RESEARCH REPORT SERIES
(*Statistics #2002-02)*

Extending the Fellegi-Holt Model of Statistical Data Editing

William E. Winkler and Bor-Chung Chen

Statistical Research Division
U.S. Bureau of the Census
Washington D.C. 20233

Report Issued: February 1, 2002

Extending the Fellegi-Holt Model of Statistical Data Editing

William E. Winkler* and Bor-Chung Chen 1/
Bureau of the Census, Washington, DC 20233-9100
william.e.winkler@census.gov (2001.09.29)

ABSTRACT

This paper provides extensions to the theory and the computational aspects of the Fellegi-Holt Model of Editing (JASA 1976). If implicit edits can be generated prior to editing, then error localization (finding the minimum number of fields to impute) can be quite rapid. In some situations, not all of the implicit edits can be generated because of the great number (> 10^30) of distinct edit patterns. The ideas in this paper are intended to determine more rapidly the approximate minimal number of fields to change in situations where not all implicit edits can be generated prior to editing. As a special case, the formal validity of Bankier's Nearest-Neighbour Imputation Method (NIM) is demonstrated.

Keywords: set-covering; integer programming; error localization

1. INTRODUCTION

Statistical data editing (SDE) are those methods that can be used to edit (i.e., clean-up) and impute (fill-in) missing or contradictory data. The result of SDE is data that can be used for intended analytic purposes. These include primary purposes such as estimation of totals and subtotals for publications that are free of self-contradictory information. The published totals do not contradict published totals in other sources. Self-contradictory information might include groups of items that do not add to desired subtotals or totals for subgroups that exceed a known proportion of the total for the entire group. The uses of the data after SDE might be preparation of variances of estimates for a number of sub-domains and micro-data analyses. If only a few published totals need to be accurate, then an efficient use of resources may be to perform detailed edits on only a few records that effect the estimated totals. If many analyses need to be performed on a large number of sub-domains or if the full set of accurate micro-data are needed, then a very large number of edits, follow-up, and corrections may be needed.

Fellegi and Holt (1976, hereafter FH) provided a seminal model for SDE. Their methods have the virtues that, in one pass through the data, an edit-failing record can be assured to satisfy all edits and that the logical consistency of the entire set of edits can be checked prior to the receipt of data. The implementations of the system have had additional advantages over traditional if-then-else rule edit systems because edits reside in easily modified tables and computer code needs no modification. FH had three goals that we paraphrase:

1. The data in each record should be made to satisfy all edits by changing the fewest possible variables (fields).
2. Imputation rules should derive automatically from edit rules.
3. When imputation is necessary, it should maintain the joint distribution of variables.

Fellegi and Holt were the first to demonstrate precisely what information was needed for correcting a record. By *correcting*, we mean changing (or filling in) values of fields so that a record satisfies all of the edits. Prior to FH, individuals were unable to account for edits that did not fail with a edit-failing record and that would fail after values in fields were changed so that the initially failing edits would no longer fail. In addition to (explicit) edits that are originally defined, FH showed that precise knowledge of implicit edits was needed. *Implicit edits* are those that can be logically derived from explicit edits. FH (Theorem 1) proved that implicit edits are needed for solving the problem of goal 1. Goal 1 is referred as the *error localization* (*EL*) problem. FH provided an inductive, existence-type proof to their Theorem 1. Their solution, however, did not deal with many of the practical computational aspects of the problem that, in the case of discrete data, were considered by Garfinkel, Kunnathur, and Liepins (1986, hereafter GKL), Winkler (1995, 1997), and Chen (1998). Because the error localization problem is NP-complete (GKL), reducing computation is the most important aspect in implementing a FH-based edit system.

The main purpose of this paper is to provide a method for EL when most, but not all, implicit edits are generated prior to editing. The algorithms are much faster than the direct integer programming methods for EL that do not use implicit edits that have been computed a priori. The speed increase is because the direct integer programming methods implicitly generate implicit edits during EL. Many implicit edits are repeatedly computed. To demonstrate our results, we build on ideas that are in or can be deduced from FH, GKL, Winkler (1997) and Chen (1998). Each of the previous papers had technical lemmas that showed how the number of computational paths could be reduced. We provide a number of technical lemmas that further reduce the number of computational paths.

Many statistical agencies have chosen to concentrate on traditional methods. These traditional methods include if-then-else rules for detecting contradictory information and various ways of imputing values of variables to replace the contradictory values. The dilemma with if-then-else rules is that they may not be straightforward to develop and may be difficult to write into computer code. If there are slight changes in the survey form and edit rules, then subsets of thousands of lines of code may need to be rewritten and debugged. The reason that FH methods are so appealing is that most of the if-then-else types of edits can be put in tables that are straightforward to modify and update. Because the source code does not need any updating, it is possible to create a FH system for editing that can be developed and maintained for different surveys by non-programmers such as subject matter specialists, statisticians, and economists.

This paper is divided into a number of sections. The second section gives background, notation, and describes some of the limitations and strengths of previous approaches. It provides an example and several insights that serve as the motivation for the approach that we have adopted. The third section provides extensive theory and computational methods that are engendered by the theory. In the fourth section, we provide some discussion. The final section consists of concluding remarks.

2. BACKGROUND AND PREVIOUS WORK

This paper only considers FH methods as they apply to discrete data. Extensions to situations for numeric data or combinations of discrete and numeric data are straightforward.

2.1. Elementary background and motivating example

The fields (variables) associated with a survey-data collection might be age, sex, marital status, and relationship to head of household. A field such as sex might take the values 'M', 'F', or 'b', representing male, female, or blank. In the computer, we might represent them by '1', '2', and '-1', respectively. A field such as marital status might take five different values and the field age might take 115 different values. A record is said to *fail* an edit if it satisfies the constraints imposed by the edit. If an edit places restrictions on the values a field can assume, then the field is said to be an *entering field* of the edit. In this paper, we only consider edits that have two or more entering fields. A record *passes* an edit if it does not fail the edit.

FH define an edit as a set of points in the product space determined by the fields that are to be edited. We consider the following three edits: E_1 = {age ≤ 15, married , . }, E_2 = {. ,not married , spouse}, and E_3 = {age ≤ 15, . , spouse}. The three fields that we consider are age, marital status, and relationship to head of household. In edit E_1 , age and marital status are entering fields. Edit E_1 places no restriction on relationship to head of household (the third field). In practice subject matter specialists with expertise in a given survey define the edits. The relationship E_1 & E_2 => E_3 means that edit E_3 can be logically derived from edits E_1 and E_2. In other words, if edit E_3 fails, then necessarily edit E_1 or edit E_2 fails. A non-redundant set of edits defined prior to editing are referred to as *explicit* edits. Edits that can be derived from explicit edits are called *implicit* edits.

Let r = { age ≤ 15, married, spouse }. Then record r fails edits E_1 and E_3 . If we only change the marital status field, then record r will now fail edit E_2. Prior to the work on FH, individuals knew that at least one value of an entering field in each failing explicit edit needed to be changed in order for a record to satisfy (not fail) all edits. However, when only fields that associated with failing explicit edits were changed, the changed record would now fail explicit edits that it did not fail originally. What FH showed (1976, Theorem 1) that all of the implicit edits are also needed for correcting a record. In other words, the implicit edits contain information about edits that do not fail with the original record but may fail after values in certain fields are changed. By *correcting*, we mean changing values in entering fields associated with failing (explicit and implicit) edits. The implication of the FH theory is that we must change at least one entering field in edits E_1 and E_3. It does not matter which fields are changed.

2.2. Background on FH Theory and previous approaches

The paper of Fellegi and Holt is seminal because it demonstrated that the implicit edits are needed for the error-localization problem (i.e., finding the minimum number of fields to impute). FH further showed that the logical consistency of edit system could be checked using the logical restraints imposed by the edits. With non-FH edit systems containing many if-then-else rules, the logical consistency is often checked via test decks. If the test decks are not appropriate, then it may not be possible that combinations of edits are mutually contradictory. During production editing, the systems can often fail. They fail because additional records may contain error patterns that are in contradiction to the logic in the existing system. In some situations, if the set of edit rules are *inconsistent* (i.e., mutually contradictory), then special ad hoc loops may be added to code. The special loops can be very difficult to develop.

