and 2 we notice that no new edits can be generated at this stage.

3 as the generating field:

The edit set available is the same as above. Using Rule 1 we know that no new edit can be constructed on 3 using edits $\{E_1, E_2, \dots, E_7\}$. Edits E_8 and E_9 cannot be combined with this subset to produce any new edits either.

Step 3: Since no new edits have been produced using the current set $\bar{E}_W = \{E_1, E_2, \dots, E_9\}$ generating in turn on fields 2, 1, and 3, no new edits can be produced using a different permutation of generating fields either. The set of original edits, E_1 and E_2 , together with the only essentially new implied edit generated, E_3 , constitute $\bar{E}_C$, the complete set of edits, i.e. $\bar{E}_C = E_1 \cup E_2 \cup E_3 \subseteq \bar{E}_W$.

Note In Example 2.11 all edits were generated by combining existing edits in pairs. This came about because the entries in field 1 in the original set $\{E_1, E_2\}$ happened to be complementary. In general one would have to generate implied edits by combining edits in pairs, triples, etc. to arrive at a complete set of edits.

II. The Minimum Weighted Fields to Impute Formulation

Section 2.1 contains a body of theory developed to facilitate identification of data records in error and it hints at a methodology for 'correcting' such records so that they pass all specified edits.

We have, thus far, not formally stated our objective. In very broad terms the objective of data editing and validation procedures may be stated as follows:

Identify and correct records in error, producing a passing record which differs from the 'true' record as little as possible, at the lowest possible cost. (2.16)

By a true record is meant the record with all the correct values placed in its various fields. This objective while desirable is also utopian. Formally, let x be the failing record and y_T the true record and let e be the vector of changes required to change x into y_T . It should be mentioned here that y_T may not be known even to the source of the data. We then have a vector equation of the form

$$x + e = y_T \quad (2.17)$$

to consider while striving for the objective (2.16). An approach to identifying e to conform to the model (2.17) would be to make as few changes, in a failing record, as possible to make the record passing. In our terminology this objective translates to "change a minimum number of fields to make the record passing." We shall refer to this problem as the Minimum Fields to Impute (MFI) problem of data editing. Clearly MFI is a special case of MWFI introduced in Section 2.1. It was indicated in Section 2.1 that the MWFI problem can be formulated as a set covering problem. It was shown in Liepins [21] that MWFI leads to the 'most likely' record, under certain assumptions.

Let $\bar{E}$ be the set of edits failed by record x and let J^0 be the set of all subsets of the index set of fields I_N . Further suppose that the underlying objective can be stated as: "find the set of

fields which is most likely in error." Equivalently, this objective can be written as the problem of finding the set $K^* \in J^0$ which maximizes over all $K \in J^0$

$$P[\{(\bar{y}_j, j \in K) \cap (y_j, j \in I_N - K)\} | x \in \bar{E}] \quad (2.18)$$

where $\bar{y}_j$ denotes the event that field j is in error and y_j the event that field j is not in error.

It was shown in Naus et al [27] that $P[\bar{y}_j | x \in \bar{E}]$ may be approximated, under certain assumptions, by $P[\bar{y}_j | \bigcup_{k \in \bar{E}} \bar{y}_k]$, where $k \in \bar{E}$ means that k enters some edit $E \in \bar{E}$. Before proceeding to state the assumptions it is useful to consider the nature of the approximation. It is clear that $x \in \bar{E} \Rightarrow \bigcup_{k \in \bar{E}} \bar{y}_k$, i.e. if the record is in error one or more of the fields that enter the edit system are in error. The converse is, however, not always true. In fact $\bigcup_{k \in \bar{E}} \bar{y}_k$ may not even imply that the record fails any edit.

Let T denote the event $x \in E_i, E_i \in \bar{E}_e, i=1, \dots, m$. T implies that at least one field that enters is in error for each of the E_i . Thus there exists a cover, and hence a prime cover, of the FEM rows corresponding to the E_i by fields in error. If $\bar{B}$ denotes the set of all prime covers by fields (in error and otherwise) of the FEM rows corresponding to the E_i , then the event that at least one of these prime covers "holds" (has all fields involved in it in error) will be denoted by C . Note that while T implies C , C does not always result in T occurring. Let $S_j \subseteq \bar{B}$ denote the set of prime covers each of which includes field j . The event that at least one of the prime covers in S_j "holds" will be denoted by H_j , while H_j' will denote the event that none of the prime covers in S_j "hold." If $\bar{S}_j \subseteq \bar{B}$

denotes the set of covers in $\bar{B}$ that do not include field j then, H_j^* will denote the event that at least one of the prime covers in $\bar{S}_j$ "holds." Naus et al assert then that, $P[\bar{y}_j|T] = P[\bar{y}_j|C]$ if

$$P[T|H_j] = P[T|\bar{y}_j \cap H_j' \cap H_j^*] = P[T|C] . \quad (2.19)$$

The condition (2.19) states that the likelihood of $x \in E_i$, $i=1, \dots, m$ given an explanation (a cover that "holds") is the same as when other explanations are available. Also the condition in the middle term of (2.19) states that this likelihood is the same even when field j is in error and yet no prime cover with field j "holds." Thus the underlying assumption formalized in (2.19) is that it is irrelevant to the likelihood of T whether or not a particular field j is in error.

The condition $Q \equiv \bar{y}_j \cap H_j' \cap H_j^*$, in (2.19), is difficult to reconcile with, for it requires that H_j be false even when field j is in error. Consider, for example, the edit $E: \{1,2\} \times A_2 \times \{1,3\}$, $A_1 = A_2 = A_3 = \{1,2,3\}$ and a record $x = \{1\} \times \{3\} \times \{3\}$. Clearly $x \in E$. Suppose fields 1 and 3 are both in error. Clearly Q does not hold. In terms of our notation the event C is precisely $\bigcup_{k \in E} \bar{y}_k$ and under the assumption (2.19) we may replace $x \in E$ in (2.18) by $\bigcup_{k \in E} \bar{y}_k$. Then (2.18) becomes

$$\max P[\{(\bar{y}_j, j \in K) \cap (y_j, j \in I_N - K)\} | \bigcup_{k \in E} \bar{y}_k] \quad (2.20)$$

Further assume that for the error model, the $\bar{y}_j$'s are independent of each other and also that the $P(\bar{y}_j)$'s are independent of the magnitude of the error. Neither assumption is particularly unrealistic for coded data. Clearly the coding scheme could be altered, thus altering the magnitude of the error. In surveys with fields which are typically not derived from other fields the independence of error occurrences is

justifiable. Then (2.20) can be rewritten, using the multiplication law of probabilities and the property of independent events, as:

$$\max_{K \in J^0} \prod_{j \in K} (P(\bar{y}_j)) \prod_{j \in I_N - K} (1 - P(\bar{y}_j))$$

such that

$$x + e \in S \quad (2.21)$$

$$e_j = 0, j \in I_N - K$$

where e_j is a proposed correction to x in field j and S is the set of acceptable records.

It will be shown that (2.21) can be transformed into MWFI. Note that the objective function in (2.21) can be written as:

$$\prod_{j \in I_N} (1 - P(\bar{y}_j)) \prod_{j \in K} (P(\bar{y}_j) / (1 - P(\bar{y}_j))) \quad (2.22)$$

where the first product term is a constant.

Taking the negative of the natural logarithm of (2.22) and setting the cost c_j of changing field j to

$$c_j = -\ln(P(\bar{y}_j)) + \ln(1 - P(\bar{y}_j)) \quad (2.23)$$

(2.21) is transformed to

$$\begin{aligned} \text{Minimize} \quad & \sum_{j=1}^N c_j \delta(e_j) \\ & x + e \in S \end{aligned} \quad (2.24)$$

$$\delta(e_j) = \begin{cases} 1, & e_j \neq 0 \\ 0, & e_j = 0 \end{cases} .$$

Clearly (2.24) seeks a change in the fields with minimum total cost to create an acceptable record from a failing record x . The cost

structure and our assumptions ensure that the solution to (2.24) results in a 'most likely' change of a record x .

We now proceed to show that the mathematical program (2.24) corresponding to the MWFI formulation can be put in the format of a set covering problem. The set to be covered is the set of edits failed by the record y . We saw in Example 2.10 that changing fields that cover this set of failed edits may not always be sufficient to ensure that this changed record $y^* \notin \bar{E}$, the originally specified set of edits. We also saw that Corollary 2.1.2 assures us, on the other hand, that if the subset of a complete set of edits that are failed by y is used in the MWFI formulation then $y^* \notin \bar{E}$.

Corollary 2.1.2 states that the set to be covered, in quest of a minimal set of fields that can be changed to result in a passing record, is the set of rows of the FEM corresponding to the subset of a complete set of edits that are failed by the record. Denoting this FEM by A and assuming that there are r edits (or equivalently r rows) in A , we have the following reformulation of (2.24).

MFI: $\text{Min } \sum_{j \in I_N} w_j$

such that

$$\sum_{j \in I_N} a_{ij} w_j \geq 1 \quad \text{for all } i = 1, 2, \dots, r \quad (2.25)$$

$$w_j = \begin{cases} 1 & \text{if field } j \text{ is to be changed} \\ 0 & \text{otherwise} \end{cases} \quad \text{for all } j \in I_N$$

and

$$a_{ij} = \begin{cases} 1 & \text{if field } j \text{ enters edit } i \\ 0 & \text{otherwise,} \end{cases} \quad i = 1, 2, \dots, r.$$

Formulation (2.25) will hereafter be referred to as the Minimum Fields to Impute problem (MFI).

The MFI is thus a set covering problem in which equal emphasis is placed on changing any of the N fields. It is likely that persons familiar with a given survey situation would be able to approximate closely the cost structure leading to the formulation of (2.24). The resultant problem, in set covering problem (SCP) format, will be known, from here on, as the MWFI problem.

$$\text{MWFI} \quad \text{Min} \quad \sum_{j \in I_N} c_j w_j$$

$$\text{such that} \quad \sum_{j \in I_N} a_{ij} w_j \geq 1, \quad i = 1, 2, \dots, r$$

$$w_j = \begin{cases} 1 & \text{if field } j \text{ is to be changed} \\ 0 & \text{otherwise} \end{cases}$$

$$\text{and} \quad a_{ij} = \begin{cases} 1 & \text{if field } j \text{ enters edit } i \\ 0 & \text{otherwise} \end{cases} \quad (2.26)$$

In this chapter we introduced the data editing and validation problem associated with coded data records in error. More specifically, the MWFI problem was introduced. A development of theory and an algorithm, largely due to Felligi and Holt [9], based on that theory for identifying a complete set of edits to facilitate solution of the MWFI problem were presented. Alternative approaches to the solution of the MWFI problem which draw upon and improve on this theory and algorithm will be presented in Chapter 3.

CHAPTER 3

ALGORITHMS FOR THE MWFI PROBLEM WITH CODED DATA

In Chapter 2, the theory leading to a procedure for constructing a complete set of edits, $\bar{E}_c$, was developed. It was noted that solving the SCP (2.26), using the original set of edits, E_e , may not always result in the solution of MWFI. Corollary 2.1.2 assures us that if $\bar{E}_c$, instead of merely E_e is used in formulating (2.26), any solution to that SCP is a solution to MWFI.

The construction of $\bar{E}_c$, as demonstrated in Example 2.11, can be a very cumbersome process even with a small number of original edits and a small number of fields. Clearly the amount of work involved in the construction of a complete set of edits, using the F-H procedure, starting from a few hundred original edits and about 100 fields would be prohibitive, even with a computer. Computational performance and an analysis of the F-H procedure, from a computational viewpoint, is a subject of Chapter 4. In the remainder of this chapter, a theory is developed to reduce the amount of work involved in constructing $\bar{E}_c$ and hence in solving MWFI. An entirely new approach to solving MWFI and the theory leading to such an algorithm is also developed.

I. Some Results Leading to Algorithm 3.1

Before proceeding to lay out a refinement of the F-H algorithm (for the generation of a complete set of edits) we need to be aware of the following results.

Lemma 3.1

Let $E_p = \bigotimes_{j \in I_N} A_{pj}$ and $E_q = \bigotimes_{j \in I_N} A_{qj}$ be any two edits such that $E_q \subset E_p$ and let E be a set of edits on the same set of fields indexed by I_N .

If $E_{q'} = E(k, \{E, E_q\}) = \bigotimes_{j \in I_N} A_{q'j}$

and

$$E_{p'} = E(k, \{E, E_p\}) = \bigotimes_{j \in I_N} A_{p'j}$$

then

$$E_{q'} \subseteq E_{p'}$$

Proof

Let $E_I = E(k, E) = \bigotimes_{j \in I_N} A_{Ij}$, then

$$E_{q'} = E(k, \{E_I, E_q\}) = \bigotimes_{j \in I_N} A_{q'j}, \text{ and}$$

$$E_{p'} = E(k, \{E_I, E_p\}) = \bigotimes_{j \in I_N} A_{p'j}$$

so that $A_{q'j} = A_{qj} \cap A_{Ij}, j \neq k, j \in I_N$ }
and $A_{q'k} = A_{qk} \cup A_{Ik} .$ } (3.1)

While $A_{p'j} = A_{pj} \cap A_{Ij}, j \neq k, j \in I_N$ }
and $A_{p'k} = A_{pk} \cup A_{Ik} .$ } (3.2)

From (3.1) and (3.2), if $E_q \subset E_p$, we have

$A_{q'j} = A_{qj} \cap A_{Ij} \subseteq A_{pj} \cap A_{Ij} = A_{p'j}, j \neq k, j \in I_N$ } (3.3)
and $A_{q'k} = A_{qk} \cup A_{Ik} \subseteq A_{pk} \cup A_{Ik} = A_{p'k}$ }

since $E_q \subset E_p \Rightarrow A_{qj} \subseteq A_{pj}$, for all $j \in I_N$.

Thus, from (3.3), we have $E_{q'} \subseteq E_{p'}$.

QED.

Although Lemma 3.1 is fairly intuitive, it is quite useful in reducing the work involved in constructing $\bar{E}_C$. In the F-H procedure we were faced with the generation of implied edits using an ever increasing number of edits. The result of Lemma 3.1 enables us to discard any dominated edits at any stage of the 'complete edit set' generation process.

If for all y in the data space, $y \in \bar{E}_A \Leftrightarrow y \in \bar{E}_B$, then the edit sets $\bar{E}_A$ and $\bar{E}_B$ are the exact same subset of D . This will be denoted by $D(\bar{E}_A) = D(\bar{E}_B)$.

The following example illustrates the fact that alternative sets of edits may be available, sometimes, for use in the SCP formulation (2.26) leading to the same MWFI problem solution.

Example 3.1

$$\bar{E}_A = E_1 \cup E_2$$

where

$$E_1 = \{1,2\} \times A_2 \times \{1,3\}$$

$$E_2 = \{2\} \times \{1,3\} \times A_3$$

and

$$\bar{E}_B = E_3 \cup E_4 \cup E_5 \cup E_6 \cup E_7$$

where

$$E_3 = \{1\} \times A_2 \times \{1,3\}$$

$$E_4 = \{1,2\} \times A_2 \times \{1\}$$

$$E_5 = \{2\} \times A_2 \times \{1,3\}$$

$$E_6 = \{2\} \times \{1\} \times A_3$$

$$E_7 = \{2\} \times \{3\} \times A_3$$

with $A_1 = A_2 = A_3 = \{1,2,3\}$.

Note that $D(\bar{E}_A) = D(\bar{E}_B)$, and that $\bar{E}_A$ is complete. There are three different situations that could arise in using $\bar{E}_A$ and $\bar{E}_B$, in turn, in the SCP formulation (2.26).

Case 1 $y \in E_1, y \notin E_2$.

The fields that enter matrix (FEM) corresponding to $\bar{E}_A$ is then $F_A = [1 \ 0 \ 1]$. Possible solutions to the SCP are then to change field 1 or field 3. (This change must result in a valid record, Corollary 2.1.2). The FEM corresponding to $\bar{E}_B$, F_B has essentially the same structure:

$$F_B = \begin{bmatrix} 1 & 0 & 1 \\ 1 & 0 & 1 \end{bmatrix} \quad \text{corresponding to} \quad \begin{bmatrix} E_3 \\ E_4 \end{bmatrix} \quad \text{or} \quad \begin{bmatrix} E_4 \\ E_5 \end{bmatrix}$$

The set covering solution in this case is the same as for $\bar{E}_A$. Clearly, whenever row 1 (row 2) of F_B is covered by a column, row 2 (row 1) is also covered. Thus F_B can be reduced to $F_B' = [1 \ 0 \ 1] = F_A$.

Case 2 $y \in E_2, y \notin E_1$

$F_A = [1 \ 1 \ 0]$, and possible SCP solutions are to change field 1 or field 2. Also $F_B = [1 \ 1 \ 0] = F_A$ corresponding to E_6 or E_7 . The SCP solutions are clearly the same as for $\bar{E}_A$.

Case 3 $y \in E_1 \cap E_2$

$$F_A = \begin{bmatrix} 1 & 0 & 1 \\ 1 & 1 & 0 \end{bmatrix}$$

corresponding to a solution of 'change field 1' (or if $c_1 > c_2 + c_3$, 'change fields 2 and 3').

It is easy to verify that the reduced F_B , $F_B' = \begin{bmatrix} 1 & 0 & 1 \\ 1 & 1 & 0 \end{bmatrix}$ leads to the same SCP solution as with F_A .

It was observed at the beginning of this chapter that even though $D(\bar{E}_c) = D(\bar{E}_e)$, a solution to (2.26) using $\bar{E}_e$ does not always solve MWFI; together with Example 3.1 this leads us to the following definition.

Definition Two sets of edits $\bar{E}_A$ and $\bar{E}_B$ are said to be sufficient for each other if and only if:

$$D(\bar{E}_A) = D(\bar{E}_B) \quad (3.3a)$$

and

the set of optimal solutions using $\bar{E}_A$ in (2.26), is identical to that using $\bar{E}_B$ for each record y . (3.3b)

In using the F-H algorithm of Chapter 2, we are faced with the task of constructing edits using all permutations of the N fields as generating fields in sequence. Theorem 3.1 establishes that the non-redundant set of implied edits generated using fields $1, 2, \dots, N$ in sequence is identical to the non-redundant set generated using any permutation $\delta(1, 2, \dots, N)$ of the N fields.

Theorem 3.1

Let $\bar{E}_e$ be the original set of explicit edits and let $\bar{E}_i$ be the set of all implied edits which can be constructed from $\bar{E}_e$ using field i as a generating field. Define

$$\bar{E}_i \equiv \bar{X}(i, \bar{E}_e)$$

and

$$\bar{S}_{ij} \equiv \bar{X}(j, \{\bar{E}_i \cup \bar{E}_e\}).$$

Then, if $\bar{S}_{ij}^*$ is the non-redundant subset of $\bar{S}_{ij}$, $\bar{S}_{ij}^* = \bar{S}_{ji}^*$.

Proof Let $\bar{E}_s = \bar{E}_e \cup \bar{E}_i \cup \bar{E}_j$ and $\bar{\sigma}_j = \bar{X}(j, \bar{E}_s)$. Then $\bar{S}_{ij} = \bar{E}_s \cup \bar{\sigma}_j$ and $\bar{S}_{ji} = \bar{E}_s \cup \bar{\sigma}_i$. An implied edit, $E_x \in \bar{S}_{ij}$, is constructed, starting from $\bar{E}_e$, and using fields i and then j as the generating fields. It is then demonstrated that an implied edit $E_y \in \bar{S}_{ji}$ can be constructed from $\bar{E}_e$, using field j and then i as the generating field so that $E_x \subseteq E_y$. It is then argued that for each $E_x \in \bar{S}_{ij}$, there exists $E_y \in \bar{S}_{ji}$ such that $E_x \subseteq E_y$, and vice versa.

To establish the theorem, we need to look at explicit representations of edits in $\bar{\sigma}_j$ and $\bar{\sigma}_i$.

Generation of an Edit in $\bar{\sigma}_j$

Let $\bar{B} \subseteq \bar{E}_e \cup \bar{E}_j$ be a set of edits involved in the generation of an arbitrary edit $E_x \in \bar{\sigma}_j$. We may assume without loss of generality that a set of r edits $\bar{W} = \{E_{q_1}, \dots, E_{q_r}\} \subseteq \bar{E}_e$, and no other edits from $\bar{B}$, are involved in the generation of E_x ; for, any edit in $\bar{E}_j$ can be generated, on field j , using a subset of the original edits $\bar{E}_e$.

Let A_m be the set of allowable values in field m , as before, and let A_m^g be the set of values appearing in $E_g \in \bar{E}_e$, in field m .

In the following, we shall assume, for simplicity of exposition (w.l.o.g.), that there are only three fields i , j and k occurring in the edits. Thus $V = E(j, \bar{W})$ can be written as

$$V = (A_i^{q_1} \cap A_i^{q_2} \cap \dots \cap A_i^{q_r}) \times (A_j^{q_1} \cup A_j^{q_2} \cup \dots \cup A_j^{q_r}) \times (A_j^{q_1} \cap A_j^{q_2} \cap \dots \cap A_j^{q_r}) \quad (3.4)$$

Let $\bar{G} \subseteq \bar{E}_j$ be another set of t edits involved in the generation of E_x . Let the indices of these edits in the set $\bar{E}_j$ be $1, 2, \dots, t$, for simplicity and w.l.o.g.; and let $\ell_1, \ell_2, \dots, \ell_t$ be the numbers of edits, from $\bar{E}_e$, involved in the construction of these t edits: i.e. $E_a = E(i, b=1, 2, \dots, \ell_a E_{s_b^a})$, $a=1, 2, \dots, t$; $E_{s_b^a} \in \bar{E}_e$. Note that many of the indices s_b^a , could be the same, as the following example illustrates.

Example 3.2 Consider the set of edits $\bar{E}_e = E_1 \cup E_2 \cup E_3$ with

$$E_1 = \{1, 2\} \times \{2, 3\} \times \{1, 2\}$$

$$E_2 = \{2, 3\} \times \{3\} \times \{1, 2\}$$

$$E_3 = \{3\} \times \{1, 2\} \times \{1\}$$

$A_1 = A_2 = A_3 = \{1, 2, 3\}$. Here (field) $i = 1$, and (field) $j = 2$.

Then $\bar{E}_1$ the set of edits constructible from $\bar{E}_e$ generating on field 1 is, $\bar{E}_1 = E_4 \cup E_5$,

where $E_4 = E(1, \{E_1, E_2\}) = A_1 \times \{3\} \times \{1, 2\}$

and $E_5 = E(1, E_1, E_3) = A_1 \times \{2\} \times \{1\}$

Similarly

$$\bar{E}_2 = E_6 \cup E_7$$

where $E_6 = E(2, \{E_1, E_2\}) = \{2\} \times \{2, 3\} \times \{1, 2\}$

and $E_7 = E(2, \{E_2, E_3\}) = \{3\} \times A_2 \times \{1\}$

Suppose 2 edits are to be chosen from $\bar{B} = \bar{E}_e \cup \bar{E}_2$ for generating V .

Let E_1 and E_6 be the 2 edits. Since $E_6 = E(2, \{E_1, E_2\})$ it suffices to choose E_1 and E_2 , from $\bar{E}_e$, for generating V ; here $r=2$, corresponding to the two edits E_1 and E_2 . In terms of the notation of the Theorem,

$$V = (A_1^1 \cap A_1^2) \times (A_2^1 \cup A_2^2) \times (A_3^1 \cap A_3^2) = E_6 \quad (3.5)$$

with $q_1 = 1$ and $q_2 = 2$.

Let $\bar{G} = \bar{E}_1 = E_4 \cup E_5$. Define $H = E(2, \{E_4, E_5\})$. Here $t = 2$ edits; namely E_4, E_5 . Now, E_4 may be derived as $E(1, \{E_1, E_2\})$; i.e. $\ell_1 = 2$, namely the two edits E_1, E_2 , and $E_5 = E(1, \{E_1, E_3\})$; i.e. $\ell_2 = 2$, namely E_1, E_3 . Then (s_1^1, s_2^1) and (s_1^2, s_2^2) correspond to the edits, (E_1, E_2) and (E_1, E_3) respectively. Thus,

$$\begin{aligned} H = & (A_1^1 \cup A_1^2) \cap (A_1^1 \cup A_1^2) \times (A_2^1 \cap A_2^2) \cup (A_2^1 \cap A_2^3) \\ & \times (A_3^1 \cap A_3^2 \cap A_3^3) = A_1 \times \{2, 3\} \times \{1\} \end{aligned} \quad (3.6)$$

Returning to the proof: define $H = E(j, \bar{G})$, then H can be written, similar to (3.6), as

$$\begin{aligned} & (A_i^{s_1^1} \cup A_i^{s_2^1} \cup \dots \cup A_i^{s_{\ell_1}^1}) \cap (A_i^{s_1^2} \cup A_i^{s_2^2} \cup \dots \cup A_i^{s_{\ell_2}^2}) \\ & \quad \cap \dots \cap (A_i^{s_1^t} \cup A_i^{s_2^t} \cup \dots \cup A_i^{s_{\ell_t}^t}) \\ H = & \times (A_j^{s_1^1} \cap A_j^{s_2^1} \cap \dots \cap A_j^{s_{\ell_1}^1}) \cup (A_j^{s_1^2} \cap A_j^{s_2^2} \cap \dots \cap A_j^{s_{\ell_2}^2}) \\ & \quad \cup \dots \cup (A_j^{s_1^t} \cap A_j^{s_2^t} \cap \dots \cap A_j^{s_{\ell_t}^t}) \\ & \times (A_k^{s_1^1} \cap A_k^{s_2^1} \cap \dots \cap A_k^{s_{\ell_1}^1}) \cap \dots \cap (A_k^{s_1^t} \cap A_k^{s_2^t} \cap \dots \cap A_k^{s_{\ell_t}^t}) \end{aligned} \quad (3.7)$$

We have then, $E_x = E(j, \bar{B} \cup \bar{G}) = E(j, \{V, H\})$ such that

$$\begin{aligned} & (A_i^{q_1} \cap A_i^{q_2} \cap \dots \cap A_i^{q_r}) \cap (A_i^{s_1^1} \cup A_i^{s_2^1} \cup \dots \cup A_i^{s_{\ell_1}^1}) \\ & \quad \cap \dots \cap (A_i^{s_1^t} \cup A_i^{s_2^t} \cup \dots \cup A_i^{s_{\ell_t}^t}) \\ E_x = & \times (A_j^{q_1} \cup A_j^{q_2} \cup \dots \cup A_j^{q_r}) \cup (A_j^{s_1^1} \cap A_j^{s_2^1} \cap \dots \cap A_j^{s_{\ell_1}^1}) \\ & \quad \cup \dots \cup (A_j^{s_1^t} \cap A_j^{s_2^t} \cap \dots \cap A_j^{s_{\ell_t}^t}) \end{aligned}$$

$$\begin{aligned}
 & \times (A_k^{q_1} \cap A_k^{q_2} \cap \dots \cap A_k^{q_r}) \cap (A_k^{s_1^1} \cap A_k^{s_2^1} \cap \dots \cap A_k^{s_{\ell_1}^1}) \\
 & \cap \dots \cap (A_k^{s_1^t} \cap \dots \cap A_k^{s_{\ell_t}^t}) \quad (3.8)
 \end{aligned}$$

Example 3.2 (continued)

From (3.5) and (3.6), with $j = 2$ we have

$$\begin{aligned}
 E_x = & (A_1^1 \cap A_1^2) \cap (A_1^1 \cup A_1^2) \cap (A_1^1 \cup A_1^3) \times (A_2^1 \cup A_2^2) \\
 & \cup (A_2^1 \cap A_2^2) \cup (A_2^1 \cap A_2^3) \times (A_3^1 \cap A_3^2 \cap A_3^3) \quad (3.9)
 \end{aligned}$$

$$E_x = \{2\} \times \{2,3\} \times \{1\} \quad (3.10)$$

Notice that E_x can be rewritten as E_y , using the distributive law of set union and intersections as follows.

$$\begin{aligned}
 & (A_1^1 \cap A_1^2 \cap A_1^1) \cup (A_1^1 \cap A_1^2 \cap A_1^2) \cap (A_1^1 \cup A_1^3) \\
 E_y = & \times (A_2^1 \cup A_2^2 \cup A_2^1) \cap (A_2^1 \cup A_2^2 \cup A_2^2) \cup (A_2^1 \cap A_2^3) \\
 & \times (A_3^1 \cap A_3^2 \cap A_3^1) \cap (A_3^1 \cap A_3^2 \cap A_3^2) \cap (A_3^1 \cap A_3^3)
 \end{aligned}$$

or,

$$\begin{aligned}
 & (A_1^1 \cap A_1^2 \cap A_1^1 \cap A_1^1) \cup (A_1^1 \cap A_1^2 \cap A_1^1 \cap A_1^3) \\
 & \cup (A_1^1 \cap A_1^2 \cap A_1^2 \cap A_1^1) \cup (A_1^1 \cap A_1^2 \cap A_1^2 \cap A_1^3) \\
 E_y = & \times (A_2^1 \cup A_2^2 \cup A_2^2 \cup A_2^1) \cap (A_2^1 \cup A_2^2 \cup A_2^1 \cup A_2^3) \\
 & \cap (A_2^1 \cup A_2^2 \cup A_2^2 \cup A_2^1) \cap (A_2^1 \cup A_2^2 \cup A_2^2 \cup A_2^3) \\
 & \times (A_3^1 \cap A_3^2 \cap A_3^1 \cap A_3^1) \cap (A_3^1 \cap A_3^2 \cap A_3^1 \cap A_3^3) \\
 & (A_3^1 \cap A_3^2 \cap A_3^2 \cap A_3^1) \cap (A_3^1 \cap A_3^2 \cap A_3^2 \cap A_3^3) \quad (3.11)
 \end{aligned}$$

E_y , in (3.11), implies the following: generate first on field 2, using edits (E_1, E_2) , (E_1, E_2, E_3) , (E_1, E_2) and (E_1, E_2, E_3) , then generate using the resultant 'edits', on field 1. Returning to the proof:

Generation of an edit in $\bar{\sigma}_i$

We now demonstrate that the edits in $\bar{E}_e$ used in deriving $E_x \in \bar{\sigma}_j$ can be used to derive $E_y \in \bar{\sigma}_i$. As in (3.11), E_x , in general, can be rewritten as E_y as follows:

Define Z = cross product term in (3.8) corresponding to field k .

$$R = (A_i^{q_1} \cap A_i^{q_2} \cap \dots \cap A_i^{q_r}), P = (A_j^{q_1} \cup A_j^{q_2} \cup \dots \cup A_j^{q_r})$$

Then,

$$\begin{aligned} & [R \cap A_i^{s_1^1} \cap A_i^{s_1^2} \cap \dots \cap A_i^{s_1^t}] \cup [R \cap A_i^{s_1^1} \cap A_i^{s_2^2} \cap A_i^{s_3^3} \cap \dots \cap A_i^{s_1^t}] \\ & \quad \cup \dots \cup [R \cap A_i^{s_1^1} \cap A_i^{s_2^2} \cap \dots \cap A_i^{s_t^t}] \\ E_x = & x [P \cup A_j^{s_1^1} \cup A_j^{s_1^2} \cup \dots \cup A_j^{s_1^t}] \cap [P \cup A_j^{s_1^1} \cup A_j^{s_2^2} \cup A_j^{s_3^3} \cup \dots \cup A_j^{s_1^t}] \\ & \quad \cap \dots \cap [P \cup A_j^{s_1^1} \cup A_j^{s_2^2} \cup \dots \cup A_j^{s_t^t}] \\ & \quad \times Z \end{aligned} \tag{3.12}$$

To show that $\bar{S}_{ij}^*$ and $\bar{S}_{ji}^*$ are identical we need to show that $E_x \subseteq E_y$. The construction of (3.12), from the edit set $\bar{E}_e$, can be achieved as follows. Denote by W_i^n the n th bracketed term, corresponding to field i , in (3.12). Construct the cross product Q_n , corresponding to the n th bracketed terms, in fields i , j and k of (3.12), by generating first on j , i.e. $Q_n = W_i^n \times W_j^n \times W_k^n$. If $n = 1, 2, \dots, h$, it is not difficult to see that not all h cross products Q_n , may be valid as edits. Example 3.2 illustrates such a situation.

