

UNITED NATIONS STATISTICAL COMMISSION and
ECONOMIC COMMISSION FOR EUROPE

CONFERENCE OF EUROPEAN STATISTICIANS

Work Session on Statistical Data Editing

(Ottawa, Canada, 16-18 May 2005)

Topic (iv): New and emerging methods, including automation through machine learning, imputation, evaluation of methods

A VARIABLE NEIGHBOURHOOD LOCAL SEARCH APPROACH FOR
THE CONTINUOUS DATA EDITING PROBLEM

Supporting Paper

Submitted by Department of Statistics, Operations Research and Computer Science,
University of La Laguna, Spain¹

Abstract

The combinatorial optimization problem that underlies in the Continuous Data Editing Problem has been shown to be \mathcal{NP} -hard. That is why the computation of the optimal solution for this problem requires an important amount of computational resources. We present in this article a new heuristic algorithm, based on a local search approach, to obtain an near-optimal solution for this combinatorial optimization problem. Some procedures of this local search approach make use of a Benders' decomposition of a linear programming model. Our computational experience on randomly generated instances solves instances up to 500 fields and 200 edits.

I Introduction

Data collected by statistical agencies may contain mistakes made during the acquisition, transcription and coding process. Therefore, since occasionally it is impossible to trace the original source, in order to improve the integrity and quality of decisions made on the basis of this information, the detection and correction of such errors becomes necessary before starting data processing. The task of identifying records containing errors as well as of identifying the specific fields causing such errors is known as the *data editing* problem. Changing these fields in order to correct such errors is known as *imputation*.

The *microdata* collected by the agencies consists of a set of records, each containing answers of a respondent to a set of queries. Each value in a record is known as *field*, and it contains either a discrete or a continuous value. Discrete values correspond to *categorical queries* (e.g., marital status), whereas continuous numbers correspond to *quantitative queries* (e.g., weight). The microdata may contain both data types.

¹Prepared by Juan-José Salazar-González (jjsalaza@ull.es) and Jorge Riera-Ledesma (jriera@ull.es), and supported by the Spanish project TIC 2002-00895 ("Ministerio de Ciencia y Tecnología") and by the HPC-Europa programme, funded under the European Commission's Research Infrastructures activity of the Structuring the European Research Area programme, contract number RII3-CT-2003-506079

The correctness of a record is determined by a set of given consistency rules called *edits*. In this sense, an edit is usually given by either a logical or a mathematical expression involving several fields.

The aim of the *Error Localization Problem* (ELP) is to modify the fewest possible number of fields that would have to be changed for the new valid record to meet the set of consistency conditions [2]. Note that this objective function aims to preserve as much of the original information as possible and it is a way of finding a “most likely” record [6]. Moreover, once the minimum number of fields is determined, an eventual phase of the editing-and-imputation would seek to give values to these fields such that the new records satisfy, in addition to the consistency rules, some specific statistical properties. However, this second phase is not addressed in this work.

The objective of the ELP is commonly extended by considering the minimization of the weighted sum of the number of fields to be changed in order to satisfy the set of edits. In other words, a non-negative weight represents the confidence in the value of its related field, and it can be thought of as a surrogate, under certain assumptions, for the objective of maximizing the product of the probabilities that a changed field is in error; see Liepins [5].

The optimization problem that underlies in the ELP has been shown to be \mathcal{NP} -hard [3] in Liepins, Garfinkel and Kunnathur [6] by a reduction to the “satisfiability” problem.

To formally introduce the optimization problem, let us suppose that there are n queries, indexed by a finite set $I := \{1, \dots, n\}$. Each record a from the microdata is a $n \times 1$ vector, say $a = [a_i : i \in I]$, whose component a_i is an entry in field i provided by a respondent. Given a set E of m edits, indexed by $J := \{1, \dots, m\}$, a record a is said to be *valid* (or *consistent*) if it satisfies every edit in E . The set of all valid records in accordance with the edit set E is denoted by \mathcal{P}_E . We shall assume through this article that the set of edits E is such that \mathcal{P}_E is not empty. To be more precise, let us consider a subset $S \subseteq I$ and a given record a . Let us denote by

$$\mathcal{P}_E(S, a) := \{y \in \mathcal{P}_E : y_i = a_i \text{ for all } i \notin S\}$$

the set of all valid records which may differ with respect to a in fields inside S . When convenient, S will be represented by a vector x such that $x_i = 1$ if $i \in S$ and $x_i = 0$ otherwise (the characteristic vector of S), and then $\mathcal{P}_E(S, a)$ will be also represented by $\mathcal{P}_E(x, a)$, i.e.