Statistical agencies have been slow to implement FH systems because of the types of Operations Research (OR) methods that are needed to implement it. With a pure FH system, there are two components: (1) integer programming algorithms in the main edit program and (2) set covering algorithms in implicit edit-generation software.

If implicit edits are generated, then standard branch-and-bound integer programs are very fast for the main edit program (Barcaroli and Ventura, 1997, 1993; Winkler 1995). Winkler (1995) introduced a greedy heuristic that is more than 100 times as fast as branch-and-bound for error-localization. If the complete set of implicit edits is available prior to editing, then the speed of the main edit program is no longer an issue. The heuristic algorithm processes on the order of 1000 records per second.

Although implicit edit-generation software can be run prior to the receipt of survey data, it still needs to run in a suitable amount of time (say, less than 24 hours). With modest size surveys, successful edit-generation algorithms have been implemented by IBM and ISTAT (see e.g., Barcaroli and Ventura, 1997, 1993) and Winkler (1995). With very large surveys such as the Italian Labour Force Survey (Barcaroli and Ventura, 1997, 1993; Winkler 1997) the amount of computation can be prohibitive. The size of the product space of possible records is enormous ($4.6 \times 10\text{^}79$) and there is a large number of edit patterns (points) associated with the edits (>>10^30). Using subsets of the Italian Labour Force Survey, Winkler (1997) extrapolated that the IBM-ISTAT algorithms would have taken at least 800 days on the largest IBM mainframe. By having the edit-generation algorithms learn from the beginning part of the computation, Winkler (1997) and Chen (1998) developed algorithms that take less than 24 hours on the same data set. The algorithms, however, are incomplete because they cannot deal effectively with complicated skip patterns. Winkler (1997) and Chen (1998) estimate that the algorithms generate at least 90 percent of the implicit edits for the Italian Labour Force Survey.

Because of the difficulty in generating implicit edits prior to editing, various methods have been developed for solving the error-localization problem directly without using implicit edits. All of the methods limit the time spent on an individual record by putting upper bounds on the amount of time for the computation or bounds on the size of certain data structures. The remaining records that exceed bounds are then given to analysts for clerical review and correction. These methods can be unsatisfactory if suitable clerical resources are not available. In a dramatic comparison, Garcia and Thompson (2000) showed that a group of ten analysts took six months to review and correct a moderate size survey that had complicated edit patterns. An FH method needed only 24 hours to edit/impute the survey and changed 1/3 as many fields as the clerical review.

There are three error-localization methods. The first and slowest method is to use direct integer programming methods such as branch-and-bound. The method can require on the order of 10 minutes per record. The second method employs variants of a cardinality constrained Chernikova algorithm (Rubin 1975, Schopiu-Kratina and Kovar 1989, Filion and Schopiu-Kratina 1993). The method is used in the GEIS system of Statistics Canada (in C, 1 second per record), in the CherryPi system of Statistics Netherlands (in Pascal, 2 seconds per record), and in the AGGIES system of the National Agriculture Statistical Service (SAS, more than one minute per record). Because the methods are somewhat slow, Statistics Netherlands (De Waal 2000) developed the LEO system that employs variants of the Fourier-Motzkin elimination method (in Pascal, at least 10 records per second). None of these methods or systems can deal, however, with large surveys having millions of records. For instance, with the U.S. Census of Manufactures, 4 percent of 2.5 million records (100,000) have edit records with failures. This large number of records drastically exceeds the capability of the aforementioned systems. It is not possible to clerically edit 100,000 records. Since most of the 100,000 are associated with small firms (companies), it seems reasonable to attain an FH system that could edit/impute all of the records automatically. Only the records associated with the largest companies would be

clerically reviewed as an additional step of the editing. For the U.S. Decennial Census, there are 300 million records (see e.g., Chen, Winkler, and Hemmig 2000).

In this paper, we demonstrate how all of the records can be error-localized when not all of the implicit edits can be generated a priori. Chen (1998) and Winkler (1997) have shown that, if most of the implicit edits are generated prior to editing, then virtually all of the records can be properly error-localized. The methods of this paper provide a means of error-localization for a small proportion of remaining records that cannot be properly corrected due to an incomplete set of implicit edits. The methods are far faster than those based on Chernikova algorithms or Fourier-Motzkin Elimination.

2.3 Bankier's Nearest-Neighbor Imputation Method

Bankier (1991, see also 1997, 2000) introduced a successful method on using (hot-deck) donor imputation that has been used for the 1996 and 2001 Canadian Censuses and will be used for the 2006 Canadian Census. As with other donor imputation systems, the method is dependent on having a large population of high quality donors. Before describing NIM, we describe how a corresponding FH edit system that uses hot-deck imputation would work. The FH edit system would determine the minimum number of fields to change. A priori matching rules would be developed to select hot-deck donors from the set of records that satisfy all edits. If there are suitable donors, then imputed fields from the hot-deck donors will maintain the univariate distributions of the respondents. Two difficulties are associated with systems (either FH or if-then-else) that use hot-deck imputation. The first is that the matching rules may not be as good as they can be. This has been noted as a problem in the 1990 U.S. Decennial Census, the 1991 Canadian Census, and the 1991 British Census. The second is that there may not be enough suitable donors. The second problem is often not as serious in a census as it is in a smaller survey.

Bankier's NIM proceeds primarily by using donors. Each edit-failing record is matched with a large subset (say 2,000) of records that satisfy all of the edits. The ones, say 40, that have the smallest deviations in terms of the number of fields differing from the edit failing record are retained. If the same edit-failing record were considered by the Fellegi-Holt method and a donor was found that was in the 2,000 records that were searched as potential donors, then NIM could necessarily get the same donor substitution as the FH method. Even if it did not have the exact same donor, it would get a solution that was optimal in terms of the weighted, minimal number of fields to impute. NIM has an effective heuristic that allows it to deal with numeric data. Age (because of the number of values it assumes) can be considered numeric. NIM has further heuristics that work somewhat as follows. Each of the 40 edit-passing records differs from the edit-failing record on a set of fields. Fast heuristics look at subsets
to determine if the record resulting from changing the values in the subset satisfy all edits. From the 40, the five best (in terms of weighted minimal number of fields changed) are selected. One of the five is randomly selected as the donor for the hot-deck imputation.

There are two crucial advantages for a NIM system. The first is that all of the imputed records satisfy all of the edits. The second is that it finds the best matching rules automatically. From the standpoint of this paper, there is another crucial insight. By considering the set of fields in a donor record that differ from the edit-failing record, it is possible to efficiently fill-in (determine the subset of fields to change) a record. The potential value states are always two. Either leave the value in a field to its value in the original edit-failing record or change it to the value in the potential donor record. In the main part of this paper, we give lemmas that characterize and

generalize ideas from NIM. The lemmas yield fast algorithms for filling in a record in situations where not all of the implicit edits can be generated. If some of implicit edits are not present, then a cover of the entering fields in the failed edits may not yield a set of fields to change that yields an edit-passing record. By a *cover*, we mean a set of fields that enter every failing edit. The lemmas give a quick way of determining additional fields that are needed for error-localization. In the context of this paper (as it was in Theorem 1 of FH), *error-localization* is determining a set of fields that can be changed so that a record satisfies all edits. In some contexts, error-localization also means determining a minimum number.