Example 3.2 (continued)

From (3.11) we see that Q_2 and Q_4 can be written as

$$Q_2 = (A_1^1 \cap A_1^2 \cap A_1^1 \cap A_1^3) \times (A_2^1 \cup A_2^2 \cup A_2^1 \cup A_2^3) \\ \times (A_3^1 \cap A_3^2 \cap A_3^1 \cap A_3^3) = Q_4$$

$$Q_2 = \phi \times A_2 \times \{1\} = Q_4$$

Thus Q_2 and Q_4 are not valid edits. Since they are not valid edits they cannot be used in the generation of implied edits (Lemma 1.1).

Notice that

$$Q_1 = Q_3 = \{2\} \times \{2,3\} \times \{1,2\} = E_y. \quad (3.13)$$

Notice also that $E_x \subset E_y$. Returning to the proof:

Given that E_x is an implied edit, note that $W_j^u \cap W_j^v \neq \phi$; for all $u, v \in \{1, 2, \dots, h\}$ in (3.12). For, otherwise the term corresponding to field j in (3.11) would be empty and E_x would not be an edit contrary to our assumption. By a similar argument $Z \neq \phi$ in (3.12).

There are then two possible situations:

Case I $W_i^n \neq \phi$, $n = 1, 2, \dots, h$ in (3.12).

In this case the Q_n , $n = 1, 2, \dots, h$ are all valid implied edits and $E_y = E(i, \{Q_1, Q_2, \dots, Q_h\}) = E_x$. Note that E_y has been constructed by first generating h edits on field j and then combining these edits on field i ; i.e. $E_y \in \mathcal{S}_{ji}$. Thus if $E_x \in \mathcal{S}_{ij}$, there exists $E_y \in \mathcal{S}_{ji}$ such that $E_x = E_y$. Clearly $\mathcal{S}_{ij} \equiv \mathcal{S}_{ji}$ and hence $\mathcal{S}_{ij}^* \equiv \mathcal{S}_{ji}^*$.

Case II $W_i^n = \phi$ for some $n \in \{1, 2, \dots, h\}$.

In this case, as we saw in the example, not all the Q_n 's are valid edits. Notice, however, that not all W_i^n , $n = 1, 2, \dots, h$, can be empty; otherwise E_x , contrary to our assumption, would not be a

valid implied edit. Since not all Q_n 's are valid edits, generation on field i using those Q_n 's which are valid edits results in $E_y = E(i, \{Q_n, n=1,2,\dots,h \mid Q_n \text{ is a valid edit}\})$, which may not be identical to E_x .

Corresponding to the empty W_i^n , the W_j^n and W_k^n , in (3.12), will not be participants in the construction of E_y . Denoting the entries in the t -th field of E_x and E_y by A_{xt} and A_{yt} respectively, we have

$$A_{xk} \subseteq A_{yk}$$

$$A_{xj} \subseteq A_{yj}$$

$$A_{xi} = A_{yi}$$

Hence $E_x \subseteq E_y$. Since $E_x \in \bar{S}_{ij}$ and $E_y \in \bar{S}_{ji}$, and since the roles of E_x and E_y could equally well have been reversed in this proof, $\bar{S}_{ij}^* \equiv \bar{S}_{ji}^*$.

QED.

Corollary 3.1.1

Let $\bar{S}_{1,2,\dots,k} = \bar{X}(k, \bar{X}(k-1, \dots, (\bar{X}(1, E_e) \cup E_e) \dots \cup E_e))$ and let $\bar{S}_{\delta(1,2,\dots,k)}$ be the set of edits generated, similar to $\bar{S}_{1,2,\dots,k}$, by using any arbitrary permutation $\delta(1,2,\dots,k)$ of the fields $1,2,\dots,k$ as generating fields in sequence. Then $\bar{S}_{1,2,\dots,k}^* \equiv \bar{S}_{\delta(1,2,\dots,k)}^*$.

Proof Follows from repeated application of Theorem 3.1 and using Lemma 3.1.

Corollary 3.1.2

Let $\bar{S}'_{1,2,\dots,k}$ be the set of essentially new implied edits

generated on fields $1, 2, \dots, k$ used as the generating sequence and let $\bar{E}_S^*$ be the non-redundant subset of the edit set

$$\bar{S} = \bar{E}_e \cup_{k \in I_N} \bar{S}'_{1,2,\dots,k}. \quad (3.14)$$

If $\bar{E}_C^*$ is the non-redundant subset of $\bar{E}_C$, the complete set of edits then $\bar{E}_S^* \equiv \bar{E}_C^*$.

Proof Follows from Corollary 3.1.1

Theorem 3.2 formalizes the intuitive notion that redundancy in an edit set $\bar{E}$ leads to redundancy in the rows of FEM's associated with $\bar{E}$ in setting up an SCP such as (2.9). Thus discarding redundant edits does not change the solution to the SCP, since discarding redundant rows does not change the solution to an SCP, (see Garfinkel [13]).

Theorem 3.2

$\bar{E}_C^*$ is sufficient for $\bar{E}_C$.

Proof We have $y \in \bar{E}_C^* \Leftrightarrow y \in \bar{E}_C$ by definition of redundancy. Thus (3.3a) is satisfied and it remains to show that (3.3b) is also true for $\bar{E}_C^*$ and $\bar{E}_C$.

In the construction of the MWFI (see (3.3b)) the matrix A is the FEM of the edits that are failed by a given record y . Let $\bar{E}_X \subseteq \bar{E}_C^*$ and $\bar{E}_Z = \bar{E}_X \cup \bar{E}_Q \subseteq \bar{E}_C$ be the set of edits failed by y ; $\bar{E}_X \cap \bar{E}_Q = \phi$. Two cases arise.

Case 1 $\bar{E}_Q = \phi$

In this case $\bar{E}_X = \bar{E}_Z$ and (3.3b) is clearly true.

Case 2 $\bar{E}_q \neq \phi$.

For each $E_g \in \bar{E}_g$, there exists $E_B \in \bar{E}_x$ such that $E_g \subset E_B$, since $\bar{E}_C^*$ is the non-redundant subset of $\bar{E}_C$. Also the FEM row corresponding to E_g , $a_g^T \geq a_B^T$, the FEM row corresponding to E_B . Thus any cover of a_B^T is automatically a cover of a_g^T and the SCP solution with a_g^T excluded from the FEM is the same as the SCP solution with a_g^T included. Thus $\bar{E}_C^*$ is sufficient for $\bar{E}_C$.

The theory developed thus far paves the way for a procedure to generate $\bar{E}_S^*$, more simply than $\bar{E}_C$, and hence solve MWFI. Such a procedure was first proposed in Liepins [21].

II. An Algorithm for the Generation of a Sufficient Collection of Edits

All possible combinations of generating fields, namely 1; 1,2; 1,2,3; ...; 1,2,...,N; 2,3; 2,3,4; ...; 2,3,...,N,...; N can be organized as a forest of n-ary (multiway) trees designated the Field Code Forest (FCF). A tree in FCF will be known as a Field Code Tree (FCT). Consider an FCF on 4 fields.

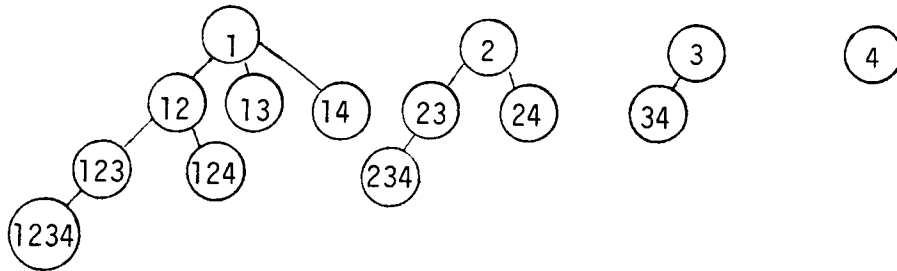


Figure 3.1 Example of an FCF

The four FCT's in the FCF have, in turn, fields 1, 2, 3 and 4 as their respective roots. Systematic traversal of the FCT's in turn, and hence the FCF, using field 1 as the generating field then

fields 2, 3 and 4 in turn as generating fields (corresponds to nodes (1,2), (1,3), and (1,4)) etc., will clearly result in $\bar{E}_S^*$ being generated (Corollary 3.1.2) after the fourth FCT has been traversed. The tree traversal strategy will be the following modification of pre-order traversal of a binary tree (see Knuth [19]).

Modified Pre-order-traversal for an n-ary tree

1. Visit the root
2. Traverse the left subtree in modified pre-order
3. Traverse the right subtree, until there are no more right subtrees, in modified pre-order.

For example, the FCT with root corresponding to field 1 will be traversed in the following order: 1, (1,2), (1,2,3), (1,2,3,4), (1,2,4), (1,3), (1,4). A refinement of this idea is included in the following algorithm for the generation of $\bar{E}_S^*$. Before specifying the algorithm some more terminology is introduced.

Definition The field code matrix (FCM), F^k , corresponding to field k and the edits $E_1, E_2, \dots, E_m$ is defined to be the boolean matrix given by:

$$F_{ij}^k = \begin{cases} 1 & \text{if edit } i \text{ includes the } j\text{th entry in field } k. \\ 0 & \text{otherwise} \end{cases} \quad \begin{matrix} i=1,2,\dots,m \\ j=1,2,\dots,|A_k|. \end{matrix}$$

Example 3.3 Consider a set of 3 edits:

$$E_1 = A_1 \times \{2\} \times \{1,3\}$$

$$E_2 = \{2,3\} \times \{A_2\} \times \{2\}$$

$$E_3 = \{1,3\} \times \{1\} \times \{1\}$$

with $A_1 = \{1,2,3,4\}$, $A_2 = \{1,2\}$, $A_3 = \{1,2,3\}$

		Entries in field 1			
Edits		1	2	3	4
	$F^1 =$				
	E_1	1	1	1	1
	E_2	0	1	1	0
	E_3	1	0	1	0

$F_{33}^1 = 1$, because $A_{31} = \{1,3\}$ which includes the third entry of field 1. We are now in a position to formally state the algorithm. An example will be given before giving formal, stepwise, justification of the algorithm.

Algorithm 3.1

0. Form an FCF corresponding to the N fields. Traversal of the trees in the FCF, i.e. generation on various field combination sequences - will have the following interpretation.
 - (a) When at the root node, i, of an FCT, use field i as the generating field only on edits which have field i entering (Rule 2.15a).
 - (b) When at a non root node, (ijk...h,q), of an FCT use field q as the generating field only on edits which have fields i,j,k,...h, uninvolved and field q entering.

The traversal of the FCT 's will be in modified pre-order.

The number of edits in $\bar{E}_e$ will be assumed, initially, to be m. $\bar{E}_M$ and $\bar{E}_S$ will denote, respectively, the sets of

edits from which new edits are to be generated at the roots of the FCT 's in the FCF and at non root nodes in an FCT. The node of the FCF currently under processing will be denoted by t .

1. Set $R = 0$; $\bar{E}_C^* = \bar{E}_e$; $\bar{E}_M = \bar{E}_e$.
2. Set $R = R+1$; If $R > N$ STOP. Otherwise set $t = \text{node } R$, denoted (R) , and set $\bar{E}_S = \bar{E}_M$.
3. If $t = (R)$ then do (3a); otherwise do (3b).

(3a) Find all prime cover solutions to the problem

$$x^T \bar{F}^R \geq 1$$

$$x_j = 0, 1, \quad j = 1, 2, \dots, g \quad (3.15)$$

where $\bar{F}^R$ is the FCM corresponding to the g edits in $\bar{E}_S$ with field R entering. Go to Step (4a).

- (3b) If $t = (i, j, k, \dots, h, q)$, then find all prime cover solutions to

$$x^T \bar{F}_{i, j, k, \dots, h}^q \geq 1$$

$$x_j = 0, 1, \quad j = 1, 2, \dots, f \quad (3.16)$$

where $\bar{F}_{i, j, k, \dots, h}^q$ is the FCM corresponding to the f edits in $\bar{E}_S$ with field q entering and fields $i, j, k, \dots, h$ uninvolved. Go to Step (4b).

- (4a) Set $\bar{E}_{Aux} = \bar{E}_S$. For each solution x generate $E = E(R, \bar{E}_x)$, $\bar{E}_x \subseteq \bar{E}_{Aux}$, and perform 4c. Go to Step 5.
- (4b) Set $\bar{E}_{Aux} = \bar{E}_S$. For each solution x generate $E = E(q, \bar{E}_x)$, $\bar{E}_x \subseteq \bar{E}_{Aux}$, and perform 4c. Go to Step 5.
- (4c) Update the $\bar{E}_M$, $\bar{E}_S$ and $\bar{E}_C^*$ sets of edit, if necessary. That is, remove from $\bar{E}_M$ each E_z such that $E_z \subset E$,

and set $\bar{E}_M = \bar{E}_M \cup E$, if there exists no $E_Z \in \bar{E}_M$ such that $E \subseteq E_Z$;

Remove from $\bar{E}_S$ each E_Z such that $E_Z \subset E$, and set $\bar{E}_S = \bar{E}_S \cup E$, if there exists no $E_Z \in \bar{E}_S$ such that $E \subseteq E_Z$;

Remove from $\bar{E}_C^*$ each E_Z such that $E_Z \subset E$, and set $\bar{E}_C^* = \bar{E}_C^* \cup E$, if there exists no $E_Z \in \bar{E}_C^*$ such that $E \subseteq E_Z$.

5. If no E , generated in Step 4, was included in $\bar{E}_S$ go to Step 6; otherwise reset t to next node in MPT and perform Step 3.
6. Terminate traversal of current subtree. If no other node exists in the modified pre-order traversal, STOP. If the next node to be visited in the traversal is the root of the next FCT then go to Step 2; otherwise reset t to the next node of the next subtree in MPT. Go to Step 3.

Example 3.4 The edits $\bar{E}_e$ are

$$E_1 = A_1 \times \{0,1\} \times \{0\} \times A_4 \times \{0,1\} \times A_6$$

$$E_2 = \{1\} \times A_2 \times \{1\} \times \{0,1\} \times A_5 \times \{2,3\}$$

$$E_3 = \{0\} \times \{1,2\} \times A_3 \times \{1,2,3\} \times A_5 \times A_6$$

$$E_4 = A_1 \times \{0,2\} \times A_3 \times A_4 \times A_5 \times \{0,1\}$$

$$E_5 = \{1\} \times A_2 \times A_3 \times \{0\} \times \{1,2\} \times A_6$$

$$A_1 = \{0,1\}, A_2 = \{0,1,2\} \times A_3 = \{0,1\}, A_4 = \{0,1,2,3\},$$

$$A_5 = \{0,1,2\}, A_6 = \{0,1,2,3\}$$

The FCF is shown in Figure 3.2.

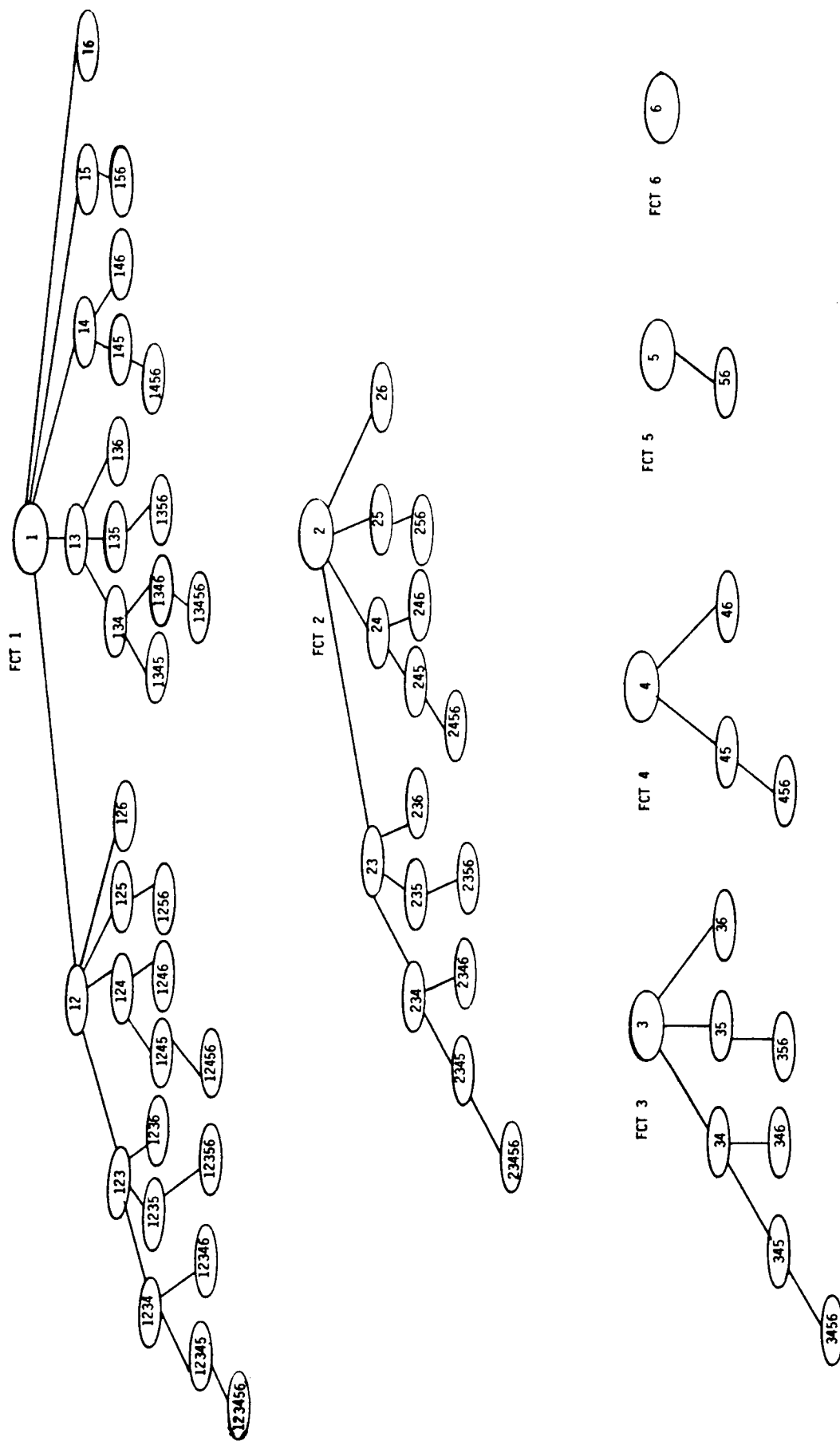


Figure 3.2 FCF on 6 Fields

Table 3.1 is a table of implied edits generated indicating in addition the edits involved in their generation and the node at which they were generated.

TABLE 3.1

Implied Edits Generated Using Algorithm 3.1 in Example 3.4

Node	Implied Edit	Generating Edits
(1)	$E_6 = A_1 \times \{1,2\} \times \{1\} \times \{1\} \times A_5 \times \{2,3\}$	E_2, E_3
(12)	$E_7 = A_1 \times A_2 \times \{0\} \times A_4 \times \{0,1\} \times \{0,1\}$	E_1, E_4
(123)	$E_8 = A_1 \times \{1\} \times A_3 \times \{1\} \times \{0,1\} \times \{2,3\}$	E_1, E_6
(16)	$E_9 = A_1 \times \{2\} \times \{1\} \times \{1\} \times A_5 \times A_6$	E_4, E_6
(2)	$E_{10} = \{0\} \times A_2 \times \{0\} \times \{1,2,3\} \times \{0,1\} \times A_6$	E_1, E_3
(2)	$E_{11} = \{0\} \times A_2 \times A_3 \times \{1,2,3\} \times A_5 \times \{0,1\}$	E_3, E_4
(25)	$E_{12} = \{1\} \times A_2 \times \{0\} \times \{0\} \times A_5 \times \{0,1\}$	E_5, E_7
(3)	$E_{13} = \{1\} \times \{0,1\} \times A_3 \times \{0,1\} \times \{0,1\} \times \{2,3\}$	E_1, E_2
(35)	$E_{14} = \{1\} \times \{0,1\} \times A_3 \times \{0\} \times A_5 \times \{2,3\}$	E_5, E_{13}
(5)	$E_{15} = \{1\} \times \{0,1\} \times \{0\} \times \{0\} \times A_5 \times A_6$	E_1, E_5
(56)	$E_{16} = \{1\} \times \{0,2\} \times \{1\} \times \{0,1\} \times A_5 \times A_6$	E_2, E_4
(56)	$E_{17} = \{1\} \times \{0\} \times A_3 \times \{0\} \times A_5 \times A_6$	E_4, E_{14}
(6)	$E_{18} = \{1\} \times \{0\} \times A_3 \times \{0,1\} \times \{0,1\} \times A_6$	E_4, E_{13}

The sequence in Table 3.1 is the sequence of generation of edits $E_6, E_7, \dots, E_{18}$. Steps 3a, 4a and 4c of the algorithm account for the generation of E_6 using edits E_2 and E_3 and the first FCT.

Traversal of that tree results in the generation of E_7 at (12) using steps 3b, 4b and 4c. No new essentially new edit can be generated using node (123) and hence the rest of the subtree, namely, subtrees with roots (1234), (12345) and (123456) can be pruned (steps 5 and 6). MPT of the remainder of the FCT results in the pruning of subtrees with roots (124), (125) and (126) before node (13) yields the new edit E_8 . No new edit is generated at (134), (135), (136), (14), and (15) and the subtrees with these nodes as roots are pruned. The node (16) results in E_9 being generated and since no other new edits are generatable from FCT 1, MPT of FCT 2 is performed and so on. In this particular instance E_M remains unchanged throughout the generation procedure and $E_C^* = \bar{E}_e \cup \{\text{the implied edits in Table 3.1}\}$.

To see how Algorithm 3.2 produces a sufficient collection $\bar{E}_C^*$ of edits from an initial set $\bar{E}_e$, let us examine the steps of the algorithm more closely.

1. Traversal of the FCF is in MPT; i.e. all combinations of generating field sequences are accommodated if there is no pruning of the tree. Thus one pass through the FCF is sufficient for generating $\bar{E}_C$ (Corollary 3.1.1).
2. Finding all prime cover solutions to (3.15) and (3.16) will identify subsets of $\bar{E}_S$ for generation of essentially new implied edits only with field $R(q)$ uninvolved. To see why we need not generate any other implied edits, consider, for the sake of simplicity, a node such as (3). If a particular subset $\bar{E} \subseteq \bar{E}_S$

of edits does not result in an essentially new implied edit on generation on field 3 but results in a simple implied edit, E_I , then the same edit set combination on field (3) cannot produce an edit with field 3 uninvolved. This is because (3) is the last node of the FCF which generates on field 3. The non inclusion of E_I in $\bar{E}_S$, $\bar{E}_M$ or $\bar{E}_C^*$ can be justified as follows: for all $E_k \in \bar{E}$, $A_{Ij} \subseteq A_{kj}$, $j \neq 3$, and for generating a new edit with $j(\neq 3)$ uninvolved, E_I is not necessary; it suffices to have $\bar{E}$.

Similar is the argument for not retaining edits which are not essentially new when generating at a node such as (123); since edits with fields 1 and 2 uninvolved would merely be a subset of $\bar{E}_M$ when generating at a later node (3).

3. The non-retention of dominated edits, in Step 4c, as part of $\bar{E}_M$, $\bar{E}_S$ for future implied edit generation is justified by Lemma 3.1. The non-redundancy of $\bar{E}_C^*$ together with item 1 leads to the conclusion that $\bar{E}_C^*$ generated by the algorithm is sufficient for the F-H complete set $\bar{E}_C$ (Corollary 3.1.2).
 4. Pruning of the tree is as follows. When at the root (j) of an FCT, if no new essentially new implied edits are generated then the FCT may be pruned. For, the rest of the nodes in the FCT do not use j as the generating field and cannot produce an essentially new implied edit, for inclusion into $\bar{E}_C^*$, with j uninvolved. Also any essentially new implied edit that can be produced at a node such as (jg) can also be produced at (g), ($g > j$).
- Computational performance of Algorithm 3.1 is taken up in Chapter 4.

IV. A Row Generation Algorithm for Solving MWFI

The F-H procedure as also Algorithm 3.1 leads to the solution of MWFI by generating the complete set of edits, $\bar{E}_c$ or its surrogate, $\bar{E}_c^*$. Both procedures guarantee that a solution to the SCP (2.26) will result in a passing record. Furthermore the generation of $\bar{E}_c$ or $\bar{E}_c^*$ is not record-specific. However, $\bar{E}_c$ ($\bar{E}_c^*$) anticipate the implied edits that will be failed by each failing record. However, not all (essentially new) implied edits failed by a record may be needed to solve MWFI. Further, the computational expenditure in arriving at $\bar{E}_c$ ($\bar{E}_c^*$) can be prohibitive (see section 4.2). A solution procedure specific to a given failing record would be more desirable under the circumstances. Such an approach was proposed in Garfinkel [12] for the solution of MWFI which is record-specific and which generates essentially new implied edits as and when necessary. This approach is formally presented here as Algorithm 3.2 with notation consistent with the rest of the chapter.

Let y be the faulty record under consideration.

Algorithm 3.2

1. Test if $y \in \bar{E}_e$. If not, STOP; if so, set $\bar{E}_y = \bar{E}_e$.
2. Solve the SCP (2.26) over the edits of $\bar{E}_y$. Denote the optimal solution to the SCP by w^* .
3. Let $J^* = \{j | w_j^* = 1\}$

Fix the values of $\hat{y}_j$, $j \notin J^*$, at y_j , but for every $j \in J^*$, let $\hat{y}_j$ assume each of the values in A_j . Test each of the $\prod_{j \in J^*} |A_j|$ possible records so defined. If any of these records is not in $\bar{E}_e$, w^* is optimal for MWFI and we STOP. If not, go to Step 4.

4. Find any prime cover to

60

$$\begin{aligned} vQ &\geq 1 \\ v &= 0,1 \end{aligned} \quad (3.17)$$

where $Q = (q_{ik})$ and

$$q_{ik} = \begin{cases} 1 & \text{if edit } i \text{ is failed by the } k\text{th record } y \\ & \text{of step 3} \\ 0 & \text{otherwise} \end{cases}$$

Denote the solution to (3.17) by v^0 and let $I^0 = \{i \mid v_i^0 = 1\}$.

5. Generate the "implied edit" E_{v^0} where E_{v^0} has entries given by

$$A_{v^0 j} = \begin{cases} \bigcap_{i \in I^0} A_{ij} & , j \notin J^* \\ A_j & , j \in J^* \end{cases} \quad (3.18)$$

Set $\bar{E}_y = \bar{E}_y \cup E_{v^0}$ and go to Step 2.