$$\mathcal{P}_E(x, a) := \{y \in \mathcal{P}_E : y_i = a_i \text{ if } x_i = 0, \text{ for each } i \in I\}$$

for all $x \in \{0, 1\}^n$. Then the ELP can be also defined as the combinatorial optimization problem

$$\min \{w^T x : \mathcal{P}_E(x, a) \neq \emptyset \text{ and } x \in \{0, 1\}^n\},$$

where $w = [w_i : i \in I]$ is the given non-negative weight vector, $a = [a_i : i \in I]$ is the original record, and $x = [x_i : i \in I]$ is the variable vector.

From now on we shall assume that each field value a_i is a real number in a known interval $[lb_i, ub_i]$ as done in Liepins, Garfinkel and Kunnathur [6]. A similar hypothesis is also assumed in McKeown [7], Schaffer [10], Garfinkel, Kunnathur and Liepins [4], and Ragsdale and McKeown [8], among other classical articles. This variation of the ELP is known as *Continuous Data Editing Problem* (CDEP). Moreover, we shall also assume that the explicit edit set E is given by a collection of linear constraints, and therefore it can be represented by a linear system $My \leq b$, where M is a $m \times n$ matrix and b is a $m \times 1$ vector. Whenever more detail is needed, the linear system will be represented by $\sum_{i \in I} m_{ij} y_i \leq b_j$ for all $j \in J$. We shall assume that $lb_i \leq a_i \leq ub_i$ for all $i \in I$, and $Ma \not\leq b$, that is, the record a does not meet the consistency rules. Under these assumptions, the set of valid records \mathcal{P}_E is given by the points of the polyhedron (see, for instance, Schrijver [11], for details on Polyhedral Theory and Polyhedral Combinatoric)

$$\mathcal{P}_E := \{y : My \leq b, lb \leq y \leq ub\}.$$

The purpose of this article is to develop a heuristic algorithm to solve this version of the CDEP. This heuristic algorithm is based on a local search approach which consist of a procedure to obtain an initial feasible solution and two families of neighborhoods for the intensification process. Some of this procedures are based on the Benders' decomposition described in the next section. Section III describes the heuristic approach in detail, Section IV studies the computational performance of this proposal, and finally the Conclusions summarizes this results.

II Model for the CDEP and Benders' Decomposition

In order to solve CDEP to optimality Riera and Salazar [9] introduces a new mixed integer linear model for CDEP which has the advantage of exploiting the bounds lb_i and ub_i to link a 0-1 variable x_i with a continuous variable y_i for each field i . Variable x_i assumes value 1 and only if the field i must be modified, and variable y_i takes the modified value of field i . Then the integer linear programming model is

$$\min \sum_{i \in I} w_i x_i, \quad (1)$$

subject to

$$\sum_{i \in I} m_{ij} y_i \leq b_j \quad \text{for all } j \in J \quad (2)$$

$$a_i - (a_i - lb_i)x_i \leq y_i \leq a_i + (ub_i - a_i)x_i \quad \text{for all } i \in I \quad (3)$$

$$x_i \in \{0, 1\} \quad \text{for all } i \in I. \quad (4)$$

Constraints (2) ensure that the new record y is in the valid set \mathcal{P}_E , and constraints (3) guarantee that the field i is not modified unless $x_i = 1$. We describe now how model (1)–(4) can be decomposed through Benders' decomposition [1] so as to take advantage of its particular structure.