2.4. Notation, additional background and technical lemmas

A record $r=(y_1,...,y_n)$ in a computer file can have n fields subject to edits. For discrete edits, y takes values in ΠZ^n, the product space of integers. Each field y_i, i=1,...,n, corresponds to a variable that is coded. For instance, y_1 might take values 1=male and 2=female. y_2 might take values 1=single, 2=divorced, and 3=married. y_3 might correspond to age and take values 0 thru 99 or 1 thru 99. We set R_n equal to the set of values that field y_n can assume and $D = \Pi R_n$. For convenience, we always assume that values in a R_n take values 1 thru k_n where the k_n integers are recodes of the k_n value states associated with field y_n. An edit E is a point set $P(E) \subseteq D$. A record r fails E is $y \in P(E)$. FH showed that an arbitrary edit E can be expressed as a union of edits E^i of a particular form. Each E^i can be expressed as ΠE_{in} where E_{in} is the set of values assumed by the nth component of the points y_n in edit E^i. This form of E^i is called the *normal form*. If E_{in} is a proper subset of R_n, then field n is said to *enter* edit E^i and edit E^i is *involved* with field n. Entering fields of an edit E are those fields that are restricted by the edit E. If $\underline{E} = \{E^1, E^2, \ldots, E^m\}$, then use $P(\underline{E})$ to denote the union $\cup \{ P(E^i): E^i \in \underline{E}\}$.

If $r^0 \in P(\underline{E})$ for some set of edits $\underline{E}$, then the EL problem is to find (or possibly minimize) $\sum_{j \in J} c_j x_j$ subject to

y in $D - P(\underline{E})$

and

$$x_j = 1 \text{ if } y_j = y_j^0 \qquad (2.1)$$
$$= 0 \text{ otherwise,}$$

where $j = 1, \ldots, m$. The coefficient c_j in the above sum is a confidence weight. In some situations, all of the c_j are set to one. The vector $x = (x_1,\ldots,x_m)$ tracks the specific fields in the original record $r^0 = (y_1^0,...,y_n^0)$ that are changed. If c_j is large, then the field j is much more unlikely to contain an error. The relative values of c_j are either decided by analysts who are familiar with the editing of the survey or using empirical data from prior editing.

We now make two restrictions that can be made without loss of generality in terms of the theory and practical application in software. The first is that every edit E^i has at least two entering fields. Let an edit E^i had only one entering field, say j. Then field j would have at least one value-state that would always result in an error regardless of the values that other fields assumed. For instance, if the j^{th} field consisted of a postal code corresponding to a U.S. State, then we would not consider any such codes that assumed invalid values. Such single-field edits are best dealt with by lookup tables associated with pre-edits in the keypunch software. Thus, while State codes can take any value, we restrict the State codes passed to the edit system of this

paper to valid ones. These valid State codes may still be used in multi-field edits because different combinations of edits may be associated with different edits in, say, different States of a national agricultural survey. Our second restriction is that, for each n, $R_n = \cup \{\tilde{E} \in \underline{E}^o \mid \neq R_n\}$ where $\underline{E}^o$ is the original set of explicit edits defined by analysts. If the union were a proper subset of R_n for some n, then any record y with a component y_n in R_n but not in the union would necessarily pass all edits. The first restriction means that we only consider value-states of fields that enter at least one edit. The second that there are no value-states of individual fields that do not enter at least one field in one edit. In practice, these restrictions could easily be checked via straightforward combinatorial routines. This would alleviate tedious, possibly error-prone checking by analysts. The restrictions facilitate our theoretical development but do not affect software development.

The following lemma of FH is the basis of generating edits in the normal form. For the remainder of the paper, we will only consider edits in the normal form because any system of discrete edits can equivalently be expressed in normal form. FH proved (1976, Theorem 2) that all of the implicit edits can be generated by successive application of the generation procedure given in Lemma 1.

Lemma 1. Let $S = \{E^j, j = 1,..., k\}$ be an arbitrary set of normal form edits such that for some field l, E_{jl} is a proper subset of R_j. Let E^* be the edit defined by:

$$E_{*i} = \bigcap_j E_{ji} \quad \text{for } i \neq l \tag{2.2a}$$

$$E_{*l} = \bigcup_j E_{jl} \tag{2.2b}$$

If $E_{*i} \neq \varnothing$ for $i \neq l$, then E^* is an implied edit in the normal form.

If a record *r* fails an implied edit E^* of the form given in Lemma 1, then *r* necessarily fails one of the edits used in generating E^*. The set S is called the *contributing* set of edits used in generating edit E^*. Field l is called the *generating field or node* of E^*. Field l necessarily enters each edit involved in the generation procedure of the lemma. If $E_{*l} = R_l$ then edit E^* is called *essentially new*. In the partial ordering of set inclusion, a normal-form edit is said to be *maximal* if it is properly included in no other normal-form edit. A normal form edit is *redundant* if it is properly included in another normal-form edit. The set of explicit edits plus the set of maximal, normal-form implicit edits is called the *complete* set of edits. The set of original explicit edits is denoted by $\underline{E}^o$ and the set of *complete* edits is denoted by $\underline{E}^c$. FH had originally defined the set of complete edits as the explicit edits plus the set of essentially new, normal-form edits. GKL noted that the proof of FH for the error-localization problem holds for the complete set as defined in this paper. Our definition of complete is the one due to GKL rather than the one due to FH. A set of edits is *consistent* if there is at least one record that fails no edit. The set of edits used in generating an implied edit will be called its *generating set*. Generating sets are not unique.

Let r^0 in $P(\underline{E})$. Then consider the set-covering problem (SCP):

$$\text{minimize } \sum_{j \in J} c_j x_j \tag{2.3}$$

$$\text{subject to } \sum_{j \in J} a_{ij} x_j \geq 1, \; E^i \text{ in } \underline{E}_F(r^0)$$

where

a_{ij} = 1 if field j enters E^i
= 0 otherwise

and $\underline{E}_F(r^0)$ is the set of edits that are failed by r^0. FH showed that the solution to (2.3) is the same as the solution to (2.1) provided the set $\underline{E}$ of edits is a complete set of edits.

In this paper, we are concerned with performing EL when the set of edit $\underline{E}$ is incomplete. Two approaches might be taken. The first is to use a heuristic to quickly determine additional fields that might be changed. Within the first approach, there are two variants. In the first variant, exemplified by NIM, the donor record yields a superset J* of the set of fields that must be changed. In the second variant, we identify a preliminary set of fields J to change (based on an incomplete set of edits $\underline{E}$. The preliminary set is also extended to a superset J* that must be changed. In each of the variants, a subset is then identified that represents the actual fields to change. In the second approach, additional implicit edits are located during the course of EL. There are two variants. In this paper, we use a method that utilizes information obtained during the edit-generation process (e.g., Winkler 1997, Chen 1998) and new ideas of this paper. In the variant due to GKL, a cutting plane algorithm (called GKL Algorithm 2) is used to identify all the failing edits during EL. Because Algorithm 2 gives significant insights into some of the information needed for reducing computation, we state it.

GKL Algorithm 2

1. Solve the SCP (2.3) and denote the solution x*.
2. Let $J = \{j \mid x_j^* = 1\}$. Fix the values of r_j for $j \notin J$ at r_{j0}, but for every j in J, let r_j assume each of the values of R_j. Test each of the $\Pi_{j \in J} \mid R_j \mid$ possible records y so defined for membership in D – P($\underline{E}$) where $\underline{E}$ is the existing set of explicit and implicit edits. If no such record is found, x* specifies a solution to (2.1). Otherwise, go to Step 3.
3. Find any prime cover v^0 to

$$vQ \geq 1 \qquad (2.4)$$

v binary

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

q_{ik} = 1 if E^i is failed by the kth record y of Step 2
= 0 otherwise.

Let $I^0 = \{i \mid v_i^0 = 1\}$.

4. Generate the implied edit E^0 given by

$$E_{0j} = \cap \{E_{ij} \;\; i \in I^0\}, \; j \in J \qquad (2.5)$$
$$= Rj, \qquad j \notin J.$$

Let $E_F(r^0) = E_F(r^0) \cup \{E^0\}$. Go to Step 1.