Before formally justifying the algorithm it is useful to look at an example.

Example 3.5 Let the record $y = (1,0,0,0,1,0)$. Also let

$c = (1, \dots, 1)$ and let the edit system $\bar{E}_e$ be as in Example 4.

Step 1: $y \in \bar{E}_e$; specifically $y \in E_1, E_4$ and E_5 .

Step 2: The SCP is

$$\begin{aligned} \min \quad & w_1 + w_2 + w_3 + w_4 + w_5 + w_6 \\ & w_2 + w_3 + w_5 \geq 1 \\ & w_2 + w_6 \geq 1 \\ & w_1 + w_4 + w_5 \geq 1 \\ & w_j = 0,1, \quad j = 1, \dots, 6 \end{aligned}$$

with $w_2^* = w_5^* = 1$ as one solution.

Step 3: $J^* = \{2,5\}$. There are nine records which can be generated and all of them fail one or more edits.

TABLE 3.2

Step 3 of Algorithm 3.2 in Example 3.5

Record	Edits Failed
100000	E_1, E_4
100010	E_1, E_4, E_5
100020	E_4, E_5
110000	E_1
110010	E_1, E_5
110020	E_5
120000	E_4
120010	E_4, E_5
120020	E_4, E_5

Step 4: The only prime cover to:

$$v_1 + v_4 \geq 1$$

$$v_1 + v_4 + v_5 \geq 1$$

$$v_4 + v_5 \geq 1$$

$$v_1 \geq 1$$

$$v_1 + v_5 \geq 1$$

$$v_5 \geq 1$$

$$v_4 \geq 1$$

(3.19)

$$v_j = 0, 1, \quad j = 1, \dots, 5 \quad \text{is } I^0 = \{1, 4, 5\}.$$

Step 5: The implied edit is

$$E_{V^0} = \begin{cases} \bigcap_{i \in J^*} A_{pi} & , \quad p \in I^0 \\ A_i & , \quad \text{otherwise} \end{cases}$$

$$E_{V0} = \{1\} \times A_2 \times \{0\} \times \{0\} \times A_5 \times \{0,1\} \quad (3.20)$$

with corresponding constraint $w_1 + w_3 + w_4 + w_6 \geq 1$ to be added to the SCP (2.26).

Step 2: A solution to the changed SCP is $w_5^* = w_6^* = 1$.

Step 3: $J^* = \{5,6\}$. There are 12 failing records.

Step 4: $I^0 = \{1,5\}$ is an index to the set of edits in $\bar{E}_e$ at least one of which is failed by each of the 12 failing records.

$$\text{Step 5: } E_{V0} = \{1\} \times \{0,1\} \times \{0\} \times \{0\} \times A_5 \times A_6 \quad (3.21)$$

and the associated SCP constraint to be added in Step 2 is

$$w_1 + w_2 + w_3 + w_4 \geq 1.$$

Step 2: A solution to the modified SCP is $w_1^* = w_2^* = 1$

Step 3: $J^* = \{1,2\}$. There are six failing records.

Step 4: $I^0 = \{1,4\}$ is a prime cover to the associated problem (3.17).

$$\text{Step 5: } E_{V0} = A_1 \times A_2 \times \{0\} \times A_4 \times \{0,1\} \times \{0,1\} \quad (3.22)$$

with associated constraint $w_3 + w_5 + w_6 \geq 1$.

Step 2: A solution to the changed SCP is $w_1^* = w_2^* = w_3^* = 1$

Step 3: A feasible and therefore optimal record, is

$$\hat{y} = (0,1,1,0,1,0).$$

It is clear that if all possible failing records, given an edit set $\bar{E}_e$, are processed through Algorithm 3.2, a sufficient set for $\bar{E}_c$ would result. In fact, a comparison with Table 3.1, of the edits (3.20), (3.21) and (3.22) reveals that they are the same as E_{12} , E_{15} and E_7 .

Justification for the validity of Algorithm 3.2 is based on the following lemmas.

Lemma 3.2

The implied edit generated in Step 5 of Algorithm 3.2 must be failed by the given record y .

Proof Assume, w.l.o.g., that $J^* = \{j | j=1, \dots, k\}$, $k \leq n$ is the set of generating fields used to construct $E_{w^*}(v^0)$ in (3.21). Since I^0 is the index set of the edits in a prime cover of all failing records generated in Step 3, at least one of the edits indexed by I^0 is failed by all records $\hat{y}$ generated in Step 3. This implies that

$$A_{v^0 j} = A_j, j \in J^*, \quad (3.23)$$

for otherwise, a feasible record, $\hat{y}$, would have been found in Step 3. Since all the records generated in Step 3 have the same entries in fields $i = k+1, \dots, N$, as the original record y , and since the edits indexed in I^0 form a (prime) cover of all these records, $A_{v^0 i} = \bigcap_{j \in J^*} A_{ji}$, $i = k+1, \dots, N$ must be such that y_i (and hence $\hat{y}_i$ for all records generated in Step 3) $\in A_{v^0 i}$, $i = k+1, \dots, N$. Together with (3.26) we then have $y \in E_{w^*}(v^0)$.

Lemma 3.2 demonstrates the validity of the cut to be added to the SCP in Step 2. The construction of E_{v^0} in Step 5 of the algorithm calls for simultaneous generation on k fields, $k > 1$. It was shown in Example 2.8 that simultaneous generation on more than one field will not always lead to a valid implied edit. The proof of Lemma 3.2 demonstrates, however, that generation on k fields, $k > 1$, will lead to a valid implied edit if a set of edits at least one of which is failed by each record generated by changing the k fields used. It

remains to be demonstrated that the solution obtained from Algorithm 3.2 is indeed the MWFI solution.

Lemma 3.3

Algorithm 3.2 solves MWFI.

Proof Assume the contrary and that x is a solution to MWFI with $c_x < c_w$, where w is the solution, and c_w its associated cost, obtained using Algorithm 3.2. Clearly, w leads to a feasible solution in that it leads to a passing record. The proof of Lemma 3.2 demonstrates that $E_{w*}(v^0)$ is a valid implied edit. The solution to MWFI is the solution to the SCP (2.26) using $\bar{E}_c$, the complete set of edits, where $\bar{E}_c = \bar{E}_e \cup \{\text{set of all essentially new implied edits}\}$. Since $E_{w*}(v^0)$ is a valid essentially new implied edit, $E_{w*}(v^0) \in \bar{E}_c$ and at the termination of Algorithm 3.2, $\bar{E}_y \subseteq \bar{E}_c$ and hence it must be true that $c_w \leq c_x$ contrary to our assumption.

QED.

Algorithm 3.2 systematically searches through all available cover solutions to the SCP in Step 2. If there is a solution to MWFI it must belong in this set of covers. If the number of fields, N , is finite so will be Algorithm 3.2. If the MWFI problem is unsatisfiable then an edit will be generated in (3.18) which will not be satisfiable in that it will lead to a SCP constraint of the form $0 \geq 1$. Computational performance of Algorithm 3.2 and its comparison with the F-H algorithm and Algorithm 3.1 is the subject of Chapter 4.

CHAPTER 4

ANALYSIS AND COMPARISON OF THE COMPUTATIONAL PERFORMANCE
OF SOME ALGORITHMS FOR SOLVING MWFI

In Chapter 3 two algorithms for solving MWFI with coded data were presented. As was mentioned in Chapter 1, the satisfiability problem, see Garey [11], is a special case of MWFI and MWFI is NP-hard. This background should be kept in mind while assessing the computational performance of Algorithms 3.1 and 3.2 being reported in this chapter. An analysis of the two algorithms precedes the presentation of the computational results in Section 4.2. The computational details, not germane to the discussions in Sections 4.1 and 4.2, are placed in Section 4.3.

I. An Analysis of Algorithms 3.1 and 3.2

It is useful to critically examine the F-H algorithm of Section 2.II before its refinement, namely Algorithm 3.1, and the row generation approach in Algorithm 3.2 are examined. The F-H procedure, with rules 1, 2, and 3 incorporated, requires the generation of implied edits using a generating sequence of fields not only in various combinations but also in all permutations of the sequence. Under the F-H procedure, implied edit generation successively on fields 1, 2 and 3 translates to generation on all 6 different permuted sequences with fields 1, 2 and 3. In terms of the effort involved, in comparison to the tree traversal procedure in Algorithm 3.1, each FCT of the FCF set up in Algorithm 3.1 would have to be processed a multiplicity of times corresponding to the various

permutations of the generating fields sequence. Further, even with rules 1, 2 and 3 included in the F-H procedure, edits which are not essentially new are carried along for future processing and thus the set of edits from which new implied edits are to be generated grows very rapidly. It is useful to note here that traversal of the FCF just once without generating edits on permuted sequences on generating fields, but carrying essentially new as well as other implied edits along, while attempting to construct a complete set of edits from the edit set in Example 3.4 took up 1 minute of CPU time on an IBM 3031 computer. Given that Example 3.4 had an explicit edit set of only 5 edits on just 6 fields, the F-H procedure for generation of a complete set of edits from large explicit edit sets is clearly very costly to adopt.

Algorithm 3.1 used for generating a collection of edits sufficient for the complete set of edits in Example 3.4 returned a sufficient set in 12 seconds of CPU time on the same computer. By definition, a set of edits $\bar{E}_S$, sufficient for a complete set of edits $\bar{E}_C$, is adequate for solving MWFI. Further, Theorem 3.1 and its corollaries demonstrated that generation of edits on permuted field sequences is unnecessary for generating $\bar{E}_S$. Algorithm 3.1, which is based on Theorem 3.1 and its corollaries will clearly involve much less computational effort for solving MWFI than the F-H procedure. However, Algorithm 3.1 in itself can be prohibitively expensive. In an MWFI problem with N fields an FCT, with root node $j \in I_N$, has 2^{N-j} nodes. The total number of nodes in the FCF is then $\sum_{j=1}^N 2^{N-j} = 2^N - 1$, and for a problem with 20 fields there are 1,048,575 nodes in the associated FCF! While Algorithm 3.1 specifies a procedure for pruning the FCT's, in the worst case all 1,048,575 nodes will have to

be visited. An edit set $\bar{E}$, obtained by stopping Algorithm 3.1 short of its normal termination, may not be sufficient for solving MWFI. This observation is crucial in the sense that computational effort expended in using Algorithm 3.1 could be substantially wasted if the algorithm is terminated prematurely.

To the computationally expensive problem of using a very large number of combinations of fields in the generating sequence (i.e. visiting a very large number of nodes in the FCF), is added the problem of finding all prime cover solutions to (3.15) or (3.16) at each node of the FCF, in Algorithm 3.1. Finding a prime cover is computationally quite inexpensive. But in a field code matrix (FCM) with k edits, and n_j entries in the field j , currently being processed, there are potentially 2^k prime covers (of the n_j entries by the k edits). Problem (3.15) is likely to be much more expensive to solve computationally than (3.16). For instance, a node such as (1234) uses only those s edits, say, which have fields 1, 2 and 3 uninvolved and field 4 entering, while a root node such as (4) uses all r edits, say, with field 4 entering and $s \leq r$. There is a parallel between step 4 of Algorithm 3.1 and the implied edit generation procedure in steps 4 and 5 of Algorithm 3.2. Algorithm 3.1, in step 4 and at nodes (q) and $(i, \dots, h, q)$, anticipates the necessity for construction of an essentially new implied edit for those failing records which cannot be made passing by changing fields $i, \dots, h$ alone without changing field q . Algorithm 3.2, in steps 3, 4 and 5, does precisely the same thing, except that it is for a single specified record at a time.

Algorithm 3.1 does not specify a solution to MWFI. It merely generates a sufficient set $\bar{E}_C^*$ which when used to set up the SCP (2.26) will result in the solution to (2.26) solving MWFI. The SCP is in itself NP-hard (see Garey [11]) and while it is one of the more 'easily' solved integer programming problems, it is by no means trivial. The problem of finding all prime cover solutions to (3.15) or (3.16) is at least as hard as an SCP in the sense that SCP is just a special case of this problem. For, if all prime covers of a set are known then the SCP is solved while it is unnecessary to know all prime covers to solve SCP. Thus the use of Algorithm 3.1 to solve MWFI involves the solution of two difficult computational problems, namely, the problem of solving, in the worst case, 2^{N-1} problems of the type (3.15) or (3.16) and the solution of the SCP (2.26). Computational performance of Algorithm 3.1 is reported in Section 4.2, and understandably the problems solved are not very large.

Algorithm 3.2 also involves the solution of SCP's such as (2.9). The number of edits, original and implied, included in formulating the SCP (2.9) using Algorithm 3.2, will usually be less than the number of edits from $\bar{E}_C^*$ included in setting up (2.26). But, if it becomes necessary for Algorithm 3.2 to generate all the implied edits in $\bar{E}_C^*$ failed by a given record x before MWFI can be solved, then a series of SCP's (2.26) would have to be solved, the last of which will have the same (effective) size as (2.26) set up with $\bar{E}_C^*$. In many data validation situations it would be undesirable to change a large proportion of fields in arriving at a passing record. In using Algorithm 3.2 there exists the option of discarding the failing record y whenever a large proportion (say 20% or

more) of the fields have been changed without arriving at a passing record. If the cost of changing the fields is non-uniform a cost cut off can be enforced, in using Algorithm 3.2, beyond which it would be unacceptable to retain the record. A similar provision can be used in solving MWFI using a sufficient collection of edits produced by Algorithm 3.1 (by enforcing a cut off cost beyond which solution of (2.26) is terminated). Such a cut off was enforced in the solution of problems reported on in Section 4.2.

A trade off between using Algorithm 3.2 or (2.26) with $\bar{E}_C^*$, for solving MWFI, is the solution of a series of 'small' SCP's the last of which results in MWFI being solved versus the solution of one 'bigger' SCP (2.26) whose solution solves MWFI. As reported in section 4.II the trade off, at least for the problems examined, appears to be in favor of Algorithm 3.2. However, it is conceivable that all the implied edits in $\bar{E}_C^*$ failed by a record y may have to be generated before MWFI is solved and given the computational complexity of SCP's, such an approach could be extremely expensive. It should be noted at this point that the edits generated in Algorithm 3.2, while processing various failing records, can be maintained as part of $\bar{E}_e$ in Step 1 of Algorithm 3.2 and eventually a set of edits $\bar{E}_e$ sufficient for $\bar{E}_C$ would have been generated resulting in the Algorithm solving MWFI in exactly one iteration for all other failing records.

Algorithm 3.2 further requires the enumeration of all records that can be constructed by changing the fields indexed by J^* (see step 3 of the Algorithm). If, for instance, $|J^*|=4$ and if each of the fields indexed by J^* has 20 permissible entries then 160,000 records need to

be generated. This in itself is a computationally expensive task. A further complication arises in having to store the information regarding the edits in $\bar{E}_e$ failed by these records in the matrix Q before the prime cover problem (3.17) can be solved in Step 4 of Algorithm 3.2. Two approaches are possible to alleviate the storage problem. The first one, presented as Device 4.1 below, attempts to reduce the number of columns in Q that need to be maintained to solve (3.17).

Device 4.1

Let the set of records on which edit failure information is currently being maintained in Q be denoted by $\bar{R}$. If any record $x \in \bar{R}$ is such that a new record $y \notin \bar{R}$ fails at least the same edits as x then y may be discarded. For, any cover of x will also be a cover of y . Conversely any record $x \in \bar{R}$ such that x fails the same edits as y , and at least one more, may be discarded; y may then be included in $\bar{R}$ in place of all these records $x \in \bar{R}$.

The column included in Q (record included in $\bar{R}$) is said to dominate the column it causes to be discarded from Q (the record excluded from $\bar{R}$). This was the device used, for the most part, in obtaining the computational results relating to Algorithm 3.2 reported in Section 4.2. Since the problems on which computational experience is being reported were quite small the column size was only as large as $|\bar{E}_e|$ or less. To better appreciate the potentially massive column dimension that Q can take on, the following notation is introduced. If $m \geq n$, $\binom{m}{n}$ will indicate $m!/(m-n)!n!$ where $n!$ denotes factorial n . The largest integer less than or equal to α will be denoted by $\lfloor \alpha \rfloor$.

The matrix Q (see Device 4.1 and step 4 of Algorithm 3.2) has rows corresponding to edits and columns corresponding to (failing) records. The (i,j) th entry in Q , $q_{ij} = 1$ if the j th record fails the i th edit and $q_{ij} = 0$ otherwise. In using Device 4.1, we need to maintain in the matrix Q all undominated columns (i.e. columns corresponding to failing records that cannot be discarded using Device 4.1). If 4 edits are failed and each (failing) record fails 2 edits then there can be a maximum of $\binom{4}{2} = 6$ columns in Q , as Figure 4.1 demonstrates. There can be no more than 6 columns because the inclusion in Q^* of any column with

$$Q^* = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 1 & 0 & 1 \end{bmatrix}$$

Figure 4.1. The Matrix Q on 4 Rows with the Maximum Number of Undominated Columns.

less than 2 one entries in it results in the discarding of at least one column of Q^* (Device 4.1). Further, any column with more than 2 one entries in it cannot be included in Q^* because it will be dominated (see Device 4.1) by an existing column in Q^* . The maximum number of undominated columns in Q , with k non-zero rows, is in fact given by the maximum number of ways in which ones can be placed in the k rows so that no column is dominated. Lemma 4.1 shows that this maximum is achieved at the value corresponding to the largest binomial coefficient corresponding to the k rows.

Lemma 4.1

If there are m non-zero rows in Q , then the maximum number of undominated columns is $\binom{m}{\lfloor m/2 \rfloor}$.

Proof $\binom{m}{\lfloor m/2 \rfloor}$ undominated columns can be in Q if each column has exactly $\lfloor m/2 \rfloor$ ones. Any column with fewer than $\lfloor m/2 \rfloor$ (but greater than 0) ones in it dominates at least one column in Q . Further $\binom{m}{\lfloor m/2 \rfloor} > \binom{m}{\lfloor m/2 \rfloor - t}$, $t \geq 1$. Also, any column with more than $\lfloor m/2 \rfloor$ ones in it is dominated by at least one column in Q and $\binom{m}{\lfloor m/2 \rfloor} \geq \binom{m}{\lfloor m/2 \rfloor + t}$, $\lfloor m/2 \rfloor + t \leq m$, $t \geq 1$.

QED.

Thus if m is the number of edits failed by the h records $\hat{y}$ generated, in step 3 of Algorithm 3.2, then the maximum number of records in $\bar{R}$ is $h^* = \text{minimum} \{h, \binom{m}{\lfloor m/2 \rfloor}\}$. In actual practice the column size of Q would be much less than h^* . But, clearly, the potential for explosive growth in the column dimension of Q necessitates the specification of a procedure for solving (3.17) if h^* is extremely large. Such a procedure is specified as Device 4.2.

Device 4.2

Maintain a set of edits, $L(E)$, failed by records $\hat{y}$, generated in Step 3, at any stage of processing in the step. Initially $L(E) = \phi$. If a new record $\hat{y}$ fails an edit $E_t \notin L(E)$ and fails no edit in $L(E)$ then set $L(E) = E_t \cup L(E)$. The edits existing in $L(E)$ cannot cover this new record without E_t added. If $|L(E)| = |\bar{E}_e|$ then no more testing against $L(E)$ of the new records, $\hat{y}$, need be done, since no other edits exist to cover any of the records $\hat{y}$. Use the edits in $L(E)$ to generate the essentially new implied edit (3.18). (Note that $L(E)$ may not correspond to a prime cover of the columns of Q .)

Example 4.1 Consider the situation in Example 3.5 and Table 3.2 which is reproduced here in Figure 4.2 for ready reference.

Record #	Record	Edits Failed
1	100000	E_1, E_4
2	100010	E_1, E_4, E_5
3	100020	E_4, E_5
4	110000	E_1
5	110010	E_1, E_5
6	110020	E_5
7	120000	E_4
8	120010	E_4, E_5
9	120020	E_4, E_5

Figure 4.2 Records Generated in Step 3 of Example 3.5

Initially $L(E) = \phi$. Record 1 fails edits E_1 and E_4 . Since neither edit is in $L(E)$ choose E_1 , arbitrarily, for inclusion in $L(E)$. Record 2 fails edits E_1 , E_4 , and E_5 . Since E_1 , already included in $L(E)$, covers record 2, E_4 or E_5 need not be included in $L(E)$ at this stage. Record 3 fails edits E_4 and E_5 neither of which is in $L(E)$. Arbitrarily choose E_4 for inclusion in $L(E)$. Records 4 and 5 are covered by edits in $L(E)$. E_5 is required to cover record 6 and hence E_5 is included in $L(E)$. At this stage $L(E) = \{E_1, E_4, E_5\}$. Records 7, 8 and 9 are all covered by the edits in $L(E)$. The implied edit generated using the edits in $L(E)$ (which happen to be a prime cover of the 9 failing records) and fields 2 and 5 as generating fields is exactly the same as the edit in (3.20).

It should be noted here that the implied edit E_{V0} generated using (3.18) is valid, i.e. if y is the record being processed, then $y \in E_{V0}$ as long as a cover to (3.17) is used to arrive at the index I^0 of the edits participating in the generation of E_{V0} . A prime cover solution to (3.17) is unnecessary for generating E_{V0} so that $y \in E_{V0}$, as can be seen from the proof of Lemma 3.2. The edits in $L(E)$ represent a cover of the failing records generated in step 3 of Algorithm 3.2. Thus the edits in $L(E)$ can be used to generate the (valid) edit E_{V0} . The difference between using Device 4.2 as opposed to Device 4.1 could be the generation of an essentially new implied edit E_I , with fields indexed by J^* uninvolved, which is dominated by E_I generated using Device 4.1. This could lead to more SCP's being solved in step 2 than is absolutely necessary. The trade off, then, is between using a tremendous amount of storage (may involve using auxiliary storage) as opposed to increasing, somewhat, the execution time.

II. Computational Performance of Algorithms 3.1 and 3.2

Algorithm 3.2 requires solution of SCP's and solution of MWFI using Algorithm 3.1 also involves the solution of SCP's. The algorithm for solving SCP's used to test the performance of Algorithms 3.1 and 3.2 is the cutting plane algorithm due to Bellmore and Ratliff [1] (see Garfinkel [13] for a concise statement of the algorithm). The computations were performed on an IBM 3031 computer at the University of Tennessee. The programming language used was FORTRAN and the programs were compiled using the IBM FORTRAN optimizing compiler. The executions of the programs were timed using the TIMETK routine supported by the UTCC. The timings are accurate to 16 milliseconds.

In testing the algorithm, it was felt that Algorithm 3.2 would be unduly favored if edits with a large proportion of fields entering were used. For, in such cases it would be 'easy' to find a passing record given a set of fields to change. This premise was upheld by the results of early computation and the density of uninvolved fields in each edit was raised to fairly high levels in subsequent testing.

The edits and records as well as the problem sizes, i.e. number of edits, number of fields, and the sum total of the permissible entries in each field (the number of 'data points') were generated using a uniform random number generator. The computational performance of Algorithms 3.1 and 3.2 are contained in Table 4.1. The following notation is used in Table 4.1.

- M : The number of edits
- N : The number of data points
- K : The number of fields
- AT1 : Average time, in seconds, per record processed using
Algorithm 3.1
- MXT1 : Maximum time, in seconds, for a record processed using
Algorithm 3.1
- AVIMP : Average number of implied edits failed in using $\bar{E}_C^*$ from
Algorithm 3.1 to set up and solve (2.26)
- MXIMP : Maximum number of implied edits failed in using $\bar{E}_C^*$ from
Algorithm 3.1 for solving (2.26)
- SCT : Time, in seconds, for generating a sufficient set for $\bar{E}_C$, $\bar{E}_C^*$,
using Algorithm 3.1

- AT2 : Average time per record processed, in seconds, using Algorithm 3.2
- MXT2 : Maximum time for a processed record, Algorithm 3.2
- AVITR : Average number of iterations required to process a record using Algorithm 3.2. (If processing was completed without encountering Step 5 then the number of iterations is considered 0.)
- MXITR : Maximum number of iterations using Algorithm 3.2 to process a record

One hundred failing records per (M,N,K) specification were randomly generated for processing problems with $N \leq 100$. The minimum proportion of uninvolved fields in any given edit generated will be denoted by UFD. It was ensured that at least 2 fields entered each edit generated. The size of each field was also a parameter denoted by MFS corresponding to the maximum number of permissible entries in each field. Note that as MFS increases the probability of locating a passing record increases. The costs for changing the fields were also generated at random using a uniform random number generator in the range 0.1 to 50. Any failing record which required changing of 40% (six) or more of its fields was discarded and computational performance on such records is not included in Table 4.1. However, it is possible that some of the problems in Table 4.1 were solved with very high cost changes, even though fewer than 40% (or six) fields were changed. Given that the number of iterations taken by Algorithm 3.2 is small in all cases it would seem that few prime covers by fields of the rows of the FEM associated with the failing record were available in almost all cases. The cut off for discarding failed records was specified in terms of fields (at an intuitively

A Comparison of the Computational Performance of Algorithms 3.1 and 3.2

77

TABLE 4.1 (continued)

S.NO	M	N	K	ATI**	MXT1	AVIMP	MXIMP	SCT	AT2	MXT2	AVTR	MXTR
16	22	82	9	3.7	.055	2.57	8	366*	1.18	11.7	.07	1
17	15	44	3	.032	.023	3.43	4	1.02	.053	.082	.01	1
18	6	68	13	.027	.032	1.3	3	.5	.022	.03	0	0
19	27	79	7	2.14	.19	6.5	14	208*	.56	1.4	.09	1
20	7	91	11	.026	.035	1.74	4	.314	.014	.048	0	0
Problem sets 16-20 had a UFD of .8 and an MFS of 20.												
21 [@]	22	500	64	-	-	-	-	-	35.8	37.2	0	0
22 [@]	91	740	92	-	-	-	-	-	1.89	2.44	0	0
23 [@]	18	497	66	-	-	-	-	-	4.94	7.5	.5	1

**ATI = (SCT + total time for solving SCP (2.26) for 100 records)/100, while MXT1 is merely the maximum time for solving SCP (2.26) for a given record.

*The number of edits in $\bar{E}^*$ exceeded 50, the maximum storage allowed for in the program. The program continued to process the FCF with the set of 50 edits. Pruning of the FCF's might have suffered on account of the non-inclusion of the essentially new edits generated beyond the cut off limit of 50. The performance of Algorithm 3.1 in these problems is included here only for comparison purposes.

- [@]Note: 1. The UFD for problem sets 21-23 was between .8 and .9.
 2. A sufficient collection $\bar{E}_C^*$ could not be generated, using Algorithm 3.1, in 30 minutes of CPU time.
 3. Only 2 records were processed per set of problems.

reasonable level) instead of a cost cut off specification since the costs were being generated at random. It should also be mentioned here that initial runs using uniform costs with both Algorithms 3.1 and 3.2 behaved very similarly to the performance pattern reported in Table 4.1.

It is apparent, in looking at MWFI problems that are quickly solved by both Algorithms 3.1 and 3.2, (problems 4,7,10,12,17,18,20) that if the time to generate a sufficient collection of edits is small, then Algorithm 3.1 is quite competitive with Algorithm 3.2. In fact, if the cost structure for changing fields is such that Algorithm 3.2 will need to iterate many times, Algorithm 3.1 would clearly be preferable. In the other problem instances Algorithm 3.2 performs much better than does Algorithm 3.1. A comparison of the number of implied edits failed in using Algorithm 3.1 to set up (2.26) and the number of iterations Algorithm 3.2 takes to process the record is quite revealing. The average number of iterations per record never exceeded 1 in using Algorithm 3.2 while at least one implied edit was failed, on an average, in every set up of (2.26) using $\bar{E}_C^*$ generated by Algorithm 3.1. In fact many implied edits are failed in problems 13, 15 and 19 on an average. This points to the presence of, perhaps, dominated or redundant constraints in the SCP (2.26) set up using $\bar{E}_C^*$ for a given failing record. There appears to be little correlation between the number of implied edits failed in the set up (2.26) and the number of iterations undertaken by Algorithm 3.2 for processing the same record.

The rapid growth in the size of the edit set using which $\bar{E}_C^*$ is to be obtained is exemplified by problem instances 2, 3, 8, 9, 14, 16, and 19. Problems 21 to 23 were such that no sufficient set of edits could

be obtained in 30 minutes of CPU time per problem. Also some of the records generated could not be processed using Algorithm 3.2 in 15 minutes of CPU time. The results in Table 4.1 corresponding to these problem instances are provided merely to highlight the fact that Algorithm 3.2 may be usable for specific records even when no sufficient set can be produced within a reasonable amount of CPU time.

Some computational considerations relevant to the use of Algorithms 3.1 and 3.2 in a production environment are pointed out in Section 4.3.

III. Computational Considerations in Coding Algorithms 3.1 and 3.2

Computational details relevant to performance of Algorithms 3.1 and 3.2 in a data processing environment are itemized below.