Let us consider a solution $x^* = [x_i : i \in I]$. Let us assume for simplicity that x^* is an integer solution, then, according to Farkas' Lemma (see, e.g., Schrijver [11]) if x^* is a non-feasible solution for the polyhedron

$$\mathcal{P}_E(x^*, a) = \left\{ y : \begin{array}{ll} \sum_{i \in I} m_{ij} y_i \leq b_j & \text{for all } j \in J \\ a_i - (a_i - lb_i)x_i^* \leq y_i \leq a_i + (ub_i - a_i)x_i^* & \text{for all } i \in I \end{array} \right\},$$

then the inequality

$$\sum_{j \in J} \alpha_j b_j + \sum_{i \in I} \beta_i (a_i + (ub_i - a_i)x_i^*) - \sum_{i \in I} \gamma_i (a_i - (a_i - lb_i)x_i^*) \geq 0 \quad (5)$$

holds for all directions of the cone $\mathcal{C}_E := \{(\alpha, \beta, \gamma) : M^T \alpha + \beta - \gamma = 0, \alpha \geq 0, \beta \geq 0, \gamma \geq 0\}$. Thus, by simple operations on (5) we get the inequality

$$\sum_{i \in I} [\beta_i (ub_i - a_i) + \gamma_i (a_i - lb_i)] x_i \geq \alpha^T (Ma - b), \quad (6)$$

which is a valid constraint which must be met in order for x^* to make it feasible. This inequality is the so-called *Benders' cut*.

Using this families of inequalities Riera and Salazar [9] propose an exact algorithm which is able to find optimal solutions for wrong records with up to 100 fields and 40 edits. The purpose of this paper is to develop, improve and analyse a near optimal algorithm to solve bigger instances. The next section describe the main details of this proposal.

III General Algorithm

This section establishes the main idea of our heuristic proposal based in a local search with two families of neighbourhoods.

A *feasible solution* for the CDEP, for the record a and the edits set E , is given by a set $S \subseteq I$, such that $\mathcal{P}_E(S, a) \neq \emptyset$. The objective value for a feasible solution S is given by $\text{Obj}(S) := \sum_{i \in S} w_i$. Let us define a feasible solution S as *minimal feasible solution* if $\mathcal{P}_E(S \setminus \{i\}, a) = \emptyset$ for all $i \in S$.

The overall approach works as follows. An initial procedure creates a first feasible solution which could be non-minimal. Then an iterative process steers a minimal solution which will be locally optimal according to two neighbourhoods. The procedure applied to a feasible solution with the purpose of producing a new solution smaller objective value is known as *Intensification*.

III.1 Initial Solution

With the purpose of building an initial feasible solution, which will be improved later by the intensification procedures, we make use of the Benders' cuts described above. The initial feasible solution is composed by mean an iterative procedure which solves a linear programming problem in each iteration. This linear programming problem, called *master problem*, is enriched with the Benders' cuts generated from the last non-feasible solution in each iteration. This procedure is repeated as long as the obtained solution is not feasible.

Note that the described procedure produces an optimal solution for the CDEP in a finite number of steps; see Riera and Salazar [9]. However, since the purpose of this initial step is to get an initial feasible solution in a very short time, this procedure, as described above would take so long, and therefore would be unappropriated for our purposes. We propose the following changes in order to improve the performance.

1. The master problem will be initialized by an initial set of constraints considering the following set-covering inequalities

$$\sum_{i \in I_j} x_i \geq 1 \quad \text{for all } j \in J(a), \quad (7)$$

where the index set $I_j \subseteq I$ corresponds to the set of fields involved in the edit j , and $J(a) \subseteq J$ is the subset of indices of the explicit edits not satisfied by the record a . Hence, for each failed edit $j \in J(a)$ the set-covering constraint (7) imposes the modification of at least one of the fields in I_j .

2. After a certain number of iterations all those variables with value 1 are fixed. This means that, for each partial non feasible solution S obtained in the previous iteration, we introduce in the master problem the constraint

$$x_i = 1 \quad \text{for all } i \in S.$$

Table 1, described in details in Section IV, shows the behaviour of our proposal for computing the initial feasible solution on instances described in Riera and Salazar [9].