GKL Algorithm 2 gives a way of finding all of the additional failing edits of the form E^0 for a record r^0. Additionally, if we take any entering field i^0 in E^0, we can iteratively expand the initial cover J to $J_1 = J \cup \{i^0\}$. At the completion of the iteration process, we have a prime cover of the failing edits for record r^0. As noted by GKL, the excessive number $\Pi_{j \in J} | R_j |$ of patterns of Step 2 typically make this procedure computationally intractable except in very small situations. Our alternative to GKL Algorithm 2 will alleviate much of the excessive computation by making use of much more of the information available from the edit-generation process of creating an incomplete set of edits.

To further prepare for the theoretical results of this paper, we need to state and explain a number of technical lemmas. Each of the technical lemmas can be deduced from ideas in FH or GKL. The technical lemmas are important because they give key insights into how computational can be reduced. The first technical lemma is just a restatement of the comments following GKL Algorithm 2.

Lemma T1. Let J be a prime cover of an incomplete set of failing edits $\underline{E}_F(r^0)$ for record r^0. Then, J can be expanded to a prime cover $J^* = J \cup \{f_1, \ldots, f_n\}$ that is a prime cover of all the failing edits for r^0.

Our goal in this paper is to find computationally tractable methods of expanding an incomplete set of edits and to quickly find prime covers for all of the failing edits for all of the records $r \in R$. The following technical lemma can be deduced from ideas in FH or in GKL.

Lemma T2. Let $\underline{E}$ be a complete set of edits. Let r^0 be an edit-failing record. Then $\underline{E}$ gives information about non-failing explicit edits for r^0 that might fail after fields in r^0 are changed.

The following technical lemma can be deduced from Theorem 1 of FH.

Lemma T3. Let $\underline{E}$ be a complete set of edits. Let r^0 be an edit-failing record. Let J be a prime cover of the edits in $\underline{E}$ that are failed by r^0. Let E^0 be an explicit edit that does not fail for record r^0 that has all of its entering fields $I^0 \subseteq J$. Then, there exist values of the fields I^0 so that the record r_1 that differs from r^0 by the changed values in the fields of I^0 satisfied all of the edits in $\underline{E}$.

The following technical lemma can be deduced from Theorem 1 of FH or more specifically and easily from GKL Algorithm 2. It can be considered a variant of Lemma T3.

Lemma T4. Let $\underline{E}$ be a complete set of edits. Let r^0 be an edit-failing record. Let J be a prime cover of the edits in $\underline{E}$ that are failed by r^0. Let E^0 be an explicit edit that does not fail for record r^0. Let I^0 be the set of entering fields that are in J. Then the fields in I^0 must be changed into a subset of the values that they can assume. If all of the records r_1 that can be obtained from r^0 by these changes in the fields of I^0 fail, then edit E^0 necessarily has one entering field f_1 that is not in J.

Technical lemmas T1-T4 will provide us with key insight in the following sections. Let J be a prime cover of the set of edits in incomplete set E that fail for record r^0. We intend to show that J can be extended to a prime cover J* of the set of all edits that fail for r^0. Part of our development will involve demonstrating how explicit edits that do not originally fail for record r^0 can fail as fields in the cover J are changed. Further elementary motivation and insights are given in section 3.1.

3. THEORY

This section contains theory and explanations that are intended to make the understanding of the computational algorithms as straightforward as possible. In the first section, we cover background on how the failing explicit and implicit edits are used to determine the exact set of fields that must be changed in an edit-failing record r. Furthermore, we show how record r is filled in. By *fill in*, we mean how new values are imputed into fields in r so that record r will no longer fail any of the explicit edits. Let the set of explicit and implicit edits E be incomplete. If r is a record that fails an implicit edit that is not in E, then we indicate what can go wrong as the record is filled in. In the second section, we provide a detailed characterization of how contradictory conditions occur when a cover J* of the failing edits of a record r does not cover a field in a failing implicit edit that is not in E. We show how to quickly determine additional fields J_1* such that changing values in fields of J* $\cup$ J_1* yields a record r' that satisfies all of the explicit edits. In the third section, we provide the main theorem that additionally shows, for any set R of records, how to quickly generate missing implicit edits "on-the-fly." In the fourth section, we show that the Nearest Neighbor Imputation Method (Bankier 1991, also 1997, 2000) can be considered a special case of the theoretical results of this paper. Because the main theorem and the computational methods represent an extension of existing FH theory, NIM is consistent with the FH Theory.

3.1. Basic Theoretical Background

FH theory gives that any prime cover J*of the entering fields of the complete set E of failing explicit and implicit edits can lead to a record r' that satisfies all edits. The record r' differs from the original record r only for the values in the fields in the cover J*. A cover is *prime* if no subset of J* is also a cover. Any prime cover J* of the failing explicit and implicit edits is said to be an *error-localization* (*EL*) solution. It satisfies the conditions (2.3) and (2.1)

Let S* be all fields that are not in J*. Start with values fixed in each field in S*. Let $f_1, f_2, \ldots, f_n$ be the fields in J*. The record r can be filled in by successively finding values for each field f_k in J* such that the existing record with values fixed for all fields in S* and for fields $f_1, \ldots, f_k$ satisfies all explicit and implicit edits that have entering fields in S* $\cup$ $\{f_1, \ldots, f_k\}$.

FH theory assures that we can find the values of fields $f_1, \ldots, f_k$ that can be used in successively filling in a record. We say that a value for a field is *assigned* if the value is fixed. With a complete set E of explicit and implicit edits, finding a cover J* and filling a record in is straightforward. If a record has values assigned in fields in S* $\cup$ $\{f_1, \ldots, f_k\}$, we need to find a value for field f_{k+1}. The fields $\{f_1, \ldots, f_n\}$ = J* are the prime cover of the failing explicit and implicit edits. We consider all explicit and implicit edits $E^1, \ldots, E^n$ that have one entering field f_{k+1} and the rest of its entering fields in S* $\cup$ $\{f_1, \ldots, f_k\}$. By FH theory, the complement of the unions of the set of values for entering fields f_{k+1} in edits $E^1, \ldots, E^n$ must not be the empty set. We can choose any value for f_{k+1} that is in the intersection of these complement sets of values for field f_{k+1} that is determined by edits $E^1, \ldots, E^n$.

By FH theory, any prime cover J* of the entering fields in a complete set $\underline{E}$ of failing explicit and implicit edits gives a proper set of starting values S* that is the set of fields that are complementary to J*. A set of starting values is called *proper* if the remaining fields in a record can be filled in so that the record satisfies all edits. Let record $r \in R$ fail an incomplete set $\underline{E}$ of explicit and implicit edits. Let J* be a cover of the entering fields in the failing edits. Let S* be a proper set of starting values (i.e., a subset of the complement of J*). Finding a proper set of starting values S* is more important than finding J*. Any subset of a proper set of starting values is also a proper set of starting values.

Let the record $r \in R$ fail a set of explicit and implicit edits $\underline{E}_F(r)$. FH theory implicitly gives information about potentially failing explicit edits. A *potentially failing edit* (explicit or implicit) is one that can fail in some situations when an entering field in an initially failing explicit or implicit edit is changed. Note that a potentially failing record does not fail initially. Assume that one field in a record is changed and a potentially failing edit now fails. Take a second entering field from the originally potentially failing edit and change the value for the record for the second field so that the values are not in the set of values of the second entering field. Then, the record will necessarily pass edits. This idea follows from Lemma T4.

We build intuition with the simplest situation. Let $r \in R$ be a record that has four fields. Assume that each field has two entering values. Let explicit edit E^1 have entering fields f_1 and f_2. Let explicit edit E^2 have entering fields f_2 and f_3. Assume that implicit edit I^1 is generated from E^1 and E^2 using field f_2. Let record r fail edits E^2 and I^1. Assume that the value in entering field f_1 for record r agrees with the value in the entering field for edit E^1. Assume that we only have explicit edits E^1 and E^2. Implicit edit I^1 is not available. Let $J^* = \{f_2\}$ be the cover of failing edit E^2 and $S^* = \{f_1, f_3, f_4\}$. Change the value in field f_2. Then edit E^1 fails.