1. The generation of a sufficient set, using Algorithm 3.1, requires the maintenance of either (a) multiple copies of the edits in $\bar{E} = \bar{E}_S \cap \bar{E}_M \cap \bar{E}_C^*$ (see Algorithm 3.1), or (b) the maintenance and update of pointers to the edits in $\bar{E}$. The course adopted in estimating the computational performance of Algorithm 3.1 was (a). Procedure (b) is less expensive in terms of storage but is more expensive in terms of execution time at least for small problems. For large problems there may be no alternative to (b). For, even with the problems documented in section 4.II the amount of storage required for using Algorithm 3.1 was close to 400K bytes.
2. Since the edit matrices and the matrices for the SCP problems, in both Algorithm 3.1 and 3.2, are likely to be rather large in reality there may be no alternative to placing them on auxiliary storage.

This would considerably worsen the execution time performance for both procedures.

3. In all likelihood the edit matrices for MWFI situations will be sparse. Sparse matrix techniques can be used for saving considerable amount of storage with some deterioration in timing as a result.
4. Coding the programs in a language with bit manipulation capability will also save a considerable amount of storage without sacrificing execution time performance.
5. The use of the Bellmore-Ratliff [1] cutting plane algorithm for solving the SCP's in using Algorithm 3.1 and 3.2 would appear justified in light of the fact that rather quick convergence to SCP solutions occurred. Given the notoriety of cutting plane procedures for solving integer programming problems it is possible that an alternative procedure, such as implicit enumeration, see Garfinkel [13], may have to be resorted to in practice.
6. For large MWFI problems, if linear programming based SCP solution procedures are adopted, many of the matrices may have to be represented in double precision resulting in significant increase in storage requirement and more than 100% deterioration in computational performance. Numerical error was a nagging problem in the SCP solution segment, even with the relatively small problems reported on in Section 4.2. The problem was overcome by carefully setting very small numbers to zero all over the program. Such a device is not expected to work in general.

7. The implied edits generated by Algorithm 3.2, which are not dominated by an edit in the current explicit set of edits, can be added to the set of explicit edits against which records are to be tested for setting up the SCP in Step 2 of the algorithm. This device should result in fewer iterations being performed in Algorithm 3.2. Additional storage is, however, likely to be needed.

CHAPTER 5

ERROR LOCALIZATION FOR CONTINUOUS DATA

The theoretical results leading to the solution of MWFI for coded data were developed in Chapters 2 and 3, and the computational performance of these algorithms was examined in Chapter 4. We now turn our attention to the processing of records in which the data is drawn from continuous domains. Examples of continuous data would be boiler pressure, distances, rate of expenditure of fuel, etc.

In this chapter we will examine two algorithms for solving the MWFI problem for continuous data. It is worth noting that many of the theoretical developments in Chapters 2 and 3 do not carry over to the continuous data validation situation. Terms defined in earlier chapters will continue to have the same meaning unless indicated otherwise in this and subsequent chapters.

I. The Least Squares Approach to Data Editing

Freund and Hartley [10] proposed the following formulation:

$$\min \quad SSC = \sum_{j=1}^N w_j (x_j - y_j)^2 + \sum_{i=1}^m u_i \left(\sum_{j=1}^N a_{ij} x_j \right)^2 \quad (5.1)$$

where x_j is the changed or corrected value of the j th datum,

y_j is the original datum,

w_j is the weight given to the j th datum, a large weight indicates more reliable data,

$\sum_{j=1}^N a_{ij} x_j$ is the deviation, from zero, of the i th consistency equation,

$$\sum_{j=1}^N a_{ij}x_j = 0, \quad i = 1, 2, \dots, m, \quad (5.2)$$

u_i is the weight of the i th consistency equation; a large weight indicating an important consistency check.

Formulation (5.1) requires the minimization of the sum of weighted squares of the weighted variation of the 'corrected' value from the original value, plus the weighted sum of squares of the variation of the consistency conditions from their required value, namely 0.

The model (5.1) and the associated solution (5.2d) below are valid even if the consistency conditions are non-homogeneous. For example, consistency conditions of the form

$$\sum_{j=1}^N a_{ij}x_j = b_i, \quad i = 1, \dots, m \quad (5.2a)$$

can be cast in the form (5.2) by treating the b_i as the coefficient $a_{i,N+1}$ associated with the variable x_{N+1} which always has the value 1. Thus (5.2a) may be rewritten as

$$\sum_{j=1}^{N+1} a_{ij}x_j = 0, \quad i = 1, \dots, m. \quad (5.2b)$$

However, consistency conditions of the form

$$\sum_{j=1}^N a_{ij}x_j \leq b_i, \quad i=1, \dots, m \quad (5.2c)$$

cannot be accommodated in formulation (5.1) if the Freund and Hartley minimization procedure, outlined below, is to be used. The violation of the i th constraint, say, in (5.2c) will result in a negative value for $s_i = b_i - \sum_{j=1}^N a_{ij}x_j$, and it is intuitively appealing to minimize the sum of squares of such deviations. However if $s_i \geq 0$, the i th constraint is not violated and it is not at all clear if it would be

desirable to minimize the sum of squares of such deviations as well.

If only deviations of the kind $s_i < 0$ are to be accommodated in the model (5.1) then the second term in (5.1) could be modified as

$\sum_{i=1}^m u_i \left(\sum_{j=1}^N a_{ij} x_j - b_i \right)_+$, where $(z)_+ = \max(0, z)$. Notice that (5.1)

with this modification in place is no longer differentiable and the Freund and Hartley procedure leading to (5.2d) can no longer be used.

In fact this modified problem is much harder to solve than is (5.1).

It is worth noting at this point that formulation (5.1) permits

erroneous records to be 'corrected' in such a way that not all of the m consistency conditions (5.2) are satisfied.

Freund and Hartley arrive at the changed data values by differentiating SSC with respect to x_j , $j \in I_N$, resulting in the vector equation

$$x = (D_w + A^T D_u A)^{-1} D_w y \quad (5.2d)$$

where D_w is an $(N \times N)$ diagonal matrix of the weights w_j , $j \in I_N$,

y is an N -vector of the original data,

x is an N -vector of 'corrected' data,

$A = (a_{ij})$ is an $(m \times N)$ matrix, and A^T is the transpose of A .

D_u is an $(m \times m)$ diagonal matrix of the weights u_i , $i = 1, \dots, m$.

The models developed in Sande [32], and in McKeown [26]

concentrate on solving an MWFI formulation for the continuous data problem. Unlike the Freund and Hartley model, these models do require that all consistency conditions be satisfied.

II. MWFI Formulation for Continuous Data

The minimization objective in (5.1) can result in most, if not all, of the original data values being changed if the u_j 's are large, while it may not be necessary to change many of the data values in the failing record to make it satisfy (5.2). It is however very likely that most of the data in a record is correct to begin with and it is intuitively appealing to attempt to change as few of the original data values in the fields of the input record as possible to make it satisfy (5.2).

It was demonstrated in Chapter 2 that the solution to the MWFI formulation for the coded data problem results, under certain assumptions, in locating the fields that 'most likely' needed to be changed in an erroneous record. The assumptions that the probability of a field being in error is independent of each other were indicated to be quite realistic for coded data collection situations. In the context of collection of continuous data, these assumptions continue to be realistic when one considers the many transitions that the data must go through between the respondent and final tabulation. Errors may occur at any stage, such as, respondent calculating a data value and/or entering a data value in error, error in transfer of data from one table to another and so on. Thus the objective of retaining as much of the original data as possible continues to be as appealing in the context of processing of continuous data records as with coded data records.

In many data collection situations involving continuous data the consistency checks to be satisfied by a record take the form of a (weighted) sum of data values equalling (exceeding or being less than) a given value. For example, total sales revenue must equal the sum of the number of units of items sold multiplied by their price; the yields per acre of wheat times the acreage corresponding to these yields should total to be greater than or equal to the total yield from all the acreage under wheat, allowing for loss due to storage, etc. Much of the business and demographic data records are such that they need to satisfy linear consistency checks. While situations involving non-linear consistency conditions also exist we shall restrict ourselves, in this work, to continuous data validation in which the consistency conditions to be satisfied by each record are expressed in linear form.

Sande [32] suggested the following formulation of the imputation problem for continuous data, under the assumption that a set of linear consistency conditions must be satisfied by every record.

$$\text{Minimize } c^T (\delta(t) + \delta(z)) \quad (5.3)$$

$$\text{subject to: } A(x^0 + t - z) \leq b \quad (5.4)$$

$$x^0 + t - z \geq 0 \quad (5.5)$$

$$t \geq 0 \quad (5.6)$$

$$z \geq 0 \quad (5.7)$$

$$z^T t = 0 \quad (5.8)$$

where N is the number of fields in the data record and x^0 , $\delta(t)$, $\delta(z)$, t , and z are N -vectors such that x^0 is the given record of continuous data,

$$\delta(t_j) = \begin{cases} 1 & \text{if } t_j > 0 \\ 0 & \text{otherwise} \end{cases} \quad j \in I_N \quad (5.9)$$

$$\delta(z_j) = \begin{cases} 1 & \text{if } z_j > 0 \\ 0 & \text{otherwise} \end{cases} \quad j \in I_N. \quad (5.10)$$

The values $t = (t_j)$, and $z = (z_j)$, $j \in I_N$, indicate the changes required to produce a consistent record from the erroneous data record x^0 . The vector $c = (c_j)$ are the weights on each field of the record denoting the degree of confidence in the validity of the entries in that field. A and b are $(m \times N)$ and $(m \times 1)$ matrices, respectively, which represent the m linear consistency conditions, $Ax \leq b$.

The complementarity condition (5.8) ensures that no more than one positive or negative change can be made to each x_j^0 in an attempt to satisfy (5.4). We shall refer to the formulation (5.3) and (5.10) as the Minimum Weighted Fields to Impute Problem for Continuous Data (MWFIC).

Procedures for the solution of MWFIC have been proposed in McKeown [26], and in Sande [32]. McKeown reformulates MWFIC as a fixed charge problem and solves it using a branch and bound technique. Sande [32] proposed the use of a modification of the Chernikova algorithm [4] to solve MWFIC. We shall examine these approaches in subsequent sections.

III. Solution of MWFIC as a Fixed Charge Problem

The (linear) fixed charge problem, see [17,37], is a mixed integer linear programming problem which can be formulated as,

$$\begin{aligned}
 \text{Minimize} \quad & z = k^T x + c^T y \\
 & Ax \leq b \\
 & y_j = \begin{cases} 1 & \text{if } x_j > 0 \\ 0 & \text{if } x_j = 0 \end{cases} .
 \end{aligned} \tag{5.10a}$$

The N -vectors k and c represent, respectively, k_j the cost per unit of x_j and c_j , the fixed charge incurred whenever $x_j > 0$, $j = 1, \dots, N$.

McKeown [26] reformulated MWFIC as a fixed charge problem. The objective function (5.3) is a sum of fixed charges incurred only when a positive, t_j , or negative, z_j , change is made to the original data value x_j , $j \in I_N$. If c is assumed to be positive then the minimization of the fixed charges ensures that the complementarity condition (5.8) is necessarily satisfied at the optimal solution.

Formally, let $t_j z_j > 0$, for some $j \in I_N$, at the optimal solution, with associated cost c^* . However, $t_j(z_j)$ could have been chosen as $|t_j - z_j|$ designating $z_j(t_j)$ equal to zero, satisfying the constraints as before and, if $c_j > 0$, a solution with $c' \leq c^* - c_j < c^*$ could have been obtained contradicting the optimality of c^* .

The fixed charge problem formulation of MWFIC is then merely (5.2) - (5.7), (5.9), and (5.10) with the stipulation that $c > 0$. This formulation of MWFIC differs from the standard fixed charge problem, (5.10a), only in that the objective function (5.3) does not include any costs for the magnitude of the changes t_j and z_j , $j=1, \dots, N$. McKeown solved MWFIC using branch and bound, by solving linear programming relaxations of MWFIC at each branch node. (For

presentations of the general branch bound technique see Geoffrion [15], or Garfinkel [13].) Example 5.1 illustrates McKeown's procedure.

Example 5.1* Consider the following set of consistency conditions.

$$x_1 \geq 1 \quad (5.11)$$

$$x_1 \leq 4 \quad (5.12)$$

$$x_2 \geq 1 \quad (5.13)$$

$$x_2 \leq 4 \quad (5.14)$$

$$x_1 + x_2 \leq 7 \quad (5.15)$$

$$x_1 + x_2 \geq 3 \quad (5.16)$$

$$-x_1 + x_2 \leq 2 \quad (5.17)$$

$$x_1 - x_2 \leq 2 \quad (5.18)$$

Figure 5.1 shows the octagon, specified by (5.11) - (5.18), within which a passing record must lie. A record, $x^0 = (5, 0.5)$, which violates the system of constraints (5.11) - (5.18) lies outside of the octagon and is marked by an asterisk in Figure 5.1. Specifically, x^0 violates the constraints (5.12), (5.13), and (5.18). The fixed charge formulation of MWFIC for the record x^0 , with $c^T = (1, 1)$ is

$$\text{Min} \quad \delta(t_1) + \delta(t_2) + \delta(z_1) + \delta(z_2) \quad (5.19)$$

$$\text{subject to:} \quad t_1 - z_1 \geq -4 \quad (5.20)$$

$$t_1 - z_1 \leq -1 \quad (5.21)$$

$$t_2 - z_2 \geq 0.5 \quad (5.22)$$

*Example is from McKeown [26] although the constraint set was first proposed in Sande [31].

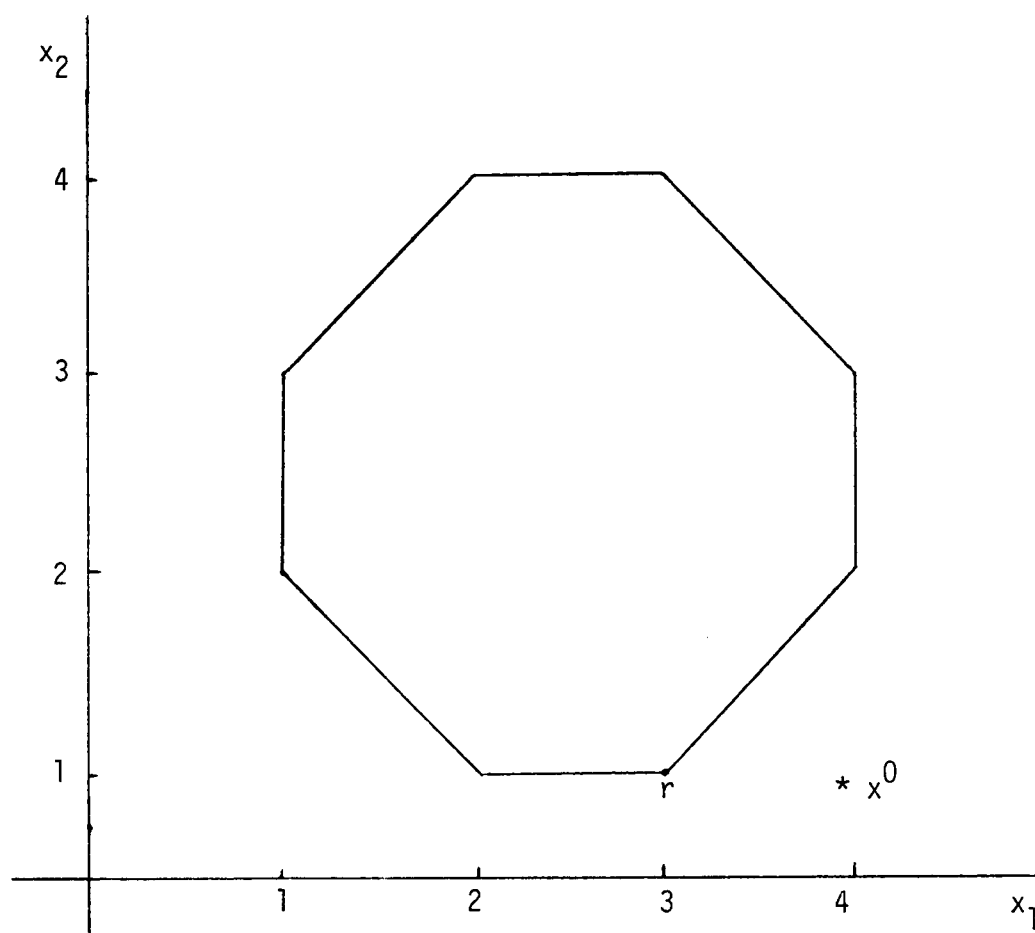


Figure 5.1 The Constraint Set of Example 5.1

$$t_2 - z_2 \leq 3.5 \quad (5.23)$$

$$t_1 + t_2 - z_1 - z_2 \leq 1.5 \quad (5.24)$$

$$t_1 + t_2 - z_1 - z_2 \geq -2.5 \quad (5.25)$$

$$-t_1 + t_2 + z_1 - z_2 \leq 6.5 \quad (5.26)$$

$$t_1 - t_2 - z_1 + z_2 \leq -2.5 \quad (5.27)$$

$$t_1, t_2, z_1, z_2 \geq 0 \quad (5.28)$$

Clearly $t = z = 0$ is infeasible; similarly z_1 or $z_2 = 0$ and $t = 0$ is infeasible; t_1 or $t_2 = 0$ and $z = 0$ is infeasible. A solu-

tion to (5.19) - (5.28) is given by $t_1 = 0$, $t_2 = 0.5$, $z_1 = 2$, $z_2 = 0$ resulting in an optimal solution value of 2. This implies that both values in x^0 need to be changed. The new record r , with $r_1 = x_1^0 - z_1$, $r_2 = x_2^0 + t_2$, is $r = (3,1)$ and it lies at a corner point of the octagon in Figure 5.1.

The new record r was picked to lie at an extreme point by design, to highlight the fact that if simplex method based procedures are used to solve the fixed change formulation of MWFIC then the corrected record will always lie at an extreme point (see Hirsch [18]). For the current example any record r^* which is located inside the octagon in Figure 5.1 would be a passing record constructed from x^0 by making an optimal cost change.

Algorithms for solving systems of homogeneous linear equations and for solving systems of homogeneous linear inequalities were developed by Chernikova [4,5]. Rubin [30] adapted these algorithms to solve the Cardinality Constrained Linear Program (CCLP). Sande [32] proposed solution of MWFIC, as a modification of CCLP, using Rubin's adaptation of Chernikova's algorithm. Rubin's algorithm for CCLP's and its adaptation for solving MWFIC will be presented in Section IV. We will assume that vectors and matrices in the following sections have only real entries.

IV. The Chernikova Algorithm and Its Modifications

Consider the problem

$$\max \quad c^T x, \quad (5.29)$$

$$\text{subject to} \quad x \in S = \{x | A_x \leq b, x \geq 0\} \quad (5.30)$$

$$|x|^+ \leq n \quad (5.31)$$

where c is $N \times 1$, A is $m \times N$, b is $m \times 1$, $|x|^+$ denotes the number of strictly positive elements of x , and n is a positive integer; also $n \leq \min \{m, N\}$. We will assume that S is bounded noting that the assumption can always be guaranteed by 'regularization,' see Charnes [3]. The problem (5.29)-(5.31) will be called CCLP.

It has been shown in Tanahashi [36] that an optimal solution to a CCLP can always be found in V , the set of vertices of S . Thus if all of the vertices of S are known, the CCLP is solvable by limiting the search for (5.29) to only those vertices for which (5.30) is satisfied. It is worth noting at this point that MWFIC, with weights $c = 1$, is very similar to a CCLP, differing from it only in that the objective is to minimize $|x|^+$.

Using the results in Burger [2], Chernikova [4] developed an algorithm for identifying all the extreme rays of a convex cone. Rubin's algorithm [30] for solving CCLP's is based on an implicit enumeration of the vertices of the polyhedron defined by S which in turn is adapted from the Chernikova procedure. Before presenting the vertex search procedure for solving MWFIC some definitions and results crucial to the validity of the procedure are presented.

Definition [33] A set S is said to be a convex set if for $x_i \in S$, $i = 1, \dots, p$

$$\sum_{i=1}^p \alpha_i x_i \in S, \quad (5.32)$$

whenever $\sum_{i=1}^p \alpha_i = 1$, $\alpha_i \geq 0$, $i=1, \dots, p$. The left hand side of (5.32) is said to be a convex linear combination of the points x_i .

Definition [16] A set of points $\bar{x}$ is said to be a ray from the origin generated by a point x if, $\bar{x} = \{\lambda x, \lambda > 0\}$ where λ is a scalar.

Definition [33] A cone G is a set of points such that if a vector $x \in G$, then $\lambda x \in G$, $\lambda \geq 0$. The cone G is thus a collection of rays generated by points in G .

Definition [28] A set of vectors is said to form a vector space if, given any two members u, v of the set, then $u+v$ and αu are also members of the set, where α is any scalar.

Definition [28] If a vector space consists of the set of all linear combinations of a finite set of vectors $x_1, x_2, \dots, x_s$, then the vectors are said to span (generate) the space.

Definition [33] A convex cone C is a cone which is a convex set.

Thus C is such that, $x \in C$, $\lambda \geq 0 \Rightarrow \lambda x \in C$ and $x_1 \in C$, $x_2 \in C \Rightarrow \alpha x_1 + (1-\alpha)x_2 \in C$, $0 < \alpha < 1$. For example, $D = \{x | Ax \geq 0\}$, where A is an $m \times N$ matrix and x is a N -vector, is a convex cone.

Definition [33] A point x in a convex set C is said to be a vertex (an extreme point) if and only if there exist no two points $x_1, x_2 \in C$ distinct from x and such that

$$x = \alpha x_1 + (1-\alpha)x_2, \quad 0 < \alpha < 1 \quad (5.33)$$

Definition A ray from the origin $\bar{x} = \{\lambda x, \lambda > 0\}$ in a convex cone G is said to be an extreme ray if x cannot be generated using (5.33) if x_1, x_2 are not themselves generators of $\bar{x}$. A vector x which generates an extreme ray $\bar{x}$ will be referred to as an extremal vector. The set of all distinct extremal vectors of a convex cone C will be referred to as a Complete Extremal System (CES) of C .

Example 5.1a Consider the convex cone G corresponding to

$$\begin{aligned} x_1 - x_2 &\geq 0 \\ x_1 &\geq 0 \\ x_2 &\geq 0 \end{aligned} \tag{5.33a}$$

in Figure 5.2.

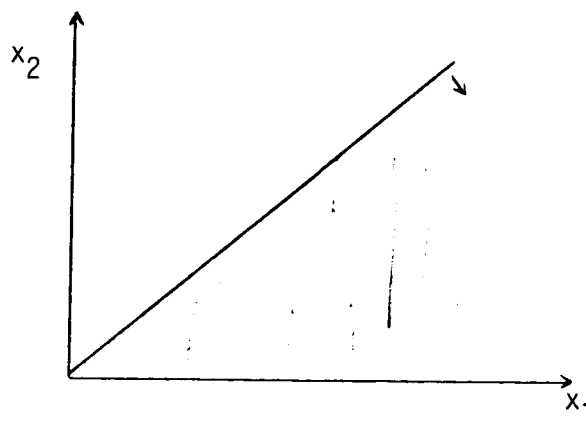


Figure 5.2 A Convex Cone

The shaded region together with its boundaries form the convex cone G . The extreme rays of G are $\bar{y}_1$ and $\bar{y}_2$ which can be generated using the vectors $y_1 = \begin{pmatrix} 1 \\ 1 \end{pmatrix}$ and $y_2 = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$ respectively. Note that $y_3 = 2y_1 = \begin{pmatrix} 2 \\ 2 \end{pmatrix}$ can also be used to generate $\bar{y}_1$. In fact any vector $t = \begin{pmatrix} t_1 \\ t_2 \end{pmatrix}$ such that $t_1 = t_2$ can generate $\bar{y}_1$, if $t_1, t_2 \in G$. A CES of G is defined by $\{y_1, y_2\}$. Other representations of the CES such as $\{y_3, y_2\}$ are also admissible.

Definition [28] A collection of N -vectors $x_1, x_2, \dots, x_s$ is said to be linearly dependent if there exist numbers $\alpha_1, \alpha_2, \dots, \alpha_s$, not all zero, such that

$$\sum_{i=1}^s \alpha_i x_i = 0. \tag{5.34}$$

If the vectors are not linearly dependent, they are said to be linearly independent. Two linearly independent vectors x_i, x_j will also be referred to as distinct vectors.

Definition [28] A basis for a vector space is a set of linearly independent vectors that spans the space. The number of vectors in a basis for a vector space is known as the dimension of the space.

Definition [28] Given a $m \times N$ matrix A , the rank of A , denoted $r(A)$, is the number of linearly independent columns (rows) of A .

Definition [28] Two non-zero N -vectors x and y are said to be orthogonal to (annihilate or annul) one another if, the scalar product of x and y , $x^T y = 0$. If vectors x and y are both orthogonal to a vector b , then x and y are said to be jointly orthogonal to b . Vectors and matrices will be assumed to be non-null (non-zero) in our discussions.

Lemma 5.1 [28]

If vectors x and y are orthogonal to each other they are also linearly independent.

Proof Assume the contrary. Then there exist numbers α_1 and α_2 at least one of which is non-zero (assume $\alpha_1 \neq 0$ w.l.o.g.) such that

$$\alpha_1 x + \alpha_2 y = 0$$

or
$$x = -\frac{\alpha_2}{\alpha_1} y .$$

If $\alpha_2 = 0$, then $x = 0$ contrary to our assumption that $x \neq 0$. If $\alpha_2 \neq 0$, then $y^T x \neq 0$ contrary to x and y being orthogonal.

QED.

Lemma 5.1(a) [28]

Given an $m \times N$ ($m \geq N$) matrix A with $r(A) = k$, the homogeneous system of equations $Ax = 0$ can never be inconsistent. Further one of the following possibilities must occur.

(a) $k = N$. Then $Ax = 0$ has a unique solution $x = 0$.

(b) $k < N$. Then $Ax = 0$ has an infinity of solutions.

Further, if $m < N$, $Ax = 0$ possesses an infinity of non-zero solutions.

Proof see Noble [28], pp. 91-92.

Definition [28] A vector space consisting of all solutions of the homogeneous equations $Ax=0$ is said to be the null space of A , where A is $m \times N$ and x is $N \times 1$.

Lemma 5.1(b)

If A is $m \times N$, the null space of A forms a vector space of dimension $N - r(A)$.

Proof see Noble [28], pp. 119-120.

Lemmas 5.2 and 5.3 and Theorem 5.1 that follow are based on the results in Burger [2] as presented in Liepins [22]. It will be assumed, unless indicated otherwise, that A is $m \times N$ and that $m \geq N$.

Lemma 5.2

If A is $m \times N$, $r(A) = N$ and x is $N \times 1$, then an N -vector $r \in C = \{x | Ax \geq 0\}$ is an extremal vector of the cone C if and only if r is orthogonal to exactly $N-1$ linearly independent rows of A .

Proof ($\Leftarrow$) If r is not extremal, then there exist linearly independent vectors $x_1, x_2 \in C$, such that

$$r = \alpha x_1 + (1-\alpha)x_2, \quad 0 < \alpha < 1. \quad (5.35)$$

Let $A_{N-1} \subseteq A$ denote the $m \times (N-1)$ matrix made up of the $N-1$ rows of A that annihilate r . Then $A_{N-1}r = \alpha A_{N-1}x_1 + (1-\alpha)A_{N-1}x_2 = 0$ implying that

$$A_{N-1}x_1 = 0 = A_{N-1}x_2. \quad (5.35a)$$

Since $r(A) = N$, (5.35a) implies that x_1 and x_2 are not distinct, thus contradicting (5.35).

($\Rightarrow$) Clearly r cannot annul more than $N-1$ linearly independent rows of A , since $r(A) = N$. We need to show that r annuls exactly $N-1$ linearly independent rows of A . Assume the contrary, i.e. r is annihilated by fewer than $N-1$ linearly independent rows of A , say $N-k$, for $k \geq 2$. Let A_{N-k} represent the matrix of the $N-k$ linearly independent rows of A annulled by r , $r(A_{N-k}) = N-k$.

Let A_N , partitioned into $\begin{pmatrix} A_{N-k} \\ A_k \end{pmatrix}$, represent a set of N linearly independent rows of A . Let a_i^T , $i=1, \dots, k$, denote the rows of A_k and

let $A_{N-1}^i = \begin{pmatrix} A_{N-k} \\ a_1^T \\ a_2^T \\ \vdots \\ a_{i-1}^T \\ a_{i+1}^T \\ \vdots \\ a_k^T \end{pmatrix}$, $i=1, \dots, k$, denote the k submatrices of A_N

with $N-1$ rows each and with A_{N-k} common to each. From Lemma 5.1(b) we know that there exist k vectors $r_1, \dots, r_k$ such that

$$A_{N-1}^i r_i = 0, \quad i = 1, \dots, k. \quad (5.35b)$$

Notice that $a_i^T r_i \neq 0$ and $a_j^T r_i = 0$, $i \neq j$, $i=1, \dots, k$. We wish to show that the r_i can be chosen to be linearly independent. Assume the

contrary. Then we can express any of the r_i , say r_k , in terms of the r_j , $j=1, \dots, k-1$, i.e.

$$r_k = \sum_{j=1}^{k-1} \alpha_j r_j, \quad (5.35c)$$

where not all of the α_j , $j=1, \dots, k-1$ are zero.

We have then,

$$A_{N-1}^k r_k = 0$$

or

$$A_{N-1}^k \sum_{j=1}^{k-1} \alpha_j r_j = 0.$$

From the construction of A_{N-1}^k and the fact that $a_j^T r_i = 0$, $i \neq j$, we have

$$\alpha_j a_j^T r_j = 0, \quad j=1, \dots, k-1. \quad (5.35d)$$

Since $a_j^T r_j \neq 0$, (5.35d) implies $\alpha_j = 0$, $j=1, \dots, k-1$ contradicting our assumption that the r_i , $i=1, \dots, k$ cannot be linearly independent.