III.2 Intensification

The aim of the intensification procedures is to change a feasible solution into a minimal feasible solution. With this purpose in mind we propose two procedures: *Reduction* and *Insertion-Reductions*.

```

function MINIMAL( $E, a, S, \sigma$ ):  $S' \subseteq S$ 
   $S' \leftarrow S$ 
  for  $i \leftarrow 1, |S|$  do
    if  $\mathcal{P}_E(S' \setminus \{\sigma(i)\}, a) \neq \emptyset$  then
       $S' \leftarrow S' \setminus \{\sigma(i)\}$ 
    end if
  end for
end function

```

Figure 1: Function MINIMAL

```

function REDUCTION( $E, a, S, \Sigma$ ):  $S' \subseteq S$ 
   $S' \leftarrow S$ 
  for all  $\sigma \in \Sigma$  do
     $S'' \leftarrow \text{MINIMAL}(E, a, S, \sigma)$ 
    if  $\text{OBJ}(S'') < \text{OBJ}(S')$  then
       $S' \leftarrow S''$ 
    end if
  end for
end function

```

Figure 2: Function REDUCTION

Reduction

The Reduction procedure is based on a simple procedure described in Figure 1, called *Minimal*. This procedure obtains a minimal feasible solution as follows:

Given a permutation $\sigma := \{\sigma(1), \sigma(2), \dots, \sigma(|S|)\}$ from the items of a feasible solution S , the procedure *Minimal* obtains a new solution S' from S extracting as many items as possible, following the order induced by σ , as long as the obtained solutions remains feasible.

This procedure induces a neighbourhood for some set $\Sigma := \{\sigma_1, \sigma_2, \dots, \sigma_l\}$ of permutations from the items of the set S . Then, the procedure *Reduction* computes the objective value for each solution obtained from each permutation σ from the set Σ , and returns that solution with minimum objective value.

Insertion-Reduction

Given a minimal feasible solution S , the procedure *Insertion-Reduction* tries to find a new minimal feasible solution with lower objective value, checking whether it is possible to replace two items i, j from S by some $l \in I \setminus S$, such that the resulting new solution is feasible. That is, decrease the objective value in one unit.

However, taking into account that the cardinality of a minimal feasible solution S usually is a low value, the procedure checking the feasibility for each item $l \in I \setminus S$ takes too much computational time. With the aim of reducing the time consumed by this procedure, we restrict the search to a subset $T \subseteq I \setminus S$ which is built from the Benders' cuts described in Section II as follows.

Since $\beta_i, \gamma_i, ub_i - a_i, a_i - lb_i$ are non-negative numbers and since x_i must be either 0 or 1, it is possible to strengthen (6) by rounding down each left-hand-side coefficient to the right-hand-side value. That is, if we denote by $\mu x \geq \eta$ the new cut currently generated by

the subproblem, then the inequality

$$\bar{\mu}x \geq 1 \text{ with } \bar{\mu}_i := \frac{\min\{\mu_i, \eta\}}{\eta} \text{ for all } i \in I \quad (8)$$

dominates the Benders' cut (6). Observe that the set-covering inequality obtained rounding up the inequality (8)

$$[\bar{\mu}]x \geq 1 \quad (9)$$

is valid as well, but weaker than (8). This last inequality means that at least a variable implied in this Benders' cut must have value 1 in order to make this system feasible, and therefore, the remaining variables not involved in (9) do not have any influence as far as feasibility concerns.

Figure 3 describes the algorithm 1-REDUCTION, in which all possible pairs $\{i, j\}$ ($i, j \in S \wedge i < j$) from a minimal feasible solution S are removed with the purpose of studying whether it is possible to restore the feasibility inserting one single item from $I \setminus S$. The function $\mathcal{B}_E(S \setminus \{i, j\}, a)$ of the non-feasible set $S \setminus \{i, j\}$, the edit set E , and the record a , returns the set of variable indices T of the non-zero coefficients of the Benders' cut (9). In other words, $\mathcal{B}_E(S \setminus \{i, j\}, a) = \{i \in I : \beta_i(ub_i - a_i) + \lambda_i(a_i - lb_i) > 0\}$.

Since the solution obtained by the procedure 1-REDUCTION has not to be feasible minimal the procedure *Insertion-Reduction* calls the procedure *Reduction* as well. Figure 4 show the full algorithm. This algorithm is repeated while the procedure 1-REDUCTION produces a reduction in the objective value.