The previous example can be extended to any situation where there is a missing implicit edit. That is, if an implicit edit is missing from a cover, then a field in a cover of the existing, incomplete set of explicit and implicit may be changed in a manner that causes a potentially failing explicit edit E^k to fail. If one of the fields in the missing implicit edit that agrees with another field in the potentially failing edit I_2 is also changed, then the edit E^k will not fail. Note that if a set of values is assigned (based on the initial set S* and the subset of values in fields in J* that have been changed) and if an implicit edit fails, then an explicit edit with entering fields in the set of assigned values will necessarily fail.

Let J* be a cover of a set $\underline{E}$ of failing explicit and first-level implicit edits. Let E^k be non-failing explicit edits that have exactly one entering field f_k agreeing with a field in J* and that fails when the value for field f_k in J* is changed. Necessarily the value placed in J* must be in the intersection of the complements of the set of values for the entering field f_k in originally failing explicit and first-level implicit edits. Let f_l be another entering field in E^k. Let I* be the complementary fields in all such non-failing explicit edits that fail after a field from J* is changed. Let S_1^* be the complement of the fields in $J^* \cup I^*$. Then S_1^* is a proper set of starting values. S_1^* may not be a maximal set of starting values. A proper set of starting values is maximal if no superset of it is proper.

Assume that an incomplete set of explicit and implicit edits exists. For the remainder of this paper, we assume that, at a minimum, the set of implicit edits must include all first-level implicit edits. There are several different ways in which error localization could be performed in the main edit program. First, if the set of incomplete edits is assumed to be nearly complete, then it may be best to take a cover J* of the failing edits in the incomplete set and fail to fill-in a record. It may be sufficient to find additional fields that need to be added to the set of fields in J*.

Second, it may be best to immediately look for fields I* to add to cover J* prior to doing error localization. Third, if J* is missing a moderate number of implicit edits, then it may be better to generate additional implicit edits based on the existing set of failing edits. The generation would be an efficient hybrid of the GKL algorithm 2 that targets only one field at a time. Suitable test decks may be good for finding additional implicit edits prior to running the main edit program.

3.2. Fundamental Characterizations

We begin by examining the simplest situations in order to build intuition. We assume that each field only assumes two values, that each edit only contains two entering fields, and that no skip patterns exist. The extensions to where each field assumes more than two values are straightforward (see e.g., Winkler 1997). The extensions to more than two entering fields in an edit are cumbersome and tedious (see Appendix). The extensions to when there are skip patterns are cumbersome and non-trivial (see Appendix).

Assume that the edit scenario involves records having n fields. The only implicit edits that we consider are maximal implicit edits (as in Winkler 1997 or Chen 1998). An edit is a *maximal implicit edit* if it is implicit and it is not properly contained in another implicit edit. GKL showed that the set of explicit and maximal implicit edits is sufficient for solving the error-localization problem. The set of implicit edits that are generated from explicit edits are called first level implicit edits. By induction, we define the nth level edit as those that are generated from a contributing set of edits in which there exists at least one edit from level n-1. If an implicit edit E^2 can be generated by an implicit edit at a given level and other edits, it is called a *successor* of E^1; E^2 is called the *predecessor* of E^1. During the edit-generation process, we proceed down edit generation trees determined by the fields (nodes) on which the implicit edits are generated. If $\underline{E}$ is a set of explicit and implicit edits, then the *lowest level* of $\underline{E}$ is the subset of edits that have no currently known successor edits.

Lemma 2. (Winkler 1997). Assume that there are no skip patterns associated with a set of edits. Then every implicit edit at level n is generated by an implicit edit from level n-1 and a set of explicit edits.

If a field only assumes two values, then an implicit edit is generated using exactly one explicit edit and an implicit edit from the previous level. Let a record r fail an explicit edit E^0 that is generated by implicit edit E^1 and explicit edit E^2 on generating field f_1. If r fails edit E^1 and the value in field f_1 is changed, then the resultant changed record fails explicit edit E^2.

Lemma 3. Let E be an implicit edit at the nth level that is generated from n-1st level edit $E^1$and another edit E^2 using field f_1. Let r be a record that fails edit E. If r fails edit E_1 and the value in field f_1 is changed, then record r fails edit E^2. If r fails edit E_2 and the value in field f_1 is changed, then record r fails edit E^1.
Proof. See appendix.

Let E be an edit at the nth level. Let record r fail E. We say that $C=\{E^1, E^2, \ldots., E^n\}$ is a *r-chain* leading to edit E if E^1 is an explicit edit (i.e., at the 0^{th} level), $E=E^n$, and record r fails all edits in C. Then the following lemma can be proved by inductively applying Lemma 3.

Lemma 4. Assume that $\underline{E}$ is an incomplete set of edits. Let E^0 be an implicit edit that is missing from $\underline{E}$. Let r be a record that fails edit E^0. There exists an r-chain leading to E^0.
Proof. See appendix.

It is not necessary for r-chains to be unique. Let record r fail an edit E^0 at level n. Let $C=\{ E^1, E^2, \ldots., E^n\}$ be an r-chain leading to E^0. Assume that $D=\{F^1, \ldots, F^{n-1}\}$ are explicit edits that satisfy the following property, for $j>1$, E^j has contributing edits E^{j-1} and F^j. By inductively applying Lemmas 2-4, we have that record r cannot fail any of the edits in D. However, if the value in a generating field is changed, then it is possible for the appropriate explicit edit to fail. If $\underline{E}$ is incomplete, then typically some of the edits in the r-chain C will not be in $\underline{E}$.

Lemma 5. Let $\underline{E}$ be an incomplete set of edits and let I be an implicit edit that is not in $\underline{E}$. Let r be a record that fails implicit edit I. Let D_1 be the set of explicit and implicit edits in $\underline{E}$ that are failed by r. Let $F=\{f_1, \ldots, f_n\}$ be any prime cover of D_1 that does not cover I. Let r' be the record obtained from r by changing each of the values in F. Then r' fails at least one explicit edit E^0.
Proof. See appendix.

Lemma 5 is just a variant of technical lemmas of section 2. Necessarily, there exists an r-chain C that goes from E^0 to I. Not all of the edits in the r-chain C need to be in $\underline{E}$.

Lemma 6. Let $\underline{E}$ be an incomplete set of edits. Let edit E^1 be at the lowest level. Let record r fail edit E^1. Let E^1 have entering fields f_1 and f_2. Let record r' be obtained from record r by changing either field f_1 or field f_2. Assume that record r' fails an explicit edit E^2 that record r did not fail. Then it is possible to generate an implicit edit E^3 with contributing edits E^1 and E^2 such that E^3 is failed for record r.
Proof. See appendix.

We could also deduce Lemma 6 from GKL Algorithm 2. The difference is that Lemma 6 yields a computationally much faster method of finding each field that must be changed. The same is true for the following lemma.

Lemma 7. Let $\underline{E}$ be an incomplete set of edits. Let edit E^1 be at the lowest level. Let record r fail edit E^1. Let E^1 have entering fields f_1 and f_2. Assume that edit E^1 is generated by implicit edit E^3 and explicit edit E^4 on field f_3. Assume that record r fails implicit edit E_3 and does not fail explicit edit E^4. Assume that f_1 enters explicit edit E^4. Let record r' be obtained from record r by changing field f_3. Then record r' may or may not fail explicit edit E^4. Assume that record r' fails explicit edit E^4. Let record r_1 differ from record r' by changing field f_1. Then record r_1 does not fail E^4, E^3, or E^1.
Proof. See appendix.