Also the r_i can be chosen so that $a_i^T r_i > 0$, $i \in I_k = \{1, \dots, k\}$. Thus $Ar_i \geq 0$, $i \in I_k$ and the $r_i \in C$ and are extremal.

The r_i , $i \in I_k$, form a basis for the null space of A , to which r also belongs. Therefore r can be expressed as, $r = \sum_{i=1}^k \alpha_i r_i$. Thus, $Ar = \sum_{i=1}^k \alpha_i Ar_i$. Partition A into $\begin{pmatrix} A_{N-k} \\ A_k \\ A_{m-N} \end{pmatrix}$ where A_k represents

k linearly independent rows of A which together with A_{N-k} form a basis. Note that A_k has a_i^T , $i \in I_k$, as its rows. Thus $Ar = \sum_{i=1}^k \alpha_i Ar_i$ can be rewritten as:

$$\begin{aligned} A_{N-k} r &= \sum_{i=1}^k \alpha_i A_{N-k} r_i = 0 \\ A_{m-N} r &= \sum_{i=1}^k \alpha_i A_{m-N} r_i \geq 0 \\ A_k r &= \sum_{i=1}^k \alpha_i A_k r_i > 0. \end{aligned} \quad (5.35e)$$

Since $a_i^T r_i > 0$, $i \in I_k$ and $a_j^T r_i = 0$, $j \neq i$, $i, j \in I_k$, (5.35e) together with the fact (based on the assumption that r annuls fewer than $N-1$ linearly independent rows of A) that r is distinct from each of the r_i , implies that $\alpha_i > 0$, $i \in I_k$. Then the r_i could have been so chosen that $\sum_{i \in I_k} \alpha_i = 1$, $\alpha_i > 0$, $i \in I_k$ and $r = \sum_{i \in I_k} \alpha_i r_i$, $r_i \in C$, $i \in I_k$ contradicting the fact that r is extremal.

Theorem 5.1

Let C be the cone $\{x | Ax \geq 0\}$, $r(A) = N$ and let $r_1, \dots, r_s$ be the complete extremal system of C . Then the CES of the cone $\bar{C} = \{x | Ax \geq 0 \text{ and } a^T x \geq 0\} \subseteq C$, where a is an N -vector, is defined by:

(a) The extremal vectors of C for which the scalar product

$$a^T r_i \geq 0, \quad i = 1, \dots, s, \quad (5.36a)$$

together with

(b) The linear combinations $v = \lambda_i r_j + \lambda_j r_i$, $\lambda_i > 0$, $\lambda_j > 0$, of all pairs of extremal vectors (r_i, r_j) such that

$$(i) \quad (a^T r_i) \cdot (a^T r_j) < 0 \quad (5.36b)$$

$$(ii) \quad r_i, r_j \text{ are jointly orthogonal to } N-2 \text{ linearly independent rows of } A. \quad (5.36c)$$

$$(iii) \quad a^T v = 0. \quad (5.36d)$$

Proof Clearly $r_i \in C$ and $a^T r_i \geq 0$ imply that $r_i \in \bar{C}$. $\bar{C} \subseteq C$ implies then that r_i is extremal in $\bar{C}$.

It remains to be shown that condition (b) leads to the construction of the remaining extremal vectors of $\bar{C}$. Note that $\lambda_i, \lambda_j > 0$ exist so that (5.36d) is true whenever (5.36b) is true. Also $Av \geq 0$, by construction of v . Thus $v \in \bar{C}$. If $(a^T r_i) \cdot (a^T r_j) > 0$ then (5.36d) is not realizable. Also, if $(a^T r_i) \cdot (a^T r_j) = 0$, v is not distinct from $r_i(r_j)$. Since $a^T r_i \neq 0$ and $a^T r_j \neq 0$, a^T is linearly

independent of all the rows of A which are jointly orthogonal to r_i and r_j . Since r_i, r_j are jointly orthogonal to $N-2$ linearly independent rows of A , so is v . Further v is orthogonal to a^T . Hence, from Lemma 5.2, v is an extremal vector of $\bar{C}$.

It remains to be shown that no other distinct extremal vectors of $\bar{C}$ exist. Assume the contrary. Let $t \in \bar{C}$ be extremal but not constructed using conditions (a) and (b). Condition (a) implies that t is not an extremal vector of C . It must therefore be true that $a^T t = 0$ and t is orthogonal to $N-2$ linearly independent rows of A . Let these rows be represented by the matrix A_{N-2} . The rows of A_{N-2} must be linearly independent of a^T , since t is extremal in $\bar{C}$. Since $t \in \bar{C} \subseteq C$ is not extremal in C , t can always be expressed as

$$t = \alpha x_1 + (1-\alpha) x_2 \quad (5.36e)$$

where $x_1, x_2 \in C$ are extremal and distinct, and $\alpha > 0$. Multiplying both sides of (5.36e) by a^T we have

$$a^T t = \alpha a^T x_1 + (1-\alpha) a^T x_2 = 0. \quad (5.36f)$$

Since t is extremal in $\bar{C}$ at least one of x_1 or x_2 is not in $\bar{C}$. (5.36f) together with $\alpha > 0$ precludes x_1, x_2 both being outside of $\bar{C}$. Assume, w.l.o.g., that $x_1 \in \bar{C}$ and $x_2 \notin \bar{C}$. Then $a^T x_2 < 0$ and $a^T x_1 > 0$, since $\alpha > 0$. Since $A_{N-2} t = 0$, (5.36e) and $x_1, x_2 \in C$ imply that

$$A_{N-2} x_1 = A_{N-2} x_2 = 0. \quad (5.36g)$$

But (5.36e), (5.36f), the extremality of $x_1, x_2 \in C$ and (5.36g) are precisely the condition (b) of the Theorem contradicting our assumption that t is not constructible using conditions (a) and (b).

Corollary 5.1.1

Beginning with the cone $C_1 = \{x | x \geq 0\}$ the complete extremal system of the cone $C_{m+1} = \{x | x \geq 0, Ax \geq 0\}$ can be constructed using

the procedure of Theorem 5.1.

Proof By induction using the result in Theorem 5.1.

The condition (5.36c) requires that extremal vectors $r_i, r_j \in C$ be jointly orthogonal to $N-2$ linearly independent rows of A , if they are to be combined to form an extremal vector for the new system, $\bar{C}$. Lemma 5.3 establishes the condition under which r_i and r_j will be jointly orthogonal to $N-2$ linearly independent rows of A . This is necessary for the recognition of such $r_i, r_j \in C$, since the presence of linearly dependent rows of A may result in $N-2$ or more rows of A being simultaneously annulled by r_i and r_j making it very difficult to recognize the satisfaction or otherwise of (5.36c).

Lemma 5.3

Two extremal vectors $r_i, r_j \in C = \{x | Ax \geq 0\}$, are jointly orthogonal to $N-2$ linearly independent rows of A if and only if there exists no other extremal vector which annuls all rows of A to which r_i, r_j are jointly orthogonal.

Proof ($\Rightarrow$) Clearly two extremal vectors r_i and r_j can be jointly orthogonal to no more than $N-2$ linearly independent rows of A . For, $r(A) = N$ precludes N linearly independent rows of A being simultaneously annihilated. Since r_i and r_j are linearly independent, joint orthogonality with $N-1$ such rows of A is also precluded. Let $a_i^T, i \in I_{N-1} \subseteq \{1, \dots, m\}$, and $a_j^T, j \in J_{N-1} \subseteq \{1, \dots, m\}$ be collections of $N-1$ linearly independent rows of A orthogonal to r_i and r_j respectively (their existence is guaranteed by Lemma 5.2), and let $a_k^T, k \in K = I_{N-1} \cap J_{N-1}$ be the collection of all linearly independent rows of A orthogonal to both r_i and r_j .

Suppose there exists no extremal vector orthogonal to all the linearly independent rows of A that r_i and r_j are jointly orthogonal to. If $|K| < N-2$ then $I_{N-1} - K \neq \emptyset$ and $J_{N-1} - K \neq \emptyset$, and K can be expanded to K^* , with additions from I_{N-1} and J_{N-1} so that K^* indexes $N-1$ linearly independent rows. One such set of the indices K^* can be chosen to index $N-2$ linearly independent rows with indices in I_{N-1} and a row indexed by $J_{N-1} - I_{N-1}$ which is linearly independent of these $N-2$ rows; such a choice can always be made since $|K| < N-2$. There exists, then, a row, say a_p^T , of A which is linearly independent of the rows indexed by K^* . Let A^* be the matrix of the rows indexed by K^* and all other rows of A except a_p^T . Note that $r(A^*) = N-1$. Thus a vector $r (\neq 0)$ exists such that $A^*r = 0$. Note that the construction of A^* renders r distinct from both r_i and r_j . Further, a_p^T is independent of the rows of A^* and r cannot be orthogonal to a_p^T (since $r(A) = N$). If $a_p^T r < 0$, $A^*v = 0$ then r may be chosen as equal to $-v$ rendering $a_p^T r > 0$, $A^*r = 0$. Thus $Ar \geq 0$ and $r \in C$. Also r is orthogonal to $N-1$ linearly independent rows of A . Hence r is extremal (Lemma 5.2) contradicting our assumption that no such extremal vector exists.

($\Leftarrow$) Conversely, if $|K| = N-2$ but there exists an extremal vector r , distinct from r_i , r_j , annulling the rows of A indexed by K then, r together with r_i , r_j and the $N-2$ linearly independent rows of A jointly orthogonal to r , r_i , and r_j amount to $N+1$ linearly independent vectors contradicting the fact that a basis in N -dimensional space cannot have more than N vectors.

QED.

Lemma 5.3 enables us to verify if the condition (5.36c) in Theorem 5.1 is satisfied. Given the cones $C = \{x | Ax \geq 0\}$ and $\bar{C} = \{x | Ax \geq 0, a^T x \geq 0\}$ we can establish if two extremal vectors $r_i, r_j \in C$, satisfying (5.36b), may be combined, as in Theorem 5.1, by verifying if there exists no other extremal vector $r \in C$ such that the products Ar , Ar_i and Ar_j have zero entries in the same coordinates in the resultant vectors. Thus Theorem 5.1 together with Lemma 5.3 provides us with a procedure for locating all the extremal vectors of a given cone C , $C = \{x | x \geq 0, Ax \geq 0\}$, starting from the extremal system of the cone $C_1 = \{x | x \geq 0\}$ and adding one row of A at a time to C_1 to produce C_2, C_3 and so on until all rows of A have been processed and no more extremal vectors can be produced using the procedure outlined in Theorem 5.1. The following algorithm due to Chernikova [4] uses precisely this procedure. We proceed to establish some notation before stating the algorithm.

Consider the matrix $\begin{pmatrix} A \\ I \end{pmatrix}$ where A is an $m \times N$ matrix and I is an $N \times N$ identity matrix. The Chernikova algorithm performs a series of transformations of this matrix that generates all the extremal vectors of the cone $C = \{x | Ax \geq 0, x \geq 0\}$. At any stage of the process, the old matrix will be denoted by $Y = \begin{pmatrix} U \\ L \end{pmatrix}$ and the new matrix being generated by $\bar{Y}$. The matrices U and L will always have m and N rows, respectively, and they will have the same number of columns, which can be different from N . In this context it is not being assumed that $m \geq N$. It is worth noting at this juncture that a nonhomogeneous system such as $By \geq b$ can be treated as a homogeneous system $Ax \geq 0$ by designating $A = (B, -b)$, $x = (y, \beta)^T$ and maintaining the (non-zero) value of β at 1 in using the Chernikova algorithm. The justification for such a device, due to Rubin [], is furnished as Lemma 5.4.

The Chernikova Algorithm

1. If any row of U has all components negative, then $x = 0$ is the only point in C .
2. If $U \geq 0$, i.e. all elements of U are non-negative, then the columns of L are the extremal vectors of C ; i.e. the ray $\bar{\ell}_j = \{x | x = \lambda \ell_j, \lambda \geq 0\}$ is an extreme ray of C , where ℓ_j denotes the j th column of L .
3. Choose any row of U , say row r , with at least one negative element.
4. Let $R = \{j | y_{rj} > 0\}$ and let $v = |R|$. Then the first v columns of the new matrix, $\bar{Y}$, are all the y_j , $j \in R$.
- 4a. If Y has only two columns and $y_{r1}y_{r2} < 0$, adjoin the column $|y_{r2}|y_1 + |y_{r1}|y_2$ to $\bar{Y}$.
Go to step 6.
5. Let $Q = \{(s,t) | y_{rs}y_{rt} < 0, s < t\}$, i.e. the set of all unordered pairs of columns of Y whose elements in row r have opposite signs. Let P be the index set of all non-negative rows of Y . For each $(s,t) \in Q$, find all $i \in P$ such that $y_{is} = y_{it} = 0$. Let this set be denoted by $P^*(s,t)$. Additional columns for $\bar{Y}$ are then created as follows:
 - (a) If $P^*(s,t) = \emptyset$, then y_s and y_t do not contribute another column to the new matrix.
 - (b) If $P^*(s,t) \neq \emptyset$, check if there is a u , $u \neq s, u \neq t$ such that $y_{iu} = 0$ for all $i \in P^*(s,t)$. 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 $\alpha_1 y_{rs} + \alpha_2 y_{rt} = 0$ (one such choice is $\alpha_1 = |y_{rt}|$, $\alpha_2 = |y_{rs}|$). Adjoin the column $\alpha_1 y_{rs} + \alpha_2 y_{rt}$ to the new matrix $\bar{Y}$.

6. When all the pairs in Q have been examined, and the additional columns (if any) have been added, we say that row r has been 'processed.' Set $Y = \bar{Y}$, the matrix produced in processing row r , and to to Step 1.

Before justifying the algorithm, the procedure is illustrated in

Example 5.2.

Example 5.2 Consider the system of homogeneous inequalities

$$Bx = \begin{pmatrix} I \\ A \end{pmatrix} x \geq 0 \quad (5.37)$$

represented by,

$$\begin{array}{rcl} x_1 & \geq & 0 \\ x_2 & \geq & 0 \\ x_3 & \geq & 0 \end{array} \quad (5.38)$$

$$\begin{array}{rcl} x_1 + x_2 - x_3 & \geq & 0 \\ 2x_1 - x_2 + x_3 & \geq & 0 \\ x_1 - 2x_2 - x_3 & \geq & 0 \end{array} \quad (5.39)$$

The initial cone is $\{x | x \geq 0\}$ to which the inequalities in (5.39) are added. Initially the matrix $U = \begin{bmatrix} 1 & 1 & -1 \\ 2 & -1 & 1 \\ 1 & -2 & -1 \end{bmatrix}$ and $L = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$ is a complete extremal system of the cone $\{x | x \geq 0\}$. The initial tableau is shown in Table 5.1.

TABLE 5.1

Initial Tableau in Example 5.2

$Y = \begin{pmatrix} U \\ L \end{pmatrix} =$	1	1	-1
	2	-1	1
	1	-2	-1
	1	0	0
	0	1	0
	0	0	1

Iteration 1

1. Steps 1 and 2 do not apply for Y .

Step 3 Row 2 of U has one negative element; choose $r = 2$.

Step 4 $R = \{j | y_{rj} > 0\} = \{1, 3\}$; i.e. columns 1 and 3 of Y are written as they are in the new matrix $\bar{Y}$.

Step 5 $Q = \{(1, 2), (2, 3)\}$, $P = \{4, 5, 6\}$

$$P^*(1, 2) = \{6\}, P^*(2, 3) = \{4\}$$

Step (5b) is applicable and 2 new columns are produced by combining existing columns 1 and 2 to produce a zero in row 2; similarly for columns 2 and 3. $\bar{Y}$ is as in Table 5.2.

TABLE 5.2

The Tableau after 1 Iteration in Example 5.2

$\bar{Y} = \begin{pmatrix} \bar{U} \\ \bar{L} \end{pmatrix}$	1	-1	3	0
	2	1	0	0
	1	-1	-3	-3

	1	0	1	0
	0	0	2	1
	0	1	0	1

Step 6 Set $Y = \bar{Y}$. Note that the matrix $\bar{U}$ has entries given by the matrix product $A\bar{L}$.

Iteration 2 Steps 1 and 2 are again not applicable.

Step 3 Choose $r=3$.

Step 4 $R = \{1\}$ and column 1 of Y is added to $\bar{Y}$ without change.

Step 4a Not applicable.

Step 5 $Q = \{(1, 2), (1, 3), (1, 4)\}$

$$P^*(1,2) = \{5\}, \quad P^*(1,3) = \{6\}, \quad P^*(1,4) = \phi$$

Column pairs (1,2) and (1,3) contribute new columns to $\bar{Y}$.

Then new $\bar{Y}$ is shown in Table 5.3.

TABLE 5.3

The Tableau after 2 Iterations in Example 5.2

$$\bar{Y} = \begin{array}{ccc} 1 & 0 & 6 \\ 2 & 3 & 6 \\ 1 & 0 & 0 \\ \hline 1 & 1 & 4 \\ 0 & 0 & 2 \\ 0 & 1 & 0 \end{array}$$

Step 6 $Y = \bar{Y}$.

Iteration 3 Step 1 is not applicable.

Step 2 $\lambda \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, \quad \alpha \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix}, \quad \text{and} \quad \delta \begin{bmatrix} 4 \\ 2 \\ 0 \end{bmatrix}, \quad \lambda, \alpha, \delta \geq 0$

are extremal vectors of the cone defined by (5.37).

Chernikova's algorithm follows from Lemmas 5.2 and 5.3, and Theorem 5.1 and its corollary. However, the steps of the algorithm need elucidation so that the correspondence between the theoretical results and the algorithm become transparent.

Step 1 At each step in the processing of the matrix A at least the row being processed is transformed into a non-negative row; i.e. a row without negative elements in it. Also, once a row of Y has become non-negative it can never have any negative elements in it, since the linear combinations in step (5b) involve only positive coefficients. The components of U at each step are also the components of AL . Thus if a row of U is all negative

then the implication is that the corresponding row of A has not been processed and more importantly $A \cdot L \not\leq 0$. Thus the 'extremal system' can never satisfy the row of A corresponding to the negative row in U and the only point in C is $x = 0$.

Step 2 If $U \geq 0$, no more extremal vectors can be generated for C . Thus the contents of L are the extremal vectors in the complete extremal system of C .

Step 3 Note that each iteration of the algorithm brings a hitherto unprocessed row of A into consideration. Since processing of a row can result in automatic processing of another hitherto unprocessed row (see Step 5), the number of iterations, taken by the algorithm cannot exceed m , the number of rows in A .

Step 4 This is a direct consequence of (5.36a), namely that the extremal vectors of the cone with row r of A , denoted a_r^T , added include the extremal vectors of the cone processed in the previous step, x_j for which $y_{rj} = a_r^T x_j \geq 0$.

Step 4a This is a special case of Step (5b) in which $u \neq s$, $u \neq t$ does not exist. (See the discussion of Step (5b) below.)

Step 5 Before explaining steps (5a) and (5b) it is useful to examine the structure of Y . Notice that $Y_1 = \begin{pmatrix} A \\ I \end{pmatrix}$ to begin with and because of the structure of I , $L = I$ is also the CES for $\{x | Ix \geq 0\}$. The transformations at each step produce new U and L matrices with the important property that, at any step, $Y = \begin{pmatrix} U \\ L \end{pmatrix} = Y_1 \cdot L$. Thus, at any step, there exist at least $N-1$ zeros in any column y_j of Y , the number of zeros can exceed $N-1$ if there are linearly dependent rows in Y_1 . The set Q is simply

the index set of all pairs of columns, (s,t) of Y corresponding to $(a_r^T \ell_s) \cdot (a_t^T \ell_t) < 0$ (condition (5.36b)).

The set of non-negative rows, P , corresponds to the set of rows of Y_1 that have been processed, explicitly or implicitly, as explained for steps 1 and 3. $P^*(s,t)$ is, then, merely the set of rows of Y_1 , in P , which are jointly orthogonal to the extremal vectors ℓ_s and ℓ_t in L . Now we are in a position to better appreciate steps (5a) and (5b).

In (5a), $P^*(s,t) = \emptyset$ implies that y_s and y_t do not contribute another column to $\bar{Y}$. Condition (5.36c) is clearly violated. Step (5b) is a direct consequence of Lemma 5.3. The choice of α_1 and α_2 to be $|y_{rt}|$ and $|y_{st}|$ respectively is merely a matter of convenience. Clearly, any other choices such as $\alpha|y_{rt}|$, and $\alpha|y_{st}|$, $\alpha > 0$ will be valid.

Step 6 This is self explanatory and valid on account of Corollary 5.1.1.

Rubin [30] modified Chernikova's algorithm to solve the CCLP (5.29) - (5.31). It was stated before that to solve the CCLP one need only look at the vertices of the polyhedron defined by (5.30). Lemma 5.4 demonstrates that the Chernikova algorithm can be used to find all vertices of a convex polyhedron $S = \{x | Ax \leq b, x \geq 0\}$.

Lemma 5.4

v is a vertex of $S = \{x | Ax \leq b, x \geq 0\}$ if and only if $\{(\frac{\lambda}{\lambda} v), \lambda \geq 0\}$ is an extremal ray of the cone $C_F = \{(x^T, \xi)^T | -Ax + b\xi \geq 0, x \geq 0, \xi \geq 0\}$, where λ and ξ are scalars, A and x are $m \times N$ and $N \times 1$ matrices respectively.

Proof Suppose v is a vertex of S but $(\begin{smallmatrix} v \\ 1 \end{smallmatrix})$ is not an extreme ray of C_F . Then there exist points $(\begin{smallmatrix} x_1^1 \\ 1 \end{smallmatrix})$, $(\begin{smallmatrix} x_1^2 \\ 1 \end{smallmatrix})$ in C_F so that $(\begin{smallmatrix} v \\ 1 \end{smallmatrix}) = \alpha(\begin{smallmatrix} x_1^1 \\ 1 \end{smallmatrix}) + (1-\alpha)(\begin{smallmatrix} x_1^2 \\ 1 \end{smallmatrix})$, $0 < \alpha < 1$. Note that x_1, x_2 are also in S . Thus v can be expressed as a convex combination of $x_1, x_2 \in S$ contradicting the assumption that v is a vertex. Conversely, if $(\begin{smallmatrix} v \\ 1 \end{smallmatrix})$ is an extreme ray of C_F but v is not an extreme point of S then β , $0 < \beta < 1$, can be chosen so that $v = \beta x_1 + (1-\beta)x_2$, $x_1, x_2 \in S$; $(\begin{smallmatrix} x_1^1 \\ 1 \end{smallmatrix})$, $(\begin{smallmatrix} x_1^2 \\ 1 \end{smallmatrix}) \in C_F$, contradicting our assumption that $\lambda(\begin{smallmatrix} v \\ 1 \end{smallmatrix})$ is an extreme ray of C_F .

QED.

The significance of Lemma 5.4 is that we can apply the Chernikova algorithm to find all extreme points of the polyhedron S , starting with the initial set up,

$$Y_1 = \begin{pmatrix} -A \\ I \end{pmatrix} \quad (5.40)$$

where A is a $m \times (N+1)$ matrix with its $(N+1)$ th column corresponding to the m -vector b and I is a $(N+1) \times (N+1)$ identity matrix. The CES of the homogeneous inequalities to which (5.40) corresponds, represents the set of vertices of S ; i.e. if β_j represents the $(N+1)$ th component of ℓ_j in L , then the vertices of S correspond to the first N components of ℓ_j , denoted $\hat{\ell}_j$, in L whenever $\beta_j > 0$. The actual vertex, x_j^* , corresponding to $\hat{\ell}_j$ is obtained by multiplying ℓ_j by a number λ so that $\lambda\beta_j = 1$. Thus $\lambda\hat{\ell}_j$ corresponds to vertex x_j^* (Lemma 5.4).

Since our goal is to solve MWFIC using modifications of the CCLP, (5.29) - (5.31), and of the Chernikova procedure, and since the number of columns of Y generated at any iteration of the process can be prohibitively large, devices for reducing the numbers of columns of Y retained at each step of the Chernikova procedure are

called for. Lemmas 5.5 and 5.6, adapted from Rubin [30], furnish such devices.

It is assumed in the following that the cost of altering variable x_i , corresponding to the i th field, is $c_i \geq 0$. The cost of altering variable x_{N+1} , corresponding to the entries in b , is set to 0. The algorithm referred to in Lemmas 5.5 and 5.6 and Theorem 5.2 is the Chernikova algorithm.

We will assume, in the following, that η is the cost associated with a feasible solution to MWFIC, and $\delta(x_j) = 1$ if $x_j > 0$; $\delta(x_j) = 0$ if $x_j = 0$.

Lemma 5.5

Suppose that in processing row i , we produce an ℓ_j with $\sum_{i=1}^N c_i \delta(\ell_{ij}) > \eta$, then any column $\bar{\ell}_k$ subsequently produced as a linear combination of ℓ_j , and some other column ℓ_t of L will also have $\sum_{i=1}^N c_i \delta(\bar{\ell}_{ik}) > \eta$.

Proof The algorithm creates new columns by taking strictly positive linear combinations of two old columns, and $L \geq 0$. Thus, if $\alpha \ell_j + \beta \ell_t = \bar{\ell}_k$, $\alpha, \beta > 0$, then $\delta(\bar{\ell}_{ik}) = \delta(\alpha \ell_{ij}) + \delta(\beta \ell_{it})$, $i=1, \dots, N$ has positive entries in every row that ℓ_j has a positive entry in and may have positive entries in rows where ℓ_j had no positive entry. Hence $\sum_{i=1}^N c_i \delta(\bar{\ell}_{ik}) > \eta$, since $c_i \geq 0$, $i \in I_N$.

Lemma 5.6

Suppose that in processing row r we encounter the following situation: $y_{rs} < 0$, $y_{rt} > 0$ and there exists k such that $y_{ik} = 0$ for all $i \in P^*(s, t)$, and $\sum_{i=1}^N c_i \delta(\bar{\ell}_{ik}) > \eta$. For any $\alpha_1 > 0$, $\alpha_2 > 0$, let

$y_\alpha = \alpha_1 y_s + \alpha_2 y_t$ and define ℓ_α as the last $N+1$ components of y_α , and $\hat{\ell}_\alpha$ as the first N components of ℓ_α . Then $\sum_{i=1}^N c_i \delta(\ell_{i\alpha}) > \eta$.

Proof Suppose $\hat{\ell}_{vk}$ is a strictly positive component of $\hat{\ell}_k$.

Since $y_{ik} = 0$ for all $i \in P^*(s,t)$, it follows that $v \notin P^*(s,t)$.

Hence at least one of $\hat{\ell}_{vs}, \hat{\ell}_{vt}$ is strictly positive; since $\alpha_1, \alpha_2 > 0$ we have $\hat{\ell}_{v\alpha} > 0$. Therefore $\sum_{i=1}^N c_i \delta(\hat{\ell}_{i\alpha}) \geq \sum_{i=1}^N c_i \delta(\hat{\ell}_{ik}) > \eta$.

Lemmas 5.5 and 5.6 enable us to keep all of Step 5 of the algorithm, for solving MWFIC. Theorem 5.2 integrates the content of Lemmas 5.5 and 5.6.

Theorem 5.2

If the algorithm ever produces a column $\bar{y}_j$ with $\sum_{i=1}^N c_i \delta(\bar{\ell}_{ij}) > \eta$, that column may be discarded from further computation.

Proof A direct consequence of Lemmas 5.5 and 5.6.

Note: It should be pointed out at this stage that the discarding of columns of Y using the provisions of Theorem 5.2 may result in non-extremal vectors being carried in the Chernikova tableau, Y . This is because with the discarding of an extremal vector the ability of the Chernikova algorithm to recognize joint orthogonality of two extremal vectors with $N-2$ linearly independent rows of A may also have been surrendered. However, if the Y matrix becomes unmanageably large testing for extremality of the components of L may be performed. From the perspective of locating a vector (or a set of vectors) with associated lowest cost for solving a CCLP (MWFIC) the presence of non-extremal solutions does not pose a problem. For, Theorem 5.2 assures us that the optimum will be found among the components (columns) of L when the Chernikova algorithm terminates.

Lemma 5.4 and Theorem 5.2 pave the way for the specification of a procedure for the implicit enumeration of all vertices of the polyhedron S , in (5.30), and hence solve the MWFIC. The use of the Chernikova procedure, as modified in Rubin, for solving MWFIC was first proposed in Sande [32].

V. A Vertex Search Algorithm for Solving MWFIC

Rubin [30] used a heuristic procedure for selecting the row r , of A , in Step 3 of the Chernikova algorithm, since the number of columns generated in using the algorithm can vary greatly depending on the choice of r at each step. He observed that the maximum number of columns of Y that a particular row r of Y can spawn is given by $Q(r) = P(r) + Z(r) + P(r) \cdot N(r)$, where $P(r)$, $Z(r)$, and $N(r)$ are, respectively, the number of positive, zero, and negative elements in row r of Y . Rubin's heuristic is to select the row r for which $Q(r)$ is minimal, breaking ties arbitrarily. We shall use this heuristic in the VS-1 algorithm.