Table 2, described in Section IV shows an important reduction of the computational time when using the restricted set $T \subseteq I \setminus S$ instead of the full set $I \setminus S$.

```

function 1-REDUCTION( $E, a, S$ ):  $S' \subseteq I$ 
   $S' \leftarrow \emptyset$ 
  for all  $\{i, j\} : i, j \in S \wedge i < j$  do
     $T \leftarrow \mathcal{B}_E(S \setminus \{i, j\}, a) \setminus \{i, j\}$ 
    for all  $l \in T$  do
      if  $\mathcal{P}_E((S \cup \{l\}) \setminus \{i, j\}, a) \neq \emptyset$  then
         $S' \leftarrow (S \cup \{l\}) \setminus \{i, j\}$ 
        Break
      end if
    end for
  end for
end function

```

Figure 3: Function 1-REDUCTION

```

function REDUCTION-INSERTION( $E, a, S, \Sigma$ ) :  $S' \subseteq I$ 
  repeat
     $S' \leftarrow S$ 
     $S \leftarrow$ 1-REDUCTION( $E, a, S$ )
    if  $|S| < |S'|$  then
       $S \leftarrow$ REDUCTION( $E, a, S, \Sigma$ )
    end if
  until  $\neg(|S| < |S'|)$ 
end function

```

Figure 4: Function REDUCTION-INSERTION

IV Computational Experience

With the purpose of evaluating our proposal we have carried out an extensive computational experience considering several aspects of our algorithm.

Since there is not a benchmark collection of test bed instances in the literature, we have conducted our experiments on two classes of instances also used in Riera and Salazar [9]. Class I contains the 25 instances proposed in Ragsdale and McKeown [8], specifically designed to fail a large number of edits. These instances have been randomly generated with the following characteristics: the number of fields is $|I| = n = 50$; the number of edits is $|J| = m = 20$; the weights are all identical ($w_i = 1$ for all $i \in I$); the right-hand side b_j is generated in the interval $[0,1000]$; the elements m_{ij} are zero with probability 0.2, and the non-zero values are generated in $[1,20]$ with probability 0.3 and in $[-20,-1]$ with probability 0.7; the record values are uniformly generated in the interval $[-10000, +10000]$, thus $lb_i = -10000$ and $ub_i = 10000$ for all $i \in I$. (These features are not explicitly specified in Ragsdale and McKeown [8] but they are in their FORTRAN code used for testing purposes and kindly provided by Cliff Ragsdale.)

Those generated records satisfying the whole edit set were removed, and the remaining records were classified according to the number of failed explicit edits into five groups: $[1, 4]$, $[5, 8]$, $[9, 12]$, $[13, 16]$ and $[17, 20]$. The random generator was repeatedly executed until we got five instances per group, as described in [8].

Because of the small size of the previously described instances, no conclusive results were obtained and therefore we found it appropriate to generate a second class of instances. Instances from Class II were generated as Class I, but doubling their size, that is, with $|I| = 100$ and $|J| = 40$. Furthermore, three families inside Class II were also generated by varying the range of the variable bounds in the intervals $[-10^3, 10^3]$, $[-10^4, 10^4]$ and $[-10^5, 10^5]$ respectively, in order to study the influence of these bounds on the performance of the algorithms. These three families are named a , b and c , respectively. Again, five random instances failing a number of edits in the ranges $[1, 8]$, $[9, 16]$, $[17, 24]$, $[25, 32]$ and $[33, 40]$ are considered.

The above described algorithm have been implemented in the C++ programming language and our experiments have been executed on a laptop computer Centrino 1.6 GHz under Windows XP. We use Cplex 9.0 to solve the linear programs and to obtain the Benders' cuts.

During this computational experience, the results are summarized averaging the values for each five instances group of each range of failing edits. The optimal values have been obtained running the exact algorithm proposed in [9] on the same computer.

The algorithm proposed in Section III.1 to obtain an initial feasible solution is studied in Table 1. This table shows, for each class of instance, the average gap (LB) between the lower bound (lb) obtained as result of the model initialized with the constraints (7) and the optimal solution (opt), computed as $\frac{opt-lb}{opt}$; the average gap (UB) between the initial feasible solution (ub) obtained by the procedure and the optimal solution (opt), computed as $\frac{ub-opt}{opt}$; the average number of iterations ($\#$) during the computation of the initial feasible solution; and the average time in seconds (Sec) consumed by the procedure.