Lemma 8. Let $\underline{E}$ be an incomplete set of edits. Let r be a record that fails some of the edits in $\underline{E}$. Then it is possible to expand $\underline{E}$ to a set of edits $\underline{E}'$ that contains all of the edits that are failed by record r.
Proof. See appendix

During the implicit edit-generation process, it is straightforward to track on what field the implicit edit was generated, what edits contributed to the implicit edit, and at what level the implicit edit was generated. For each implicit edit, it is possible to find an (non-unique) r-chain leading back to an explicit edit at the 0^{th} level. Because we are interested in implicit edits that may be missing from the set of edits that we have generated, we need only consider implicit edits that are at the end of an r-chain. In other words, if $\underline{E}$ is the existing set of explicit and implicit edits, then we only consider implicit edits $\underline{E}^f$ in E that are at the lowest level of the r-chains.

If we have an incomplete set $\underline{E}$ of explicit and implicit edits, then Lemma 6 gives us a very quick way to determine additional fields that would yield an EL solution. If we generate and store the additional implicit edits that are obtained by repeatedly applying Lemma 6, then we have all of the implicit edits that a record r fails. We can then reapply the main EL routine to get a more efficient solution than would be obtained by adding fields f_E to the fields of the cover of the incomplete set $\underline{E}$. We can think of this as a quick, heuristic approach to finding an EL solution.

3.3. Main Theorem

Let us assume that incomplete set $\underline{E}$ contains most of the implicit edits. It is very rapid to expand $\underline{E}$ to $\underline{E}$' with implicit edits that are found via Lemma 7. For a given set of records R, if $\underline{E}$ is expanded to $\underline{E}$', then set $\underline{E}$' would necessarily contain all implicit edits that fail for R. The generalization of Lemma 7 to situations in which a field can assume more than two values is straightforward (see the appendix). The generalization to situations when skip patterns are present (on the survey form and in the set of edits) is not straightforward. It involves a series of technical lemmas and results that extend Winkler (1997).

We are now in the position to state the main theorem.

Theorem 1. Let $\underline{E}$ be an incomplete set of edits. Let R be a set of records that fail edits in $\underline{E}$. Let $r \in R$ be a record that fails at least one implicit edit that is not in $\underline{E}$. Let $\underline{E}_F(r)$ be the set of implicit edits in $\underline{E}$ that are failed by record r. Let $F=\{f_1, \ldots, f_n\}$ be the set of fields that are a prime cover of $\underline{E}_F(r)$. Then it is possible to very quickly find a set of fields $F_r =\{f_{n+1}, \ldots, f_q\}$such that $F \cup F_r$ is an EL solution for r. Furthermore, it is possible to find the implicit edits $\underline{E}_{mr}$ that are failed by r that are not in $\underline{E}_F(r)$.

Corollary 1. Let $\underline{E}$ be an incomplete set of edits. Let R be a set of records that fail edits in E. Then $\underline{E}$ can be expanded to a set of edits $\underline{E}$' that contain all of the implicit edits that fail for records in R.

In the situation when there are no skip patterns, the algorithm is straightforward.

Algorithm NS1 (no skip patterns)

1. Let $E_F(r^0)$ be the set of failing edits for record r^0. Solve the SCP (2.3) and denote the solution x*.

2. Let $J = \{j \mid x_j^* = 1\}$. Fix the values of y_j for $j \notin J$ at y_{j0}. For each $j \in J$, let $E_j^c = \cap \{F_{ji}^c , j \in J, E^i \in E_F(r^0)\}$ be the intersection of the complements of the jth entering fields. If every value f_j in E_j^c yields a newly failing explicit edit $E^{k(j)}$, then generate a new implicit

edit $I^{k(j)}$ that fails for record r^0. If, for all $j \in J$, there exists no such $E^{k(j)}$, then x* is a solution to (2.1); else let $\underline{E}_1 = \{I^{k(j)}, j \in J\}$ be the set of new implicit edits. Let $E_F(r^0) = E_F(r^0) \cup \underline{E}_1$. Go to Step 1.

An alternative, much faster algorithm is

Algorithm NS2 (no skip patterns)

1. Let $E_F(r^0)$ be the set of failing edits for record r^0. Solve the SCP (2.3) and denote the solution x*.

2. Let $J = \{j \mid x_j^* = 1\}$.

3. Fix the values of y_j for $j \notin J$ at y_{j0}. For each $j \in J$, let $E_j^c = \cap \{F_{ji}^c, j \in J, E^i \in E_F(r^0)\}$ be the intersection of the complements of the jth entering fields. If every value f_j in E_j^c yields a newly failing explicit edit $E^{k(j)}$, then let d(j) be a complementary entering field in $E^{k(j)}$ that is not in J. If, for $j \in J$, there exists no such $E^{k(j)}$, then go to Step 4; else let $\underline{J}_1 = \{ d(j), j \in J\}$ and $J = J \cup J_1$. Go to Step 2.

4. Find a subset of J that yields a solution to (2.1).

Algorithm NS2 is far faster than Algorithm NS1 because new implicit edits do not need to be computed at intermediate stages of the algorithm. Algorithm NS2 will generally not yield a minimal solution to (2.1). Algorithm NS1 can yield a minimal solution to (2.1) if a branch-and-bound or similarly appropriate algorithm is applied to (2.3) with the new set of implicit edits that fail for record r^0. Algorithm NS1 is far faster than GKL Algorithm 2 because it does not require enumerating all the $\Pi_{j \in J} \mid R_j \mid$ possible records y associated with the cover J and finding all existing edits that fail for them.

We can think of a set of records as a set of points in the product space of the values assumed by all of the fields. With a large survey such as the Italian Labour Force Survey (Barcaroli and Ventura, 1997, 1993; Winkler 1997), the product space of slightly more than 100 fields may consist of on the order of $4.6 \times 10\text{^}79$ points. For this survey, there are six hundred explicit edits. There may be as many as 6,000 implicit edits. For this set of data, we extrapolate that conventional set-covering (e.g. GKL) would need between 800 and 8,000 days on an exceptionally fast computer to generate all of the implicit edits. The heuristic algorithms of Winkler (1997) and Chen (1998) generate at least 90% of the implicit edits in less than 24 hours. Because a survey will not contain nearly as many records as there are in the product space or in the complete set of implicit edits, a more practical day-to-day procedure may be as follows. Generate 90% of the implicit edits using the heuristic algorithms. For a given set of survey records, generate the remaining implicit edits "on-the-fly" in the main edit program.

3.4. The Nearest-Neighbour Imputation System

Bankier (1991, see also 1997, 2000) introduced the Nearest-Neighbour Imputation Method (NIM). NIM has always been suitable for nearly discrete data fields such as age that are very similar to numeric fields. Bankier (2000) shows how NIM is extended to situations involving general continuous data. For convenience, we will only consider the discrete data situation.

NIM has two stages. In the first, an edit-failing record r_0 is compared with a large set of edit-passing records. Using a metric that counts the number of fields that differ between the records, a group G of edit-passing records that differ on the smallest number of fields is obtained. Each record r in G differs from record r_0 on a number of fields F_r. Fast heuristics determine the approximate minimal number of fields F_r' in F_r that can be changed and still yield an edit-passing record. The final imputation for record r_0 is obtained by simple random sampling of those records that are closest in terms of the number of fields in the sets F_r'.

In potential donor records, NIM considers all of the fields that differ from the edit-failing record. Assume the initial number of differing fields is N. To determine the approximate minimum number of fields to change, NIM first determines whether single fields can be dropped. This is equivalent to determining EL solutions consisting of N-1 fields. If a solution can be found having N-1 fields, then NIM may look for solutions having N-2 fields and so on. Although NIM is the direct inspiration for the approach used in this paper, NIM methods can be considered a special case of methods used in Lemmas 3-5. In NIM, there is a direct computational advantage because the value of the field that must be substituted is already known. The following is another corollary to the theorem of this paper.