The VS-1 Algorithm

0. Obtain a feasible solution (t', z') to the MWFIC. Record the solution and its associated cost η .
1. Find the vertices of the polyhedron defined by (5.4) - (5.7) in the MWFIC formulation, using the Chernikova algorithm with the following modifications:
 - (i) In Step 3, choose r so that $Q(r) = P(r) + Z(r) + N(r) \cdot P(r)$ is minimal, breaking ties arbitrarily.
 - (ii) Any column of Y with $\sum_{i=1}^N c_i \delta(x_{ij}) > \eta$ should be discarded.
 - (iii) Any column of Y , at a given stage in the processing, say

y_j , such that $u_j = 0$ and ℓ_j with $\beta_j = 0$ can be discarded. (β_j is the $(N+1)$ st component of ℓ_j).

- (iv) At any stage of processing if ℓ_j and ℓ_k produce $\ell_t = \alpha \ell_j + \beta \ell_k$, $\alpha, \beta > 0$, with $t_k z_k > 0$, then it may be discarded. For, ℓ_t can only produce other vertices which violate the complementarity condition (5.8).
- (v) If the cost associated with each ℓ_j in L is equal to η , (t', z') is optimal and processing may be terminated; if the procedure terminates normally, i.e. with $U \geq 0$, and there exist ℓ_j such that $\beta_j > 0$ then the optimal solution may be found as follows.

The optimal solution (t^*, z^*) corresponds to,

$$(t^*, z^*) = (t', z') \text{ if } \eta \leq \min_{\ell_j \in L} \{c(\hat{\ell}_j)\} \quad (5.40a)$$

where $c(\hat{\ell}_j)$ denotes the cost associated with $\hat{\ell}_j$; where $\hat{\ell}_j$ is the N -vector corresponding to the first N components of ℓ_j . Otherwise

$$(t^*, z^*) = \{\lambda_j \hat{\ell}_j \mid \lambda_j \beta_j = 1, j \in J^*, \lambda_j \text{ a scalar}\}$$

$$\text{where } J^* = \{k \mid c(\hat{\ell}_k) = \min_{\ell_j \in L} \{c(\hat{\ell}_j)\} < \eta\}. \quad (5.40b)$$

Example 5.3 Let us solve the problem outlined in (5.19) - (5.28).

Rewriting the constraint set in the format required for input into the Chernikova algorithm we have

$$t_1 - z_1 + 4 \geq 0 \quad (5.20')$$

$$-t_1 + z_1 - 1 \geq 0 \quad (5.21')$$

$$2t_2 - 2z_2 - 1 \geq 0 \quad (5.22')$$

$$-2t_2 + 2z_2 + 7 \geq 0 \quad (5.23')$$

$$-2t_1 + 2z_1 - 2t_2 + 2z_2 + 3 \geq 0 \quad (5.24')$$

$$2t_1 - 2z_1 + 2t_2 - 2z_2 + 5 \geq 0 \quad (5.25')$$

$$2t_1 - 2z_1 - 2t_2 + 2z_2 + 13 \geq 0 \quad (5.26')$$

$$-2t_1 + 2z_1 + 2t_2 - 2z_2 - 5 \geq 0 \quad (5.27')$$

$$t_1, t_2, z_1, z_2 \geq 0 \quad (5.28')$$

If $c_1 = 1$ and $c_2 = 2$, i.e. cost of changing field 1 is 1 and 2 for changing field 2, then (5.19) may be rewritten as

$$\delta(t_1) + \delta(z_1) + 2\delta(t_2) + 2\delta(z_2) \quad (5.19')$$

We know that $t_2 = .5$, $z_1 = 2$ is feasible to (5.20') - (5.28').

The cost of this solution is $\eta = 2 + 1 = 3$. (5.41)

The initial set up for the Chernikova algorithm is in Table 5.4.

TABLE 5.4

Initial Setup in Example 5.3

	1	-1	0	0	4
	-1	1	0	0	-1
	0	0	2	-2	-1
	0	0	-2	2	7
	-2	2	-2	2	3
	2	-2	2	-2	5
	2	-2	-2	2	13
$Y = \begin{pmatrix} U \\ L \end{pmatrix} =$	-2	2	2	-2	-5
	1	0	0	0	0
	0	1	0	0	0
	0	0	1	0	0
	0	0	0	1	0
	0	0	0	0	1

where the rows of L correspond to the variables t_1, z_1, t_2, z_2 and x_{N+1} , the variable corresponding to b .

$Q(1) = 4$ has the minimum column generation potential. Choosing $r = 1$ in the new $\bar{Y}$, we write columns 2, 3 and 5 out to $\bar{Y}_2$. Columns 1 and 5 are combined to produce the last column of $\bar{Y}$.

TABLE 5.6

The Tableau after 2 Iterations in Example 5.3

		0	0	3	0
		0	0	0	3
		2	-2	-1	-1
		-2	2	7	7
		-2	2	5	11
		2	-2	3	-3
		-2	2	11	5
		2	-2	-3	3
$\bar{Y}_2 =$		-----			
	t_1	0	0	0	0
	z_1	0	0	1	4
	t_2	1	0	0	0
	z_2	0	1	0	0
	b	0	0	0	1
Cost		2	2	1	1

Row 3 with $Q(3) = 4$ promises the smallest column growth in processing $\bar{Y}_2$.

Choosing $r = 3$, we have column 1 retained in the new $\bar{Y}$. Columns (1,2), (1,3), and (1,4) can be combined to produce three new columns of $\bar{Y}_3$. Combination (1,2) violates the complementarity condition and is discarded (since it will never again participate in generation of new columns, Step 1 (iv)). The new $\bar{Y}$ is in Table 5.7.

TABLE 5.7

The Tableau after 3 Iterations in Example 5.3

	0	6	0
	0	0	6
	2	0	0
	-2	12	12
	-2	8	20
	2	8	-4
	-2	20	8
$\bar{Y}_3 =$	2	-4	8
t_1	0	0	0
z_1	0	2	8
t_2	1	1	1
z_2	0	0	0
b	0	0	1
Cost	2	3	3

Choose $r=8$; all eligible rows have the same $Q(\cdot)$ value. Columns 1 and 3 are retained in the new $\bar{Y}$. Columns 1 and 2 can be combined; however, columns 2 and 3 cannot be combined as the conditions of Step (5b) are not met. The new $\bar{Y}$, $\bar{Y}_4$ is shown in Table 5.8.

Choose $r=4$; $Q(4) = 4$ is minimal. Columns 2 and 3 are retained. Columns 1 and 2 can be combined, similarly for 1 and 3.

TABLE 5.8

The Tableau after 4 Iterations in Example 5.3

		0	0	6
		0	6	0
		2	0	4
		-2	12	8
		-2	20	4
		2	-4	12
		-2	8	16
		2	8	0
$\bar{y}_4 =$		-----		
	t_1	0	0	0
	z_1	0	8	2
	t_2	1	1	3
	z_2	0	0	0
	b	0	1	0

	Cost	2	3	3

The new $\bar{Y}$ is shown in Table 5.9. At this point we can terminate processing, in the knowledge that the four columns of $\bar{Y}$ cannot ever produce a column with lower cost; since $t_2 = .5$, $z_1 = 2$, $n = 3$ is feasible it is also optimal.

A discussion of Algorithm VS-1 is part of Chapter 6. However, certain observations are in order at this juncture.

1. The contents of Theorem 5.2 are equally valid with strict inequality replaced with greater than or equal to, if the following modification is made to VS-1:

TABLE 5.9

Final Tableau in Example 5.3

	0	6	0	6
	6	0	6	0
	0	4	12	12
	12	8	0	0
	20	4	8	04
	-4	12	8	20
	8	16	-4	8
$\bar{y}_5 =$	8	0	20	8
	0	0	0	0
	8	2	8	2
	1	3	7	7
	0	0	0	0
	1	0	1	0
	Cost	3	1	3

Step 1 (vi) If termination of the Chernikova procedure occurs on account of anything but Step 2 of the Chernikova algorithm being satisfied, then (t', z') is optimal.

Step 1 (ii) Change ' $>$ ' to ' $\geq$ '.

2. With these modifications processing could have been terminated at $\bar{y}_3$ in Example 5.3.

Step 1 (vi) is justified thus: if termination occurs on account of Step 1 of the Chernikova procedure, $t = z = 0$ is the only solution. But $t = z = 0$ is infeasible since the input record violates at least

one constraint in the MWFIC. If, however, termination occurs on account of there being no columns in the new $\bar{Y}$ on account of the application of the modified step 1 (ii) of VS-1, then clearly (t', z') is optimal.

An alternative procedure, VS-2, is presented in Section VI, for the special case where the costs of changing the N fields, c_j , $j \in I_N$, are uniform.

VI. A Threshold Algorithm, VS-2, for MFIC

The performance of VS-1 is heavily dependent, among other things, on a 'good' initial feasible solution being obtained. If the c_j , $j \in I_N$, are uniform then a series of CCLP's can be solved corresponding to the number of positive components in $\hat{x}_j$, the first N components of $\hat{x}_j$, the j th column of L , being restricted to 1, 2, etc., for all x_j in L , until a feasible solution is found for MFIC. If n is the integer value at which a feasible solution is obtained then n corresponds to the optimal solution to MFIC and the components of $\hat{x}_j$, in that feasible solution, for any x_j in L , correspond to the optimal solution. The algorithm and an example are presented below.

Algorithm VS-2

1. Set up the matrix $Y = \begin{pmatrix} U \\ L \end{pmatrix}$ corresponding to the constraint set, (5.4) - (5.7), of the MFIC.
2. Set $n = 1$.
3. Apply the Chernikova procedure to the matrix Y with the following provisions:
 - (a) In step 3, choose r so that $Q(r) = P(r) + Z(r) + P(r) \cdot N(r)$ is minimal, breaking ties arbitrarily.

- (b) Any column of Y with $|\hat{x}_j|^+ > \eta$ should be discarded: set $\hat{Y} = Y$ before any columns are discarded from Y .
- (c) Any column of Y , at a given stage in the processing, say y_j , such that $u_j = 0$ and violates the complementarity condition (5.8) can be discarded.
4. (a) If the Chernikova procedure terminates when the condition in Step 2 of the procedure is met, namely $U \geq 0$, and if $K^* = \{j | \beta_j > 0\} \neq \emptyset$, then the vectors $\hat{x}_j$, $j \in K^*$, corresponding to the first N components of x_j in L , characterize the vertices of the constraint set (5.4)-(5.7). Any one of these vertices corresponds to the optimal solution; STOP.
- (b) If the condition in step 4(a) is not met then set $\eta = \eta + 1$ and $Y = \hat{Y}$. If $\eta \leq N$ go to step 3; if $\eta > N$, (5.4) - (5.8) are infeasible; STOP.

Example 5.4 illustrates the procedure in VS-2.

Example 5.4 Consider the following set of consistency conditions which a record must satisfy.

$$x_1 - x_2 \geq -4 \quad (5.41)$$

$$x_2 + x_3 \geq 7 \quad (5.42)$$

A record $x^0 = (0, 5, 0)$ violates both (5.41) and (5.42). Recasting the constraints (5.41) and (5.42) in the form (5.19') - (5.27') we have

$$t_1 - z_1 - t_2 + z_2 - 1 \geq 0 \quad (5.43)$$

$$t_2 - z_2 + t_3 - z_3 - 2 \geq 0 \quad (5.44)$$

$$t_1, t_2, t_3, z_1, z_2, z_3 \geq 0 \quad (5.45)$$

With $c_1 = c_2 = c_3 = 1$, the MFIC objective is to minimize

$$\text{minimize } y_1 + y_2 + y_3 + z_1 + z_2 + z_3 \quad (5.46)$$

Using the VS-2 algorithm, we have

Step 1

TABLE 5.10

The Initial Tableau in Example 5.4

	1	-1	-1	1	0	0	-1
	0	0	1	-1	1	-1	-2
$Y = \begin{pmatrix} U \\ L \end{pmatrix} =$	1						
		1					
			1				
				1			
					1		
						1	
							1
Cost	1	1	1	1	1	1	0

Step 2 $n = 1$.

Step 3 Choose row 1 for processing, i.e. $r=1$. The only columns of Y that survive the provision in (5.3b) of VS-2 are 1,4,5 and 6 and the combinations of columns (1,7), (4,7).

Note $\hat{Y} = Y$, since some generatable columns have been discarded on account of condition (3b). The new $\bar{Y}$ is in Table 5.11.

TABLE 5.11

The Tableau after 1 Iteration in Example 5.4

	1	1	0	0	0	0
	0	-1	1	-1	-2	-3
$\bar{Y} =$	1	0	0	0	1	0
	0	0	0	0	0	0
	0	0	0	0	0	0
	0	1	0	0	0	0
	0	0	1	0	0	0
	0	0	0	1	0	1
	0	0	0	0	1	0
Cost	1	1	1	1	1	1

Set $Y = \bar{Y}$. On processing row 2, $r = 2$, of $\bar{Y}$, we get (since $\eta = 1$),

TABLE 5.12

The Tableau after 2 Iterations in Example 5.4

	1	0
	0	1
$\bar{Y} =$	1	0
	0	0
	0	0
	0	0
	0	1
	0	0
	0	0

i.e. only columns 1 and 3 survive; all combinations violate
cost $\leq \eta$.

No further processing of $\bar{Y}$ is possible. The two vectors

$$\begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} \text{ and } \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} \text{ are not feasible to (5.43) - (5.45), since the}$$

(N+1)th component (the 7th component) of the vectors from

which these were derived is zero. [Step 4(b)].

Step 4(b) $Y = \hat{Y}$, $n = n + 1 = 2$.

Step 3 Process row 1; note $n = 2$

Skipping columns which will not contribute to future processing, we have

TABLE 5.13

The Tableau after 3 Iterations in Example 5.4

$\bar{Y} =$	1	1	0	0	0	0	0
	0	-1	1	-1	1	-2	-3
	1	0	0	0	1	1	0
	0	0	0	0	0	0	1
	0	0	0	0	1	0	0
	0	1	0	0	0	0	1
	0	0	1	0	0	0	0
	0	0	0	1	0	0	0
	0	0	0	0	0	1	0
Cost	1	1	1	1	2	1	2

Set $Y = \bar{Y}$. Set $r=2$; columns 1, 3, and 5 are retained as they are; the other columns are contributed by (2,3), (3,6), (3,8), and (5,6).

TABLE 5.14

The Final Tableau in Example 5.4

		1	0	0	1	0	0	0
		0	1	1	0	0	0	0
$\bar{y} =$	t_1	1	0	1	0	1	0	3
	z_1	0	0	0	0	0	0	0
	t_2	0	0	1	0	0	0	2
	z_2	0	0	0	1	0	1	0
	t_3	0	1	0	1	2	3	0
	z_3	0	0	0	0	0	0	0
	b	0	0	0	0	1	1	1
Cost		1	1	2	2	2	2	2
feasible								

Set $Y = Y$. The three alternate optima correspond to

$$\hat{\ell}_5 = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 2 \\ 0 \end{bmatrix}, \hat{\ell}_6 = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 3 \\ 0 \end{bmatrix}, \text{ and } \hat{\ell}_7 = \begin{bmatrix} 3 \\ 0 \\ 2 \\ 0 \\ 0 \\ 0 \end{bmatrix} \text{ corresponding to the}$$

solutions: $t_1=1, t_3=2; z_2=1, t_3=3$; and $t_1=3, t_2=2$ respectively.

In this chapter we examined the formulation of MWFIC. The theory leading to the Chernikova algorithm for generating all extremal vectors of a convex cone was also developed. Rubin's adaptation of the Chernikova procedure for identifying all vertices of a convex polyhedron for solving CCLP's was also examined. Algorithm VS-1, based on Sande [32], for solving MWFIC was specified together with an algorithm, VS-2, for solving MFIC. Examples 5.3 and 5.4, while

illustrating VS-1 and VS-2, also serve to illustrate the fundamental weakness of vertex enumeration as a means of solving MWFIC (MFIC). Chapter 6 begins with an analysis of algorithms VS-1 and VS-2. A different and faster algorithm for solving MWFIC and MFIC is also developed in Chapter 6.

CHAPTER 6

A CUTTING PLANE ALGORITHM FOR THE MWFIC
PROBLEM WITH CONTINUOUS DATA

In Chapter 5 a modification of the Chernikova procedure for solving MWFIC was presented. A threshold algorithm (VS-2), which progressively relaxes the constraints on retaining columns in the Chernikova tableau while producing a feasible and hence an optimal solution, was also developed for solving MFIC. In this chapter an algorithm, similar to the row generation approach in Garfinkel [12], for solving MWFIC, is developed. This algorithm solves both MWFIC and MFIC using, basically, a different approach from the algorithms presented in Chapter 5. Computational experience with this new algorithm is also given in this chapter.

Before proceeding to a detailed discussion of the new algorithm, an analysis of the algorithms of Chapter 5 is taken up.

I. Analysis of Algorithms VS-1 and VS-2

Methods for identifying all vertices of polyhedral sets were surveyed in Matheiss [25]. It was reported therein that the largest problem successfully completed using Chernikova's algorithm, on an IBM 370/168 computer with 512K (bytes) of main memory, had a constraint matrix with 25 rows and 8 columns. While it is not inconceivable that problems with associated constraint matrices larger than 25x8 could be processed using the Chernikova procedure, within 512K memory, the

tremendous amount of storage required in using the procedure is exemplified by the results in [25]. A constraint set with 10 rows and 20 columns could potentially have 6.7×10^{11} vertices, although in general the number of vertices feasible to the constraint set is much less. It was reported in [25] that about 8,100 feasible vertices were enumerated, on the average, using the Chernikova procedure, in about 30 minutes of CPU time.

Algorithms VS-1 and VS-2 of Chapter 5 are based on the Chernikova procedure for solving MWFIC and MFIC respectively. While total enumeration of the vertices of the polyhedron defined by the constraint set associated with the problem, MWFIC or MFIC, is not attempted, the rapid expansion of the column size of the matrix $Y = \begin{pmatrix} U \\ L \end{pmatrix}$ is all too evident in the examples in Chapter 5. In using the VS-1 algorithm a 'good' initial feasible solution is necessary to contain the rapid growth in the column dimension of Y . The VS-2 algorithm, on the other hand, is more appealing as a procedure for solving MFIC's as opposed to VS-1. Both procedures, however, suffer from the doubling of the number of variables from the original N , namely x_j , $j \in I_N$, to $2N$ on account of the modification in the values x_j^0 , $j \in I_N$, of a given record to $x_j^0 + t_j - z_j$, $j \in I_N$, in the MWFIC formulation. Further, non-negativity of the x_j , $j \in I_N$, forces the upper bounding of the z_j , causing additional rows to be included in Y .

In light of the findings in Matheiss [25] and the fact that the column and row sizes of Y are increased substantially by the introduction of the positive (t), and negative (z), corrections together with the non-negativity constraints on the x_j , $j \in I_N$, it appears that only very

small MWFIC and MFIC problems would be computationally feasible if the VS-1 or VS-2 procedure is used. Most practical problems are, however, considerably larger than could be efficiently solved using VS-1 or VS-2.

The algorithm to be presented in this chapter is based on a set covering approach similar to the row generation approach of Garfinkel [12] presented in Chapter 3. It is also similar to the VS-2 algorithm in that feasibility and hence optimality is achieved simultaneously. Some definitions and necessary theoretical developments are presented before stating the algorithm.

II. Some Results Leading to a Cutting Plane Algorithm for MWFIC

As in Chapter 5, we assume that a record, x , must satisfy a set of linear constraints to be called a passing record. We will further assume that the constraint set is of the form $S = \{x | Ax \leq b\}$, where A is $m \times N$, x is $N \times 1$ and b is $m \times 1$.

A record y is said to be a failing record if and only if $y \notin S$. The index set of constraints $\{1, 2, \dots, m\}$ will be denoted by I_m , while the index set of fields will continue to be denoted by I_N .

Definition If a_i^T is the i th row of A , then E_i such that $E_i = \{x | a_i^T x > b_i\}$ will be referred to as an edit. If $x \in E_i$, x is said to fail E_i . Note that if x fails E_i , $x \notin S$.

Definition $E_e = \bigcup_{i \in I_m} E_i$ is defined to be the edit set associated with S . Note that $x \in S$ if and only if $x \notin E_e$. Also, $x \in E_e$ if and only if $x \in E_i$, for some $i \in I_m$.

Definition A field k is said to enter the edit E_i if $a_{ij} \neq 0$; otherwise k is said to be uninvolved in E_i . For example, if the i th constraint in S is

$$2x_1 + 3x_4 - 5x_6 \leq -7$$

with $I_N = \{1, \dots, 6\}$, then fields 2, 3, and 5 are uninvolved in E_i , while fields 1, 4, and 6 enter E_i . Fields that enter matrices (FEM) are defined similar to (2.8).

The FEM associated with an edit set $\bar{E}$ is denoted by $F = (f_{ij})$ where

$$f_{ij} = \begin{cases} 0 & \text{if } a_{ij} = 0 \\ 1 & \text{otherwise} \end{cases} \quad (6.1)$$

In this chapter our goal continues to be one of locating a least cost change of the entries in a failing record to make it pass all the edits in $\bar{E}_e$. It was observed in Section 2.3 that locating such a least cost change is related to the location of the fields most likely in error in the record. It was also observed in Chapter 2 that this error localization problem can be formulated as an SCP, (see (2.9)), based on the edits that are failed by the record. As with coded data, solution of such an SCP does not always result in locating a set of fields that can be changed to render the record passing, as Example 6.1 illustrates. A realization of the record x will be denoted by $x^0 = (x_1^0, x_2^0, \dots, x_N^0)$.

Example 6.1 Consider the constraint set $S = \{x | Ax \leq b, x \geq 0\}$, where $Ax \leq b$ is represented by

$$2x_1 + x_2 \leq 5 \quad (6.2)$$

$$-x_2 + x_3 \leq 4 \quad (6.3)$$

$$-x_1 + x_3 \leq -2 \quad (6.4)$$

and $I_N = \{1, 2, 3\}$. (Note that there are 3 more edits associated with $x \geq 0$).

Let the record x^0 be $(3, 0, 5)$. The edits associated with (6.2) - (6.4) are

$$E_1 : \quad 2x_1 + x_2 > 5 \quad (6.2a)$$

$$E_2 : \quad -x_2 + x_3 > 4 \quad (6.3a)$$

$$E_3 : \quad -x_1 + x_3 > -2 \quad (6.4a)$$

Clearly, x^0 fails E_1 , E_2 , and E_3 . The associated FEM is then

$$F = \begin{bmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 1 \end{bmatrix} \quad (6.5)$$

If the cost of changing the three fields is assumed to be uniform then the SCP associated with the record x is:

$$\begin{aligned} \text{Min} \quad & w_1 + w_2 + w_3 \\ & w_1 + w_2 \geq 1 \\ & w_2 + w_3 \geq 1 \\ & w_1 + w_3 \geq 1 \end{aligned} \quad (6.6)$$

$$w_j = \begin{cases} 1 & \text{if field } j \text{ is to be changed} \\ 0 & \text{otherwise,} \end{cases} \quad j \in I_N.$$

An SCP solution is $w_1 = w_2 = 1$ but it is easy to verify that changing x_1^0 and x_2^0 without changing x_3^0 will not result in a passing record. In particular (6.4) requires that $x_1 \geq 7$, if x_3^0 is left unchanged at 5, while (6.2) requires $x_1 \leq 5/2$, given that $x_2 \geq 0$.

Example 6.1 illustrates that an SCP formulation, using the original set of edits E_e , will not always result in a set of fields being located, which can be changed to arrive at a passing record. As with coded data, this situation points to the existence of new edits, in

this case implied by (6.2) - (6.4) and the non-negativity of the x_j , $j \in I_N$.

The continuous data situation under consideration, while similar to the coded data problem, poses some difficulty in the generation of new constraints using $\bar{E}_e$. Fellegi and Holt [9] treat the continuous data problem as, essentially, being no different from the coded data situation and they attempt to express continuous data edits in normal form. For example, they express an edit such as $x_1 + x_2 - x_3 > 0$ in normal form as $\{x_1: x_1 > x_3 - x_2\} \times \{x_2\} \times \{x_3\}$, for all x_2, x_3 . Representations of edits, in the continuous data situation, such as these, while theoretically sound, are impractical to adopt in attempting to generate implied edits using Lemma 2.1. The procedure in Lemma 2.1 requires the use of set unions and intersections in generating an implied edit, while it is quite difficult to express the components of E in the form of sets of values for x_1 , x_2 and x_3 . Given that the implied edit generation procedures in Chapters 2 and 3 are dependent on the existence of edits in normal form, we need to specify alternative procedures to generate implied edits (constraints) with continuous data.

Definition Given a constraint set S and the associated set of edits $\bar{E}_e$, an edit $E_r: a_r^T x > b_r$ (constraint $S_r: a_r^T x \leq b_r$) is said to be an implied edit (constraint) if,

- (i) $a_{rj} \neq 0$, for at least one $j \in I_N$
- (ii) $x \in E_r (x \notin S_r) \Rightarrow x \in \bar{E}_e (x \notin S)$.

A procedure was specified in Fellegi and Holt [] for generating implied edits with continuous data. We need one more definition before stating Theorem 6.1.

Definition An edit E_r is said to be an essentially new implied edit of a set of edits $\bar{E}$, generated on field j , if and only if E_r is implied by $\bar{E} = \{E_i\}$, $i \in I_m$, with $a_{rj} = 0$, and $a_{ij} \neq 0$, for all $i \in I_m$. Field j will be referred to as the generating field and edits such as E_r will be denoted by $G(j, \bar{E})$.

Example 6.2 Consider the edits E_1 and E_2 of Example 6.1. Let $\bar{E} = E_1 \cup E_2$. $E_4: 2x_1 + x_3 > 9$ is implied by $\bar{E}$, which is easily seen by adding (6.2) and (6.3). Further E_3 is essentially new since $a_{42} = 0$, while a_{12} and a_{22} are both non-zero.

Theorem 6.1

An essentially new implied edit $E_t = G(j, \{E_r, E_s\})$ can be generated by taking a linear combination of E_r and E_s if and only if $a_{rj}a_{sj} < 0$.

Proof ($\Leftarrow$) By definition, $E_t = G(j, \{E_r, E_s\})$ requires that a_{rj} and a_{sj} be non-zero. If $a_{rj}a_{sj} > 0$ then no non-negative linear combination of E_r and E_s can result in E_t and negative linear combinations will not result in a valid edit (inequality). Thus, if $E_t = G(j, \{E_r, E_s\})$ can be constructed then it must be that $a_{rj}a_{sj} < 0$.

($\Rightarrow$) If $a_{rj}a_{sj} < 0$ then $\alpha, \beta > 0$ can be chosen so that $E_t = \alpha E_r + \beta E_s$. One such choice of α and β is $\alpha = |a_{sj}|$, $\beta = |a_{rj}|$. E_t may be written as,

$$E_t : \alpha \sum_{\substack{k \in I_N \\ k \neq j}} a_{rk} x_k + \beta \sum_{\substack{k \in I_N \\ k \neq j}} a_{sk} x_k > \alpha b_r + \beta b_s. \quad (6.7)$$

We need to show that if $x^0 \in E_t$, then $x^0 \in E_r \cup E_s$. If $x^0 \in E_t$ but $x^0 \notin E_r \cup E_s$ then x^0 must be such that $\sum_{k \in I_N} a_{rk} x_k^0 \leq b_r$ and $\sum_{k \in I_N} a_{sk} x_k^0 \leq b_s$ leading to $x^0 \notin E_t$.

Hence E_t is an essentially new implied edit.

QED.

The essentially new implied edit E_3 of Example 6.2 is derivable from edits E_1 and E_2 using field 2 as the generating field. It was seen in Example 6.1 that changing fields 1 and 2 of record x^0 without changing field 3 will not lead to a passing record. In Lemma 6.1 it is demonstrated that if a field j cannot be changed to make the record passing then there exists an (essentially new implied) edit with field j uninvolved which must be failed by x^0 .

Lemma 6.1

If a record $x^0 \in E = E_r \cup E_s$ and changing the entry in field k of x^0 , x_k^0 , will not result in $x \notin E$ then,

$$(i) \quad a_{rk} = 0 \quad \text{or} \quad a_{sk} = 0$$

or

(ii) an essentially new implied edit

$$E_t : \sum_{\substack{j \in I_N \\ j \neq k}} (|a_{rk}|a_{sj} + |a_{sk}|a_{rj})x_j > |a_{rk}|b_s + |a_{sk}|b_r \quad (6.8)$$

can be constructed from E_r and E_s so that $x \in E_t$.

Proof

We have two cases to consider

Case 1 $a_{rk}a_{sk} \geq 0$.

If $a_{rk}a_{sk} > 0$ then there exists an x_k^* so that $x^* = (x_1^0, \dots, x_{k-1}^0, x_k^*, x_{k+1}^0, \dots) \notin \bar{E}$, contradicting the fact that $x^* \in \bar{E}$ for all x_k . Such an x_k^* could have been produced by choosing x_k^* so that $a_{rk}x_k^* \leq b_r - \sum_{\substack{j \in I_N \\ j \neq k}} a_{rj}x_j^0$ and $a_{sk}x_k^* \leq b_s - \sum_{\substack{j \in I_N \\ j \neq k}} a_{sj}x_j^0$. Since $a_{rk} \cdot a_{sk} > 0$ such an x_k^* can always be chosen. If $a_{rk} \cdot a_{sk} = 0$ then (i) is clearly true.