Table 2 shows the behaviour of the algorithm *Insertion-Reduction* described in Section III.2. In particular, it shows the clear reduction in both the number of iterations and the computing time with the improvement based of the Benders' cuts. For each instance this procedure has been run with and without that improvement. Although the number of studied pairs coincides for both options, the number of iteration and the computing time show an important reduction. This table shows, for each class of instances, the average number of pair evaluations ($\#Eval$), which is common for the improved and non-improved algorithm; the ratio ($\#it$) between the average number of insertion evaluations with (it) and without

(it') improvement computed as $\frac{it'-it}{it}$; the ratio (Sec) between the average computing times consumed in both cases.

Table 3 describes the intensification procedure behaviour. With this purpose, it shows the average gap between the optimal and heuristic solutions after each phase. First, the average gap between the initial feasible solution and the optimal solution (IFS), computed as $\frac{ifs-opt}{opt}$. Second, the average gap between the optimal solution and the heuristic solution after the procedure *Reduction* (MEP). And finally, the average gap between the optimal solution and the final heuristic solution (INT). In addition, it also shows the computing time consumed by the intensification procedure in seconds (Sec).

Table 4 compares the exact algorithm and our heuristic proposal. The two first columns show the goodness of the solution provided by the heuristic. The first column (Abs) shows the absolute difference between the optimal and the heuristic solution, and the second column shows the gap between the heuristic and the optimal solution computed as the column IFS in Table 3. The main reason of showing the absolute difference Abs is that, since the optimal values is usually very small, a single unit difference with the heuristic result appears as a big increase in the gap IFS . The following two columns compare both computational time: average time consumed by the heuristic algorithm (Heu) and average time consumed by the exact algorithm (Opt).

V Conclusions

The computational experience shows a competitive behaviour of this proposal on these families of instances, since it obtains good quality solutions in a very short time. This solutions are not more than one unit far over the optimal in average, and the computing time never overcomes half a second. This computational experience also shows that the Benders' decomposition applied to the produce the initial feasible solution and to restrict the search set in the *Insertion-Reduction* produce an important reduction in the computing time.

References

- [1] J. F. Benders. Partitioning procedures for solving mixed-variables programming problems. *Numerische Mathematik*, 4:238–252, 1962.
- [2] I. P. Fellegi and D. Holt. A systematic approach to automatic edit and imputation. *Journal of the American Statistical Association*, 71:17–35, 1976.
- [3] M. R. Garey and D. S. Johnson. *Computers and intractability, a guide to the theory of NP-Completeness*. W. H. Freeman and Company, New York, 1979.
- [4] R. S. Garfinkel, A. S. Kunnathur, and G. E. Liepins. Error location for erroneous data: continuous data, linear constraints. *SIAM Journal on Scientific and Statistical Computing*, 9:922–931, 1988.
- [5] G. E. Liepins. A rigorous, systematic approach to automatic data editing and its statistical basis. Technical report ORNL/TM -7126, Oak Ridge National Laboratory, 1980.
- [6] G. E. Liepins, R. S. Garfinkel, and A. S. Kunnathur. Error localization for erroneous data: a survey. *TIMS/Studies in the Management Sciences*, 19:205–219, 1982.
- [7] P. G. McKeown. A mathematical programming approach to editing of continuous survey data. *SIAM Journal on Scientific and Statistical Computing*, 5:785–797, 1984.

Table 1: Initial feasible solution.

Class I				Class IIa				Class IIb				Class IIc			
LB	UB	#	Sec	LB	UB	#	Sec	LB	UB	#	Sec	LB	UB	#	Sec
0.00	0.57	5.8	0.03	0.51	0.33	12.2	0.14	0.17	0.45	9.2	0.07	0.17	0.67	20.0	0.13
0.15	0.10	3.6	0.02	0.48	0.66	15.8	0.15	0.21	0.35	12.8	0.09	0.20	0.37	12.0	0.08
0.00	0.10	2.2	0.01	0.60	0.29	16.2	0.18	0.32	0.33	13.4	0.10	0.29	0.20	13.8	0.09
0.12	0.08	2.6	0.02	0.68	0.29	18.6	0.20	0.33	0.43	18.8	0.16	0.31	0.43	19.8	0.14
0.04	0.12	1.8	0.02	0.66	0.22	18.6	0.24	0.39	0.24	13.8	0.12	0.33	0.30	20.4	0.17
0.06	0.19	3.2	0.02	0.59	0.36	16.3	0.18	0.28	0.36	13.6	0.11	0.26	0.39	17.2	0.12

Table 2: Insertion-Reduction.