Corollary 2. The Nearest-Neighbor Imputation Method (NIM) is consistent with the (computational) extensions of the Fellegi-Holt model of statistical data editing as detailed in this paper.

If there is a very large set of suitable donors, then NIM will get solutions (in terms of the number of fields changed) that are as good as those obtained by FH. NIM will automatically find the best nearest-neighbor matching rules. NIM drastically reduces computation because it only considers computational paths associated with the available donors. It only considers changing the values of the fields in F_r between the existing value in record r_0 and potential donor record r.

4. DISCUSSION

As with the GKL paper, this paper primarily deals with computational extensions of the FH model. In a roundabout way, Bankier's NIM procedure provides key insights that eventually led to the computational improvements of this paper. Let a record r in R fails some of the explicit edits. Let a donor record r_1 that satisfies all of the edits. Let J* be the set of fields that differ between r and r_1. Then J* is necessarily a superset of the EL solution. The NIM method works in a top-down method. The heuristic method of this paper works in a bottom-up method in finding a superset J_1* of the EL solution. In both situations, an EL solution might be equal to the sets J* or J_1*. NIM is more straightforward because it limits the computational paths to either changing a field to the value in the donor record or leaving at the value in the original record. The heuristic method of this paper must deal with more of the possible changes in field values than those of NIM.

The main theorem of this paper provides a slower method of finding all failing implicit edits for a set of records R. It further allows finding minimal EL solutions.

5. CONCLUDING REMARKS

This paper describes theoretical and computational extensions of the Fellegi-Holt model of statistical data editing. The main application is in determining the approximate minimum

number of fields to impute in situations when not all implicit edits can be generated a priori. As a special case, a theoretical justification for Bankier's Nearest-Neighbour Imputation Method is given.

1/ This paper reports the results of research and analysis undertaken by Census Bureau staff. It has undergone a Census Bureau review more limited in scope than that given to official Census Bureau publications. This report is released to inform interested parties of research and to encourage discussion.

REFERENCES

Barcaroli, G., and Venturi, M. (1993), "An Integrated System for Edit and Imputation of Data: an Application To the Italian Labour Force Survey," *Proceedings of the 49th Session of the International Statistical Institute*, Florence, Italy, September 1993.

Barcaroli, G., and Venturi, M. (1997), "DAISY (Design, Analysis and Imputation System): Structure, Methodology, and First Applications," in J. Kovar and L. Granquist, (eds.) *Statistical Data Editing, Volume II*, U.N. Economic Commission for Europe, 40-51.

Bankier, M. (1991), "Alternative Method of Doing Quantitative Variable Imputation," Statistics Canada Memorandum.

Bankier, M., Houle, A.-M., Luc, M. and Newcombe, P. (1997), "1996 Canadian Census Demographic Variables Imputation," *American Statistical Association, Proceedings of the 1997 Section on Survey Research Methods*, 389-394.

Bankier, M. (2000), "2001 Canadian Census Minimum Change Donor Imputation Methodology," U.N. Economic Commission for Europe Work Session on Statistical Data Editing, Cardiff, UK, October 2000 (also available at http://www.unece.org/stats/documents/2000.10.sde.htm).

Chen, B.-C. (1998), "Set Covering Algorithms in Edit Generation," *American Statistical Association, Proceedings of the Section on Statistical Computing*, 91-96 (also available as Statistical Research Division Report rr98/06 at http://www.census.gov/srd/www/byyear.html).

Chen, B.-C., Winkler, W. E., and Hemmig, R. J. (2000), "Using the DISCRETE edit system for ACS Surveys," Statistical Research Division Report rr00/03 at http://www.census.gov/srd/www/byyear.html .

Chernikova, N.V. (1964), "Algorithm for Finding a General Formula for the Non-negative Solutions of System of Linear Equations," *USSR Computational Mathematics and Mathematical Physics*, **4**, 151-158.

Chernikova, N.V. (1965), "Algorithm for Finding a General Formula for the Non-negative Solutions of System of Linear Inequalities," *USSR Computational Mathematics and Mathematical Physics*, **5**, 228-233.

De Waal, T. (1996), "CherryPi: A computer program for automatic edit error localization," Paper presented at the UN Work Session on Statistical Data Editing, 4-7 November 1996, Voorburg, the Netherlands.

De Waal, T. (1997), "A recipe for applying CherryPi to the edit process," Paper presented at the UN Work Session on Statistical Data Editing, 14-17 October 1997, Prague, Czech Republic (also available at http://www.unece.org/stats/documents/1997.10.sde.htm).

De Waal, T. (2000), "New Developments in Automatic Edit and Imputation at Statistics Netherlands," U.N. Economic Commission for Europe Work Session on Statistical Data Editing, Cardiff, UK, October 2000 (also available at http://www.unece.org/stats/documents/2000.10.sde.htm).

Draper, L. and Winkler, W.E. (1997), "Balancing and Ratio Editing with the new SPEER system," *American Statistical Association, Proceedings of the 1997 Section on Survey Research Methods*, 570-575 (also available as Statistical Research Division Report rr97/05 at http://www.census.gov/srd/www/byyear.html).

Fellegi, I. P. and Holt, D. (1976), "A Systematic Approach to Automatic Edit and Imputation," *Journal of the American Statistical Association*, **71**, 17-35.

Filion, J.-M., and Schopiu-Kratina, I. (1993), "On the Use of Chernikova's Algorithm for Error Localization," *Statistics Canada Technical Report.*

Garfinkel, R. S., Kunnathur, A. S. and Liepins, G. E., (1986), "Optimal Imputation of Erroneous Data: Categorical Data, General Edits," *Operations Research*, **34**, 744-751.

Garcia, M. and Thompson, K. J. (2000), "Applying the Generalized Edit/Imputation System AGGIES to the Annual Capital Expenditures Survey," *Proceedings of the International Conference on Establishment Surveys, II*, 777?-789.

Kovar, J.G., MacMillan, J.H. and Whitridge, P. (1991), "Overview and Strategy for the Generalized Edit

and Imputation System", Statistics Canada, Methodology Branch Working Paper BSMD 88-007E (updated in 1991).
Little, R. A., and Rubin, D. B., (1987), *Statistical Analysis with Missing Data*, John Wiley: New York.
Nemhauser, G. L. and Wolsey, L. A., (1988), *Integer and Combinatorial Optimization*, John Wiley: New York.
Rubin, D.S. (1975), "Vertex Generation in Cardinality Constrained Linear Programs," *Operations Research*, **23**, 555-565.
Schiopu-Kratina, I. And Kovar, J.G. (1989), "Use of Chernikova's Algorithm in the Generalized Edit and Imputation System," Statistics Canada, Methodology Branch Working Paper BSMD 89-001E.
Winkler, W. E. (1995), "Editing Discrete Data," *American Statistical Association, Proceedings of the Section on Survey Research Methods*, 467-472 (also available as Statistical Research Division Report rr97/04 at http://www.census.gov/srd/www/byyear.html).
Winkler, W.E. (1997), "Set-Covering and Editing Discrete Data," *American Statistical Association, Proceedings of the Section on Survey Research Methods*, 564-569 (also available as Statistical Research Division Report rr98/01at http://www.census.gov/srd/www/byyear.html).
Winkler, W. E. (1999), "The State of Statistical Data Editing," in *Statistical Data Editing*, Rome: ISTAT, pp. 169-187 (also available at http://www.census.gov/srd/www/byyear.html).
Winkler, W. E., and Petkunas, T. (1997), "The DISCRETE Edit System," in J. Kovar and L. Granquist, (eds.) *Statistical Data Editing, Volume II*, U.N. Economic Commission for Europe, 56-62.

APPENDIX

This appendix provides proofs of the main lemmas of this paper and additional explanation that connects the lemmas. Let r be a record that fails an implicit edit I. Let E be an explicit edit that is failed by r. Let r' be a record that is obtained from record r by changing a value in an entering field from edit E. A basic idea is that implicit edit I gives information about an explicit edit E that does not fail originally with record r but may fail with record r'.