Case 2 $a_{rk}a_{sk} < 0$.

From Theorem 6.1, a linear combination of E_r and E_s will

result in an essentially new implied edit E_t , with field k uninvolved, as represented in (6.8). We need to show that $x^0 \in E_t$. Assume the contrary. Then $x^0 \notin E_t$ or equivalently,

$$\sum_{\substack{j \in I_N \\ j \neq k}} (|a_{rk}|a_{sj} + |a_{sk}|a_{rj}) x_j^0 \leq |a_{rk}|b_s + |a_{sk}|b_r. \quad (6.9)$$

Since $|a_{rk}|a_{sk} + |a_{sk}|a_{rk} = 0$ we may rewrite (6.9) as

$$\sum_{j \in I_N} (|a_{rk}|a_{sj} + |a_{sk}|a_{rj}) x_j^0 \leq |a_{rk}|b_s + |a_{sk}|b_r. \quad (6.10)$$

If x^0 is such that $x^0 \in E_r \cap E_s$ then, clearly, (6.10) does not hold and $x^0 \in E_t$. In the following we assume w.l.o.g. that $a_{rk} < 0$ and $a_{sk} > 0$. If $x^0 \in E_s$ but $x^0 \notin E_r$, then $x^0 \notin E_r$ implies that,

$$|a_{sk}|a_{rk} x_k^0 + |a_{sk}| \left(\sum_{\substack{j \in I_N \\ j \neq k}} a_{rj} x_j^0 \right) - |a_{sk}|b_r = \delta \leq 0. \quad (6.11)$$

Since we assumed that $x^0 \notin E_t$, (6.10) holds and from (6.11) may be rewritten as

$$|a_{rk}| a_{sk} x_k^0 + \left(\sum_{\substack{j \in I_N \\ j \neq k}} |a_{rk}| a_{sj} x_j^0 \right) + \delta \leq |a_{rk}| b_s \quad (6.12)$$

or,

$$b_s^* = b_s - \sum_{\substack{j \in I_N \\ j \neq k}} a_{sj} x_j^0 - a_{sk} x_k^0 \geq \frac{\delta}{|a_{rk}|}. \quad (6.13)$$

Decreasing x_k^0 increases the value of δ and clearly δ can be made equal to zero by a suitable choice of x_k^0 . The inequality (6.13) is preserved as long as δ is defined by (6.11). $x^0 \in E_s$ implies that $b_s^* < 0$ while the fact that x_k^0 exists such that $\delta=0$ implies that $b_s^* \geq 0$, from (6.13), contradicting $x^0 \in \bar{E}$ for all x_k^0 . Thus $x^0 \in E_t$.

QED.

Example 6.3 Consider the edit set

$$E_1 : x_1 + x_2 > 8 \quad (6.14)$$

$$E_2 : -x_1 + x_3 > 6 \quad (6.15)$$

$$E_3 : -x_1 > 0, \quad E_4 : -x_2 > 0, \quad E_5 : -x_3 > 0 \quad (6.16)$$

A record $x^0 = (1, 7, 8)$ fails E_2 . If we look at the associated SCP two covers are possible solutions. If the cost of changing field 1 is 1 and of changing field 3 is 2 then $w_1 = 1$ is the lowest solution. However, changing x_1^0 alone will not result in $x^0 \notin \bar{E} = E_1 \cup E_2$. For, (6.14) and (6.15), with $x_2^0 = 7$ and $x_3^0 = 8$, imply

$$x_1 > 1 \quad (6.17)$$

and

$$x_1 < 2 \quad (6.18)$$

Adding E_1 and E_2 we may generate an essentially new implied edit,

$$E_3 : x_2 + x_3 > 14 \quad (6.19)$$

Note that $x^0 \in E_3$.

Intuitively it is evident that the result of Lemma 6.1 should hold if k fields are to be changed instead of just one. That is, an implied edit E_I , with k fields uninvolved, should be constructible so that $x^0 \in E_I$. Motivation for developing such a device stems from a similar procedure in Step 5 of Algorithm 3.2. If each of the k fields enters at least one edit in $\bar{E}$ then it is shown in Theorem 6.2 that E_I can be constructed. Theorem 6.2 is based on a result in Kohler [20] pointed out by Liepins [23].

Theorem 6.2

Suppose a record x^0 fails each edit in $\bar{E} = \{E_i\}$, $i=1, \dots, r$, $\bar{E} \subseteq E_e = \{x | Ax > b\}$, the set of m ($\geq r$) originally specified edits and $x^0 \notin E_e - \bar{E}$. Let the field indices $I_k = \{1, \dots, k\}$, $I_k \subseteq I_N$ be a prime cover of the rows of the FEM set up with the edits $\bar{E}$. Partition A and x into $A = [B \ C]$ and $x = \begin{pmatrix} y \\ z \end{pmatrix}$ with $B = (A_1, \dots, A_k)$, the k columns of A indexed by I_k , $C = (A_{k+1}, \dots, A_N)$, the other $N-k$ columns of A , $y = \begin{pmatrix} x_1 \\ \vdots \\ x_k \end{pmatrix}$ and $z = \begin{pmatrix} x_{k+1} \\ \vdots \\ x_N \end{pmatrix}$. If changing the fields indexed by I_k cannot result in $x^0 \notin E_e$ then there exists an m vector p so that

$$p^T B = 0, \quad p^T b - p^T C z^0 < 0, \quad \text{and } p \geq 0 \quad (6.20)$$

where z^0 is the instance of z corresponding to x^0 .

Proof E_e can be written as $By + Cz > b$ which with $p \geq 0$ implies that

$$p^T (By + Cz) > p^T b. \quad (6.21)$$

Consider $P: \quad n = \max 0y, \quad By \leq b - Cz^0 = k$. From duality theory the dual D of P may be written as $D: \quad \delta = \min p^T k, \quad p^T B = 0, \quad p \geq 0$. If the only solution to D is $p = 0$ then clearly $By \leq k$ is feasible contradicting the fact that there exists no y such that $By \leq k$. If $p \geq 0, p \neq 0$ is a solution to D , two possibilities arise. (i) δ is bounded or (ii) δ is unbounded.

Case (i) If δ is bounded then there exists a y such that $By \leq k$ and again we have a contradiction.

Case (ii) If δ is unbounded then P is infeasible, implying that there exists no y such that $By \leq k$.

Thus Case (ii) must hold so that $p^T k < 0, p^T B = 0, p \geq 0$ is feasible. QED.

Theorem 6.2 established the existence of an essentially new implied edit E_t with at least k fields uninvolved, namely,

$$E_t : p^T C z > p^T b \quad (6.22)$$

whenever a given failing record x cannot be made to pass all edits in E_e by changing the k fields (that are uninvolved in (6.22)) in isolation of the rest of the fields. Example 6.4 illustrates the content of Theorem 6.2.

Example 6.4 Consider the edit system E defined in Example 6.1.

$$E_1 : 2x_1 + x_2 > 5 \quad (6.2a)$$

$$E_2 : -x_2 + x_3 > 4 \quad (6.3a)$$

$$E_3 : -x_1 + x_3 > -2 \quad (6.4a)$$

$$E_4 : -x_1 > 0 \quad (6.23a)$$

$$E_5 : -x_2 > 0 \quad (6.23b)$$

$$E_6 : -x_3 > 0 \quad (6.23c)$$

The record x^0 was $(3,0,5)$ and it was observed that changing fields 1 and 2 without changing field 3 will not result in a passing record. In the terminology of Theorem 6.2, we have B , C and b as in Table 6.1.

TABLE 6.1

Initial Setup in Example 6.4

$B =$	2	1	$C =$	0	$b =$	5
	0	-1		1		4
	-1	0		1		-2
	-1	0		0		0
	0	-1		0		0
	0	0		-1		0

A solution to (6.20) is given by $p^T = [1 \ 1 \ 2 \ 0 \ 0 \ 0]$ yielding,

$$E_7 : 3x_3 - 5 > 0 . \quad (6.23d)$$

Thus x_3^0 needs to be changed before x^0 can be made to pass E .

Solution of (6.20) is simplified by the removal of the strict inequality as established in the following lemma.

Lemma 6.2

A solution q^* to

$$\begin{aligned} q^T B &= 0 \\ q^T C z^0 - q^T b &\geq \alpha \\ q &\geq 0 \\ \alpha &> 0 \text{ and constant, solves (6.20).} \end{aligned} \quad (6.24)$$

Proof If q^* is a solution to (6.24) so that

$$q^{*T} C z^0 - q^{*T} b = \alpha^* \geq \alpha > 0 \text{ then clearly } p = q^* \text{ solves (6.20) as well.}$$

QED.

Thus it suffices to locate q for $\alpha=1$ in arriving at an essentially new implied edit E_t , defined by (6.20), so that $x \in E_t$. Example 6.5 illustrates the implied edit construction procedure suggested in Lemma 6.2.

Example 6.5 Consider the situation in Example 6.4.

Using the content of Lemma 6.2, we need a solution to

$$[q_1 \ q_2 \ q_3 \ q_4 \ q_5 \ q_6] \begin{bmatrix} 2 & 1 \\ 0 & -1 \\ -1 & 0 \\ -1 & 0 \\ 0 & -1 \\ 0 & 0 \end{bmatrix} = 0 \quad (6.25)$$

and

$$[q_1 \dots q_6] \begin{bmatrix} 0 \\ 1 \\ 1 \\ 0 \\ 0 \\ -1 \end{bmatrix} x_3^0 - [q_1 \dots q_6] \begin{bmatrix} 5 \\ 4 \\ -2 \\ 0 \\ 0 \\ 0 \end{bmatrix} \geq 1.$$

Rewriting (6.25) in equation form, by the introduction of a surplus variable, s_1 , in the inequality and substituting $x_3^0=5$, we have

$$\begin{aligned} 2q_1 - q_3 - q_4 &= 0 \\ q_1 - q_2 - q_5 &= 0 \\ -5q_1 + q_2 + 7q_3 - 5q_6 - s_1 &= 1 \\ q_j &\geq 0, \quad j \in I_6 \\ s_1 &\geq 0. \end{aligned} \quad (6.26)$$

Note that a basic solution is $q_1 = q_2 = 0$, $s_1 = -1$, which is infeasible.

(6.26) with q_1 eliminated from the second row and q_1 and q_2 eliminated from row 3 looks thus:

$$\begin{aligned} q_1 - \frac{1}{2}q_3 - \frac{1}{2}q_4 &= 0 \\ q_2 - \frac{1}{2}q_3 - \frac{1}{2}q_4 + q_5 &= 0 \\ \frac{9}{2}q_3 - \frac{5}{2}q_4 + q_5 - 5q_6 - s_1 &= 1. \end{aligned} \quad (6.26a)$$

Introducing q_3 into the basis and removing s_1 , we obtain a basic feasible solution involving q_1 , q_2 and q_3 .

$$\begin{aligned}
 q_1 \quad & -\frac{7}{18}q_4 + \frac{1}{9}q_5 - \frac{5}{9}q_6 - \frac{1}{9}s_1 = \frac{1}{9} \\
 q_2 \quad & -\frac{7}{18}q_4 + \frac{10}{9}q_5 - \frac{5}{9}q_6 - \frac{1}{9}s_1 = \frac{1}{9} \\
 q_3 \quad & -\frac{5}{9}q_4 + \frac{2}{9}q_5 - \frac{10}{9}q_6 - \frac{2}{9}s_1 = \frac{2}{9}
 \end{aligned} \tag{6.26b}$$

(6.25b) represents the basic feasible solution to (6.25) with

$q_1 = q_2 = \frac{1}{9}$, $q_3 = \frac{2}{9}$, $q_4 = q_5 = q_6 = 0$. This produces an implied edit $q^T C z - q^T b > 0$,

$$[1 \ 1 \ 2 \ 0 \ 0 \ 0] \begin{bmatrix} 0 \\ 1 \\ 1 \\ 0 \\ 0 \\ -1 \end{bmatrix} x_3 - [1 \ 1 \ 2 \ 0 \ 0 \ 0] \begin{bmatrix} 5 \\ 4 \\ -2 \\ 0 \\ 0 \\ 0 \end{bmatrix} > 0$$

$$\text{i.e.} \quad 3x_3 - 5 > 0. \tag{6.27}$$

The given record with $x^0 = (3, 0, 5)$ does fail (6.27). The introduction of this edit into the SCP for solving the MFIC in Example 6.1 immediately points to changing x_3 to be less than or equal to $5/3$. The modified SCP is then,

$$\begin{aligned}
 \text{Min} \quad & w_1 + w_2 + w_3 \\
 & w_1 + w_2 \geq 1 \\
 & w_2 + w_3 \geq 1 \\
 & w_1 + w_3 \geq 1 \\
 & w_3 \geq 1 \\
 & w_1, w_2, w_3 = 0, 1.
 \end{aligned} \tag{6.28}$$

A solution to (6.28) is $w_1 = w_3 = 1$, i.e. change fields 1 and 3. The constraint set S rewritten to determine the feasibility of such a change is

$$\begin{aligned}
x_1 &\leq 5 - 2x_2^0 = 5 \\
x_3 &\leq 4 + x_2^0 = 4 \\
-x_1 + x_3 &\leq -2 \\
x_1, x_3 &\geq 0
\end{aligned} \tag{6.29}$$

Clearly, a solution to (6.29) is to have $x_1 = 5$, $x_3 = 3$. Thus a feasible record is $x' = (5, 0, 3)$. The procedure outlined in Example 6.5 is formalized in the following algorithm.

III. A Cutting Plane Algorithm for Solving MWFIC

Given a set of m constraints $S = \{x | Ax \leq b\}$ and the associated edit set $\bar{E}_e = \{x | x \notin S\}$, if $x = (x_j)$, $j \in I_N$ is such that $x \notin S$ then a passing record x' can be produced, if one exists, from x by using the following procedure. The cost of changing field j is assumed to be $c_j \geq 0$.

Algorithm 6.1

1. If $x^0 \in \bar{E}_e$ go to step 1a; if not STOP.
- 1a. Let $\bar{E} = \bigcup_{i \in I_k} E_i$, $E_i \in \bar{E}_e$ denote the set of k edits failed by x^0 .
2. Construct the FEM, F , corresponding to $\bar{E}$ and solve the SCP

$$\begin{aligned}
\text{Min } & \sum_{j \in I_N} c_j w_j \\
& \sum_{j \in I_N} f_{ij} w_j \geq 1, \quad i \in I_k
\end{aligned} \tag{6.30}$$

$$w_j = \begin{cases} 1 & \text{if field } j \text{ is changed} \\ 0 & \text{otherwise} \end{cases} .$$

Solve (6.30) to obtain $w^* = (w_j^*)$, $j \in I_N$.

3. Partition the matrix A into $A = [B \ C]$, where B corresponds to the columns indexed by $\{j | w_j^* = 1\}$, and C contains the rest of the

columns of A . Let x be similarly partitioned into $x = \begin{pmatrix} y \\ z \end{pmatrix}$.

Solve

$$By \leq b - Cz^0 \quad (6.31)$$

for y , where $z_j^0 = x_j^0$, $j \notin \{j | w_j^* = 1\}$. If a solution exists to (6.31), STOP; we have obtained a feasible and hence optimal solution. If no solution exists to (6.31), then go to Step 4.

4. Solve for q , the system of inequalities

$$\left. \begin{array}{l} q^T B = 0 \\ q^T C z^0 - q^T b \geq 1 \\ q \geq 0 \end{array} \right\} \quad (6.32)$$

Let q^* be a solution to (6.32). Generate the implied edit

$$E_q : q^T C z > q^T b \quad (6.33)$$

and let

$$\bar{E} = E \cup E_q \quad (6.34)$$

Go to Step 2.

Theorem 6.3

Algorithm 6.1 solves MWFIC.

Proof Assume the contrary. Let an optimal solution be to change the fields indexed by $J^* \subseteq I_N$, with associated cost c^* , and let c be the cost of the solution obtained using Algorithm 6.1. The edits constructed using (6.32) are a set of valid implied edits, say $\bar{E}_I$. The optimality of c^* implies that $x^* \notin \bar{E}_I$. However no solution to the SCP (6.30) with $\bar{E}_I$ included has value lower than c contradicting $x^* \notin \bar{E}_I$ and $c^* < c$.

QED.

Computational experience with Algorithm 6.1 is the content of the next section.

IV. Computational Experience with Algorithm 6.1

Algorithm 6.1 was coded in ASCII FORTRAN and was tested on a UNIVAC 1100/81 computer at the University of Wisconsin-Milwaukee. As with Algorithm 3.2, the set covering solution required in Algorithm 6.1 was obtained using the Bellmore-Ratliff Algorithm [1]. The determination of feasibility of (6.31) was done using the simplex method. The system (6.32) was also solved using the simplex method. The following heuristic procedure was used to assign weights (costs) to the q_i , $i = 1, \dots, m$, associated with the rows of B.

$$\text{Cost of } q_i = \text{number of non-zero elements in the } i\text{th row of B.} \quad (6.35)$$

Given that many distinct q 's exist as solutions to (6.32), the device (6.35) attempts to locate a q which excludes rows of B with a large number of non-zero components from participating in the construction of the new implied edit (6.33). What is being attempted here is the construction of an implied edit (6.33) which is failed by the existing record, with as many fields uninvolved as possible, so that a deep cut is added to the SCP in (6.30), resulting possibly in fewer number of iterations of the algorithm. While no appreciable gains were noticed while using this heuristic as opposed to not using it with problems reported on in Table 6.2, it is submitted here that the structure of the problems would appear to largely determine the number of iterations the algorithm must go through before the record can be made passing. Further the use of the procedure did not increase the CPU time for solution of the problems reported in Table 6.2.

The abbreviations used in Table 6.2 are as below.

AVEDIT : Average number of edits failed
 MAXED : Maximum number of edits failed
 AVITER : Average number of iterations to get a passing record
 MAXITER : Maximum number of iterations to get a passing record
 AVTIME : Average CPU time in milliseconds to process a record
 MAXTIME : Maximum CPU time in milliseconds to process a record

Each row of Table 6.2 corresponds to the processing of 100 failing records. The constraint matrices as well as the records were generated using a uniform random number generator. All problems generated were assumed to have no equality constraints.

TABLE 6.2
Computational Performance of the Cutting Plane
Algorithm for MWFIC

S.No.	No. of Edits	No. of Fields	AVEDIT	MAXED	AVITER	MAXITER	AVTIME	MAXTIME
1	8	4	5	6	3.3	4	145	290
2	11	6	4	8	2.1	11	210	4357
3	4	4	3	3	9.3	11	691	1275
4	13	4	4	7	6.9	9	559	877
5	7	13	3	5	2.8	6	176	345
6	9	7	6	7	8.6	12	1822	4485
7	9	10	5	6	5.3	9	1900	9780
8	12	7	7	10	4.3	8	1567	6244
9	7	7	4	5	2.0	2	153	268
10	12	9	6	7	6.5	11	1317	4676
11	7	10	4	5	2.4	6	311	3090
12	4	18	2	4	0.0	0	18	25

No clear relationship appears to exist between the average number of edits failed and the average solution time. Problem sets 1,2,3,5 and 6 illustrate the situation. Further, no clear relationship seems to exist between the average proportion of edits failed to the number of original edits and the CPU times. The number of fields do not also seem to have any significant correlation with solution times. The average number of iterations per record in this series of problems is (surprisingly) high pointing either to the algorithm generating shallow cuts or to the problems intrinsically being very tightly constrained. Observation of the intermediate processes, namely the mechanism and the nature of generation of the cuts in many instances during the computation points to the original constraint system being very tightly bound rather than to the algorithm generating shallow cuts. Solution of problems with 20 edits and 20 fields was attempted with no success in under 1 minute of CPU time.

It is worth mentioning at this juncture that problems of sizes comparable to the problem sets 5 and 12 of Table 6.2 solved at the Bureau of the Census [16], using McKeown's branch and bound technique on a UNIVAC 1100/81 computer, took an average of 6,000 milliseconds per record. However, even Algorithm 6.1 can clearly not be used with larger problems unless further research is conducted towards exploiting problem structure if any, and efficient procedures devised to reduce the execution time to less than 5 seconds per record. A heuristic procedure proposed in Chapter 7 appears to perform very well in terms of both execution time and accuracy of solutions.

CHAPTER 7

A HEURISTIC ALGORITHM FOR SOLVING MWFIC

Algorithms for solving MWFIC (and MFIC) were proposed in Chapters 5 and 6. An analysis of the VS-1 and VS-2 algorithms proposed in Chapter 5 revealed that they would be of little practical value since they require excessive amounts of both computer time and storage to solve even small problems. Further the intermediate solutions obtained in using VS-1 or VS-2 are usually not feasible to the constraint set taken in its entirety. Thus the VS-1 or VS-2 procedure cannot be halted before their logical termination to obtain a feasible approximate solution. The performance of Algorithm 6.1, while encouraging on small problems, is not acceptable on problems with size larger than 20 edits and 20 fields. Since Algorithm 6.1 progresses through infeasible solutions and terminates whenever a feasible (and hence optimal) solution is obtained, no approximate feasible solutions can be obtained prior to normal termination. A heuristic procedure for solving MWFIC (and MFIC) is proposed in this chapter which on testing was found to work well on problems which could not be efficiently solved using Algorithm 6.1. A comparative analysis of the computational performance of the heuristic procedure against the performance of Algorithm 6.1 is also included in this chapter. The notation and terms defined in Chapters 5 and 6 continue to have the same interpretation as before, in this chapter, unless otherwise defined.

I. Analysis of Algorithm 6.1

Algorithm 6.1 is an iterative procedure for processing a failing record and is aimed at generating only those implied edits (and not necessarily all such) that are failed by the record. Computationally, the most expensive step of the algorithm is Step 2 in which an SCP is to be solved at each iteration. Algorithm 6.1 is likely to perform very well (and in the test problems reported in Table 6.1 it does) whenever the number of iterations and hence the number of SCP's that have to be solved is very small. In tightly constrained edit systems it is probable that many implied edits may have to be generated before MWFIC associated with a given failing record is solved. Also, with a large number of edits being failed (10 or more) by the record and a large number of fields (10 or more) even two iterations can be very expensive computationally, since the associated SCP becomes very expensive to solve. Thus alternative procedures, even if suboptimal, need to be explored to 'solve' such problems. It is worth noting at this stage that given the continuous nature of the data associated with the constraint set in an MWFIC, a feasible solution to MWFIC can be obtained at very low computational expenditure by using procedures based on the simplex method. Once such a feasible solution is obtained, it is again computationally inexpensive to seek an adjacent extreme point solution with lower associated MWFIC solution cost. This is the motivation behind the heuristic proposed in this chapter.

II. The SWIFT Heuristic for the Fixed Charge Problem

Cooper and Drebes [7], Steinberg [34], Denzler [8], and Cooper [6] have developed approximate heuristic adjacent extreme

point algorithms for the general fixed charge problem, (5.10a). These algorithms follow a generic procedure of starting from an initial feasible solution and proceeding to decrease the solution value by moving to an adjacent extreme point if possible. Walker [38] extended their work and incorporated into his SWIFT heuristic, procedures for jumping from the current extreme point solution to extreme points not immediately adjacent to the current solution by forcing 2 basic variable changes at each iteration of the heuristic in an attempt to reduce the solution value. The SWIFT heuristic used to solve problems, see [38], with 5 linear constraints and 10 variables solved the associated fixed charge problem in under 6 seconds of computer time on IBM 360/65 and CDC 1604 computers. Eight fixed charge problems with 15 constraints and 30 variables were solved optimally using between 18 and 60 seconds of CPU time on the same computers. The SWIFT heuristic begins processing with an initial basic feasible solution. On the other hand an MWFIC problem formulated for a failing record will be primal infeasible and also will contain upper bound constraints. The heuristic proposed in Section III is similar in spirit to the SWIFT and other heuristics using an adjacent extreme point search for locating 'better' solutions.

III. A Heuristic Algorithm for Solving the Continuous Data Error Localization Problem

The MWFIC problem defined in (5.3) - (5.8) is restated here for ready reference.

$$\text{Minimize } f = c^T \delta(t) + \delta(z) \quad (7.1)$$

$$\text{subject to: } A(x^0 + t - z) \leq b \quad (7.2)$$

$$x^0 + t - z \geq 0 \quad (7.3)$$

$$t^T \cdot z = 0 \quad (7.4)$$

$$t, z \geq 0, \quad (7.5)$$

where x^0 is the N-vector of field values in the failing record, t and $-z$ are N-vectors of positive and negative corrections to be applied to the values in x^0 to enable the record to satisfy the constraints defined by $S = \{x | Ax \leq b, x \geq 0\}$, where A is $m \times N$ and b is $m \times 1$. Also,

$$\delta(t_j) = \begin{cases} 1 & \text{if } t_j > 0 \\ 0 & \text{otherwise} \end{cases} \quad j \in I_N \quad (7.6)$$

$$\delta(z_j) = \begin{cases} 1 & \text{if } z_j > 0 \\ 0 & \text{otherwise} \end{cases} \quad j \in I_N, \quad (7.7)$$

and c is the N-vector of costs of changing the fields indexed by I_N . The problem (7.1)-(7.5) is a mixed integer linear program with the integer variables, $\delta(t)$ and $\delta(z)$, affecting only the objective function (7.1) directly. The constraints (7.2)-(7.5) may be rewritten as,

$$A(t - z) + s = b - Ax^0 = b^0 \quad (7.7)$$

$$z \leq x^0 \quad (7.8)$$

$$t^T \cdot z = 0 \quad (7.9)$$

$$t, z, s \geq 0 \quad (7.10)$$

where it is assumed that the MWFIC is formulated for a failing record x^0 with $x_j^0 \geq 0, j \in I_N$. In (7.7) some of the components of b^0 will be negative if x^0 is a failing record. The heuristic for solving (7.7)-(7.10) is stated below and is illustrated with an example before it is

explained. The heuristic handles the complementarity condition (7.9) implicitly.

Heuristic 7.1

0. Start with an initial basic (infeasible) solution to (7.7)-(7.10) with the slack variables $s_i, i \in I_m$ all in the basis. Generate a basic feasible solution (one procedure that can be used efficiently is specified in Section 7.5). Let the feasible solution set be denoted by y^* with an associated cost, in (7.1), of f^* . Record this solution in $G = (y^*, f^*)$.
- 1(a) Let S' denote the set of slack variables not currently in the solution y^* .
- 1(b) If $S' = \emptyset$ go to Step 2. Otherwise introduce a slack variable $s^0 \in S'$ into the basic feasible solution y^0 (removing an existing basic variable) if the associated cost in (7.1) f^0 of y^0 satisfies $f^0 < f^*$; set $y^* = y^0$, $f^* = f^0$, and $G = (y^*, f^*)$. Repeat Step 1(b) after setting $S' = S' - \{s^0\}$.
- 2(a) Let V denote the set of non-basic variables which are not slack variables.
- 2(b) If $V = \emptyset$ STOP, G contains the heuristic solution. Otherwise include $p \in V$ in the basis if the associated solution y^p has cost, in (7.1), of $f^p < f^*$; set $y^* = y^p$, $f^* = f^p$, $G = (y^*, f^*)$. Set $V = V - \{p\}$. Repeat Step 2(b).

Note: Whenever the cost of a solution is to be determined and $t_j > 0$ and $z_j = x_j$, some $j \in I_N$, only one of $\delta(t_j)$ and $\delta(z_j)$ is set to 1 and only if $|t_j - z_j| > 0$.

Example 7.1 The constraint set $Ax \leq b$ is represented by

$$x_1 + x_2 - x_3 + x_4 \leq 10 \quad (7.11)$$

$$2x_1 + 3x_3 - 4x_4 \leq 2 \quad (7.12)$$

$$3x_2 + 2x_3 + 2x_4 \leq 14 \quad (7.13)$$

$$-3x_1 - 2x_3 + 6x_4 \leq -1 \quad (7.14)$$

$$x_1, x_2, x_3, x_4 \geq 0 \quad (7.15)$$

The costs of changing fields 1 to 4 are 4, 2, 8 and 12 respectively;

i.e. $c = (4, 2, 8, 12)$. The record $x^0 = (0, 3, 2, 2)$ does not satisfy

(7.13) and (7.14). Rewriting the constraint set in the form (7.7),

(7.8), (7.10) for the record x^0 , we have

$$t_1 - z_1 + t_2 - z_2 - t_3 + z_3 + t_4 - z_4 + s_1 = 7 \quad (7.16)$$

$$2t_1 - 2z_1 + 3t_3 + 3z_3 + 4t_4 + 4z_4 + s_2 = 4 \quad (7.17)$$

$$3t_2 - 3z_2 + 2t_3 - 2z_3 + 2t_4 - 2z_4 + s_3 = -3 \quad (7.18)$$

$$-3t_1 + 3z_1 - 2t_3 + 2z_3 + 6t_4 - 6z_4 + s_4 = -9 \quad (7.19)$$

$$t, z, s, \geq 0 \quad (7.20)$$

$$z_1 \leq 0, z_2 \leq 3, z_3 \leq 2, z_4 \leq 2 \quad (7.21)$$

Notice that a simplex tableau set up with the constraints (7.16)-(7.19) implicitly taking into consideration the non-negativity and the upper bound constraints on the t, z and s variables with only the slack variables s_1 to s_4 in the basis will be primal infeasible. Using the Phase I procedure in the simplex method, see [24], we arrive at the following abbreviated tableau (note that all costs are assumed zero at this stage).