Class I			Class IIa			Class IIb			Class IIc		
#Ev	#it	Sec	#Ev	#it	Sec	#Ev	#it	Sec	#Ev	#it	Sec
2.0	0.09	0.41	23.8	0.12	0.32	7.8	0.12	0.36	8.6	0.13	0.32
5.6	0.10	0.34	49.0	0.12	0.28	16.8	0.12	0.31	15.6	0.13	0.31
8.8	0.10	0.40	56.2	0.13	0.30	22.4	0.16	0.37	14.0	0.13	0.34
11.0	0.11	0.32	110.8	0.16	0.38	22.6	0.14	0.32	25.8	0.14	0.32
11.8	0.09	0.36	132.4	0.19	0.41	31.8	0.16	0.34	18.6	0.14	0.32
7.8	0.10	0.37	74.4	0.14	0.34	20.3	0.14	0.34	16.5	0.11	0.32

Table 3: Intensification.

Class I				Class IIa				Class IIb				Class IIc			
IFS	MEP	INT	Sec	IFS	MEP	INT	Sec	IFS	MEP	INT	Sec	IFS	MEP	INT	Sec
0.57	0.50	0.10	0.04	0.33	0.27	0.05	0.35	0.45	0.45	0.17	0.15	0.67	0.67	0.07	0.23
0.10	0.10	0.00	0.03	0.66	0.56	0.45	0.67	0.35	0.26	0.04	0.25	0.37	0.33	0.08	0.23
0.10	0.10	0.05	0.04	0.29	0.24	0.09	0.89	0.33	0.29	0.19	0.33	0.20	0.20	0.04	0.23
0.08	0.08	0.04	0.05	0.29	0.17	0.08	0.97	0.43	0.37	0.07	0.45	0.43	0.37	0.11	0.35
0.12	0.08	0.00	0.05	0.22	0.15	0.08	1.27	0.24	0.24	0.07	0.43	0.30	0.20	0.10	0.38
0.19	0.17	0.04	0.04	0.36	0.28	0.15	0.83	0.36	0.32	0.11	0.32	0.39	0.35	0.08	0.28

Table 4: Overall Algorithm.

Class I				Class IIa				Class IIb				Class IIc			
Abs	Gap	Heu	Opt	Abs	Gap	Heu	Opt	Abs	Gap	Heu	Opt	Abs	Gap	Heu	Opt
0.2	0.10	0.04	0.04	0.4	0.05	0.35	2.59	0.6	0.17	0.15	0.48	0.2	0.07	0.23	0.39
0.0	0.00	0.03	0.02	3.0	0.45	0.67	83.85	0.2	0.04	0.25	6.94	0.4	0.08	0.23	4.30
0.2	0.05	0.04	0.03	0.8	0.09	0.89	129.88	1.0	0.19	0.33	8.10	0.2	0.04	0.23	1.68
0.2	0.04	0.05	0.16	1.0	0.08	0.97	307.72	0.4	0.07	0.45	17.56	0.6	0.11	0.35	2.46
0.0	0.00	0.05	0.16	1.0	0.08	1.27	165.14	0.4	0.07	0.43	26.70	0.6	0.10	0.38	3.67
0.1	0.04	0.04	0.08	1.2	0.15	0.83	137.83	0.5	0.11	0.32	11.95	0.4	0.08	0.28	2.50

[8] C. T. Ragsdale and P. G. McKeown. On solving the continuous data editing problem. *Computers & Operations Research*, 23:263–273, 1996.

[9] J. Riera-Ledesma and J. J. Salazar-González. Algorithms for the automatic data editing. *Statistical Journal of the United Nations Economic Commission for Europe*, 20(3–4):255–264, 2003.

[10] J. Schaffer. Procedure for solving the data-editing problem with both continuous and discrete data types. *Naval Research Logistics*, 34:879–890, 1987.

[11] A. Schrijver. *Theory of linear and integer programming*. John Wiley & Sons, 1986.