Proof of Lemma 3. This follows from identical reasoning to that given in paragraph seven of section 3.1. ■

Proof of Lemma 4. Assume that E^0 would be generated at level n if it had been generated and that $n \geq 1$. By Lemma 3, E^0 is generated by an implicit edit E^0 (n-1) at the n-1st level and explicit edit E^1 (n-1). Because r fails E_0 it must necessarily fail either E^0 (n-1) or E^1 (n-1). Set E^2 (n-1) equal E^0 (n-1) or E^1 (n-1) depending on which one failed. If E^2 (n-1) is equal E^1 (n-1), then we have constructed an r-chain going from E^0 to E^1 (n-1). For each j, $0<j<n-1$, assume that the r-chain has not already terminated and let E^0 (j) = E^2 (j) be the implicit edit at level j that fails. Then there exists implicit edit E_0 (j-1) at level j-1 and explicit edit E^1 (j-1) that are used to generate E^0 (j). Let E^2 (j-1) be equal to E^0 (j-1) or E^1 (j-1) depending on which one failed. If E^2 (j-1) = E^1 (j-1), then the r-chain terminates; else we continue the induction until j-1= 0. ■

By Lemma 2, it is straightforward to extend Lemmas 3 and 4 to the situations in which an implicit edit is generated from one implicit edit and multiple explicit edits. This is the situation when a field can assume more than two value states. It is also straightforward to extend the two lemmas to situations in which edits have three or more entering fields.

Proof of Lemma 5. Record r' necessarily fails implicit edit I because no entering field in I was changed. By Lemma 4, there is an r'-chain leading from some explicit edit E to implicit edit I. Record r' necessarily fails explicit edit E. ■

Proof of Lemma 6. Assume r' differs from r on field f_1. Let E^2 have entering fields f_3 and f_4. Necessarily either field f_3 or field f_4 must equal f_1. Assume that $f_1 = f_3$. Generate edit E^3 on field f_1. E^3 has entering fields f_4 and f_2. Record r fails implicit edit E^3. ■.

Proof of Lemma 7. This is proved by the reasoning of section 3.1. We have changed a prime cover of fields in edits E^4, E^3, and E^1 such that the resultant record r_1 fails none of them. ■

The proof of Lemma 7 depends implicitly on the logical consistency of the edit system as described by FH. Assume edits E^4, E^3, and E^1 from Lemma 7 were contained in a complete set of edits $\underline{E}$. Let record r fail E^4, E^3, and E^1. One of the prime covers of $\underline{E}$ could contain fields f_3 and f_1. Let r' differ from r by generating field f_3. If record r' fails explicit edit E^4, then necessarily by changing the complementary entering field f_1 in E^4 we obtain a record r_1 that must not fail E^4.

Proof of Lemma 8. Let $\underline{E}'$ be the subset of $\underline{E}$ of edits at the lowest level. Define sets of edits at the lowest level that record r fails as follows. Let $\underline{E}_0(r)$ be the subset of $\underline{E}'$ of edits that record r fails. Let E^1 be any edit in $\underline{E}_0(r)$. By Lemma 5B, if the entering fields of E^1 are changed, then either no explicit edit fails or an explicit edit that did not originally fail will now fail. In the former case, define $\underline{E}_1(r) = \underline{E}_0(r) \setminus \{E^1\}$. In the latter case, define $\underline{E}_1(r) = \underline{E}_0(r) \cup \{E^2\} \setminus \{E^1\}$, where E^2 is the successor implicit edit to edit E^1 that is failed by record r. Choose edit E^3 in $\underline{E}_1(r)$. If E^3 has no successor, let $\underline{E}_2(r) = \underline{E}_1(r) \setminus \{E^3\}$. If E^3 has successor E^4 that is failed by record r, let $\underline{E}_2(r) = \underline{E}_1(r) \cup \{E^4\} \setminus \{E^3\}$. For $j > 2$, let E^{2j+1} be an edit in $\underline{E}_j(r)$. If E^{2j+1} has no successor, define $\underline{E}_{j+1}(r) = \underline{E}_j(r) \setminus \{E^{2j+1}\}$. If E^{2j+1} has successor E^{2j+2}, define $\underline{E}_{j+1}(r) = \underline{E}_j(r) \cup \{E^{2j+2}\} \setminus \{E^{2j+1}\}$. Because record r can only fail a finite number of edits, there must exist a value of j for which $\underline{E}_{j+1}(r)$ is the null set. Let $\underline{E}' = \underline{E} \cup (\cup_j \{E^{2j+2}$ for j for which the edit exists$\}$). ■

All of the proofs involve the properties of the values of generating fields and of the entering fields of the contributing edits used in generating an implicit edit. Because the characteristics on an implicit edit at a given level depend on an implicit edit from the immediately preceding level and an explicit edit, the lemmas are straightforward to extend to fields that assume more than two values and to edits that have two or more entering fields. We now proceed to the situation in which skip patterns occur. Prior to proving the extensions for skip patterns, we need to provide background that characterizes skip patterns.

To summarize the previous work when there are no skip patterns. Assume that record r fails edits in an incomplete set $\underline{E}$. Let S be the set of starting values and let J be a prime cover of the fields of the failing edits in $\underline{E}$. As we sequentially change the values in the fields in J, it is possible to find a failing explicit edit E^1 that has at least one entering field that is not in J. By taking the complementary entering field in E^1, we can either expand J to a larger set or generate an additional implicit edit that is not in $\underline{E}$. When skip patterns are present, we wish to quickly find failing explicit edits as the values in the fields in J are changed.

To provide intuition, we describe a typical situation in which skip patterns occur. We will always assume that there are no skip patterns within skip patterns. In other words, our skip patterns are first-level skip patterns. Higher-level skip patterns are extremely rare in real

surveys. For a large labor-force survey, a question may ask age of the respondent. If the age is less than or equal fifteen, the there is a skip to a section of questions on education. If the age is greater than fifteen, then there is a skip to a section of labor-force questions. There is an explicit assumption that individuals less than or equal fifteen will not be working and that they will be in school. Individuals over fifteen that are answering the work-force questions are generally assumed to not be in school. Equivalently, if an individual is answering work-force questions, they are assumed to be over fifteen.

As shown by Winkler (1997), if no skip patterns exist, then each implicit edit at level n can be generated by an implicit edit at level n-1 and a set of explicit edits. If first-level skip patterns exist, then there are implicit edits at level n that can only be generated by an implicit edit at level n-1, at least one first-level implicit edit, and, possibly, additional explicit edits. In the following lemma, we assume that each field assumes just two value states. The extension to more than two value states is straightforward.

Lemma 9. Let $r \in R$ fail some edits in incomplete set $\underline{E}$. Let E^1 be a failing edit at the lowest level of an existing edit-generation tree. Let there exist a missing implicit edit E^2 that fails for r and that will not be covered by any prime cover J of the failing edits in $\underline{E}$. Further, assume E^2 can only be generated from E^1 and a first-level implicit edit E^3 on generating field f_0. Fix a prime cover J that contains field f_0. If f_0 is changed, then there exists explicit edit E^4 that fails.
Proof. Let E^3 be generated from E^{3a} and E^{3b} on some field f_1. If f_0 is changed, then edit E^3 fails. Because the entering fields in E^2 are not covered by J, the entering fields in E^3 that are complementary to f_0 are also not covered.
Note that E^3 need not be observed. As E^3 fails, then either E^{3a} or E^{3b} fails. Assume E^{3a}. Then f_0 enters E^{3a} that is the desired explicit edit E^4. If a complementary entering field f_2 in E^{3a} is also in J, then when f_2 is changed E^{3a} will no longer fail. If f_2 is not in J, then we can expand J with field f_2. ■

In the proof of the prior lemma, we do not need to observe E^2 or E^3. As values in the fields in J are changed, an explicit edit will eventually fail.