Table 7.1
Initial Simplex Tableau in Example 7.1

Basic Variable	Solution Value	Non-basics				z_1	s_2	s_2	s_4
		t_2	t_3	t_4					
s_1	29/10	-7/2	0	0	0	14/5	-3/2	11/5	
z_4	3/10	-3/2	0	-1	0	26/35	-1/2	2/5	
z_3	6/5	0	-1	0	0	-3/5	0	-2/5	
t_1	16/5	3	0	0	-1	-8/5	1	-7/5	

In Table 7.1 the upper bounded variables, z_1 and z_2 are at their lower bounds. The Table 7.1 only has columns corresponding to t_j , $j=1, \dots, 4$, which are non-basic (z_j , $j \in \{1, \dots, 4\}$ if t_j is basic) since the column corresponding to z_j , $j \in \{1, \dots, 4\}$ is merely the negative of the t column. The solution in Table 7.1 corresponds to having x_4^0 reduced to $x_4^0 - z_4 = 2 - 3/10 = 17/10$, x_3^0 reduced to $4/5$ from 2 and x_1^0 increased by $b_1 = 16/5$ to $16/5$. Since $c_1 = 4$, $c_3 = 8$ and $c_4 = 12$ the total cost of changing the three original field values is $f^* = 24$.

We now apply Step 1 of Heuristic 7.1. The slack variable s_2 can be introduced into the basis maintaining feasibility and causing z_4 to leave the basis with a resultant reduction of f^* to 12 from 24. The resultant tableau is shown in Table 7.2. The current solution is $G = (\{s_1, s_2, z_3, t_1\}, 12)$.

In repeating Step 1(b) of the heuristic we find that no more slack variables can be introduced to reduce f^* ; i.e. $S' = \emptyset$ and hence we proceed to execute Step 2, beginning with $V = \{t_2, t_3, t_4, z_1, z_2, z_4\}$. On iterating through Step 2(b) we find that V becomes empty before any original variable can be located to improve upon G . We stop processing

Table 7.2

Simplex Tableau after Application of Step 1 of Heuristic 7.1

Basic Variable	Solution Value	Non-basics					
		t_2	t_3	t_4	z_1	s_3	s_4
s_1	23/13	28/13	0	7/13	0	5/13	9/13
s_2	21/52	-105/52	0	-35/26	0	-35/52	7/13
z_3	75/52	-63/52	-1	-21/26	0	-21/52	-1/13
t_1	50/13	-12/52	0	-28/13	-1	-1/13	-7/13

and we have as a feasible solution to the MWFIC associated with x^0 , $s_1 = 23/13$, $s_2 = 21/52$, $z_3 = 75/52$, $t_1 = 50/13$ at a cost of 12. That is, the record x^0 needs to be modified to $x^* = (50/13, 3, 29/52, 2)$.

The optimal solution to this problem arrived at using Algorithm 6.1 also indicates that x_1 and x_3 should be changed. The algorithm required solution of 4 SCP's before arriving at the optimal solution.

Before proceeding to a more detailed analysis of Heuristic 7.1 some of its steps need elucidation and justification. In Step 1(b) a slack variable s^0 is introduced into the basis only if (a) the resulting solution is feasible and (b) the cost of this solution is lower than the cost of the previously recorded solution G. Item (b) above requires that s^0 displace only an original variable (t or z) and not a slack variable from the basis. Thus cycling back and forth between slack variables introduced and exiting from the basis is avoided and a strict decrease, if possible, is effected in the value of the objective (7.1). Since Heuristic 7.1 terminates processing at a local optimum (possibly global optimum), if more than one slack variable in S' is a candidate for entry into the basis, the

slack variable s^m with minimum associated solution cost in (7.1) should be introduced into the basis. It is possible that with s^m and s^0 as two candidates for entry into the basis with $f^m < f^0 < f^*$ choosing s^0 may be disadvantageous if Heuristic 7.1 terminates with no further reduction in objective value. Introducing s^m in this situation would result in a better heuristic solution value. There are significant overheads associated with this procedure in that all the slack variables in S' must be tried at every iteration in Step 1(b). In a computational environment this device may need to be invoked only if heuristic objective values are found unacceptably high (according to some prespecified cost measure). A similar procedure can be adopted in Step 2(b) of the heuristic for introduction of t or z variables into the basis. Notice, however, that when the heuristic terminates a local optimum is indeed obtained since an exhaustive search for the introduction of a currently non-basic variable into the basis with lower objective value terminates unsuccessfully.

IV. A Comparative Analysis of Heuristic 7.1 Versus Algorithm 6.1

The Heuristic 7.1 terminates at an extreme point of the polyhedron defined by the constraint set $A(t-z) \leq b^0$, $z \leq x^0$, $t, z \geq 0$ in such a way that no other adjacent extreme point solution has associated with it a lower objective value. Since MWFIC is a special case of the general fixed charge problem and since an optimal solution to a fixed charge problem can always be found among the extreme points of the linear constraint set associated with the problem [18] the heuristic solution may result in optimality. At the least Heuristic 7.1 locates a local optimum solution in the sense of no adjacent extreme point solution

being better. As in Example 7.1 it may terminate with its solution as the true optimum solution for a given MWFIC. Heuristic 7.1 does not involve the solution of SCP's or other NP-complete problems and since the manipulations involved are basically the same as in using the upper bounded simplex method, Heuristic 7.1 is likely to be a quick algorithm for locating a local optimum solution. If the local optimum also happens to be the true optimum then the heuristic's CPU time performance is likely to be quite superior to that of Algorithm 6.1. However, local optimum solutions located by Heuristic 7.1 can be significantly at variance with (in fact they can be arbitrarily far from) the true solution located by Algorithm 6.1. The following computational analysis includes consideration of the deviation of the heuristic solutions from those identified by Algorithm 6.1.

The programming language used to code Heuristic 7.1 is ASCII FORTRAN (level 9R.2). The same language was used to code Algorithm 6.1. The computer programs for both the heuristic and the algorithm were compiled and executed on a UNIVAC 1100/81 computer at the University of Wisconsin-Milwaukee. Each row of Table 7.3 corresponds to a set of 100 MWFIC problems solved for the given combination of number of fields and number of edits. The MWFIC problems were generated at random using a uniform random generator to generate the number of edits, number of fields, the coefficients of the variables in the associated constraint set, the costs of changing the original field values and finally 100 failing records for each constraint set. The MWFIC constraint sets were generated as follows. The minimum number of edits and fields was restricted to be at least 4 each. The

constraints were generated initially with all coefficients non-zero. The signs of the coefficients were generated at random as were those of the right hand side values of each constraint. All constraints generated were assumed to be inequalities of type $\leq$. The uninvolved field density in each constraint, i.e. the proportion of the number of zero coefficients to the total number of fields, was varied to be between 0 and 0.9 in an uniformly random fashion with the additional restriction that at least 2 fields per constraint must have non-zero coefficients. These density restrictions were enforced by suitably zeroing out the coefficients in the previously generated matrix of coefficients (with 100% non-zero coefficients). Empirically it was found that such a procedure results in very tightly bound constraint sets being constructed and the placement of the zero coefficients is such that parallel constraints generally do not occur.

The execution times are accurate to within 0.2 milliseconds. Feasible solutions to the MWFIC problems, required in Step 0 of Heuristic 7.1, were obtained using a modification of the upper bounded simplex method applied to a primal infeasible linear programming problem (see Appendix 7.1 for details). The constraint set (7.7)-(7.10) with an objective function to maximize 0 is dual feasible. However, the presence of upper bound constraints and of complete degeneracy makes the use of a dual simplex method type of procedure a trial and error process for obtaining primal feasibility. Also, the procedure outlined in Appendix 7.1 permits the implicit processing of N of the variables since only one of the two columns corresponding to (t_j, z_j) ,

$j \in I_N$, need be maintained while the other can be obtained by taking the negative of the existing column. The notation used in Table 7.3 is as follows.

Applicable to Algorithm 6.1 and Heuristic 7.1

AVEDIT : Average number of edits failed by a given record

MAXED : Maximum number of edits failed by a given record

Specific to Algorithm 6.1 only

AVITER : Average number of iterations to get a passing record

MAXITER : Maximum number of iterations to get a passing record

AVTIME : Average time, in milliseconds, to get a passing record

MAXTIME : Maximum time, in milliseconds, to get a passing record

Specific to Heuristic only

Define $DEV = (f - f^*)/f^*$, where f and f^* are the objective values obtained by the heuristic and by the algorithm respectively.

AVDEV : Average of DEV taken over 100 records processed

MAXDEV : The maximum value of DEV for a set of 100 MWFIC's

AVT2 : Average time taken by the heuristic to process a record

MAXT2 : Maximum time taken by the heuristic to process a record

PCTVAR : The number (percentage) of MWFIC problems out of the 100 generated in which the heuristic objective value was different from the optimal solution value obtained using Algorithm 6.1

The objective values obtained by Heuristic 7.1 was, on the average, within 10% of the optimal objective value in 14 of the 19 sets of problems reported in Table 7.3. However, problem sets 2, 9 and 12

Table 7.3

Comparison of the Computational Performance of
Heuristic 7.1 and Algorithm 6.1

SNO	# OF EDITS	# OF FIELDS	AV EDIT	MAX EDIT	AV ITER	MAX ITER	AV TIME	MAX TIME	AV- DEV	MAX- DEV	AV T ₂	MAX T ₂	PCTVAR
1	8	4	5	6	3.3	4	145	290	.003	.28	24	32	1
2	11	6	4	8	2.1	11	210	4357	3.14	12.6	42	62	98
3	4	4	3	3	9.3	11	691	1275	.001	.06	10	14	1
4	13	4	4	7	6.9	9	559	877	0	0	131	165	0
5	7	13	3	5	2.8	6	176	345	0	0	31	38	0
6	9	7	6	7	8.6	12	1822	4485	.002	.04	36	42	4
7	9	10	5	6	5.3	9	1900	9780	.096	.24	29	30	94
8	12	7	7	10	4.3	8	1567	6244	.005	.46	41	58	1
9	7	7	2	4	.4	3	37	316	1.26	23.6	12	17	93
10	12	9	6	7	6.5	11	1317	4676	.024	.18	58	79	18
11	7	10	4	5	2.4	6	311	3090	.56	1.6	20	24	91
12	4	18	2	4	0	0	18	25	1.14	11.5	9	14	16
13	6	4	4	5	4.9	9	231	604	.01	.88	17	17	2
14	5	4	2	3	5.4	7	267	390	.024	.8	9	9	5
15	4	4	2	4	5.6	10	186	533	.03	.76	12	18	4
16	7	8	5	7	2.0	5	264	1176	.04	.26	26	35	59
17	7	7	4	5	2.0	2	153	268	.08	.21	13	16	38
18	7	7	2	3	0	0	19	21	.502	4.2	13	17	71
19	7	7	3	3	2.0	2	125	145	0	0	11	12	0

demonstrate that the heuristic can perform very poorly on occasion (refer to the MAXDEV column). In instances such as these, it may be useful to include an amendment to Heuristic 7.1 as a rider to Step 2.

Rider to Step 2: If the objective value f^* is unacceptable (on termination of the heuristic) according to some predetermined cost cut off value $\bar{f}$ (i.e. $f^* > \bar{f}$) then pivot out of the present solution to

a different one with equal or higher objective value. Let the basic variable departing the basis be k . Repeat Steps 1 and 2 introducing, if at all necessary, the variable k into the basis only as a last resort in the first iteration.

On problems where the deviation of the heuristic value from the optimum is considerable (e.g., 2,9,12) the heuristic solutions are obtained in less than 0.1 second of CPU time, thus permitting the use of a procedure such as the one in Rider to Step 2 above while keeping the execution time acceptable (low). In roughly 50% of the problem sets documented in Table 7.3 the heuristic generates solutions with values almost the same as the optimum in less than 10% of the time taken by Algorithm 6.1 to generate the optimum. The heuristic performed very well in test runs on problems with 35 edits and 35 fields furnishing a local optimum solution in under 1 second of CPU time. It was, however, not possible to judge the accuracy of these solutions since no solution could be obtained using Algorithm 6.1 on such problems.

The heuristic may be used to generate an upper bound to solution values in using McKeown's [26] branch and bound scheme mentioned in Chapter 5. Also, the heuristic may be usable with the VS-1 algorithm if all extreme point optimal solutions to a given error localization problem is desired.

V. An Algorithm for Locating a Feasible Solution to MWFIC

An algorithm for obtaining a basic feasible solution, if one exists, starting from a (linear programming problem) set up which is

neither primal feasible nor dual feasible can be found in Garfinkel [14]. The algorithm used in the computer program, to obtain a basic feasible solution in Step 0 of Heuristic 7.1, is motivated by the procedure in Garfinkel [14] and is a modification of the simplex method with upper bounded variables. In the following discussions we will be dealing with an abbreviated tableau of the form:

$$\begin{array}{c|c} & y_{0j}, j \in R \\ \hline z_{i0}, i \in B & \end{array} \quad (7.22)$$

corresponding to the problem,

$$\begin{aligned} \max \quad & 0x \\ \text{Ax} \quad & \leq b \\ 0 \leq x \leq d \end{aligned} \quad (7.23)$$

where A is $m \times N$, x is $N \times 1$ and b and d are $m \times 1$. In (7.22) B is the set of basic variables and R is the set of non-basic variables at any stage of processing of (7.22). The set R is further partitioned, as in the upper bounded simplex method, see [], into $R_1 = \{j | j \in R, x_j = 0\}$ and $R_2 = \{j | j \in R \text{ and } x_j = d_j\}$. The value of the i th basic variable x_{B_i} may then be obtained from

$$x_{B_i} = y_{i0} - \sum_{j \in R_1} y_{ij} x_j - \sum_{j \in R_2} y_{ij} x_j, \quad i = 1, \dots, m \quad (7.24)$$

where $y_{ij}, j=0, \dots, N, i=1, \dots, m$ are the entries in the tableau at any iteration. The current value of x_{B_i} can then be obtained as

$$x_{i0} = y_{i0} - \sum_{j \in R_2} y_{ij} u_j, \quad i = 1, \dots, m. \quad (7.25)$$

We shall adapt the pivot rules for the upper bounded simplex method, see, for instance, Garfinkel [13], for the purpose of obtaining a

basic feasible solution given that some of the z_{i0} are initially negative. In the initial tableau associated with Step 0 of Heuristic 7.1 the basic variables are all the slack variables only.

Determination of an entering non-basic variable

Let T_1 and T_2 be the sets of rows in the tableau (7.22) with $z_{i0} \geq 0$ and $z_{i0} < 0$ respectively. If $T_2 = \emptyset$ then the tableau contains a feasible solution and we may proceed to Step 1 of Heuristic 7.1. If $T_2 \neq \emptyset$, select a row $r \in T_2$ for processing. Partition the columns corresponding to the non-basic variables into $R_3 = \{j | y_{rj} \geq 0\}$ and $R_4 = \{j | y_{rj} < 0\}$.

Step 1 If $R_3 \neq \emptyset$ then let k be the index of any column such that

$$y_{rk} = \min_{\substack{j \in R_4 \\ j \in R_1}} y_{rj}; \quad \text{go to Step 2.} \quad (7.26)$$

If $R_3 = \emptyset$ then select an index ℓ such that

$$y_{r\ell} = \max_{\substack{j \in R_3 \\ j \in R_2}} y_{rj}; \quad \text{go to Step 3.} \quad (7.27)$$

Determination of a departing basic variable

Step 2 A first basic variable to reach zero can be determined by computing

$$\theta_{tk} = \min_{p \in T_1} \{z_{p0}/y_{pk}, y_{pk} > 0\}. \quad (7.28)$$

A first basic variable to reach its upper bound (excluding x_k) is given by x_{B_s} which can be determined from

$$\alpha_{sk} = \min_{p \in I_m} \{(z_{p0} - d_p)/y_{pk}, y_{pk} < 0\}. \quad (7.29)$$

If all basic variables are non-upper bounded and $y_{pk} < 0$ for all $p \in I_m$ then (7.29) needs to be modified to

$$\alpha_{sk} = \max_{p \in I_m} \{z_{p0}/y_{pk}, y_{pk} < 0\}. \quad (7.29a)$$

The index of the departing variable is then determined from the index corresponding to

$$\min \{\theta_{tk}, \alpha_{sk}, d_k\}. \quad (7.30)$$

Go to Step 4.

Step 3 A first basic variable to reach zero can be determined from the index for which

$$\hat{\theta}_{t\ell} = \max_{p \in T_1} (z_{p0}/y_{p\ell}, y_{p\ell} < 0) \quad (7.31)$$

holds. Note $\hat{\theta}_{t\ell} \leq 0$ and represents the change allowed in x_ℓ . Thus if $d_\ell < -\hat{\theta}_{t\ell}$, x_ℓ reaches zero before x_{B_t} . A first basic variable to reach its upper bound may be determined from

$$\hat{\alpha}_{s\ell} = \max_{p \in I_m} ((z_{p0} - d_p)/y_{p\ell}, y_{p\ell} > 0). \quad (7.32)$$

If all basic variables are non-upper bounded and $y_{p\ell} \geq 0$ for all $p \in I_m$ then (7.32) needs to be modified to

$$\hat{\alpha}_{s\ell} = \min_{p \in I_m} (z_{p0}/y_{p\ell}, y_{p\ell} > 0). \quad (7.32a)$$

The index of the departing variable is then determined by the index for which $\max \{\hat{\theta}_{t\ell}, \hat{\alpha}_{s\ell}, -d_\ell\}$ holds. Go to Step 4.

Step 4 The change of basis is effected as with the upper bounded simplex method (see [13] for details).

The Steps 1-4 of the procedure outlined above guarantee that the cardinality of T_1 is non-decreasing. By choosing a lexicographically determined pivot element we can ensure a lexicographic increase in the vector z'_0 determined by $(z_{r0}, \{z_{i0} \mid i \in T_1\})$ and hence ensure that at each Step $|T_1|$ increases. Thus the procedure is finite.

CHAPTER 8

EXTENSIONS AND FUTURE RESEARCH

In earlier chapters the theory, and some solution procedures based on that theory, for solving the minimum weighted fields to impute problem with categorical (MWFI) and with continuous (MWFIC) data was presented. More and more areas of business, government and industry are moving towards computerized information systems and the need for sound data validation techniques is likely to be felt acutely with this growth. The problems that have been solved, as presented in Chapters 2 to 7, are by no means the only ones existing in the area of data validation. Further, the procedures developed thus far for solving MWFI and MWFIC can get prohibitively expensive if large problems (say 200 edits and 100 fields) are to be solved. It would not be incorrect to say that the problems in data validation attacked thus far, in this and other works, barely scratch the surface of a rather fertile area for research. In this chapter some of the problems that would appear to require solutions as a follow up of the research presented in this work are briefly enumerated and explained.

The solution of MWFIC with an additional constraint added to it, namely, that all variable values must be integer cannot be arrived at using procedures such as Algorithm 6.1 or Heuristic 7.1. We shall refer to this problem as the Minimum Weighted Fields to Impute Problem with Discrete data (MWFID). An adaptation of the Chernikova [4]

algorithm for the solution of MWFID has been proposed in Liepins [22]. This adaptation is impractical in that the number of columns in even the initial set up of the tableau is rendered prohibitive by having to include one column for each integer value a variable can take on. Efficient procedures need to be developed for MWFID since a considerable amount of data generated in surveys and by accounting procedures is inherently discrete with linear algebraic constraints associated with such data.

The error localization problems with categorical, continuous and discrete data have been treated as separate and distinct so far. In most survey situations, however, these types of data are likely to be encountered simultaneously and in an interrelated fashion. We shall refer to the minimum weighted fields to impute problem in such mixed data situations as MWFIM. As long as the type of data and the type of edit are the same, namely, edits with categorical data only, continuous data only, etc. the error localization problem would be separable into categorical, continuous and discrete data problems and can be attacked separately. However, this may not always be the case. Consider the situation in Example 8.1.

Example 8.1: Let x_1 , x_2 and x_3 be the variables corresponding to the values associated with fields 1, 2 and 3 which contain continuous data and let y_4 correspond to the values associated with field 4 which contains categorical data, such that $A_4 = \{1,2,3,4\}$. An edit

$$E : \text{ If } y_4 = 1 \text{ or } 3 \text{ then a record } R \text{ with} \\ x_1^0 - x_2^0 \leq x_3^0 \text{ is failing} \quad (8.1)$$

cannot be separated into a categorical data portion and a continuous data segment without losing the content of E.

MWFIM's such as one defined on the data in Example 8.1 cannot be solved using the (theory and) procedures in Chapters 2 to 7. It is not clear whether implied edits would need to be generated as a means to solving this problem. It is also not readily apparent if set covering based procedures, such as Algorithms 3.2 and 6.1 can be used to attack MWFIM.

Another area of future research is in the development of procedures to speed up Algorithm 3.2 and Algorithm 6.1. It was mentioned in Chapter 4 that the enumeration of all records generatable on a given set of fields in Step 3 of Algorithm 3.2 can be very expensive. It will be useful to have access to procedures that can streamline Step 3 using perhaps the knowledge of the edit failure patterns of records processed in the past, thus speeding up Algorithm 3.2.

Both Algorithms 3.2 and 6.1 perform well for small problems. Given that the determination of feasibility in Algorithm 6.1 (Step 3) involves the solution of a rather easily solved linear programming problem, heuristic procedures to lower bound the MWFIC solution values resulting in fewer iterations, and hence fewer SCP solutions, should considerably speed up the algorithm. Such a device may result in large MWFIC's being solved rather quickly and is well worth exploring. Algorithm 3.2, for solving MWFI generates a "strong" SCP constraint in Step 5 using the prime cover solution in Step 4 of the algorithm. On the other hand Algorithm 6.1, for solving MWFIC, constructs an SCP

constraint in Step 5 which may eliminate no more than the SCP solution just processed from recurring. Development of devices that result in deeper cuts (stronger SCP constraints) being generated in Step 5 of Algorithm 6.1 is needed.

Error localization has been the underlying objective in all the procedures presented in this work, while Algorithms 3.2, 6.1, Heuristic 7.1 and the VS-1 and VS-2 algorithms do obtain a passing record as a byproduct. However, Algorithms 6.1, VS-1, VS-2 and also Heuristic 7.1 seek to locate corner points of the polyhedron defined by the constraint set as a means to solving MWFIC. Thus, if the passing record obtained by these procedures is accepted as such, then the corrected data set will get biased with a surfeit of feasible records located on the boundary of the polyhedron. More imaginative imputation techniques are clearly needed and further research needs to be conducted in this area. Algorithm 3.1 may produce passing records with identical values imputed into the fields that are to be changed in different records. Randomization techniques based on the knowledge of the probability distribution of the values in each field of the records could be an acceptable solution and is worth investigating.

Error localization and imputation techniques given different observed data distributions and error models need to be developed. The impact of data imputation, in fields identified by error localization techniques presented in this work, on the observed data distribution also needs to be investigated. While a considerable amount of theory needs to be developed, it is also felt that empirical testing of the techniques is crucial to their adoption in practice and such testing is likely to be very time consuming.

The main issue in data validation, however, is likely to be the effective and objective evaluation of the four alternatives outlined in Chapter 1 namely, whether to keep the faulty record, discard the record, change the record to make it consistent or if possible ask the respondent to revise the record. Procedures for assessing the value of data (to the agency receiving the information for example), the cost of accepting inconsistent data, the cost of imputing data which is consistent but may be biasing the distribution of data, etc. need to be developed to assist in such evaluation.

LIST OF REFERENCES

LIST OF REFERENCES

1. Bellmore, M., and Ratliff, H. D., "Set Covering and Involuntary Bases," Management Science 18, 194-206 (1971).
2. Burger, V. E., "Über homogene lineare Urgleichung systeme," Z. Angew Math. Mech. 36, 135-139 (1956).
3. Charnes, A., Cooper, W. W., Management Models and Industrial Applications of Linear Programming Vol. I, 413, Wiley, New York (1961).
4. Chernikova, N. V., "Algorithm for Finding General Formula for the Non-negative Solutions of a System of Linear Equations," U.S.S.R. Computational Mathematics and Mathematical Physics 4, 151-158 (1964).
5. Chernikova, N. V., "Algorithm for Finding General Formula for the Non-negative Solutions of a System of Linear Inequalities," U.S.S.R. Computational Mathematics and Mathematical Physics 5, 228-233 (1965).
6. Cooper, L., "The Fixed Charge Problem-I: A New Heuristic Method," Computers & Mathematics with Applications 1, 89-95 (1975).
7. Cooper, L., and Drebes, C., "An Approximate Solution Method for the Fixed Charge Problem," Naval Research Logistics Quarterly 8, 101-113 (1967).
8. Denzler, D. R., "An Approximate Algorithm for the Fixed Charge Problem," Naval Research Logistics Quarterly 16, 411-416 (1969).
9. Fellegi, P., and Holt, D., "A Systematic Approach to Automatic Edit and Imputation," JASA 71, 17-25 (1976).
10. Freund, R. J., and Hartley, H. O., "A Procedure for Automatic Data Editing," JASA 62, 341-352 (1967).
11. Garey, M. R., and Johnson, D. S., Computers and Intractability: A Guide to the Theory of NP-Completeness, Freeman Press, San Francisco (1979).
12. Garfinkel, R. S., "An Algorithm for Optimal Imputation of Erroneous Data," Working Paper #83, College of Business Administration, The University of Tennessee (1979).
13. Garfinkel, R. S., and Nemhauser, G. L., Integer Programming, Wiley, New York (1972).

14. Garfinkel, R. S., and Yu, P. L., "A Primal-Dual Surrogate Simplex Algorithm: Comparison with the Criss-Cross Method," Working Paper #22, College of Business Administration, The University of Tennessee (1975).
15. Geoffrion, A. M., and Marsten, R. E., "Integer Programming Algorithms: A Framework and State-of-the-Art Survey," Management Science 18, 465-491 (1972).
16. Greenberg, B., Personal communication (1982).
17. Hillier, F. S., and Lieberman, G. J., Operations Research, Holden-Day, California (1974).
18. Hirsch, W. M., and Dantzig, G. B., "The Fixed Charge Problem," Naval Research Logistics Quarterly 15, 413-424 (1968).
19. Knuth, D. E., The Art of Computer Programming: Sorting and Searching, Vol. 3, Addison Wesley, Reading, Mass. (1973).
20. Kohler, D. A., "Projections of Convex Polyhedral Sets," Report ORC 67-29, Operations Research Center, University of California, Berkeley (1967).
21. Liepins, G. E., "Refinements to the Booleau Approach to Automatic Data Editing," Oak Ridge National Laboratories/TM-7156 (1978).
22. Liepins, G. E., "Algorithms for Error Localization of Discrete Data," Oak Ridge National Laboratories (1981).
23. Liepins, G. E., Personal communication (1982).
24. Luenberger, D. G., Introduction to Linear and Non-linear Programming, Addison-Wesley, Reading, Massachusetts (1973).
25. Matheiss, T. H., and Rubin, D. G., "A Survey and Comparison of Methods for Finding All Vertices of Convex Polyhedral Sets," Math. of OR 5, 167-185 (1980).
26. McKeown, P. G., "A Mathematical Programming Approach to Editing of Economic Survey Data," Working Paper, The University of Georgia, (undated).
27. Naus, J. I., Johnson, T. G., and Montalvo, R., "A Probabilistic Model for Identifying Errors in Data Editing," JASA 67, 943-950 (1973).
28. Noble, B., Applied Linear Algebra, Prentice-Hall, New Jersey (1969).

29. Pritzker, L., Ogus, J., and Hansen, M. H., "Computer Editing Methods - Some Applications and Results," Bulletin of the International Statistical Institute 41, 442-465 (1965).
30. Rubin, D. S., "Vertex Generation and Cardinality Constrained Linear Programs," Operations Research 23, 555-565 (1975).
31. Sande, G., "Diagnostic Capabilities for a Numerical Edit Specification Analysis," Statistics Canada (1976).
32. Sande, G., "An Algorithm for the Fields to Impute Problems of Numerical and Coded Data," Statistics Canada (1978).
33. Simmonard, M., Linear Programming, 380-401, Prentice-Hall, New Jersey (1966).
34. Steinberg, D. I., "The Fixed Charge Problem," Naval Research Logistics Quarterly 17, 217-236 (1970).
35. Terry, M. E., "The Principles of Statistical Analysis Using Large Electronic Computers," Bulletin of International Statistical Institute 40, 547-552 (1963).
36. Tanahashi, K., and Luenberger, D. G., "Cardinality Constrained Linear Programming," Stanford University (1971).
37. Wagner, H. M., Principles of Operations Research, Prentice-Hall, New Jersey (1975).
38. Walker, W. E., "A Heuristic Adjacent Extreme Point Algorithm for the Fixed Charge Problem," Management Science 22, 587-596 (1976).

VITA

Anand S. Kunnathur was born and raised in Delhi, India. He holds a Master's degree in Physics from The University of Delhi and a Master's degree in Applied Mathematics from York University, Toronto, Canada. He entered the Graduate School of The University of Tennessee, Knoxville in September 1977 and expects to receive a Ph.D. in Management Science in December 1982. He has taught at The University of Tennessee, Knoxville for four years as a graduate teaching assistant and has worked as a consultant with the Oak Ridge National Laboratories, Tennessee for two years. He joined the faculty of The University of Wisconsin-Milwaukee in Fall 1981.

He is married to the former N. V. Jayashree of Mysore, India.